

Manuale di Signal Processing Scientifico

Edizione di luglio 2024

Un manuale illustrato con software gratuito e fogli di calcolo da scaricare

Un progetto di
Tom O'Haver
Professore Emerito
[Dipartimento di Chimica e Biochimica](#)



orcid.org/0000-0001-9045-1603



Accesso online alle ultime versioni

Libro (in inglese) in formato PDF:

<https://terpconnect.umd.edu/~toh/spectrum/IntroToSignalProcessing.pdf>



Libro (in italiano) in formato PDF:

https://github.com/BravoBaldo/IntroToSignalProcessing_it



Indirizzo Web: <http://bit.ly/1NLOILR>



Tool Interattivi Matlab: <http://bit.ly/1r7oN7b>



Link per il download del software: <http://tinyurl.com/cey8rwh>



Esempi animati: <https://terpconnect.umd.edu/~toh/spectrum/ToolsZoo.html>



Leggendo questo libro su un computer o un tablet connessi a Internet, si può toccare o cliccare (Ctrl-Click) su qualsiasi numero di pagina nel testo per saltare direttamente a quella pagina. Con [Microsoft Word 365](#), le animazioni GIF verranno eseguite automaticamente. Nella maggior parte dei visualizzatori PDF gratuiti, è necessario cliccare sui link dei GIF per visualizzare le animazioni. È inoltre possibile cliccare sugli indirizzi https, sui nomi dei software o sulle figure per visualizzare, ingrandire o scaricare tali elementi.

Ci sono domande o suggerimenti? Scrivere all'indirizzo toh@umd.edu

C'è un gruppo Facebook a cui unirsi: "Pragmatic Signal Processing"

Ringraziamenti. Grazie ai miei [studenti laureati](#), molti dei quali hanno lavorato con le tecniche qui descritte, a [M. Farooq Wahab](#) per i suoi numerosi contributi e per le tante fruttuose discussioni, a [Baldassarre Cesaran](#) per l'attenta lettura e correzione tipografica di questo testo, al [Dr. Raphael Attié](#) della NASA/Goddard Space Flight Center per le correzioni, a [Diederick](#) dell'Università di Hong Kong per i contributi al codice, a [Yuri Kalambet](#) di Ampersand, Ltd. per i tanti suggerimenti, correzioni e idee, e ai molti corrispondenti di posta elettronica che hanno fornito suggerimenti, posto domande, rilevato errori, inviato esempi dei loro dati e che mi hanno mostrato nuove aree di applicazione che hanno ampliato l'ambito di questo lavoro.

Tabella dei Contenuti

Introduzione	9
Aritmetica del segnale	12
Aritmetica dei segnali nei Fogli di Calcolo	14
Aritmetica e plottaggio dei segnali in Matlab	15
Import dei dati in Matlab/Octave e in Python.	18
Le Versioni di Matlab.	19
Matematica in Python.....	20
GNU Octave	20
Spreadsheet o Matlab/Python?	21
Segnali e rumore	22
Il limite del rilevamento	25
Media di insieme	27
Distribuzione in frequenza del rumore casuale	28
Dipendenza dall'ampiezza del segnale	29
La distribuzione di probabilità del rumore casuale	30
Rappresentazione del rumore casuale negli Spreadsheet	31
Funzioni random in Matlab e in Python	32
Differenza tra script e funzioni	33
I Live Script.....	35
Funzioni utente relative ai segnali e al rumore.....	36
Il ruolo della simulazione e della modellazione.	38
Smoothing.....	38
Algoritmi di smoothing	39
Riduzione del rumore	40
Effetto della distribuzione in frequenza del rumore	41
Effetti agli estremi e il problema dei punti persi	41
Esempi di smoothing	42
Il problema con lo smoothing	43
Ottimizzazione dello smoothing	45

Quando si dovrebbe applicare lo smoothing ad un segnale?.....	46
Quando NON si dovrebbe applicare lo smoothing ad un segnale?	46
Gestire gli spike e i valori anomali.....	47
Media dell'Insieme	48
Condensare i segnali sovra-campionati	49
Lo smoothing con gli spreadsheet	49
Lo smoothing in Matlab e in Octave	52
Smoothing segmentato	52
Altre funzioni di smoothing.....	53
Altre funzioni per la riduzione del rumore.....	54
Live Script per lo smoothing	55
Confronto delle prestazioni dello smoothing	56
Differenziazione	58
Proprietà Fondamentali delle Derivate dei Segnali	60
Applicazioni della Differenziazione	62
Rilevamento dei picchi	64
Spettroscopia Derivativa (o Differenziale).....	64
Analisi delle Tracce	66
Derivate e Rumore: L'importanza dello Smoothing.....	68
Differenziazione nei fogli di calcolo	70
Differenziazione in Matlab e in Python.....	71
Peak Sharpening	75
Sharpening con derivata pari	75
Simmetrizzazione della derivata prima con area costante (“de-tailing”)	77
Il metodo della Legge della Potenza	79
Peak Sharpening per fogli di calcolo Excel e Calc.....	81
Sharpening dei picchi con Matlab e Octave	83
Analisi delle armoniche e la Trasformata di Fourier.	87
Dettagli Software	96
Matlab e Octave.....	97
Spettro di Fourier della potenza segmentato nel tempo.	98
Osservazione dello spettro di Frequenza con iSignal.....	99
Visualizzazione della frequenza.	100
Miglioramento del segnale	101
Dimostrazione che lo spettro di Fourier di una Gaussiana è anche una Gaussiana	102
Convoluzione di Fourier.....	103
Semplici vettori di convoluzione a numeri interi	104
Dettagli software per la convoluzione.....	105
Convoluzione sequenziale multipla.....	106
Deconvoluzione di Fourier	108
Software per la deconvoluzione	111
Matlab e Octave.....	111

Riduzione del rumore nei segnali deconvoluti	115
Riduzione del rumore in eccesso mediante aggiunta al denominatore.....	116
Deconvoluzione per la misura dell'area dei picchi	118
Deconvoluzione sequenziale multipla.....	118
Deconvoluzione segmentata.....	119
Live script del tool per l'Auto-deconvoluzione	120
Deconvoluzione interattiva con iSignal.....	121
Filtro di Fourier	122
Software per il filtraggio di Fourier.....	123
Wavelet e wavelet denoising	126
Visualizzazione ed analisi.....	128
Riduzione del rumore con le wavelet	130
Integrazione e misura dell'area di un picco	133
La gestione dei picchi sovrapposti.....	135
Misura dell'area di un picco con gli spreadsheet.....	138
Sharpening per la misura dell'area di picchi sovrapposti	138
Misura dell'area di un picco con Matlab e Octave	139
Rilevamento automatico di più picchi	141
Misura dell'area col curve fitting iterativo.....	146
Correzione del background/linea di base	146
Picchi asimmetrici e ampliamento del picco: taglio verticale rispetto al curve fitting.....	149
Approssimazione delle curve A: Quadrati minimi lineare	154
Esempi di approssimazioni polinomiali	155
Affidabilità dei risultati col curve fitting	159
Propagazione algebrica degli errori	159
Simulazione Monte Carlo	161
Il metodo Bootstrap	162
Confronto dei metodi per la previsione dell'errore	163
Effetto del numero dei punti sulla precisione dell'approssimazione con i minimi quadrati..	163
Trasformare relazioni non-lineari	164
Semplice approssimazione di picchi Gaussiani e Lorentziani con trasformazione dei dati ..	165
Dettagli matematici e software per i quadrati minimi lineare	169
Spreadsheet per i quadrati minimi lineari.....	170
Applicazioni per la calibrazione analitica e la misura.....	172
Matlab e Octave.....	172
Approssimazione dei picchi con Gaussiana o Lorentziana singola.....	178
Approssimazione delle curve B: Spettroscopia Multicomponente.....	179
Calibrazione multivariata col metodo dei minimi quadrati classico (CLS)	180
Calibrazione inversa dei minimi quadrati (ILS)	182
Software per la spettroscopia a lunghezze d'onda multiple	183

Spreadsheet (Fogli di calcolo)	183
Matlab/Octave e Python	185
I Quadrati Minimi Classici in Python.....	189
Approssimazione delle curve C: Approssimazione iterativa non-lineare.....	190
Spreadsheet e programmi autonomi	192
Matlab e Octave.....	195
Approssimazione dei picchi	195
Funzione semplificata di approssimazione del picco per scopi generali.....	197
Tipi di forma variabile.....	197
Funzioni di Fitting per Matlab e Octave.....	199
Accuratezza e precisione dei parametri del picco	202
a. Modelli errati.	202
b. Correzione del background.....	211
c. Rumore casuale nel segnale.....	214
d. Errori nell'approssimazione iterativa.....	217
Un caso difficile.	218
Approssimazione dei segnali soggetti ad ampliamento [broadening] esponenziale.	220
L'Effetto dello Smoothing prima dell'analisi dei minimi quadrati	226
Ricerca e Misura dei Picchi	227
Semplice rilevamento dei picchi	228
Misura del picco Gaussiano.....	230
Ottimizzazione della ricerca dei picchi	232
In che modo ‘findpeaksG’ è diverso da ‘max’ in Matlab o ‘findpeaks’ nel <i>Signal Processing Toolkit?</i>	233
Ricerca degli avvallamenti	233
Accuratezza delle misure dei picchi	234
Ricerca del picco con l'approssimazione iterativa.....	234
Confronto delle funzioni per la ricerca dei picchi	237
Inizio e fine del picco	239
Utilizzo della tabella dei picchi	242
Script dimostrativi	243
Identificazione del Picco	244
Tool Live Script per il Rilevamento del Picco	245
<i>iPeak</i> : Rilevatore di picchi interattivo azionato tramite tastiera	246
<i>iPeak</i> keyboard Controls (version 8.1):.....	260
<i>iPeak</i> Funzioni demo	261
Peak Finder con Template di Fogli di Calcolo	265
Spettrofotometria di Assorbimento Quantitativo Iperlineare	268
Background.....	270
Implementazione dello spreadsheet.....	274
Implementazione Matlab/Octave: La funzione fitM.m	275
Funzione demo per Octave o Matlab.....	277

TFitDemo.m: Demo interattiva per il metodo Tfit	278
Statistiche dei metodi a confronto (TFitStats.m, per Matlab o Octave)	279
Confronto delle curve analitiche (TFitCalDemo.m, per Matlab o Octave)	280
Applicazione su una miscela a tre componenti	281
Tutorial, Casi di Studio e Simulazioni.....	283
Il rumore con smoothing può essere confuso con l'effettivo segnale?	283
Segnale o Rumore?	284
Il tesoro sepolto	287
Il Torneo: un confronto dei metodi	291
Schemi di media di insieme in un segnale continuo.....	293
Analisi delle Armoniche dell'Effetto Doppler	295
Misura degli spike	296
Deconvoluzione di Fourier rispetto all'approssimazione della curva [curve fitting] (<i>non è la stessa cosa</i>).....	299
Rumore di digitalizzazione - l'aggiunta di rumore può davvero aiutare?.....	300
Quanto in basso si può scendere? Prestazioni con rapporti segnale/rumore molto bassi.....	302
Elaborazione dei segnali nella ricerca dell'intelligenza extraterrestre	304
Perché misurare l'area anziché l'altezza del picco?	306
Utilizzo delle macro per estendere le capacità degli spreadsheet.....	308
Passeggiate aleatorie (random walk) e correzione della linea di base.....	310
Modulazione e rilevamento sincrono.	312
La misura di un picco sepolto.....	315
Segnale e Rumore nel Mercato Azionario (aggiornato)	319
Misura del rapporto segnale/rumore nei segnali complessi	322
Gestione di segnali ad ampio raggio: Elaborazione segmentata	326
Calibrazione della Misura.....	329
Precisione numerica del software	332
Elaborazione miniaturizzata del signal processing: La Famiglia del Raspberry Pi	335
Elaborazione batch	337
Il signal processing in tempo reale	339
Sharpening del picco	343
Gestione degli array di dati variabili negli spreadsheet.....	345
Illuminare l'invisibile: Simulazione di strumenti col computer	347
Chi usa questo libro, il relativo sito web, i documenti e il software?	350
La Legge dei Grandi Numeri.....	353
Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo	354
La sfida del picco misterioso	358
Sviluppo di Script Live Matlab e App	360
Uso della modellazione del segnale reale per determinare l'accuratezza della misura.	364
Dettagli software del signal processing.	366
Smoothing, differenziazione e analisi del segnale interattivamente (iSignal).....	366

Filtro di Fourier gestito da tastiera	380
Peak Fitter Matlab/Octave.....	384
Funzione a riga di comando in Matlab/Octave: peakfit.m	384
Esempi	389
Come trovare gli argomenti di input corretti per peakfit?	401
Lavorare con la matrice dei risultati dell'approssimazione "FitResults"	401
Script dimostrativo per peakfit.m	402
Approssimazione di picchi in dati multi-colonna.....	403
Gestire i segnali complessi con molti picchi	403
Ricerca automatica e Approssimazione dei picchi	404
Peak Fitter Interattivo Gestito da Tastiera (ipf.m)	405
<i>ipf</i> controlli da tastiera (Versione 13.4): Ottenuti premendo il tasto K	407
Esempi pratici con dati sperimentali reali:	409
Istruzioni sull'uso di ipf.m (versione 13.4).....	411
Demoipf.m.....	418
Tempo di esecuzione dell'approssimazione ed altri compiti del signal processing	420
Suggerimenti e consigli per il Curve Fitting Iterativo.....	420
Estrazione delle equazioni per i modelli migliori.....	423
Come aggiungere un nuovo profilo in peakfit.m, ipf.m, iPeak, o iSignal	424
Quale usare? <i>peakfit</i> , <i>ipf</i> , <i>findpeaks</i> ..., <i>iPeak</i> o <i>iSignal</i> ?	425
Tool Live Script Peak Fitter	427
Python: un linguaggio alternativo gratuito, open-source	429
Smoothing a media mobile del segnale	429
Trasformazione di Fourier e (de)convoluzione	431
I Quadrati Minimi Classici	431
Rilevamento dei picchi	432
Approssimazione ai Quadrati minimi interattiva	432
Intelligenza Artificiale ed Elaborazione dei Segnali	434
IA come assistente di un programmatore.	434
Fogli di calcolo per le Curve di Calibrazione Analitiche	438
Background.....	438
Compilazione dei fogli di calcolo per diversi metodi di calibrazione.....	439
Confronto dei metodi di calibrazione	442
Istruzioni per l'utilizzo dei template per la calibrazione.....	443
Domande frequenti (tratte da email e query sui motori di ricerca)	445
Catalogo delle funzioni, script e spreadsheet per il signal processing	451
Funzioni per il profilo dei picchi (per Matlab e Octave).....	451
Aritmetica del Segnale	453
Segnali e Rumore	455
Smoothing.....	457
Differenziazione e sharpening	459
Analisi delle armoniche	461

Convoluzione e deconvoluzione di Fourier.....	462
Filtro di Fourier	463
Wavelet ed eliminazione del rumore	464
Misura dell'area del picco	464
I minimi quadrati lineare	466
Ricerca e Misura dei Picchi	468
Spettrosopia Multicomponente	474
Approssimazione iterativa non lineare e approssimazione dei picchi	475
Funzioni <i>interattive</i> operanti da tastiera	479
Spettrofotometria di Assorbimento Quantitativo Iperlineare	480
File MAT (per Matlab e Octave) e i file Testo (.txt).....	480
Spreadsheet (per Excel e OpenOffice Calc)	480
Epilogo	484
Come è nato questo libro.....	484
Chi ha bisogno di questo software?.....	485
Organizzazione	485
Metodologia.....	486
L'influenza di Internet.....	486
Lo Stile	486
Criterio di selezione della piattaforma software.....	487
Risultati.....	488
Impatto	489
Il Futuro	489
Riferimenti.....	489
Pubblicazioni che citano questo libro, i programmi e/o la documentazione	493

Introduzione

L'interfacciamento di strumenti di misura con piccoli computer per l'acquisizione diretta dei dati è diventata ormai una pratica standard nel moderno laboratorio di scienze. I computer vengono utilizzati per l'acquisizione, l'elaborazione e l'archiviazione dei dati, utilizzando i metodi numerici digitali. Sono disponibili tecniche in grado di trasformare i segnali in formati più utili, rilevare e misurare picchi, ridurre il rumore, migliorare la risoluzione di picchi sovrapposti, compensare artefatti strumentali, testare ipotesi, ottimizzare strategie di misura, diagnosticare difficoltà nelle misurazioni, visualizzare e scomporre segnali complessi nelle loro componenti. Queste tecniche spesso facilitano misure difficili estraendo molte più informazioni dai dati a disposizione. Molte di queste tecniche utilizzano laboriose procedure matematiche che non erano nemmeno praticabili prima dell'avvento dei computer. È importante apprezzare le capacità, *così come le limitazioni*, di tali tecniche. Negli ultimi decenni, tuttavia, l'archiviazione e l'elaborazione digitale è diventata molto meno costosa e letteralmente milioni di volte più capace, riducendo il costo dei dati originali e rendendo le complesse tecniche digitali per l'elaborazione dei segnali più pratiche e necessarie. Le approssimazioni e le scorciatoie, un tempo inevitabili per comodità matematica, non sono più

necessarie (p. es. 135, 190, 268). E non è solo l'evoluzione dei computer: ora ci sono nuovi materiali, nuovi strumenti, nuove tecniche di costruzione, nuove capacità di automazione. Abbiamo laser, fibre ottiche, superconduttori, super-magneti, ologrammi, tecnologia quantica, nanotecnologie e ora anche la promessa dell'intelligenza artificiale (pag. 434). I sensori sono più piccoli, economici e più veloci che mai; si possono effettuare misure su gamme più ampie di velocità, temperature, pressioni e posizioni. Le persone si portano appresso smartphone e fitness trackers ovunque vadano, registrano la frequenza cardiaca, ecc., creando nuovi insiemi di dati che prima non c'erano. Come hanno scritto Erik Brynjolfsson e Andrew McAfee in *The Second Machine Age* (W. W. Norton, 2014): "... man mano che i dati diventano più economici, il collo di bottiglia è sempre più la capacità di interpretare e utilizzare i dati". [Kate Keahey](#), una Senior Scientist dell'[Argonne National Laboratory](#), scrive che "Il software è una parte vitale nel mondo della ricerca, e la maggior parte dei ricercatori beneficeranno dalla comprensione delle possibilità, delle limitazioni e dei presupposti per la sua costruzione".

Questo libro copre solo argomenti di base relativi a segnali mono-dimensionali, non dati bidimensionali come le immagini. Utilizza un approccio *pragmatico* e la matematica è limitata agli aspetti più elementari del calcolo, della statistica e delle matrici. Si usano argomenti logici, analogie, grafici e animazioni per spiegare le idee, anziché tanta matematica formale. Elaborazione dei dati senza matematica? Non proprio! La matematica è essenziale, così come lo è la tecnologia per i cellulari, il GPS, la fotografia digitale, il Web, i videogiochi e le auto moderne. Ma tali strumenti si possono iniziare ad utilizzare senza comprenderne tutti i dettagli matematici e software. Vederli in funzione probabilmente farà venir voglia di capire *come* funzionano. Tuttavia, alla fine, non è sufficiente sapere come far funzionare il software, così come il saper usare un word processor o un sequencer MIDI non fa diventare scrittori o musicisti. Si *inizia* con cose che funzionano; spetta al lettore decidere poi se sia necessario approfondire gli argomenti più avanzati.

Perché questo piccolo documento si chiama "elaborazione dei segnali" anziché "elaborazione dei dati"? Per "segnaile" si intendono i dati numerici delle serie temporali x,y registrati da strumenti scientifici, dove x può essere il tempo o un'altra quantità come l'energia o la lunghezza d'onda, come avviene nei vari tipi di spettroscopia. Questi dati vengono talvolta chiamati "linee ondulate" [squiggly line]. Non ci si occupa tanto di dati categorici. In altre parole, siamo orientati ai dati che si traccerebbero su un foglio di calcolo con i grafici a dispersione anziché grafici a barre o a torta.

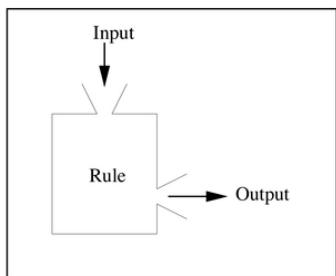
Alcuni degli esempi provengono dalle mie aree di ricerca nella chimica analitica, ma queste tecniche sono state utilizzate in un'ampia gamma di applicazioni. [Oltre 750 articoli](#), tesi e brevetti hanno citato questo libro e il suo software, dal mondo accademico, industriale, ambientale, medico, ingegneristico, scienza della terra, spazio, militare, finanziario, agricoltura, comunicazioni e persino musica e scienze del linguaggio. I suggerimenti e i dati sperimentali inviati dai lettori di centinaia di lettori hanno contribuito a plasmare la scrittura e lo sviluppo del software. Sono stati fatti molti sforzi per rendere questo documento conciso e comprensibile; esso è stato [accolto molto positivamente da molti lettori](#).

Al momento, questo lavoro non copre l'elaborazione delle immagini, il riconoscimento di pattern né l'analisi dei fattori. Per questi argomenti e per una matematica più rigorosa sull'argomento trattato, si faccia riferimento alla vasta letteratura sull'elaborazione dei segnali, sulla statistica e la chemiometria.

In questo lavoro viene descritta un'ampia gamma di applicazioni e connessioni, alcune potenzialmente interessanti, come gli investimenti in borsa (pagina 319), i pregiudizi cognitivi umani (pagina 353), il guasto di un veicolo spaziale della NASA (pagina 72), l'aggiunta di un tipo di rumore per ridurne un altro (pagina 300), lo studio dell'erosione delle spiagge causata dalla sabbia portata dal vento (pagina 296), l'intelligenza artificiale (pagina 434), una tecnica che amplia i

limiti classici della misurazione in spettroscopia (pagina 268), l'intelligibilità del parlato digitalizzato (pagine 99 e 381), computer in miniatura (pagina 335) e un modo semplice per creare app GUI interattive (pagina 35). L'elenco delle citazioni (pagina 493 nel PDF) testimonia una gamma di applicazioni davvero sbalorditiva.

Questo lavoro fa un uso considerevole di [Matlab](#), un ambiente di calcolo numerico commerciale e proprietario ad alte prestazioni e un linguaggio di programmazione di "quarta generazione" ampiamente utilizzato nella ricerca (riferimenti 14, 17, 19, 20 a pagina 489), *Octave*, un'alternativa gratuita a Matlab che esegue quasi tutti i programmi e gli esempi di questo tutorial (pagina 20) e [Python](#) (pagina 429), un linguaggio open source molto popolare. C'è una buona ragione per cui Matlab e Python sono così popolari nella scienza e nell'ingegneria; sono potenti, veloci,



relativamente facili da apprendere e possono gestire facilmente variabili vettoriali e matriciali. Entrambi si basano su *funzioni*, chiamate anche *subroutine*, che sono moduli di codice autonomi che svolgono un compito specifico (pagina 488). Le funzioni di solito prendono i dati (l'input), li elaborano e restituiscono un risultato (l'output). Un esempio banale è $a=\sqrt{b}$, che prende il valore di b , ne calcola la radice quadrata e la assegna alla variabile a . Una volta che una funzione è stata scritta, può essere riutilizzata più volte. Le funzioni si possono anche "chiamare" dall'interno di altre funzioni. Matlab è *dotato di molte funzioni native per eseguire attività di elaborazione dati* come la matematica matriciale, filtri, trasformate di Fourier, convoluzione e deconvoluzione, regressione multilineare e ottimizzazione. Si possono scaricare dei potenti 'toolbox' da [Mathworks](#) e [funzioni fornite da utenti](#) di terze parti gratuitamente. In pratica, *si possono scrivere le proprie funzioni personalizzate*. Una volta scritte o scaricate, le funzioni possono essere inserite nel "[path di ricerca](#)" di Matlab, per essere utilizzate *come qualsiasi altra funzione Matlab*. (Digitare "help path"). Le funzioni Matlab possono eseguire qualsiasi attività, tra cui disegnare grafici e interagire con l'utente. Matlab può interfacciarsi con C, C++, Java, Fortran e [Python](#). Ci sono molti esempi di codice in questo testo da Copiare e Incollare (o trascinare) sulla riga di comando di Matlab/Octave per eseguirli subito o modificarli.

La maggior parte delle tecniche trattate in questo lavoro possono essere eseguite in normali fogli di calcolo come *Excel* di Microsoft o *Calc* di OpenOffice/LibreOffice (11, 22, 23), che possono essere scaricati gratuitamente dai loro siti web, <https://sourceforge.net/projects/octave/> e <https://www.libreoffice.org/>.

Si possono scaricare tutti gli script Matlab/Octave e script Python o le funzioni e i template da <http://tinyurl.com/cey8rwh> gratuitamente; questi hanno ricevuto un [feedback dagli utenti estremamente positivo](#). Se si tenta di eseguire uno di questi script o funzioni e si riceve l'errore "missing function", si cerchi l'elemento mancante in <http://tinyurl.com/cey8rwh>, scaricandolo nel path di Matlab/Octave. (Digitare "help path" per ulteriori informazioni sul [path di ricerca](#)).

Se non si conosce Matlab, si può leggere la pagina 15 e le successive per un rapido avvio. Matlab è specificamente adatto per i metodi numerici, le manipolazioni di matrici, il disegno di funzioni e dati, creazione di algoritmi e di interfacce utente e distribuzione su dispositivi portatili come i tablet - in pratica le esigenze del [calcolo numerico per scienziati e ingegneri](#). Matlab è [blandamente tipizzato e digitato dinamicamente](#), è meno strutturato, nel senso formale, di altri linguaggi, e tende ad essere preferito da scienziati ed ingegneri e meno apprezzato da informatici e programmatore professionisti. Python è diverso in molti [dettagli](#), è un po' più difficile da installare e richiede diversi "pacchetti" aggiuntivi, ma ha il grande vantaggio di essere *gratuito*. Si tenga presente che si può utilizzare un chatbot di intelligenza artificiale come aiuto nella conversione di Matlab in Python e

viceversa (pag. 434).

Ci sono diverse versioni di Matlab, comprese quelle stand-alone per studenti e home a basso costo, completamente funzionanti che girano in un [browser web \(si veda il grafico sotto\), e delle app per iPad e iPhone](#). Si veda <https://www.mathworks.com/pricing-licensing.html> per i prezzi e le limitazioni di utilizzo.

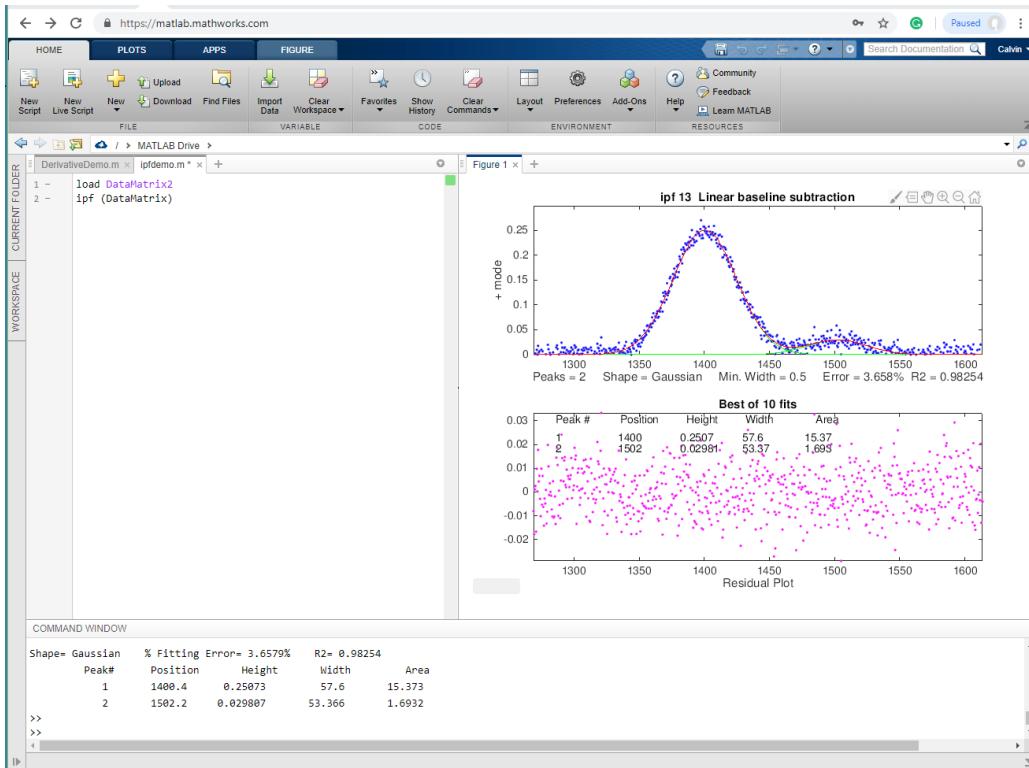


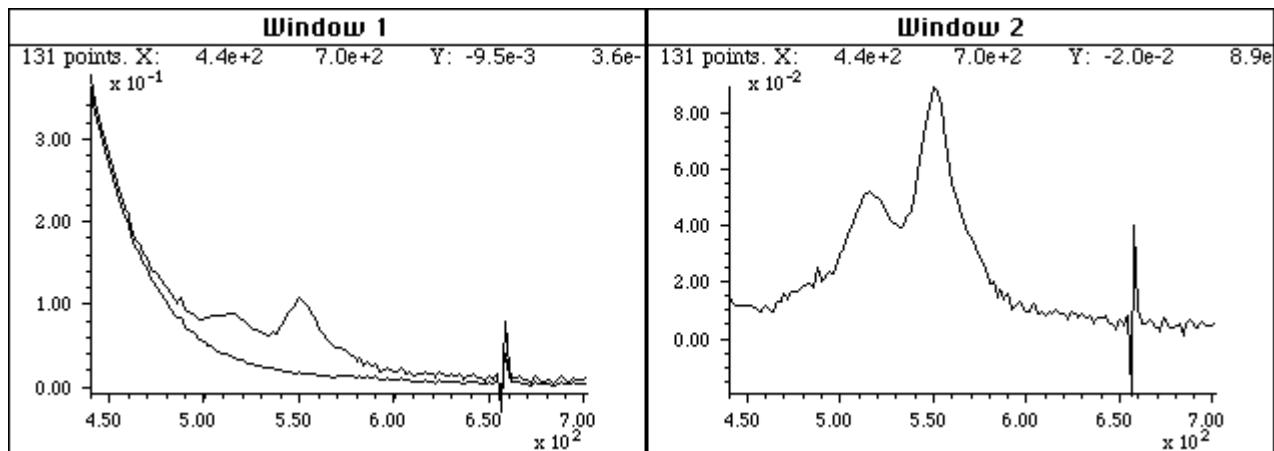
Figura 1. Matlab Online che esegue il Peak Fitter Interattivo (ipf.m) in Chrome su un PC Windows

Ci sono delle alternative a Matlab, in particolare, *Octave*, che è essenzialmente un clone di Matlab, ma ci sono anche Scilab, FreeMat, Julia e Sage, che sono per lo più, o in qualche modo, compatibili col linguaggio MATLAB. Per una discussione sulle altre possibilità, si veda <http://www.dspguru.com/dsp/links/matlab-clones>.

Se si sta leggendo questo libro *online*, su un computer connesso ad Internet, si può cliccare su qualsiasi indirizzo Web, sui nomi del software scaricabile oppure sulle animazioni, per vederle o scaricarle. Per l'elenco completo di tutto il software, si veda la pagina 451 o <http://tinyurl.com/cey8rwh>.

Aritmetica del segnale

Le operazioni più elementari del signal processing sono quelle che coinvolgono l'aritmetica semplice dei segnali: somma punto-per-punto, sottrazione, moltiplicazione e divisione di due segnali o di un segnale ed una costante. Nonostante la semplicità matematica, queste operazioni possono risultare molto utili. Ad esempio, nella parte sinistra della figura sotto (Window 1) la curva in alto è lo spettro di assorbimento ottico di un estratto da un campione di scisti bituminosa, un tipo di roccia da cui si ricava petrolio.

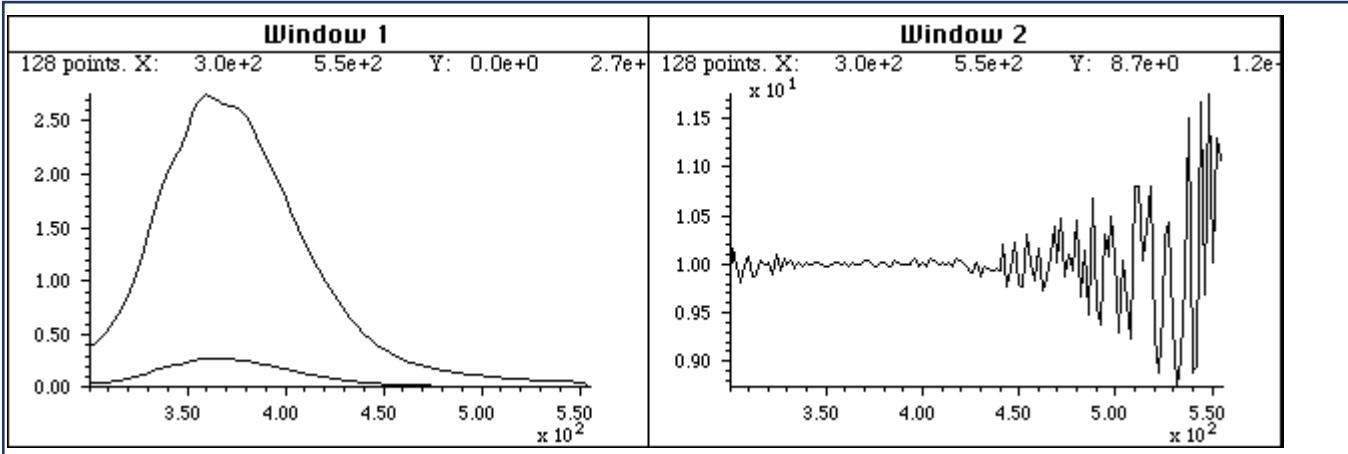


Una semplice sottrazione punto-punto di due segnali consente di sottrarre il background (la curva in basso a sinistra) da un campione complesso (la curva in alto a sinistra), ottenendo un'immagine più chiara di ciò che realmente è il campione (a destra). (Asse-X = lunghezza d'onda in nm; Asse-Y = assorbanza).

Questo spettro ottico mostra due bande di assorbimento, a 515 nm e a 550 nm. Questi picchi sono dovuti ad una classe di fossili molecolari di clorofilla chiamati *porfirine*, che vengono usate come “geo-marcatori” nell'esplorazione petrolifera. Queste bande sono sovrapposte ad un assorbimento di fondo causato dai solventi estrattori e da composti non porfirinici nello scisto. La curva inferiore è lo spettro di un estratto di scisto non contenente porfirina, che mostra solo l'assorbimento del background. Per ottenere lo spettro dell'estratto di scisto senza il background, questo (curva sotto) viene semplicemente sottratto dallo spettro campione (curva superiore). La differenza appare nella “Window 2” a destra (si noti il cambio della scala sull'asse Y). In questo caso, la rimozione del background non è perfetta, perché lo spettro del background è stato misurato su un altro campione di scisto. Tuttavia, funziona abbastanza bene da evidenziare le due bande e risulta più facile misurarne con precisione le assorbanze e le lunghezze d'onda. (Grazie al Prof. David Freeman della Univ. of Maryland per gli spettri degli estratti di scisti bituminosi).

In questo esempio, e nel seguente, si è ipotizzato che i due segnali in Window 1 avessero gli *stessi valori sull'asse x* - in altre parole, che entrambi gli spettri fossero stati digitalizzati nello stesso insieme di lunghezze d'onda. La sottrazione o la divisione di due spettri non sarebbe valida se i due spettri fossero stati digitalizzati su intervalli di lunghezze d'onda diversi o con intervalli differenti tra punti adiacenti. I valori dell'asse x devono corrispondere punto per punto. In pratica, questo è molto spesso il caso dei set di dati acquisiti all'interno di un esperimento su uno strumento, ma si deve fare attenzione se si modificano le impostazioni dello strumento o se si combinano i dati provenienti da due esperimenti o da due strumenti diversi. È possibile utilizzare la tecnica matematica dell'*interpolazione* per modificare la frequenza di campionamento (intervallo dell'asse x) o per equalizzare intervalli di segnali sull'asse x non equidistanti; i risultati sono generalmente solo approssimazioni ma spesso abbastanza corretti nella pratica. Excel può eseguire i calcoli utilizzando la funzione [Previsione](#). Matlab e Octave hanno funzioni native per l'interpolazione, tra cui [interp1.m](#), vedere l'[esempio 1 \(grafico\)](#) e l'[esempio 2 \(grafico\)](#).

A volte è necessario sapere se due segnali hanno la stessa forma, ad esempio confrontando un segnale ignoto con uno di riferimento. Molto probabilmente le ampiezze dei due segnali saranno diverse. Pertanto, una sovrapposizione o sottrazione diretta dei due segnali non sarà utile. Una possibilità consiste nel calcolare il *rapporto* punto per punto dei due segnali; se hanno la stessa forma, il rapporto sarà una costante. Per esempio, si esamini questa figura:



I due segnali a sinistra hanno la stessa forma? Certamente non hanno lo stesso aspetto, ma potrebbe semplicemente essere perché uno è molto più debole dell'altro. Il rapporto dei due segnali, mostrato nella parte destra (Window 2), è relativamente costante da 300 a 440 nm, con un valore di 10 ± 0.2 . Ciò significa che la forma di questi due segnali è pressoché identica su questo intervallo dell'asse x.

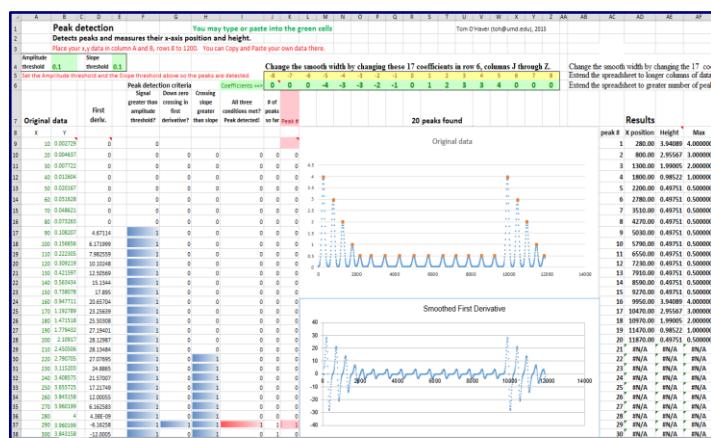
La parte sinistra (Window 1) mostra due segnali sovrapposti, uno dei quali è molto più debole dell'altro. Ma hanno la stessa forma? È difficile da dire. È molto più chiaro se si osserva il *rapporto* dei due segnali, mostrato nella parte destra (Window 2), che è relativamente costante da $x=300$ a 440 , con un valore di 10 ± 0.2 . Ciò significa che la forma di questi due segnali è la stessa, entro $\pm 2\%$ circa, su questo intervallo dell'asse x, e che la curva superiore è circa 10 volte più intensa di quella inferiore. Al di sopra di $x=440$ il rapporto non è nemmeno approssimativamente costante; questo è causato dal *rumore*, che è l'argomento del prossimo paragrafo (pagina 22).

Quando si dividono due vettori punto per punto, *anche un singolo zero* nel vettore denominatore arresterà il programma con una **division by zero error**. Un numero particolarmente piccolo ma finito nel denominatore non fermerà il programma ma genererà un numero enorme nel risultato. Entrambi i problemi possono essere generalmente evitati applicando una piccola quantità di smoothing (pag. 38) del denominatore o utilizzando la funzione Matlab/Octave [rmz.m](#) (rimuovi gli zeri) che sostituisce gli zeri con i numeri diversi da zero più prossimi. La relativa funzione [rmnan.m](#) rimuove i NaN (“Not a Number”) e gli Inf (“Infinite”) dai vettori, sostituendoli con numeri finiti reali vicini.

Calcoli e grafici on-line. [Wolfram Alpha](#) è un sito Web gratuito ed una app per smartphone ed è uno strumento di calcolo e sorgente di informazioni estremamente utile, con capacità di matematica simbolica, di [disegno](#), di vettori e gestioni di matrici, di analisi statistiche e di dati e molti altri argomenti. [Statpages.org](#) può eseguire una vasta gamma di test e calcoli statistici. Esistono diversi siti Web specializzati nella stampa di dati, tra cui [Plotly](#) e [Grapher](#). Tutti questi richiedono una connessione Internet affidabile e sono utili quando si lavora su un dispositivo mobile o un computer su cui non è installato il software richiesto. Nella versione PDF di questo libro, si può digitare **Ctrl-Click** su questi link per aprirli nel browser.

Aritmetica dei segnali nei Fogli di Calcolo

Gli spreadsheet, i fogli di calcolo, più diffusi, come *Excel* o *Calc di Open Office*, sono destinati principalmente ad applicazioni aziendali e finanziarie, ma hanno funzioni integrate per molte operazioni matematiche comuni, nomi di



variabili, grafici x,y, formattazione del testo, matematica con matrici, ecc. Le celle possono contenere valori numerici, testo, espressioni matematiche o riferimenti ad altre celle. Si può rappresentare uno spettro con una riga o una colonna di celle. Si può rappresentare un insieme di spettri come un blocco rettangolare di celle. Si possono dare dei nomi alle singole celle o a gruppi di celle, per poi farvi riferimento, per nome, nelle espressioni matematiche. È possibile copiare espressioni matematiche in un intervallo di celle, con i riferimenti che possono o meno cambiare a piacere. È possibile creare grafici di vari tipi (incluso l'importantissimo grafico *x-y* o *scatter*) selezionandolo dal menù. Per una dimostrazione, c'è questo video su YouTube:

<http://www.youtube.com/watch?v=nTlkkbQWpVk>. Sia *Excel* che *Calc* danno la possibilità di "progettare moduli" con un set completo di oggetti per l'interfaccia utente come pulsanti, menù, cursori e caselle di testo; si possono usare per creare interfacce grafiche accattivanti per applicazioni per utenti finali, come quelle create per insegnare ai corsi di chimica analitica su <http://terpconnect.umd.edu/~toh/models/>. Le ultime versioni sia di *Excel* (2013) che di *OpenOffice Calc* (3.4.1) possono aprire e salvare entrambi i formati di file (.xls e .ods, rispettivamente).

Semplici spreadsheet in entrambi i formati sono compatibili con l'altro programma. Tuttavia, ci sono piccole differenze nel modo in cui alcune operazioni vengono interpretate e per questo motivo la maggior parte dei fogli di calcolo viene fornita in formato .xls (per *Excel*) e in .ods (per *Calc*). Vedere "Differenze tra il formato Spreadsheet Open-Document (.ods) ed Excel (.xlsx)".

Fondamentalmente, *Calc* è in grado di eseguire quasi tutto quello che può fare *Excel*, ma *Calc* è scaricabile gratuitamente e segue di più lo stile estetico di Windows. *Excel* è più "Microsoftese" ed è più veloce di *Calc*. Se si ha accesso ad *Excel*, se ne raccomanda l'uso.

Se si lavora su un tablet o un smartphone, si può usare l'app *Excel mobile*, *Numbers* per iPad, o diversi altri "spreadsheet mobile". Queste app possono svolgere le attività di base ma non hanno le capacità più elaborate dei normali computer. Salvando i propri dati nel "cloud" (p.es. iCloud o SkyDrive), queste app sincronizzano automaticamente le modifiche tra dispositivi mobili, computer desktop e portatili in entrambe le direzioni, rendendole utili per l'inserimento dei dati sul campo.

Aritmetica e plottaggio dei segnali in Matlab

In Matlab (e nel suo clone GNU *Octave*) o in Python, l'aritmetica è molto simile a qualsiasi altro linguaggio: ad es. $(a+b)/c$. In Matlab e in Python (pag. 20), una singola variabile può rappresentare un singolo valore "scalare", un *vettore* di valori (come uno spettro o un cromatogramma), una *matrice* (un array rettangolare di valori, come un insieme di spettri) o un insieme di matrici *multiple*. Tutte le operazioni e le funzioni matematiche standard vi si adeguano. Ciò facilita notevolmente le operazioni matematiche sulle forme d'onda del segnale. La sottrazione di due segnali **a** e **b**, come a pagina 13, si può eseguire semplicemente scrivendo **a-b**. Allo stesso modo, il rapporto tra due segnali in Matlab, come a pagina 14, è "**(a ./b)**". Quindi, "/" significa dividere punto-per-punto e ".*" significa moltiplicare punto-per-punto. Il "*" di per sé indica la moltiplicazione tra *matrici*, utilizzabile per eseguire moltiplicazioni ripetute senza usare i cicli. Per esempio, se **x** è un vettore.

```
A=[1:100]'*x;
```

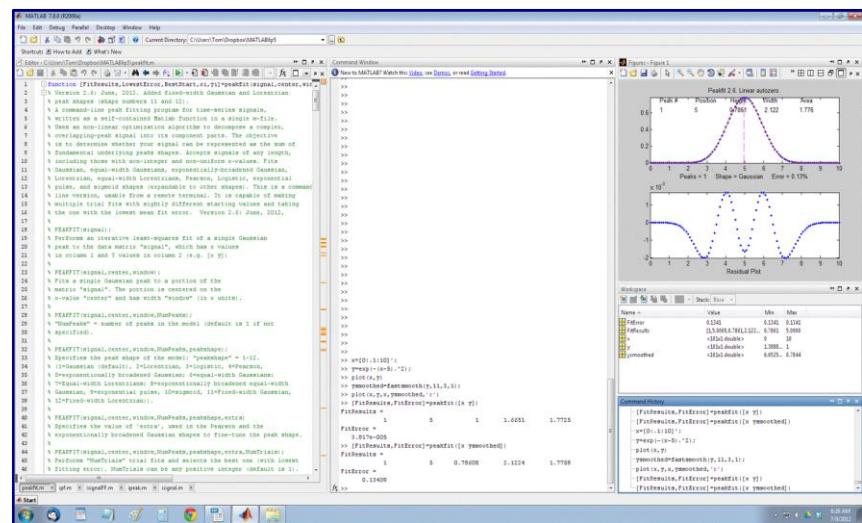
crea una matrice **A** in cui ciascuna colonna è **x** moltiplicato per i numeri 1, 2,...100. È equivalente a scrivere un ciclo "for" come questo, ma più compatto da scrivere e più veloce da eseguire:

```
for n=1:100;
A(:,n)=n.*x;
end
```

Plottaggio (disegno) dei dati. Se si hanno le ampiezze di un segnale nella variabile **y**, queste si possono disegnare digitando semplicemente "**plot(y)**". E se si ha un vettore **t** della stessa lunghezza contenente i tempi in cui ciascun valore di **y** è stato ottenuto, si può plottare **y** rispetto a **t** digitando "**plot(t,y)**". Due segnali **y** e **z** si possono plottare sullo stesso asse temporale per confrontarli digitando "**plot(t,y,t,z)**". (Matlab assegna automaticamente diversi colori a ciascuna curva, ma si può controllare il colore e lo stile della linea aggiungendo altri simboli; per esempio, "**plot(x,y,'r.',x,z,'b-')**" disegnerà **y** rispetto a **x** con puntini rossi e **z** rispetto a **x** con una linea blu. È possibile suddividere una finestra più aree di disegno più piccole mettendo **subplot(m,n,p)** prima del comando **plot** nella p^a sezione di una griglia di disegni m-per-n. (Se si sta leggendo online, si può cliccare [qui](#) per un esempio di un 'subplot' 2x2. Si può anche selezionare, copiare e incollare, o trascinare, una o più righe dei codici di esempio, nell'editor di Matlab o di Octave oppure direttamente sulla riga di comando e premere **Enter** per eseguirlo immediatamente). In Matlab, digitare "help plot" per altre opzioni per il disegno. In Python, "**import matplotlib.pyplot as plt**" abilita un [plottaggio simile a quello di Matlab](#).

Per avere dei grafici con una **qualità da pubblicazione**, cliccare sulla finestra di una Figura, poi cliccare su **File > Export setup**, scegliere la dimensione, la risoluzione, il colore, i font, ecc, poi cliccare su **Export** e selezionare il formato del file (p.es. TIF, eps, ecc.). Si può utilizzare anche [PlotPub](#), una libreria scaricabile gratuita, facile da usare, che consente una grande flessibilità nella scelta dei dettagli del grafico e crea grafici bellissimi in Matlab esportabili in EPS, PDF, PNG e TIFF con la regolazione della risoluzione. Ecco un esempio ([script](#), [grafico](#)).

La funzione **max(y)** restituisce il valore massimo di **y** e **min(y)** ne restituisce il minimo. I singoli elementi in un vettore sono indicati con un *numero d'indice*; per esempio, **t(10)** è il 10° elemento nel vettore **t**, e **t(10:20)** è il vettore dei valori di **t** dal 10° al 20° elemento. È possibile trovare il numero dell'indice dell'elemento più vicino a un dato valore in un vettore utilizzando la funzione [val2ind.m](#). Per esempio, **t(val2ind(y,max(y)))** restituisce il tempo del massimo di **y**, e **t(val2ind(t,550):val2ind(t,560))** è il vettore dei valori di **t** tra 550 e 560 (assumendo che **t** contenga valori in questo intervallo). I dati delle *unità* di tempo nel vettore **t** possono essere

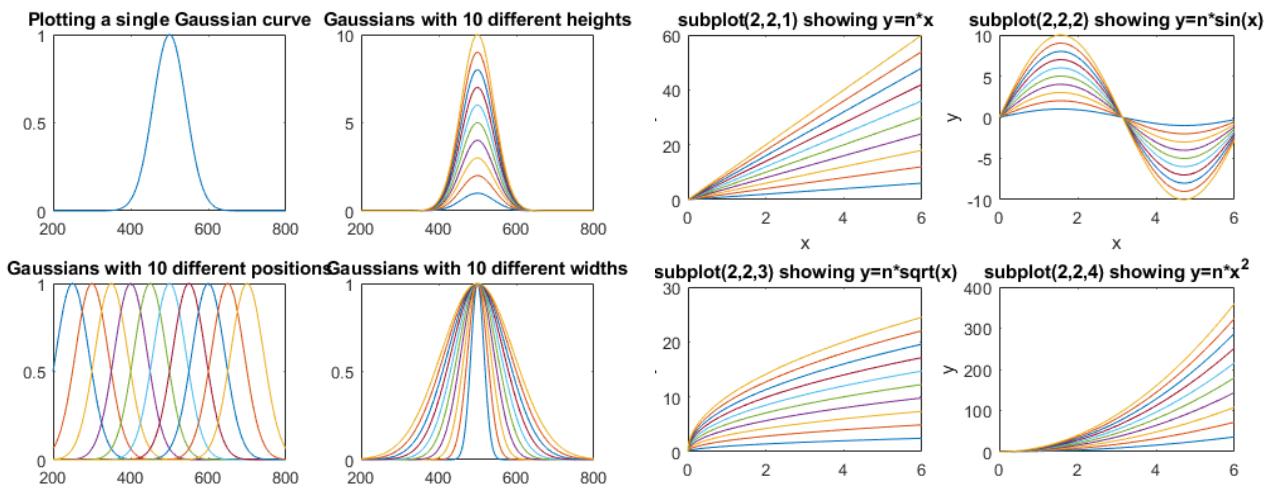


qualsiasi cosa: microsecondi, millisecondi, ore, qualsiasi unità di tempo.

Una variabile Matlab può anche essere una *matrice*, un insieme di vettori della stessa lunghezza raggruppate in un array rettangolare. Per esempio, le letture delle intensità di 10 diversi spettri ottici, ciascuno acquisito nello stesso insieme di 100 lunghezze d'onda, potrebbero essere raggruppate

in una matrice **S** 10x100. **S(3,:)** corrisponde al terzo di questi spettri e **S(5,40)** è l'intensità della 40° lunghezza d'onda del 5° spettro. Gli script Matlab [plotting.m](#) (a sinistra) e [plotting2.m](#)(a destra) mostrano come plottare più segnali utilizzando una matrice e i "subplot".

Vedere [TimeTrial.txt](#) per i dettagli. Sarà utile pre-allocare lo spazio di memoria per la matrice **A** aggiungendo l'istruzione **A=zeros(100,100)** prima del ciclo. Anche così, la notazione matriciale è più veloce del loop.



In Matlab/Octave, "/" non è lo stesso di "\\". Digitando "b\|a" verrà calcolata la "divisione matriciale a sinistra", in pratica, il rapporto medio ponderato delle ampiezze dei due vettori (una specie di soluzione per l'approssimazione con i quadrati-minimi). Il punto qui è che *Matlab non richiede di trattare vettori e matrici come raccolte di numeri*; sa quando si ha a che fare con le matrici o quando il risultato di un calcolo sarà una matrice e di conseguenza regola i calcoli. Cfr. https://www.mathworks.com/help/matlab/matlab_prog/array-vs-matrix-operations.html.

Probabilmente gli errori più comuni che si faranno in Matlab/Octave riguarderanno la punteggiatura, come il mischiare punti, virgole, due-punti e punti-e-virgola o parentesi, quadre e graffe; si può digitare "help punct" al prompt di Matlab e leggere il file di help fino allo sfinimento. *Le piccole cose possono significare molto* in Matlab. Un altro errore comune è quello di confondere le righe e le colonne di vettori e matrici. (Una confessione: commetto ancora questo tipo di errori). Ecco un [file di testo](#) che fornisce esempi di normali operazioni tra vettori e matrici e gli errori in Matlab e in Octave. Per i neofiti, si consiglia di leggere questo file e di provare gli esempi. Scrivere in Matlab è un processo per tentativi ed errori, con l'accento sull'*errore*. Iniziare con cose semplici, farle funzionare e poi progredire.

Ci sono molti esempi di codice in questo testo da copiare, incollare e modificare sulla riga di comando di Matlab/ Octave, ed è il miglior modo per imparare. Nella versione PDF di questo libro, si possono selezionare per poi copiare e incollare o selezionare e trascinare, qualsiasi esempio di codice su singola o più linee nell'editor di Matlab o di Octave o direttamente sulla riga di comando e poi premere **Enter** per eseguirlo immediatamente). Ciò è particolarmente conveniente se si esegue Matlab e si legge dal sito web o dal libro sullo stesso computer; posizionare le finestre in modo che Matlab condivida lo schermo con questo sito web (p.es. Matlab a sinistra e il browser web a destra

Peakfit m Version 8 No baseline correction

Peaks = 1 Shape = Gaussian Min. Width = 0.67587 Error = 1.785% R2 = 0.99762

Single fit

Peak #	Position	Height	Width	Area
1	-0.02152	272.3	2.32	670.8

Residual Plot

Command History

```
>> p1=(x.^2);
>> c1g;
>> plot(x,p1);
>> plot(x,p1,'*');
>> hold on;
>> plot(x(1:24),Height1.*gaussian(x(1:24),Position1,Width1));
>> PositionError1;
>> PositionError1=100.* (position1-MeasuredPosition1). / po
>> c1g;
>> PositionError1=100.* (position1-MeasuredPosition1). / po
>> num2str(PositionError1);
>> MeasuredHeight2;
>> help lorentzfit;
>> %----- 3/19/17 SISE AM -----
>> MeasuredHeight2;
>> N,X)=hist(randn(size(1:100));
>> peakfit([X;N]);
>> [N,X]=hist(randn(size(1:100));
>> peakfit([X;N]);
>> [N,X]=hist(randn(size(1:100));
>> peakfit([X;N]);
>> peakfit([X;N]);
```

Workspace

Name	Value	Min	Max
HeightError1	-0.2025	-0.2025	-0.2025
HeightError2	-2.7998	-2.7998	-2.7998
HeightError3	1.0000	1.0000	1.0000
MeasuredHeight1	0.2096	0.2096	0.2096
MeasuredHeight2	45.0334	45.0334	45.0334
MeasuredPosition1	59.7232	59.7232	59.7232
MeasuredPosition2	10.0553	10.0553	10.0553
MeasuredWidth1	10.46	10.2746	10.2746
MeasuredWidth2	-0.0741	-0.0743	-0.0743
PositionError1	0.4914	0.4616	0.4616
PositionError2	-0.3535	-0.3535	-0.3535
WidthError1	-2.7496	-2.7496	-2.7496
WidthError2	1.2841	1.2192	1.2192
Y	1.2841	1.2192	1.2192
ans	1	1	1
height1	0.2000	0.2000	0.2000
height2	0	0	0
meas	0	0	0
position1	45	45	45
position2	60	60	60
range1	<1x10 double>	35	50
range2	<1x10 double>	57	69

Help text for peakfit.m:

Matlab and Octave have built-in functions that can be used for calculating, measuring and plotting signals and noise, including mean, max, min, std, kurtosis, skewness, plot, hist, histfit, rand, and randn. Just type "help" and the function name at the command >> prompt, e.g. "help mean". Most of these functions apply to vectors and matrices as well as scalar variables. For example, if you have a series of results in a vector variable 'Y', mean(Y) returns the average and std(Y) returns the standard deviation of all the values in Y. For vectors, std computes sqrt(mean(Y.^2)). You can subtract a scalar number from a vector (for example, v = v-min(v)) sets the lowest value of vector v to zero. If you have a set of signals in the rows of a matrix S, where each column represents the value of each signal at the same value of the independent variable (e.g. time), you can compute the ensemble average of those signals just by typing "mean(S)", which computes the mean of each column of S. Note that function and variable names are case sensitive.

As an example of the "randn" function in Matlab/Octave, it is used here to generate 100 normally-distributed random numbers. Then the "hist" function computes the "histogram" (probability distribution) of those random numbers, then the downloadable function peakfit.m fits a Gaussian function (plotted with a red line) to that distribution:

```
>> [N,X]=hist(randn(size(1:100));
>> peakfit([X;N]);
```

If you change the 100 to 1000 or a higher number, the distribution becomes closer and closer to a perfect Gaussian and its peak falls closer to 0.00. The "randn" function is useful in signal processing for predicting the uncertainty of measurements in the presence of random noise. For example by using the Monte Carlo or the bootstrap methods that will be described in a later section. (You can copy and paste, or drag and drop, these two lines of code into the Matlab or Octave editor or into the command line and press Enter to execute it).

The difference between scripts and functions: You can also create your own user-defined scripts and functions in Matlab or Octave to automate commonly-used algorithms. Scripts and functions are simple text files saved with a ".m" file extension to the file name. The difference between a script and a function is that a function definition begins with the word "function"; a script is just any list of Matlab commands and statements. For a script, all the variables defined and used are listed in the workspace window. For a function, on the other hand, the variables are internal and private to that function; values can be passed to the function through the input arguments, and values can be passed from the function through the output arguments, which are both defined in the first line of the function definition. That means that functions are a great way to package chunks of code that perform useful operations in a form that can be used as components in other programs without worrying that the variable names in the function will conflict and cause errors. Scripts and functions can call other functions, scripts must have those functions in the Matlab path; functions, on the other hand, can have all their required sub-functions defined within the main function itself and thus can be self-contained. If you run one of my scripts and get an error message that says "Undefined function...". you need to download the specified function from [functions.html](#) and place it in the Matlab/Octave path).

For writing or editing scripts and functions, Matlab and the latest version of Octave have an internal editor. For an explanation of a function and a simple worked example, type "help function" at the command

come mostrato di seguito). O, ancora meglio, alcuni computer desktop hanno *due* uscite per monitor, quindi se ne possono usare due contemporaneamente per espandere il desktop orizzontalmente.

Suggerimento: *Se tentando di eseguire uno di questi script o funzione e si riceve l'errore "missing function", si cerchi l'elemento mancante in <http://tinyurl.com/cey8rwh>, scaricandolo nel path di ricerca e riprovare.*

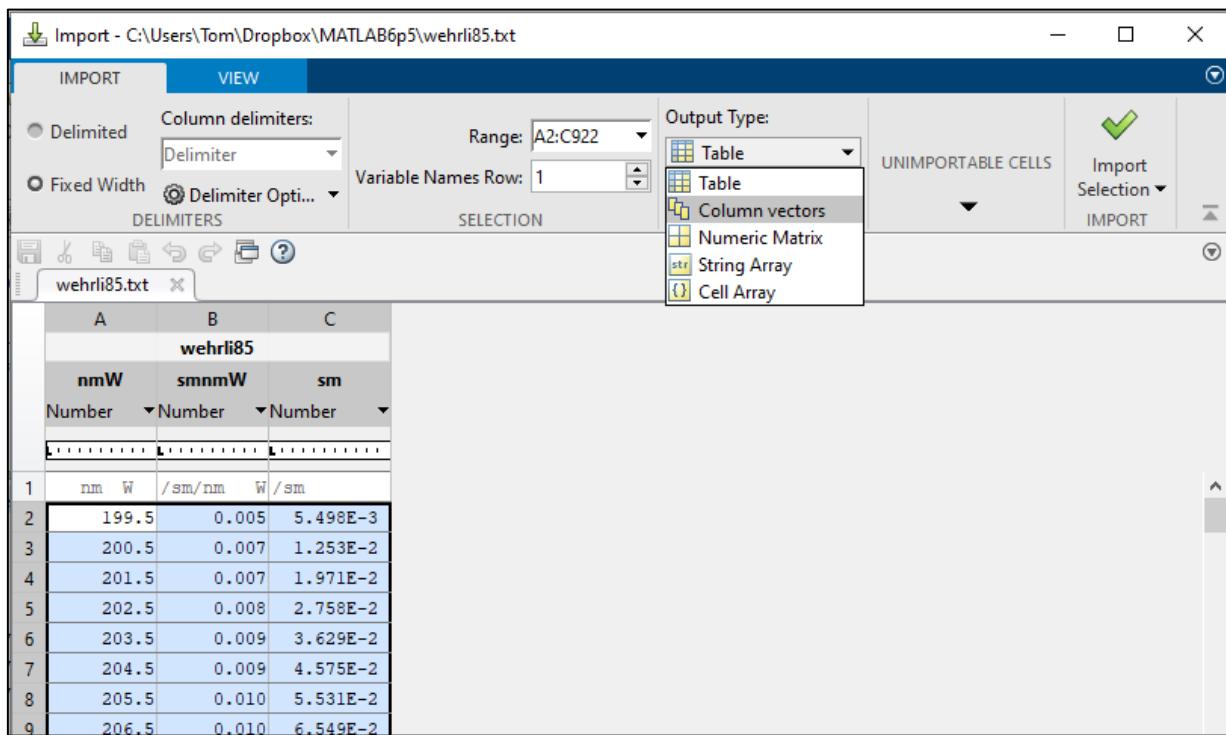
Una cosa che si noterà in Matlab è che la *primissima volta* che si esegue uno script o una funzione, e *solo* la prima volta, c'è un piccolo ritardo prima dell'esecuzione, mentre Matlab compila il codice in linguaggio macchina. Questo, però, avviene solo la *prima* volta; dopo di ciò, l'esecuzione inizierà immediatamente. (Per un'esecuzione più veloce, è disponibile separatamente il "Matlab Compiler" che consente di condividere i programmi come applicazioni *autonome*, separate dall'ambiente Matlab. "Matlab Compiler SDK" consente la creazione di librerie shared C/C++, assembly di Microsoft .NET, classi Java e pacchetti Python partendo dai programmi Matlab). Si può anche fare qualche disegno in *real-time* in Matlab/Octave; cfr. pagina 339.

Import dei dati in Matlab/Octave e in Python.

Si possono importare i propri dati in Matlab o in Octave utilizzando il pulsante "Import data" nel tab Home o le funzioni [xlsread](#) o [importdata](#) sulla riga di comando o in uno script. I dati si possono importare da file di testo (.txt), CSV (comma separated values), da diversi formati di immagini e suoni o da spreadsheet. Per esempio, le righe seguenti leggeranno le prime due colonne del file csv "Sample_5.0ppm.csv" nella cartella corrente a le assegneranno ai vettori x5 e y5, ovvero le variabili vettoriali indipendente e dipendente, rispettivamente:

```
mydata=xlsread('Sample_5.0ppm.csv');  
x5=mydata(:,1);  
y5=mydata(:,2);
```

Lo script "[xlsreadDemo.m](#)" fornisce un semplice esempio di lettura di un file "xlsx" di un foglio di calcolo a più colonne. Per fogli di calcolo più complessi, Matlab ha utilissimo *Import Wizard* (clic su **File > Import Data**) che offre un'anteprima nel file di dati, analizza il file cercando per colonne e per righe di dati numerici e relative etichette, e dà la possibilità di selezionare, ri-etichettare le variabili e di scegliere di importarle come vettori, matrici o tabelle. Si può anche cliccare sulla piccola freccia accanto a "Import selection" e *Matlab scriverà uno script che eseguirà queste operazioni*, che si possono modificare per altri tipi di file e formati.



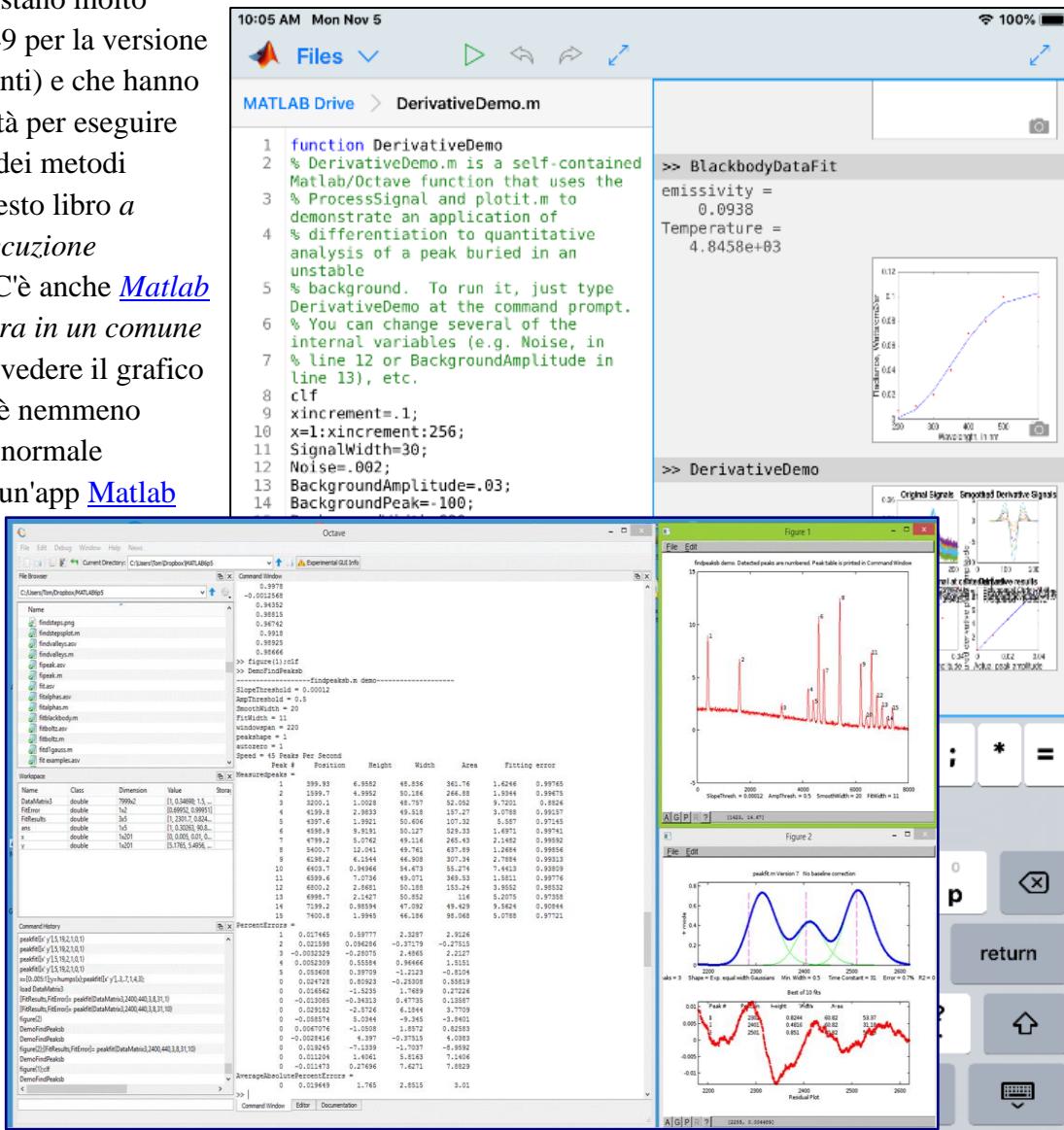
JCAMP-DX è un modulo standard per lo scambio di spettri infrarossi e relative informazioni chimiche e fisiche tra sistemi di dati spettrometrici di diversa provenienza. La funzione jcampread di Matlab può importare tali dati. Per un esempio, vedere [ReadJcampExample.m](#). È anche possibile importare dati *approssimativi* da grafici a linee o grafici *stampati* utilizzando la funzione nativa "ginput" che ricava i dati numerici dalle coordinate del clic del mouse o utilizzando applicazioni più automatizzate come "[Data Thief](#)" o "[Figure Digitizer](#)" in Matlab File Exchange. Ovviamente, i risultati non saranno accurati come quelli che si ottengono accedendo ai dati originali in un file. Matlab versione R2013a o più recente può anche [leggere dei sensori](#) dal proprio telefono iPhone o Android via Wi-Fi. Per leggere i segnali in output da vecchi strumenti analogici, c'è bisogno di un [convertitore analogico-digitale](#), una [scheda col microcontroller Arduino](#) o un [voltmetro USB](#).

Mathworks dispone di [strumenti di acquisizione dati](#) per Matlab. **Nota:** L'addizione, la sottrazione, la moltiplicazione e la divisione di due segnali digitali richiedono che abbiano lo *stesso numero di punti*. Se necessario, si possono togliere alcuni punti al segnale più lungo o aggiungerne a quello più corto (di solito degli zeri, detto "zero filling").

[Python può importare dati](#) nei formati testo, CSV, JSON, Matlab e diversi altri, utilizzando il pannello *Variable Explorer* sul desktop di *Spyder*, o tramite il pacchetto [Pandas Data Analysis](#) scaricabile a parte.

Le Versioni di Matlab.

La versione standard *commerciale* di Matlab è costosa (più di \$2000) ma ci sono le versioni *student* e *home* che costano molto meno (solo \$49 per la versione base per studenti) e che hanno tutte le capacità per eseguire uno qualsiasi dei metodi descritti in questo libro *a velocità di esecuzione comparabili*. C'è anche [Matlab Online](#), che gira in un comune browser web (vedere il grafico su 12). Non c'è nemmeno bisogno di un normale computer: c'è un'app [Matlab](#)



[Mobile](#) gratuita che esegue un'interfaccia Matlab su iPhone e su iPad connessi a Internet (illustrati a [lato](#)). Richiede solo una licenza di base per studenti e utilizza tutte le funzioni standard, oltre a qualsiasi funzione o script del libro (o anche quelle scritte dagli utenti) e *precedentemente caricate* nel proprio account sul [cloud di Matlab](#). Tutte queste versioni hanno una velocità di calcolo sono per lo più entro un fattore 2 l'una dall'altra, come mostrato in [TimeTrial.txt](#).

Matematica in Python

In Python, dopo aver importato numpy come np, le funzioni matematiche di base sono molto simili a quelle di Matlab: **len(d)** , **np.sum(d)** , **np.mean(d)** , **np.std(d)** , **np.sqrt(d)** , **max(d)** , **min(d)** . La funzione esponenziale è indicata come ****** anziché **^** come in Matlab.

Consultare la pag. 427 per [altri esempi](#).

GNU Octave

Octave è una alternativa gratuita a Matlab che è "per lo più compatibile". DspGURU dice che Octave è "...un clone maturo e di alta qualità di Matlab. Ha il più alto grado di compatibilità con Matlab tra tutti i cloni." Tutto quanto detto in precedenza su Matlab funziona anche in Octave. Infatti, *le versioni più recenti di quasi tutte le funzioni, gli script, le demo e gli esempi di Matlab in questo documento funzioneranno nell'ultima versione di Octave senza modifiche*. Le funzioni interattive azionate dai tasti *iPeak* (pagina 245), *iSignal* (pagina 366), *ipf.m* (pagina 405) e *ifilter.m*

richiedono *versioni separate* per Octave, che utilizzano tasti diversi per il pan e lo zoom. Se si prevede di utilizzare Octave, ci si assicuri di avere la versione corrente. C'è una [FAQ](#) che può aiutare nel porting di programmi Matlab in Octave. Vedere “[Differenze tra Octave & Matlab](#)”. Ci sono versioni Windows, Mac e Unix di Octave. La versione Windows è scaricabile da [Octave Forge](#). Ci sono molti help online: Google "GNU Octave" o si vedano i video YouTube per un aiuto. Per le applicazioni specifiche al signal processing, su Google "signal processing octave".

[Octave Version 6.4.0](#) è ora disponibile per il [download](#). La documentazione è online; vedere <https://www.octave.org>. Quasi tutti gli script e le funzioni girano su Octave. Tuttavia, è ancora computazionalmente circa 5 volte più lento, mediamente, rispetto all'ultima versione di Matlab, a seconda dell'attività (sono disponibili confronti specifici per diverse attività di signal processing in [TimeTrial.txt](#)). Conclusione: Matlab è migliore, ma se non è alla portata, Octave fornisce la maggior parte delle funzionalità allo 0% del costo. Nota: il vecchio Octave 3.6 può funzionare anche su un Raspberry Pi, un computer in miniatura a basso costo descritto alla pag. [335.](#)

Spreadsheet o Matlab/Python?

Per l'elaborazione dei segnali, i linguaggi per computer come Matlab/Octave o Python sono più veloci e potenti rispetto all'utilizzo di uno spreadsheet, ma si può sicuramente dire che gli spreadsheet più spesso disponibili di Matlab, Octave o Python sui computer dei tecnici. Per cominciare, gli spreadsheet sono più facili da usare, ed offrono una presentazione e un'interfaccia flessibile. I fogli di calcolo sono migliori per l'immissione manuale dei dati; si possono facilmente distribuire su dispositivi portatili come smartphone e tablet (p.es. utilizzando *Google Sheets*, *iCloud Numbers* o l'app *Excel*). *Gli spreadsheet sono concreti e a più basso livello, mostrando ogni singolo valore esplicitamente in una cella.* Al contrario, Matlab/Octave e Python ad un livello più alto di astrazione, dato che una singola variabile può essere un numero, un vettore o una matrice e la punteggiatura o qualsiasi funzione può fare molta differenza. È molto potente ma difficile da padroneggiare *all'inizio*. In Matlab e in Octave le funzioni e i gli script (“file m”) sono semplici file di testo con l'estensione “.m” (o “.py” nel caso di Python), quindi *questi file possono essere aperti ed esaminati con un qualsiasi editor di testo, anche sui dispositivi che non hanno tali programmi installati*, il che facilita la traduzione degli script e delle sue funzioni in altri linguaggi. Inoltre, le funzioni definite dall'utente possono chiamare *altre funzioni native o definite dall'utente, che a loro volta possono chiamare altre funzioni, e così via, consentendo la costruzione di funzioni di alto livello con strati molto complessi*. Fortunatamente, Matlab e Python possono facilmente analizzare file Excel “.xls” e “.xlsx” ed importarne le righe e le colonne in variabili vettoriali/matriciali.

Utilizzando l'analogia con i circuiti elettronici, gli spreadsheet sono come *componenti discreti* elettronici, dove ogni resistore, condensatore, induttanza e transistor è un'entità discreta e macroscopica direttamente visibile e manipolabile. Un linguaggio di programmazione basato su funzioni come Matlab/Octave è più simile alla *micro-elettronica*, dove le funzioni (i "file-m" che iniziano con "function...") sono i "chip", che racchiudono complesse operazioni in un unico pacchetto *con i pin di I/O documentati* (gli *argomenti* di input e output delle funzioni) collegabili ad altre funzioni, ma che *nascondono i dettagli interni* (a meno che non interessi guardare il codice, cosa che è sempre fattibile). Per esempio il "timer 555", è un timer, un generatore di impulsi ed un oscillatore ad 8-pin, introdotto nel lontano 1972, usato ancora oggi diventando il [circuito integrato più popolare mai prodotto](#). Quasi tutta l'elettronica ora è fatta con i chip, perché è *più facile capire il numero relativamente basso di input e output* di un chip che avere a che fare col gran numero di componenti all'interno. Gran parte di Matlab/Octave è scritto in Matlab/Octave stesso, utilizzando

funzioni più semplici per costruire quelle più complesse. Ci si possono scrivere nuove funzioni che estendono il linguaggio in qualsiasi direzione sia necessario (pag. 33).

Alla fine si ha che gli spreadsheet sono più facili all'inizio, ma, per attività più complesse, l'approccio Matlab/ Octave/ Python è computazionalmente più veloce, può gestire set di dati molto più grandi e può fare di più con meno sforzo. Ciò è dimostrato dal confronto di entrambe le piattaforme per la *spettroscopia multicomponente* trattata a pagina 179 ([RegressionDemo.xls](#) rispetto a Matlab/Octave [CLS.m](#)). Ancora più evidenti sono i diversi approcci per *trovare e misurare i picchi nei segnali*, che vengono trattati nella sezione iniziale a pagina 227 (p.es. uno spreadsheet da 250 Kbyte rispetto a uno script Matlab da 7 Kbyte che fa la stessa cosa ma è *50 volte più veloce*). Se si ha una grande mole di dati da trattare automaticamente con un processo personalizzato a più passi, senza intervento manuale e il più velocemente possibile, allora Matlab è ottimo. È molto più semplice scrivere uno script in Matlab che *automatizzerà l'elaborazione in autonomia di grossi volumi di dati* memorizzati in diversi file sul computer, come mostrato nell'esempio a pagina 337.

I programmi per spreadsheet, Matlab/Octave e Python hanno un enorme vantaggio sui programmi commerciali e freeware come SPECTRUM; questi si possono *ispezionare e modificare* personalizzando le routine per esigenze specifiche. È facile apportare semplici modifiche con poca o nessuna conoscenza della programmazione. Per esempio, si possono facilmente modificare le etichette, i titoli, i colori o lo stile delle linee nei grafici nei programmi Matlab o Octave, per i propri scopi: utilizzare **Find...** per cercare "title()", "label()" o "plot()". Il codice contiene *commenti che indicano i posti dove si possono fare specifiche modifiche*: cercare la parola "change". Si è invitati a modificare a piacimento gli script e le funzioni. La [licenza software](#) racchiusa nei commenti di tutto il mio codice Matlab/Octave è molto liberale.

Segnali e rumore

Le misure sperimentali non sono sempre perfette, anche quando si usano strumenti moderni e sofisticati. Vengono riconosciuti due tipi principali di errori di misurazione: (a) *errore sistematico*, in cui in ogni misura è costantemente inferiore o superiore al valore corretto di una certa percentuale o quantità, e (b) *errore casuale*, in cui ci sono variazioni imprevedibili nel segnale misurato da momento a momento o da misura a misura. Quest'ultimo tipo di errore è spesso chiamato *rumore*, per analogia col rumore acustico. Ci sono molte sorgenti di rumore nelle misure fisiche, come vibrazioni strutturali, correnti d'aria, fluttuazioni della corrente, radiazioni vaganti emesse da dispositivi elettrici, elettricità statica, interferenze da trasmissioni radio o TV, turbolenza nel flusso di gas o di liquidi, movimento termico casuale delle molecole, radiazione di fondo da elementi radioattivi naturali, la stessa natura quantistica di base della materia e dell'energia e il rumore di digitalizzazione (l'arrotondamento a un numero fisso di cifre), e "raggi cosmici" dallo spazio (sul serio). Poi, ovviamente, c'è l'onnipresente "errore umano", che può essere un fattore importante ogni volta che gli esseri umani sono coinvolti nel funzionamento, nella regolazione, nella registrazione, nella calibrazione o nel controllo degli strumenti e nella preparazione dei campioni per la misurazione. Se è presente un errore casuale, una serie di misure ripetute produrrà risultati che non sono tutti uguali ma piuttosto variano o si disperdonano attorno a un valore medio, che è la somma dei valori divisa per il numero di valori "d": `sum(d) ./length(d)` o semplicemente `mean(d)` in notazione Matlab/Octave. Il modo più comune per misurare la quantità di variazione o dispersione di un insieme di valori di dati consiste nel calcolare la *deviazione standard*, "std" che è la radice quadrata della somma dei quadrati delle deviazioni dalla media diviso il numero di punti meno uno: `sqrt(sum((d-mean(d)).^2) ./ (length(d)-1))`, in notazione Matlab/Octave. Questi si calcolano più facilmente con la funzione interna `mean(d)` e `std(d)`, dove `d` è il vettore dei dati. Un fatto fondamentale delle variabili casuali è che quando si

combinano, si devono calcolare i risultati *statisticamente*. Ad esempio, quando vengono addizionate due variabili casuali, la deviazione standard della somma è la "somma quadratica" (la radice quadrata della somma dei quadrati) delle deviazioni standard delle singole variabili. In Matlab, la funzione "randn(1,n)" restituisce n numeri casuali con una deviazione standard di 1. Perciò:

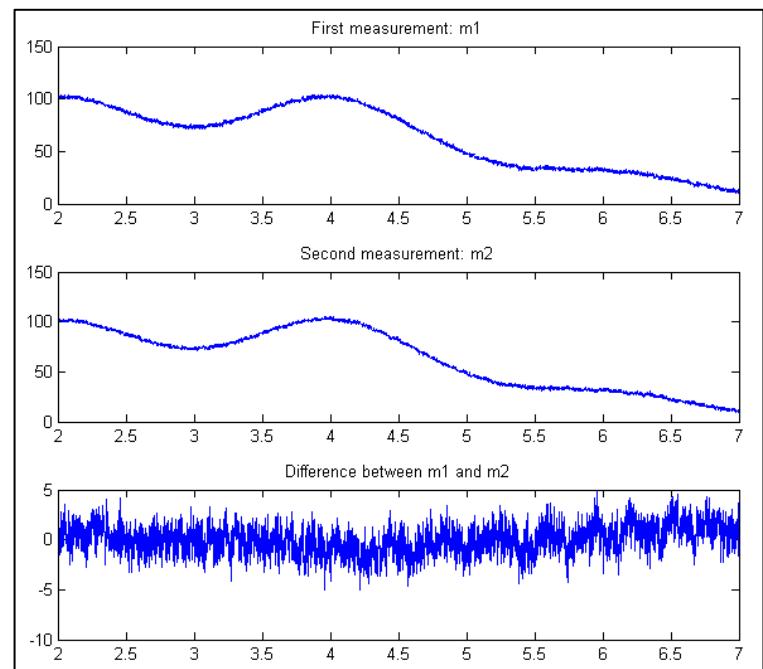
$$\lim_{n \rightarrow \infty} (std(randn(1, n))) = 1$$

$$\lim_{n \rightarrow \infty} (std(randn(1, n) + randn(1, n))) = sqrt(2)$$

Questo è dimostrato dalle serie di comandi Matlab/Octave [a questo link](#). Da provare.

Il termine ‘segnale’ ha due significati. In senso più generale, può significare l’*intera registrazione* dei dati, incluso il rumore e altri artefatti, come nel “segnale originale” prima dell’applicazione dell’elaborazione. Ma può anche indicare solo la parte *desiderabile* o *importante* dei dati, il *vero segnale sottostante* che si cerca di misurare, come nell’espressione “rapporto segnale-rumore”. Un problema fondamentale nella misura del segnale è distinguere il vero segnale dal rumore. Per esempio, si supponga di voler misurare la media del segnale in un certo tempo o l’altezza di un picco o l’area del picco che si ottiene dai dati. Nello spettro di assorbimento sulla metà destra della figura a pagina 13, le parti “importanti” sono probabilmente i picchi di assorbimento situati a 520 e a 550 nm. L’altezza o la posizione di uno di questi picchi potrebbe essere considerata il segnale, a seconda dell’applicazione. In questo esempio, l’altezza del picco più grande è di circa 0,08 unità di assorbanza. Ma come misurare il rumore? Nel caso eccezionale che si abbia un sistema fisico e uno strumento di misura, *entrambi* completamente stabili (*eccetto* per il rumore casuale), un modo semplice per isolare e misurare il rumore consiste nel *registrare due segnali m1 ed m2 dello stesso sistema fisico*. Sottraendo queste due registrazioni, la parte del segnale verrà annullata. Quindi la deviazione standard del rumore nei segnali originali è data da $sqrt((std(m1-m2)^2)/2)$, dove “sqrt” è la radice quadrata e “std” è la deviazione standard. (La derivazione di questa espressione è basata sulle regole per la propagazione dell’errore matematico ed è approfondita in <https://terpconnect.umd.edu/~toh/spectrum/Derivation.txt>). Lo script Matlab/Octave “SubtractTwoMeasurements.m” mostra questo processo quantitativamente e graficamente (sotto)..

Ma si supponga che le misure non siano così riproducibili o che si abbia solo *una* delle registrazioni di questo spettro senza altri dati. In questo caso, si può tentare di stimare il rumore nel segnale registrato, basandosi sull’*assunto* che le *fluttuazioni a breve termine* visibili nel segnale - le piccole irregolarità casuali sovrapposte al segnale regolare e fluido - siano *rumore* e non parti del vero segnale sottostante. Ciò dipende da una certa conoscenza dell’origine del segnale e dalle possibili forme che potrebbe assumere. Gli esempi nella sezione precedente (pagina 13) sono gli spettri di assorbimento di soluzioni liquide nell’intervallo di lunghezze d’onda da 450 nm a 700 nm. Queste soluzioni normalmente mostrano picchi ampi e lisci con una larghezza tra i 10 e i 100 nm, quindi quelle piccole oscillazioni devono essere *rumore*. In questo caso, tali fluttuazioni hanno una deviazione standard di circa 0.001. Spesso il modo migliore per misurare il

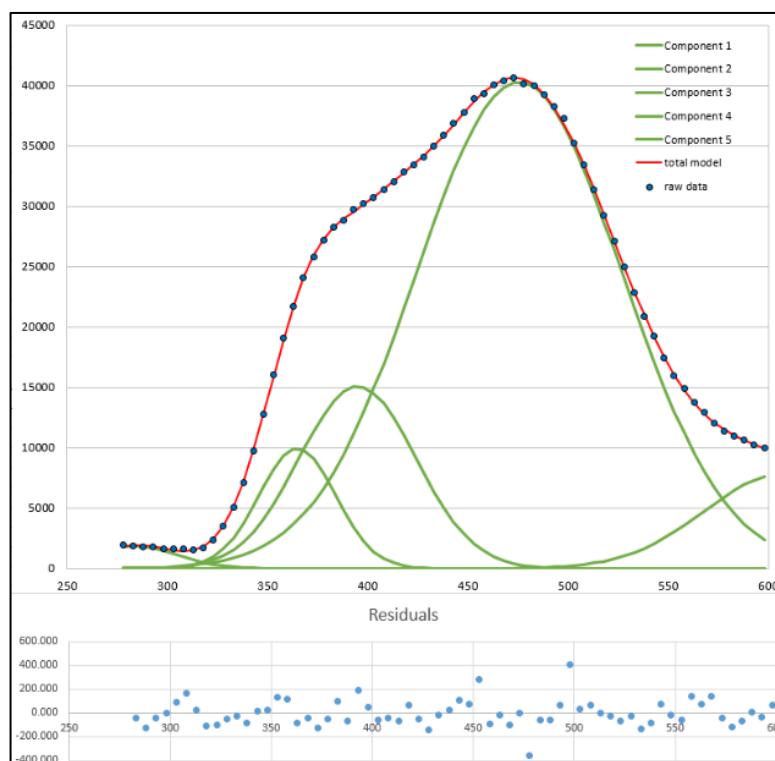


rumore è individuare una regione sulla linea di base dove il segnale è piatto e qui calcolare la deviazione standard. Questo è facile da fare con un computer se il segnale è digitalizzato. La cosa importante è che si deve conoscere abbastanza sulla misurazione e i dati che genera per riconoscere i tipi di segnali riproducibili, in modo da avere la possibilità di distinguere quale sia il *segnale* e quale il *rumore*.

È importante comprendere che le deviazioni standard calcolate su un piccolo insieme di misure possono essere molto più alte o molto inferiori rispetto alla deviazione standard effettiva su un numero maggiore di misure. Per esempio, la funzione Matlab/Octave `randn(1,n)`, dove n è un intero, restituisce n numeri casuali che hanno *mediamente* una media di zero e una deviazione standard di 1.00 se n è grande. (In Python, la funzione random è `np.random.rand(n)`). Ma se n è piccolo, le deviazioni standard saranno diverse ogni volta che si richiamerà questa funzione; per esempio, se $n=5$, la deviazione standard `std(randn(1,5))` può variare casualmente da 0.5 a 2 o anche più. Questa è la [Legge dei Grandi Numeri](#) (pag. 353); è per la natura inevitabile dei piccoli insiemi di numeri casuali che la loro deviazione standard sia solo un'*approssimazione molto approssimativa* della deviazione standard della reale “popolazione” in esame.

Un modo rapido ma approssimativo per stimare visivamente l'ampiezza del rumore è l'intervallo *da picco-a-picco*, che è la differenza tra i valori più alti e quelli più bassi tra due regioni in cui il segnale è piatto. L'intervallo picco-picco di $n=100$ numeri casuali distribuiti normalmente è circa 5 volte la deviazione standard, come si può dimostrare eseguendo più volte questa riga di codice Matlab/Octave: `n=100; rn=randn(1,n); (max(rn)-min(rn))/std(rn)`. Per esempio, i dati nella metà destra della figura a pagina 27 ha un picco al centro con un'altezza di circa 1.0. Anche il rumore da picco a picco sulla linea di base è di circa 1.0, quindi la deviazione standard del rumore è circa 1/5° di quello, o 0.2. Tuttavia, tale rapporto varia con il logaritmo di n ed è più vicino a 3 quando $n = 10$ ed a 9 quando $n = 100000$. Al contrario, la deviazione standard diventa sempre più vicina al valore vero all'aumentare di n . È meglio calcolare la deviazione standard, se possibile.

Oltre alla deviazione *standard*, è anche possibile misurare (ma non usuale) la deviazione *mediana assoluta* (mean absolute deviation "mad"). La deviazione standard è maggiore di quella mediana



assoluta perché quella standard pesa maggiormente le deviazioni più ampie. Per una variabile casuale normalmente distribuita, la deviazione mediana assoluta è in media l'80% di quella standard: $\text{mad}=0.8*\text{std}$.

La *qualità* di un segnale è spesso espressa quantitativamente come il *rapporto segnale-rumore* (S/N ratio o SNR), che è il rapporto dell'ampiezza dell'effettivo segnale (p.es. l'ampiezza media o l'altezza del picco) con la deviazione standard del rumore. Quindi il rapporto S/N dello spettro in figura a pagina 13 è circa $0.08/0.001 = 80$ ed il segnale a pagina 27 ha un rapporto S/N di

$1.0/0.2 = 5$. Quindi diremmo che la qualità del primo è migliore perché ha un rapporto S/N

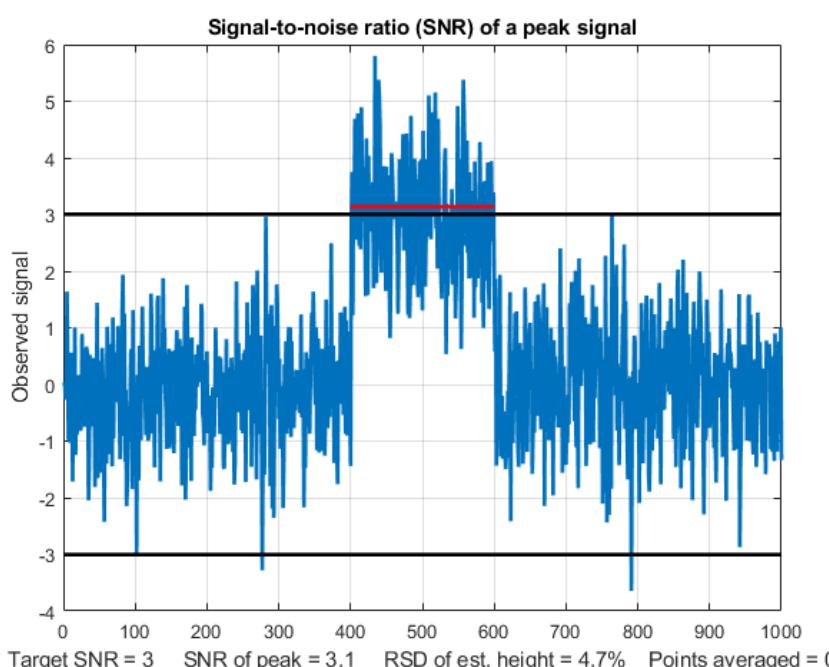
maggiori. Misurare il rapporto S/N è molto facile se il rumore si può misurare separatamente, in assenza del segnale. A seconda del tipo di esperimento, è possibile acquisire letture del solo rumore, per esempio su un segmento della linea di base prima o dopo la presenza del segnale. Tuttavia, se l'ampiezza del rumore dipende dal livello del segnale, allora lo sperimentatore deve tentare di produrre un livello costante del segnale per consentire la misura del rumore sul segnale. In alcuni casi, è possibile utilizzare "l'approssimazione iterativa" (pag. 190) per approssimare accuratamente il segnale mediante una funzione matematica di smoothing (come una polinomiale o la somma ponderata di un numero di semplici funzioni a forma di picco). Il rumore può quindi essere isolato sottraendo l'approssimazione dal segnale sperimentale senza smoothing. Per esempio, il grafico precedente, mostra un *segnalet sperimentale di dati reali* (punti blu scuro) che non arriva mai sulla linea di base per consentire una misura semplice del rumore. Ma il segnale può essere approssimato approssimando un modello (linea rossa) costituito da 5 funzioni di picco Gaussiano sovrapposte (pagina 192). La differenza tra i dati originali e il modello, mostrata in basso (celeste), è una buona misura del rumore casuale nei dati. (In alcuni casi potrebbe essere possibile determinare la deviazione standard di misure ripetute dell'oggetto, come ad esempio le altezze o le aree dei picchi, piuttosto che provare a stimare il rumore da una singola registrazione dei dati).

Il limite del rilevamento

Il "limite del rilevamento" è definito come il segnale più piccolo che è possibile rilevare in modo affidabile in presenza di rumore. Nell'analisi chimica quantitativa, è solitamente definito come la concentrazione che produce il più piccolo segnale rilevabile (Riferimento 92). Un segnale che sia al di sotto del limite di rilevamento non può essere rilevato in modo affidabile; ovvero, se la misura viene ripetuta, il segnale verrà spesso "perso nel rumore" e riportato come nullo. Un segnale al di sopra del limite di rilevamento verrà rilevato in modo affidabile e raramente o mai riportato come nullo.

Il valore più comune di rapporto segnale/rumore per un rilevamento affidabile è 3. Ciò è illustrato nella figura a lato (creata dallo script Matlab/Octave [SNRdemo.m](#)).

Questa figura mostra un segnale rumoroso sotto forma di un impulso rettangolare. Definiamo il "segnale" come l'ampiezza media del segnale durante l'impulso, indicato dalla linea rossa, che è circa 3. Definiamo il "rumore" come la deviazione standard del rumore casuale sulla linea di base prima e dopo l'impulso, che è circa 1.0, quasi 1/5 del rumore da picco a picco



della linea di base (linee nere). Pertanto, il rapporto segnale/rumore (SNR) in questo caso è circa 3, che è una definizione comune di SNR al limite di rilevamento. Ciò significa che i segnali inferiori a questo dovrebbero essere segnalati come "non rilevabili".

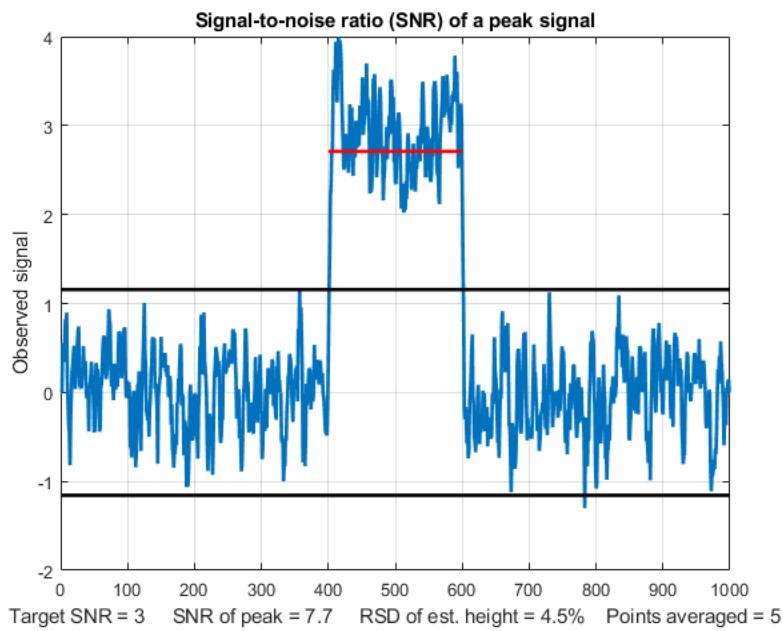
Ma c'è un problema. Il segnale qui è chiaramente rilevabile a occhio; infatti, dovrebbe essere possibile rilevare visivamente segnali *più piccoli* di questo. Come può essere? La risposta è "fare la

media". Guardando questo segnale, si sta *inconsciamente stimando la media dei punti* sull'impulso del segnale e sulla linea di base e la personale capacità di rilevamento visivo viene migliorata da questa media. Senza tale media, guardando solo i *singoli* punti nel segnale, all'incirca solo la metà di questi punti isolati soddisfarebbe il criterio dell'SNR=3. Si può vedere, nel grafico sopra, che diversi punti sul picco del segnale sono un realtà *più bassi* di alcuni punti più alti sulla linea di base. Ma questo non è un problema nella pratica, perché qualsiasi software scritto correttamente includerà la media che raddoppia la media visiva che tutti noi facciamo.

Nello script [SNRdemo.m](#), il numero di punti mediati è regolato dalla variabile "AveragePoints" nella riga 7. Se lo si imposta a 5, il risultato (mostrato di seguito) mostra che tutti i punti del segnale

(ciascuno dei quali è ora la media di 5 punti originali) stanno al di sopra dei punti più alti della linea di base.

Questo grafico rappresenta più da vicino *come che giudichiamo* quando guardiamo un segnale come quello nel grafico precedente, che ha una netta separazione tra il segnale e la linea di base. L'SNR del picco è migliorato da 3.1 a 7.7 e *il limite di rilevamento si ridurrà di conseguenza*. Come regola pratica, per il tipo più comune di rumore casuale, questo diminuisce all'incirca della radice quadrata del numero di punti mediati (in questo caso, $\sqrt{5}=2.2$). Valori più alti miglioreranno ulteriormente



l'SNR e ridurranno la deviazione standard relativa del segnale medio, ma il *tempo di risposta* – che è il tempo impiegato dal segnale per raggiungere il valore medio – diventerà sempre più lento all'aumentare del numero di punti mediati. Ciò viene mostrato da [un altro grafico, con 100 punti mediati](#). Con un segnale molto più basso pari a 1.0, il segnale originale è [non rilevabile visivamente](#), ma con una media di 100 punti, la [precisione del segnale è buona](#); in questo caso la media digitale batte la media visiva. Si osserverebbe un comportamento simile se il segnale fosse un picco arrotondato anziché un rettangolo.

In [SNRdemo.m](#), il rumore è costante e indipendente dall'ampiezza del segnale, che è il caso più comune. Nella variante [SNRdemoHetero.m](#), il rumore nel segnale è direttamente proporzionale al livello del segnale o alla sua radice quadrata, di conseguenza il limite del rilevamento dipende dal rumore costante della linea di base ([grafico](#)). Vedere pagina 29. Nella variante [SNRdemoArea.m](#), è l'*area* del picco che viene misurata anziché la sua altezza, in cui si ottiene un SNR migliorato della radice quadrata dell'ampiezza del picco ([grafico](#)).

Un esempio di applicazione pratica di un segnale come quello illustrato nelle figure precedenti potrebbe essere l'accensione di una spia o di un cicalino quando il segnale supera la soglia di 1,5. Non funzionerebbe se si usasse il segnale *originale non filtrato* della pagina precedente; non c'è un valore di soglia che non verrebbe mai superato dalla linea di base ma superato sempre dal segnale. Soltanto il segnale *mediato* attiverebbe in modo affidabile l'allarme al di sopra della soglia di 1.5 e non lo attiverebbe mai al di sotto di 1.5.

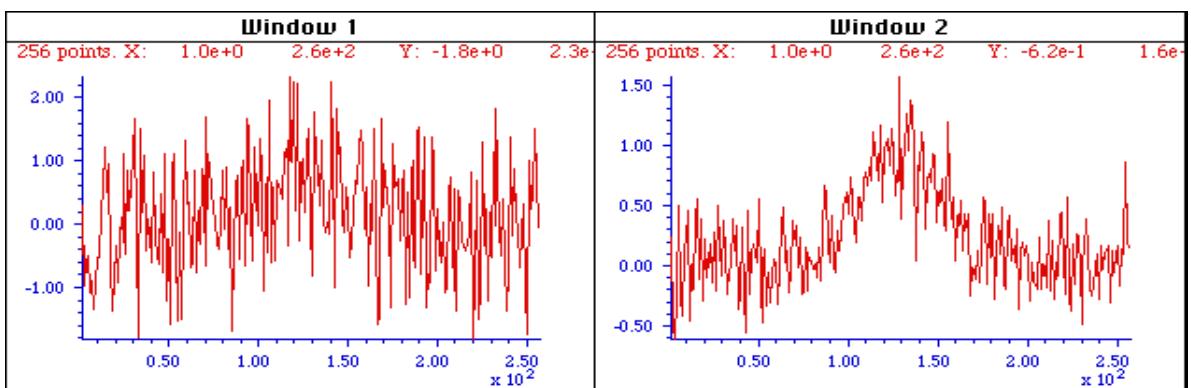
Si sentirà anche il termine "Limite di determinazione", che è il segnale o la concentrazione più bassa che raggiunge una precisione minima accettabile, definita come la deviazione standard

relativa dell'ampiezza del segnale. Il limite di determinazione viene definito con un rapporto segnale/rumore molto più alto, diciamo 10 o 20, a seconda dei requisiti delle proprie applicazioni. La media, come quella fatta qui, è la forma più semplice di "smoothing", che viene trattata nel prossimo capitolo (pag. 38).

Media di insieme

Una cosa fondamentale che distingue davvero il segnale dal rumore è che il rumore casuale non è lo stesso da una misura del segnale all'altra, mentre il vero segnale è (idealmente) riproducibile.

Quindi, se il segnale può essere misurato più di una volta, si può usare questo fatto misurando il segnale più e più volte, il più velocemente possibile e *sommando* tutte le misure punto per punto, e poi dividendo per il numero di segnali mediati. Questa è chiamata *media d'insieme [ensemble averaging]* ed è uno dei metodi più potenti per migliorare i segnali, quando applicabile. Affinché funzioni correttamente, il rumore deve essere casuale ed il segnale deve avvenire nello stesso tempo per ogni ripetizione. Si osservi l'esempio in figura.



In Window 1 (a sinistra) c'è una singola misura del segnale molto rumoroso. C'è un ampio picco al centro del segnale, ma è difficile misurarne la posizione, la larghezza e l'altezza in modo accurato perché il rapporto S/N è pessimo. In Window 2 (a destra) c'è la media di 9 misure ripetute del segnale, mostra chiaramente il picco emergente dal rumore. Il miglioramento atteso nel rapporto S/N è 3 (la radice quadrata di 9). Spesso è possibile mediare centinaia di misure, ottenendo un miglioramento molto più sostanziale. Il rapporto S/N nel segnale medio risultante nell'esempio è circa 5.

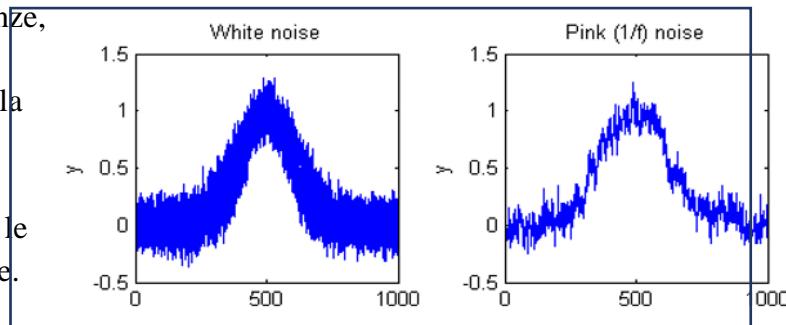
Lo script Matlab/Octave [EnsembleAverageDemo.m](#) illustra graficamente la tecnica per un insieme di 500 segnali. (If you are reading this online, [cliccare per il grafico](#)). Altri esempi vengono mostrati nei link di queste animazioni video, [EnsembleAverage1.wmv](#) o [EnsembleAverageDemo.gif](#), che mostrano la media d'insieme di 1000 ripetizioni di un segnale, migliorando il rapporto S/N di circa 30 volte. È anche possibile ridurre il rumore della digitalizzazione facendo la media d'insieme, ma solo se sono presenti, o sono state aggiunte, piccole quantità di rumore casuale al segnale; vedere pagina 300.

Animazione visuale di una media di insieme. Questi 17 secondi di filmato ([EnsembleAverage1.wmv](#)) mostrano la "ensemble averaging" di 1000 ripetizioni di un segnale con un pessimo rapporto S/N. Il segnale di per sé consiste in tre picchi posizionati a $x = 50, 100$ e 150 , con altezze di 1, 2 e 3 unità. I picchi del segnale sono sepolti nel rumore casuale la cui deviazione standard è 10. Pertanto, il rapporto S/N del picco più piccolo è 0.1, che è troppo basso per essere persino visto come segnale e tanto meno per essere misurato. Il video mostra il segnale medio accumulato mentre si eseguono 1000 misure. Alla fine, il rumore si riduce (in media) della radice quadrata di 1000 (circa 32), in modo che il rapporto S/N dei picchi più piccoli finisce per essere di circa 3, quanto basta per rilevarne affidabilmente la presenza. Se si sta leggendo online, click [qui](#) per scaricare un breve video (2 MByte) in formato WMV.

Distribuzione in frequenza del rumore casuale

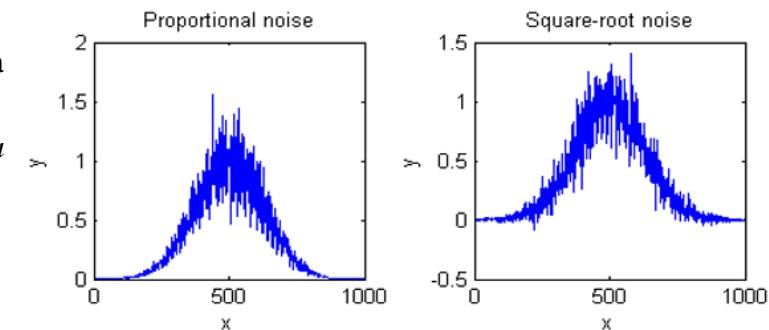
Talvolta il segnale e il rumore possono essere parzialmente distinti in base alle componenti in frequenza: per esempio, il segnale può contenere soprattutto parti a bassa frequenza e il rumore può essere localizzato a frequenze più alte o distribuito su una gamma molto più ampia di frequenze.

Questa è la base del filtraggio e dello smoothing (pag. 38). Nelle figure precedenti, il picco stesso contiene principalmente componenti a bassa frequenza, mentre il rumore è (apparentemente) casuale e distribuito su una gamma molto più ampia di frequenze. La frequenza del rumore è caratterizzata dal suo spettro delle frequenze, spesso descritto in termini di colore del rumore. Il *rumore bianco* è casuale ed ha la stessa potenza su tutta la gamma di frequenze. Il suo nome deriva dalla *luce bianca*, che ha la stessa luminosità a tutte le lunghezze d'onda nella regione del visibile. Il rumore nei segnali dell'esempio



precedente, e nella metà sinistra della figura, è *bianco*. Nel campo acustico, il rumore bianco suona come un *sibilo*. Nelle misure scientifiche, il rumore bianco è molto comune. Ad esempio, il rumore di quantizzazione, il rumore di Johnson-Nyquist (termico), il rumore fotonico e il rumore prodotto da singoli spike hanno tutti una distribuzione in frequenza bianca, ed hanno tutti in comune la loro origine in eventi istantanei quantizzati in discreto, come il flusso dei singoli elettroni e fotoni.

Un rumore che ha un carattere più evidente alle basse frequenze, cioè che ha più potenza alle basse frequenze anziché alle alte, è spesso chiamato "rumore rosa". Nel campo acustico, il rumore rosa suona come un *ruggito*. (Una sotto-specie frequente di rumore rosa è il "rumore 1/f", dove la potenza del rumore è inversamente proporzionale alla frequenza, illustrata nel quadrante in alto a destra della figura a lato). Il rumore rosa è più fastidioso del rumore bianco perché una *data deviazione standard del rumore rosa ha un maggior effetto sull'accuratezza della misura rispetto alla stessa deviazione standard del rumore bianco* (come dimostrato dalla funzione Matlab/Octave noisetest.m, che genera la figura a lato).



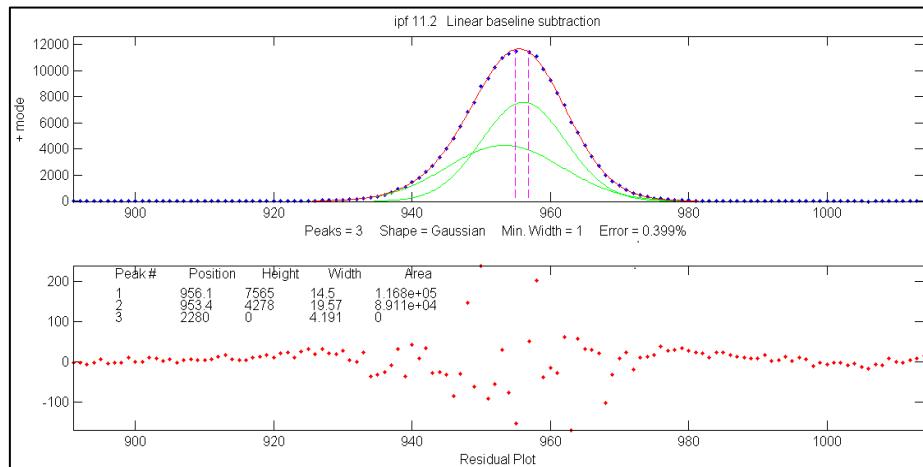
Inoltre, l'applicazione dello smoothing e del filtraggio passa-basso (pagina 38) per ridurre il rumore è più efficace per il rumore bianco che per quello rosa. Quando è presente il rumore rosa, a volte è utile applicare delle tecniche di modulazione, per esempio, il "chopping" ottico o la modulazione a lunghezza d'onda nelle misure ottiche, per convertire i segnali in corrente continua (DC) in segnali in corrente alternata (AC), aumentando così la frequenza del segnale verso una regione delle frequenze dove il rumore è più basso. In tali casi, si è soliti utilizzare un amplificatore lock-in, o il suo equivalente digitale, per misurare l'ampiezza del segnale. Un altro tipo di rumore misurato a bassa frequenza è il rumore *browniano*, dal nome del botanico Robert Brown. Viene anche chiamato "rumore rosso" [o marrone] (per analogia con il rumore rosa) o "passeggiata aleatoria", ed ha una potenza inversamente proporzionale al *quadrato* della frequenza. Questo tipo di rumore si osserva nei segnali sperimentali e può interferire seriamente con l'accuratezza delle misure. Cfr. pagina 310: *Passeggiate aleatorie [Random walks] e correzione della linea di base*.

Al contrario, il rumore che ha più potenza alle frequenze *alte* è detto rumore "blu". Questo tipo di rumore si incontra meno spesso nel lavoro sperimentale, ma può verificarsi in segnali elaborati che sono stati soggetti a una sorta di processo di differenziazione (pag. 58) o che sono stati deconvoluti da un processo di [blurring] o di ampliamento (pag. 107). Il rumore blu è *più facile* da ridurre con lo

smoothing (pag. 28) e ha un effetto minore sulle approssimazioni dei minimi quadrati rispetto alla quantità equivalente di rumore bianco.

Dipendenza dall'ampiezza del segnale

Il rumore può anche essere caratterizzato dal modo in cui varia con l'ampiezza del segnale. Il rumore costante di "sottofondo" è indipendente dall'ampiezza del segnale. D'altro canto il rumore può aumentare con l'ampiezza del segnale, che è un comportamento spesso osservato nella spettroscopia di emissione, nella spettroscopia di massa e nello spettro in frequenza dei segnali. I



nomi tecnici per questi due tipi di comportamenti sono rispettivamente *Omoschedastico* e *Eteroschedastico*, rispettivamente. Un modo per osservarlo consiste nel selezionare un segmento di segnale in cui l'ampiezza del segnale varia molto,

approssimare il segnale a un polinomio, o a un modello a picchi multipli (pag. 195), e osservare come i residui variano con l'ampiezza del segnale. Il segnale viene mostrato a lato, nel pannello superiore, mostra un po' di rumore visibile. La differenza tra quel segnale (i puntini blu) e il modello approssimato (la linea rossa) da un'operazione di approssimazione dei minimi quadrati (pag. 190) è mostrata nel pannello inferiore, lasciando il rumore facilmente visibile (puntini rossi). Chiaramente il rumore in questo caso *aumenta* con l'ampiezza del segnale. In altri casi, il rumore potrebbe aumentare con la radice quadrata del segnale, oppure potrebbe essere indipendente dall'ampiezza del segnale come nell'esempio a pagina 24.

Spesso, c'è un mix di rumori con diversi comportamenti. In spettroscopia ottica, si riconoscono tre tipi di rumore, in base alla loro origine e a come variano con l'intensità della luce: *rumore fotonico*, *rumore del rivelatore* e *rumore flicker (fluttuazioni)*. Il rumore fotonico (spesso il rumore limitante negli strumenti che usano rilevatori con foto-moltiplicatori) è *bianco* ed è proporzionale alla *radice quadrata* dell'intensità della luce. Il rumore del rilevatore (spesso il rumore limite negli strumenti che utilizzano rivelatori a fotodiodi a stato solido) è *indipendente* dall'intensità della luce e quindi l'SNR è direttamente proporzionale all'intensità della luce. Il rumore da sfarfallio [flicker], provocato dall'instabilità della sorgente di luce, vibrazioni, errori di posizionamento della cella campione, turbolenza del campione, dispersione della luce per delle particelle sospese, polvere, bolle, ecc., è direttamente proporzionale all'intensità della luce (e solitamente è *rosa* anziché *bianco*), quindi il rapporto S/N dello sfarfallio non diminuisce aumentando l'intensità della luce. In pratica, è probabile che il rumore totale osservato sia un contributo di tutti e tre i tipi di dipendenza dall'ampiezza, nonché un misto di rumori bianchi e rosa.

Solo in pochissimi casi speciali è possibile eliminare completamente il rumore, quindi di solito bisogna accontentarsi aumentando il più possibile il rapporto S/N. La soluzione, in qualsiasi sistema sperimentale consiste nel capire le possibili fonti del rumore, suddividere il sistema nelle sue parti e misurare il rumore generato in ognuna di esse, separatamente, per poi cercare di ridurre o compensare il più possibile ciascuna sorgente di rumore. Ad esempio, nella spettroscopia ottica, il rumore da sfarfallio della sorgente può essere spesso ridotto o eliminato utilizzando la

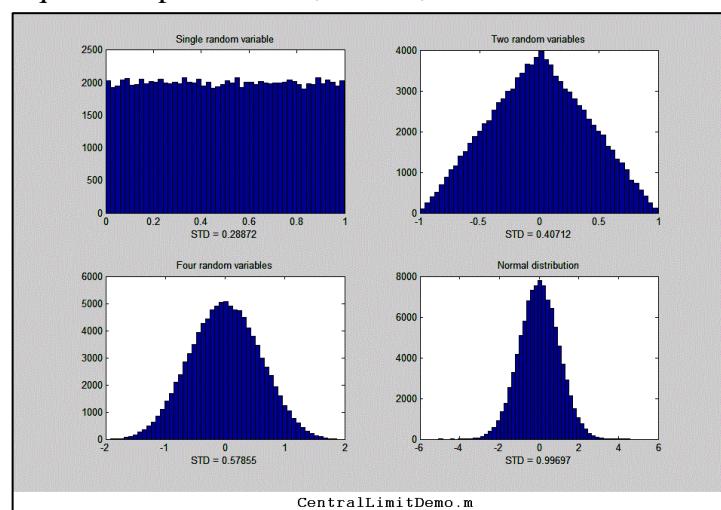
stabilizzazione feedback, scegliendo una sorgente di luce migliore, utilizzando un campione standard interno http://en.wikipedia.org/wiki/Internal_standard o usando un tipo di strumento appositamente progettato come quelli a doppio-fascio [double-beam], a doppia lunghezza d'onda, derivativo o a modulazione della lunghezza d'onda (pagina 312). L'effetto del rumore fotonico e di quello del rilevatore si possono ridurre aumentando l'intensità della luce al rilevatore, e il rumore elettronico si può talvolta ridurre raffreddando o migliorando il rilevatore e/o l'elettronica. Il rumore a schema fisso, negli array di rilevatori, si può correggere via software. Solo il *rumore fotonico* può essere previsto dai primi principi (p.es. come viene fatto in questi fogli di calcolo che simulano il comportamento del segnale-rumore limitato dal rumore fotonico della [spettrofotometria ultravioletta-visibile](#), della [spettrofotometria di fluorescenza](#) e della [spettrofotometria della emissione atomica](#)).

La distribuzione di probabilità del rumore casuale

Un'altra proprietà che distingue il rumore casuale è la sua distribuzione di probabilità, la funzione che descrive la probabilità che una variabile casuale rientri in un certo intervallo di valori. Nelle misure fisiche, la distribuzione più comune è chiamata una *curva normale* (detta anche curva a “campana” o a “pagliaio”) ed è descritta da una funzione

Gaussiana http://en.wikipedia.org/wiki/Gaussian_function, $y=e^{-(x-\mu)^2 / (2\sigma^2)} / (\sqrt{2\pi}\sigma)$, dove μ è il valore medio (media) e σ (σ) è la deviazione standard. In questa distribuzione, gli errori di rumore più comuni sono piccoli (cioè vicini alla *media*) e gli errori diventano meno comuni quanto maggiore è la loro deviazione dalla media. Allora perché questa distribuzione è così comune? Il rumore osservato nelle misure fisiche è spesso la somma bilanciata di molti eventi casuali non osservati, ognuno dei quali ha una distribuzione di probabilità sconosciuta, per esempio, alle proprietà cinetiche di gas o liquidi o al comportamento quantistico di particelle fondamentali come fotoni o elettroni. Ma quando molti di questi eventi si combinano per formare la variabilità complessiva di una quantità osservata, la distribuzione di probabilità risultante è quasi sempre *normale*, ovvero, descritta da una funzione Gaussiana. Questa comune osservazione

è riassunta nel *Teorema centrale del limite*.



Matlab/Octave). Il grafico in alto a sinistra della figura mostra la distribuzione di probabilità, detta “istogramma”, di quella variabile casuale. In seguito, si uniscono i due insiemi di tali variabili casuali indipendenti e uniformemente distribuite (sottraendoli in modo che la media sia centrata sullo zero). Il risultato (mostrato nel grafico in alto a destra nella figura) ha una distribuzione *triangolare* tra -1 e +1, col punto più alto a zero, perché ci sono molti modi per far sì che la differenza tra due numeri casuali sia piccola, ma un solo modo affinché la differenza sia 1 o -1 (ciò capita solo se un numero è esattamente zero e l'altro è esattamente 1). In seguito, si combinano *quattro* variabili casuali indipendenti (in basso a sinistra); la distribuzione risultante ha un intervallo totale da -2 a +2, ma è anche ancora *meno* probabile che il risultato sia prossimo a 2 o a -2 e molti *più* modi che portino il risultato ad essere piccolo, quindi la distribuzione è più stretta e più

Una simulazione può dimostrare come questo comportamento si manifesti naturalmente. Nell'esempio a lato si comincia con un gruppo di 100.000 numeri casuali *uniformemente distribuiti* che hanno la stessa probabilità di avere un dato valore entro certi limiti, tra 0 e +1, in questo caso (come la funzione "rand" nella maggior parte degli spreadsheet e in

arrotondata, e sta già iniziando ad assomigliare alla distribuzione normale Gaussiana (generata utilizzando la funzione “randn” e mostrata per riferimento in basso a destra). Se si combinano sempre più variabili casuali uniformi e indipendenti, la distribuzione di probabilità combinata si approssima sempre di più alla Gaussiana mostrata per confronto in basso a destra. Il punto importante è che *la distribuzione Gaussiana emergente che si osserva qui non è forzata da ipotesi precedenti; sorge naturalmente*. (Si può scaricare uno script Matlab per questa simulazione da <http://terpconnect.umd.edu/~toh/spectrum/CentralLimitDemo.m>).

Sorprendentemente, *le distribuzioni dei singoli eventi non contano affatto*. Si possono modificare le singole distribuzioni in questa simulazione, sostituendo la funzione *rand* con versioni modificate come *sqrt(rand)*, *sin(rand)*, *rand^2*, *log(rand)*, ecc., per ottenere altre singole distribuzioni *radicalmente non normali*. Ma sembra che non importa quale sia la distribuzione della singola variabile casuale, nel momento in cui si combinano anche solo quattro di esse, la distribuzione risultante è già visivamente vicina alla normale. Le osservazioni macroscopiche del mondo reale sono spesso il risultato di *milioni o miliardi* di singoli eventi microscopici, quindi qualunque sia la distribuzione di probabilità dei *singoli* eventi, le osservazioni macroscopiche *combinate* si avvicinano a una distribuzione normale quasi perfettamente. È su questa comune aderenza alle distribuzioni normali che si basano le comuni procedure statistiche; l'uso della media, della deviazione standard σ , delle approssimazione dei minimi quadrati, degli intervalli di confidenza, ecc., sono tutti basati sull'*assunzione* di una distribuzione normale. Ma solitamente è un'ipotesi molto buona.

Anche così, gli errori sperimentali e il rumore non sono *sempre* normali; a volte ci sono errori molto grandi che cadono ben oltre il range “normale”. Sono detti “valori anomali” e possono avere un grande effetto sulla deviazione standard. In questi casi, è possibile utilizzare lo “[scarto interquartile](#)” (IQR), definito come la differenza tra i quartili superiore e inferiore, invece della deviazione standard, perché *lo scarto interquartile non è influenzato da qualche valore anomalo*. Per una distribuzione *normale*, lo scarto interquartile è pari a 1.34896 volte la deviazione standard. Un modo rapido per verificare la distribuzione di un ampio insieme di numeri casuali è calcolare sia la deviazione standard che lo scarto interquartile; se sono più o meno uguali, la distribuzione è probabilmente normale; se la deviazione standard è *molto* più ampia, l'insieme dei dati probabilmente contiene valori anomali e la deviazione standard *senza* i valori anomali si può stimare meglio dividendo lo scarto interquartile per 1.34896.

L'importanza della distribuzione normale sta nel fatto che se si conosce la deviazione standard (di solito data dal simbolo “ σ ”) di un valore misurato, si può prevedere la probabilità che la misura possa essere errata di una certa quantità. Circa il 68% dei valori ricavati da una distribuzione normale sono entro un σ di distanza dalla media; il 95% dei valori si trova entro 2σ , e il 99.7% entro 3σ . Questa è nota come la [regola 3-sigma](#). Ma il vero problema pratico è questo: le deviazioni standard: *le deviazioni standard sono difficili da misurare con precisione a meno che non si disponga di un numero elevato di campioni*. Vedere “*La Legge dei Grandi Numeri*” ([pagina 353](#)).

Le tre caratteristiche del rumore discusse nei paragrafi precedenti - la distribuzione della frequenza, la distribuzione dell'ampiezza e la dipendenza dal segnale - sono reciprocamente indipendenti; in linea di principio un rumore può avere qualsiasi combinazione di queste proprietà.

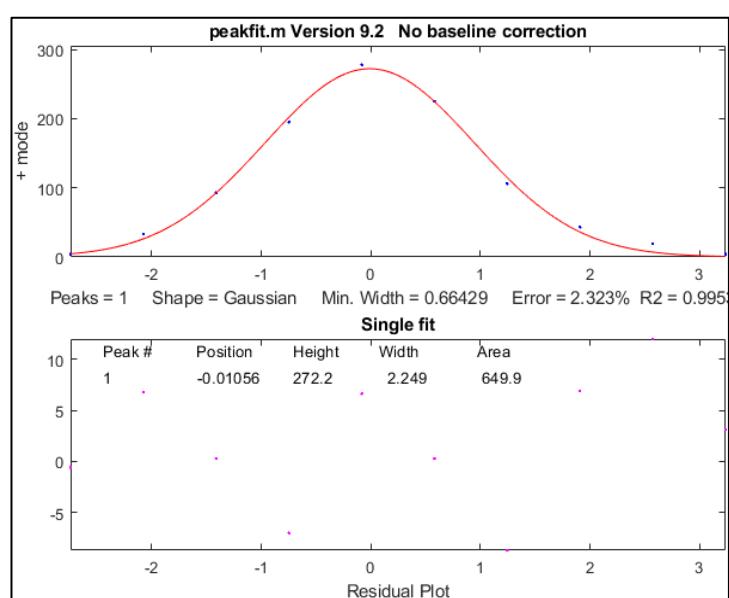
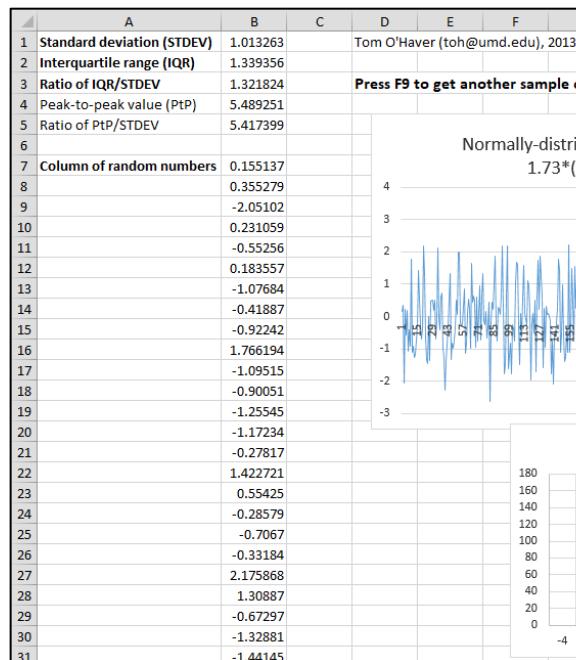
Rappresentazione del rumore casuale negli Spreadsheet

Gli spreadsheet popolari, come *Excel* o *Open Office Calc*, hanno delle funzioni utilizzabili per calcolare, misurare e disegnare segnali e rumore. Per esempio, la formula della cella per calcolare

un punto su un picco **Gaussiano** è `amplitude*EXP(-1*((x-position)/(0.60056120439323*width))^2)`, dove 'amplitude' è l'altezza massima del picco, 'position' è la posizione del massimo sull'asse x, 'width' la larghezza totale a metà altezza del picco (FWHM) (che è pari a σ per 2.355), e 'x' è il valore della variabile indipendente in tale punto. La formula della cella per un picco **Lorentziano** è `amplitude/(1+((x-position)/(0.5*width))^2)`. Altre funzioni utili comprendono AVERAGE, MAX, MIN, STDEV, VAR, RAND, e QUARTILE. La maggior parte dei fogli di calcolo hanno solo una funzione per i numeri casuali *distribuiti uniformemente* (RAND) e non una funzione di numeri casuali *distribuiti normalmente*, ma è molto più realistico simulare errori che sono distribuiti normalmente. Ma non c'è da preoccuparsi, si può utilizzare il Teorema Centrale del Limite per creare approssimativamente numeri casuali normalmente distribuiti combinando diverse funzioni RAND, per esempio, questa strana espressione `SQRT(3)*(RAND()-RAND()+RAND()-RAND())` crea numeri casuali quasi normali con una media di zero, una deviazione standard molto prossima a 1 e un intervallo massimo di ± 4 . Viene usato questo trucco nei [modelli di spreadsheet che simulano il funzionamento di strumenti analitici](#). (L'espressione `SQRT(2)*(RAND()-RAND()+RAND()-RAND()+RAND()-RAND())` funziona allo stesso modo, ma ha un intervallo massimo maggiore). Per creare numeri casuali con una deviazione standard diversa da 1, basta semplicemente *moltiplicare* per quel numero. Per creare numeri casuali con una media diversa da zero, basta semplicemente *addizionare* quel numero.

Lo *scarto interquartile* (IQR) può essere calcolato in un foglio di calcolo sottraendo il terzo quartile dal primo (ad esempio, **QUARTILE(B7: B504,3) - QUARTILE(B7: B504, 1)**).

I fogli di calcolo [RandomNumbers.xls](#), per Excel, e [RandomNumbers.ods](#), per OpenOffice, (schermata di seguito), e lo script Matlab/Octave [RANDtoRANDN.m](#), dimostrano tutti questi fatti. La stessa tecnica viene usata nello spreadsheet [SimulatedSignal6Gaussian.xlsx](#), che calcola e disegna un segnale simulato composto da un massimo di 6 bande Gaussiane sovrapposte con in più del rumore bianco casuale.



Funzioni random in Matlab e in Python

Matlab e Octave hanno funzioni utili per calcolare, misurare e disegnare segnali e rumore, tra cui mean, max, min, std,

kurtosis, skewness, plot, hist, rand e randn. Basta digitare "help" e il nome della funzione al prompt dei comandi, p.es., "help mean". *La maggior parte di queste funzioni si applica a vettori e matrici nonché a variabili scalari.* Per esempio, se si ha una serie di risultati in una variabile vettoriale 'y', mean(y) restituisce la media e std(y) restituisce la deviazione standard di tutti i valori in y. Per i vettori, std calcola sqrt(mean(y.^2)). Si può sottrarre un numero scalare da un vettore (per esempio, $v = v - \min(v)$) setta a zero il numero più basso del vettore v). Se si ha un gruppo di segnali nelle righe di una matrice S, in cui ogni colonna rappresenta il valore di ciascun segnale allo stesso valore della variabile indipendente (p.es. il tempo), si può calcolare la media d'insieme di tali segnali semplicemente digitando "mean(S)", che calcola la media di ciascuna colonna di S. Si noti che i nomi delle funzioni e delle variabili sono "case-sensitive". (È possibile aprire il codice di qualsiasi funzione, selezionandone il nome, e aprendola con "open..").

La funzione "randn" in Matlab/Octave genera numeri casuali normalmente distribuiti con una media di zero e una deviazione standard di 1: p.es., randn(1,100) restituisce un vettore di 100 di questi numeri. (In Python, dopo aver importato "numpy" come "np", la sintassi è simile:
`np.random.randn(100)`).

Nell'esempio seguente, "rand" viene usato per generare 100 numeri casuali, di seguito la funzione "hist" di Matlab calcola l'*istogramma* (distribuzione della probabilità) di questi numeri casuali, poi la funzione **peakfit.m** (pagina 384, [link per il download](#)) approssima una funzione Gaussiana (disegnata con una linea rossa) a tale distribuzione.

```
[N,X]=hist(randn(size(1:100)));
peakfit([X;N]);
```

Se si cambia il 100 in 1000 o in un numero ancora più alto, la distribuzione di questi numeri diventa *sempre più prossima* ad una Gaussiana perfetta e il suo picco si avvicina a 0.00. Questa è una [animazione MP4](#) che dimostra l'emergere graduale di una distribuzione Gaussiana normale man mano che il numero di campioni "randn" aumenta da 2 a 1000. Da notare quanti campioni siano necessari prima che la distribuzione normale sia ben formata. La funzione "randn" viene utilizzata nel signal processing prevedere l'incertezza delle misure in presenza di rumore casuale, per esempio utilizzando i metodi Monte Carlo o bootstrap che verranno descritti in una prossima sezione (pagine 161, 162). (Nota: Nella versione PDF di questo libro, si può selezionare per poi copiare e incollare o selezionare e trascinare, esempi di codice su singola o più linee nell'editor di Matlab o di Octave o direttamente sulla riga di comando e poi premere **Enter** per eseguirlo immediatamente).

Differenza tra script e funzioni

Se ci si accorge di scrivere ripetutamente la stessa serie di comandi Matlab, si prenda in considerazione la creazione di uno *script* o di una *funzione* per risparmiare il codice sul computer e riutilizzarlo facilmente in seguito senza il pericolo di errori tipografici o di maldestre operazioni di copia e incolla. È estremamente utile crearsi in proprio script e funzioni in Matlab o in Octave per automatizzare gli algoritmi usati frequentemente.

In Matlab, script e funzioni sono semplici file di testo salvati con l'estensione ".m" aggiunta al nome. La differenza tra uno script e una funzione è che la definizione di una funzione comincia con la parola 'function'; uno script è solo un elenco di comandi e istruzioni Matlab. Per uno *script*, tutte le variabili definite ed usate vengono elencate nella finestra di lavoro e condivise con altri script. Per una *funzione*, invece, le variabili sono *private e interne alla funzione*; i valori si possono passare alla funzione tramite le variabili di *input* (chiamate *argomenti*), e i valori si possono prelevare dalla

funzione tramite le variabili di *output*, entrambi definiti nella prima riga della definizione della funzione.

```
[variabili di output] = NomeFunzione(variabili di input)
```

Ciò significa che le funzioni costituiscono un ottimo modo per impacchettare blocchi di codice per eseguire operazioni utili in una forma che possa essere riutilizzata, come componenti in *altri* script e funzioni, *senza preoccuparsi che i nomi delle variabili vadano in conflitto causando errori*. Quando si scrive una funzione, la si può salvare sul computer e può essere utilizzata proprio come le funzioni native di Matlab. Oppure la si può caricare nel proprio account Matlab, e la si potrà usare su un tablet o su uno smartphone. Qui c'è un semplicissimo esempio, una funzione che calcola la deviazione standard relativa di un vettore x, [rsd.m](#):

```
function relstddev=rsd(x)
% Deviazione standard relativa del vettore x
relstddev=std(x)./mean(x);
```

In Python, la stessa funzione verrebbe codificata

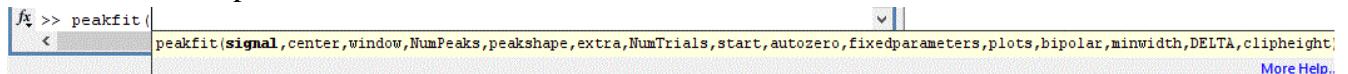
```
def rsd(x):
# Deviazione standard relativa del vettore x
return np.std(x)/np.mean(x)
```

Script e funzioni possono chiamare altre funzioni. Nelle versioni più vecchie di Matlab, devono avere tali funzioni memorizzate nel [path di ricerca](#) di Matlab; le funzioni, d'altra parte, *possono avere le loro sotto-funzioni necessarie all'interno della funzione principale stessa e quindi si possono auto-contenere*. Se si scrive uno script o una funzione che richiama una o più funzioni utente, e si invia il tutto a qualcun altro, ci si accerti di includere tutte le funzioni richiamate. (È meglio rendere tutte le tue funzioni *autonome* includendo tutte le sotto-funzioni). Se uno degli script restituisce un messaggio di errore che dice, "Undefined function...", si deve scaricare la funzione specificata da <http://tinyurl.com/cey8rwh> e metterla nel [path di ricerca di Matlab/Octave](#). **Nota:** in Matlab R2016b o successivo, si POSSONO includere le funzioni negli script; basta posizionarli alla fine dello script e aggiungere un'ulteriore istruzione "end" a ciascuna funzione aggiunta. Vedere https://www.mathworks.com/help/matlab/matlab_prog/local-functions-in-scripts.html.

Per avere la descrizione di una funzione, digitare "help FunctionName" al prompt, dove FunctionName è il nome della funzione o, in Python, "help(FunctionName)". Per scrivere o modificare script e funzioni, Matlab, [l'ultima versione di Octave](#) e Python/Spyder hanno tutti degli editor inclusi. Quando si scrivono e proprie funzioni o script, si dovrebbero sempre aggiungere tante "righe di commento", iniziando col carattere % (o # in Python) spiegando quello che si sta facendo. *Si sarà felici in seguito di averlo fatto*. Il primo gruppo di righe di commenti, fino alla prima riga vuota che inizia con un %, viene usato come "file di help" per quello script o quella funzione. Digitando "help FunctionName" vengono visualizzate le righe di commento per quella funzione o script nella finestra di comando, proprio come avviene per le funzioni e gli script nativi. È una buona idea aggiungere anche uno o più esempi di operazioni che gli utenti possono copiare e incollare sulla riga di comando. Ciò renderà i propri script e funzioni molto più facili da capire ed usare, sia da altre persone che da se stessi in futuro. *Resistere alla tentazione di omettere questo passaggio*. Man mano che si sviluppano funzioni personalizzate per il proprio lavoro, si creerà un "toolkit" [cassetta degli attrezzi] che diventerà utilissima per i colleghi e per se stessi in futuro, *se si commenteranno abbondantemente*. Lo diciamo per esperienza personale. Ma non sempre si seguono questi consigli.

Matlab ha un help utilissimo per le funzioni: quando si digita il nome di una funzione nell'editor di Matlab, se ci si ferma un attimo dopo aver digitato la parentesi aperta subito dopo il nome della funzione, Matlab mostrerà un pop-up con l'elenco di tutte le possibili variabili di input come promemoria. *Questo, funziona anche per le funzioni scaricate e per ogni nuova funzione creata in*

proprio! È particolarmente utile quando la funzione ha così tante possibili variabili di input che è difficile ricordarle tutte. Il popup *resta sullo schermo mentre si digita*, evidenziando ciascuna variabile a turno, per ricordare dove ci si trova:

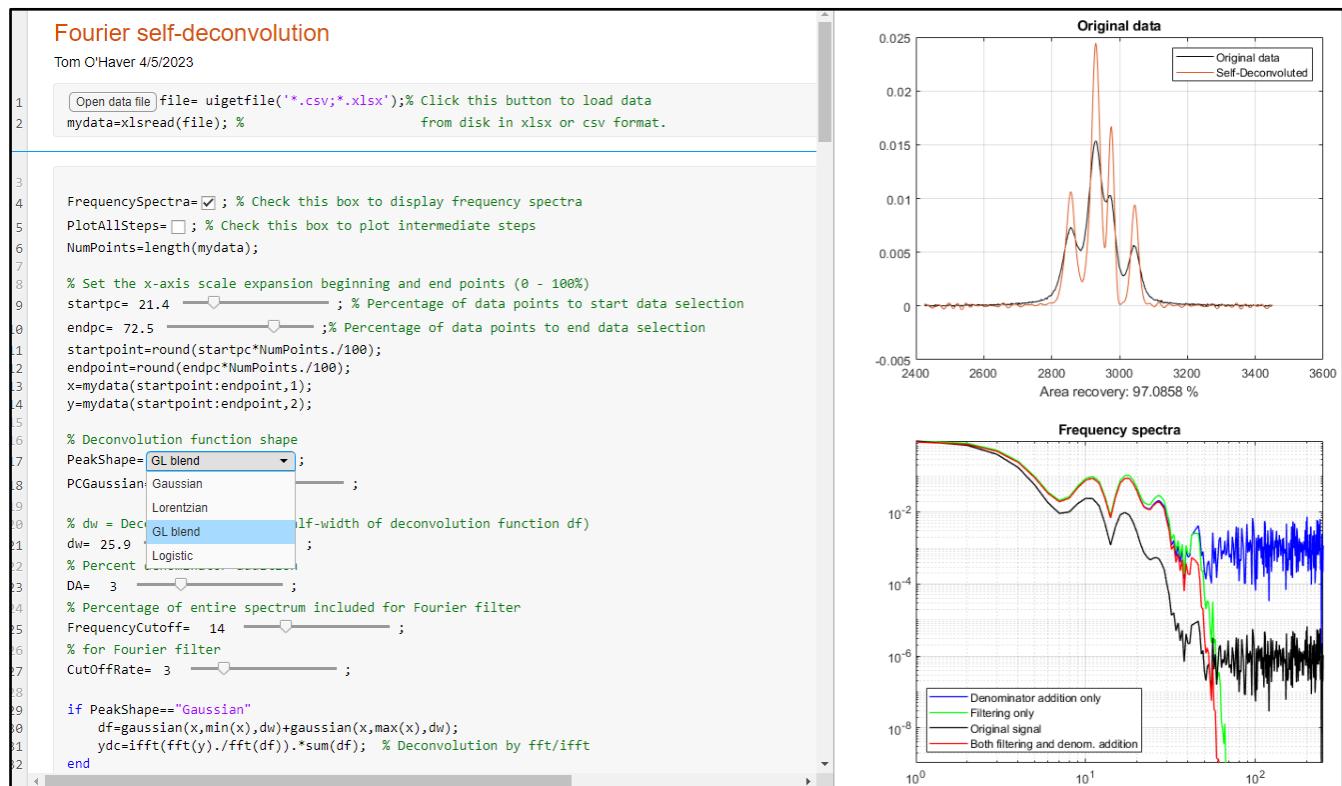


Questa caratteristica è spesso trascurata, ma è molto utile. Cliccando sul link “[More Help...](#)”, a destra, appare l'help per quella funzione in una finestra separata. Nota: Octave non ha questa funzione.

I Live Script

Sia Matlab che Python hanno *alternative interattive* agli script convenzionali. I [Live Script](#) in Matlab sono documenti interattivi che combinano codice, output e testo formattato e controlli interattivi in un unico ambiente chiamato Live Editor. (I Live Script sono disponibili a partire da MATLAB R2016b). Cfr. pag. 360. Python ha i [Jupyter Notebooks](#) che vengono utilizzati per creare una narrazione interattiva attorno al codice. Entrambi semplificano la creazione di documenti interattivi condivisibili con dispositivi di interfaccia utente grafica come menu a discesa, caselle di controllo e dispositivi di scorrimento per regolare i valori numerici in modo interattivo. In Matlab, si può semplicemente aprire uno script convenzionale (.m) nel Live Editor e [inserire gli oggetti dell'interfaccia direttamente nello script](#) dove sarebbero andati i numeri nelle istruzioni di assegnazione. Salvandolo diventa un file .mlx.

Di seguito è mostrato un esempio di una porzione di Live Script. Questo esempio mostra quattro tipi di controlli interattivi. La riga 1 mostra un pulsante che apre un **browser di file** che consente di navigare verso un file specifico, in questo caso un file di dati da elaborare. Le righe 5 e 6 mostrano **caselle di controllo**, utilizzate per abilitare o disabilitare sezioni di codice opzionali. Diverse righe mostrano **cursori numerici**, utilizzati per controllare le variabili continue. La riga 17 mostra un **menu a discesa** che consente scelte multiple, mostrato nello screenshot seguente.



I Live Script producono un output grafico in piccole finestre sul lato destro della finestra del Live Editor, dove si può copiare, eseguire la panoramica, lo zoom ed esportare in file png come al solito utilizzando il mouse. Si può anche convertire qualsiasi elemento grafico dei Live Script in una finestra standard facendo clic sul suo angolo in alto a destra; la finestra standard può poi essere esportata in altri formati grafici, espansa a schermo intero, stampata, ecc.

Altri esempi di Live Script includono il versatile **strumento di smoothing dei dati** mostrato a pagina 55, uno strumento per la **differenziazione** (pagina 74), lo **script di auto-deconvoluzione** mostrato sopra (pagina 120) e uno **strumento per il rilevamento dei picchi** (pagina 245). Questi Live Script sono sorprendentemente facili da creare nell'ambiente Matlab modificando uno script convenzionale e uno **strumento di approssimazione dei picchi** a pagina 427. Consultare la pagina 360 per ulteriori dettagli sullo sviluppo di Live Script e App Matlab.

Funzioni utente relative ai segnali e al rumore.

Ecco alcuni esempi di funzioni-utente create per il toolkit di signal processing di questo libro. Queste non sono funzioni native; le si devono scaricare e metterle nel path di Matlab per usarle.

Data plotting: [plotit.m](#), una funzione facile da usare per disegnare e approssimare dati x,y in matrici o in vettori separati. Per gestire più facilmente segnali *molto grandi*, [plotxrange.m](#) (`[xx,yy,irange] = plotxrange(x,y,x1,x2)`) estrae e disegna i valori di vettori x,y solo per valori di x nel range dei valori di x specificato; [segplot.m](#) (`[s,xx,yy] = segplot(x,y,NumSegs,seg)`) divide i segnali in "NumSegs" segmenti di uguale lunghezza e li disegna contrassegnandoli con linee verticali, ognuno etichettato con un suo numerino in basso e restituisce un vettore 's' degli indici dei segmenti e il sotto-insieme xx,yy, dei valori nel segmento numero 'seg'.

Profili dei picchi. Di seguito sono riportati i link a diverse funzioni Matlab per i profili dei picchi che si incontrano normalmente in chimica analitica come [Gaussiana](#), [Lorentziana](#), [log-normale](#), [Pearson 5](#), [Gaussiana esponenzialmente allargata](#), [Lorentziana esponenzialmente allargata](#), [impulso esponenziale](#), [sech2 NEW](#), [sigmoide](#), [mix di Gaussiana/Lorentziana](#), [Gaussiana biforcata](#), [Lorentziana biforcata](#)), [profilo Voigt](#), [triangolare](#) ed altre. Consultare la pag. 451 e seguenti per una lista più completa. Lo script [Sech2ShapeComparison.m](#) [NEW](#) confronta le forme dell'impulso gaussiano, Lorentziano e sech2, mostrando l'impulso sech2 nella posizione intermedia tra Gaussiano e Lorentziano ([grafico](#)).

[peakfunction.m](#), una funzione che genera uno di quei tipi di picco specificandolo con un numero.

[ShapeDemo](#) mostra graficamente le 16 forme di picco di base, mostrando i picchi di forma variabile come linee multiple. ([grafico](#) a pagina 411)

Generatori di rumore. Ci sono diverse funzioni per simulare i diversi tipi di rumore ([bianco](#), [rosa](#), [blu](#), [proporzionale](#) e [radice quadrata](#)).

Funzioni Varie di Matlab:

[stdev.m](#), una funzione per la deviazione standard che lavora sia in Matlab che in Octave (la funzione std.m nativa si comporta in modo diverso tra Matlab e Octave); [rsd.m](#), per la deviazione standard *relativa*.

[PercentDifference.m](#), calcola semplicemente la differenza percentuale tra due variabili.

[IQrange.m](#) calcola l'intervallo interquartile (spiegato sopra).

halfwidth.m per misurare l'intera larghezza a metà del massimo dei picchi con smoothing con qualsiasi profilo.

<https://terpconnect.umd.edu/~toh/spectrum/ExpBroaden.m> applica l'ampliamento esponenziale a qualsiasi vettore di serie temporali.

rmnan.m rimuove le voci "not-a-number" dai vettori, utile per ripulire i file di dati reali; rmz.m rimuove gli zeri dai vettori, sostituendoli coi numeri prossimi ma diversi da zero.

val2ind.m restituisce l'indice del valore dell'elemento del vettore x che sia più prossimo ad un particolare valore. Questa è una semplice funzione ma è più utile di quanto si immagini. Cercare in questo documento “val2ind” per i diversi esempi dell'uso pratico di questa funzione.

Queste funzioni sono utili nella modellazione e nella simulazione di segnali analitici e per il test delle tecniche di misurazioni (pag. 38). Nella versione PDF di questo libro, si può fare click o ctrl-click su tali link per ispezionare il codice o, col click-destro e selezionando "Save link as...", per scaricarlo sul proprio computer. Dopo aver scaricato queste funzioni e averle inserite nel path di ricerca, si possono usare come qualsiasi altra funzione nativa. Per esempio, si può disegnare un picco Gaussiano con l'aggiunta di rumore bianco digitando `x=[1:256];`

`y=gaussian(x,128,64) + whitenoise(x); plot(x,y)`. Lo script plotting.m, mostrato nella figura a pagina 17, usa la funzione gaussian.m per mostrare le differenze di *altezza, posizione e larghezza* di una curva Gaussiana. Lo script SignalGenerator.m chiama diverse di queste funzioni scaricabili per creare il grafico di un segnale realistico generato al computer con molteplici picchi, su una linea di base variabile, con in più del rumore casuale; si potrebbe provare a modificare le variabili nei posti indicati per farlo assomigliare ai propri tipi di dati. Tutte queste funzioni sono compatibili con l'ultima versione di Octave senza modifiche. Per un elenco completo delle funzioni e degli script scaricabili sviluppate per questo progetto, si vada a pagina 451 o sul Web alla pagina <http://tinyurl.com/cey8rwh>.

La funzione noisetest.m di Matlab/Octave mostra l'aspetto e l'effetto dei diversi tipi di rumore. Disegna picchi Gaussiani con quattro diversi tipi di rumore aggiunto: il rumore bianco costante, quello rosa (1/f) costante, il rumore bianco proporzionale e quello bianco radice quadrata, quindi approssima una Gaussiana a ciascun set di dati rumorosi e calcola la media e la deviazione standard dell'altezza, della posizione, della larghezza e dell'area del picco per ciascun tipo di rumore. Digitare "help noisetest" al prompt dei comandi. Lo script Matlab/Octave SubtractTwoMeasurements.m (pag. 23) mostra la tecnica di sottrazione di due diverse misure di una forma d'onda per estrarre il rumore casuale (ma funziona solo se il segnale è stabile, a parte il rumore).

iSignal (pagina 366) è un gruppo di moduli Matlab multi-uso scaricabili sviluppati per combinare molte delle tecniche descritte qui; iSignal può disegnare segnali controllando pan e zoom, misurare le ampiezze del segnale e del rumore nelle regioni selezionate del segnale e calcolare il rapporto S/N dei picchi. È gestito semplicemente da tastiera. Altre funzionalità di iSignal includono lo smoothing (pag. 38), la differenziazione, lo sharpening e il troncamento [de-tailing] dei picchi, la deconvoluzione, la misura dei picchi ai quadrati minimi, ecc.

Altre funzioni interattive in questo gruppo comprendono *iPeak*, pagina 246, orientato al rilevamento dei picchi, e *ipf.m*, pagina 407, specializzato nell'approssimazione iterativa della curva. Queste funzioni sono ideali per le esplorazioni iniziali dei segnali complessi perché facilitano la selezione delle operazioni e la regolazione dei controlli con semplici pressioni di tasti. Funzionano anche se si

esegue [Matlab Online in un browser web](#), ma non funzionano su [Matlab Mobile](#). Notare che le versioni per Octave, [ipfoctave.m](#), [ipeakoctave.m](#), [isignaloctave.m](#) e [ifilteroctave.m](#), usano i tasti < e > (con e senza shift) per il pan e lo zoom.

Per i segnali che contengono schemi di forme d'onda ripetitivi in un segnale continuo, con nominalmente lo stesso profilo ad eccezione del rumore, la funzione interattiva di rilevamento dei picchi *iPeak* (pag. 245), ha una funzione di media di insieme [ensemble averaging] (**Shift-E**) che può calcolare la media di tutte le forme d'onda ripetute. Funziona rilevando un singolo picco di riferimento in ciascuna forma d'onda per sincronizzare le ripetizioni (e quindi non richiede che le ripetizioni siano equamente distanziate o sincronizzate con un segnale di riferimento esterno). Per utilizzare questa funzione, prima si regolano i controlli di rilevamento del picco per rilevare *solo un picco in ciascuno schema ripetuto*, uno zoom per isolare uno qualsiasi di questi pattern, e si preme **Shift-E**. La forma d'onda media viene mostrata in Figura 2 e salvata come "EnsembleAverage.mat" nella directory corrente. Cfr. [iPeakEnsembleAverageDemo.m](#) per una dimostrazione. Vedere pagina 322: *Misurare il Rapporto Segnale/Rumore di Segnali Complessi* per altri esempi di rapporto segnale/rumore nei calcoli con Matlab/Octave.

Il ruolo della simulazione e della modellazione.

Una simulazione è un'imitazione del funzionamento di un processo o di un sistema reale nel tempo. Le simulazioni richiedono l'uso di *modelli*, che rappresentano le caratteristiche o i comportamenti rilevanti del sistema o del processo in esame, mentre la *simulazione* rappresenta l'evoluzione del modello nel tempo. L'[articolo su Wikipedia sulla simulazione](#) elenca 27 aree molto diverse in cui vengono applicate la simulazione e la modellazione. Nell'ambito della misura scientifica, sono state ampiamente impiegate simulazioni di strumenti di misura (pagina 347) o di tecniche di elaborazione del segnale. Un segnale simulato può essere sintetizzato con modelli matematici per forme di segnale (pagina 451) in combinazione con appropriati tipi di rumore casuale altrettanto simulati (pagina 22), entrambi basati sulle comuni caratteristiche dei segnali reali. Ma è importante rendersi conto che un segnale simulato non è un "falso" segnale, in quanto *non è destinato a ingannare*. Piuttosto, i segnali simulati si possono usare per testare l'accuratezza e la precisione di una tecnica di elaborazione proposta, usando dati simulati i cui veri parametri in esame sono noti (cosa che riguarda i segnali reali). Inoltre, è possibile testare la robustezza e la riproducibilità di una tecnica proposta creando più segnali con gli stessi parametri del segnale in esame ma con l'aggiunta di imperfezioni, come rumore casuale, linee di base diverse da zero o spostate, picchi di interferenza, distorsione del profilo, ecc. Ad esempio, lo script [CreateSimulatedSignal.m](#) mostra come creare un modello realistico di un segnale multi-picco basato sulle caratteristiche misurate di un segnale sperimentale. Vedremo molte applicazioni di questa idea, ad esempio nelle pagine 302 e 329. E la simulazione è applicabile anche nei casi più sofisticati. A pagina 354, si descrive una relazione tecnica commerciale pubblicata contenente un esempio dettagliato di un'applicazione pratica della cromatografia di liquidi con array di diodi rilevatori per separare tre isomeri chimici simili. Con queste informazioni si è stati in grado di creare simulazioni realistiche "data-based" dei dati ottenuti nell'esperimento, che hanno permesso di "ripetere" l'esperimento numericamente, in diverse condizioni sperimentali, per esplorare i limiti di applicabilità di quel metodo ad altre applicazioni potenzialmente più impegnative.

Smoothing

In molti esperimenti scientifici, le vere ampiezze del segnale (valori sull'asse y) cambiano piuttosto uniformemente in funzione dei valori sull'asse x, mentre molti tipi di rumore appaiono come bruschi

e casuali cambiamenti dell'ampiezza da un punto all'altro nel segnale. In quest'ultima situazione può essere utile, in alcuni casi, tentare di ridurre il rumore mediante un processo detto *smoothing*. Nello smoothing, i punti di un segnale vengono modificati in modo che i singoli punti che risultano *più alti* di quelli immediatamente adiacenti (presumibilmente a causa del rumore) vengono ridotti, e quelli che risultano *più bassi* di quelli adiacenti vengono aumentati. Questo porta naturalmente a un segnale più fluido (e una risposta più lenta ai gradini del segnale). Se il vero segnale è effettivamente regolare, questo non verrà molto distorto dallo smoothing, ma il rumore ad alta frequenza verrà ridotto. In termini di componenti di frequenza del segnale, un'operazione di smoothing agisce come un filtro passa-basso, riducendo le componenti alle alte frequenze e facendo passare quasi indenni quelle a bassa frequenza. Se il segnale e il rumore vengono misurati per tutte le frequenze, il rapporto segnale-rumore verrà migliorato dallo smoothing, di una quantità che dipende dalla distribuzione in frequenza del rumore. (Lo smoothing può essere contrastato dal *wavelet denoising*, pagine 126 e 57, che riduce anch'esso il rumore ma non necessariamente rende il segnale completamente 'liscio').

Algoritmi di smoothing

I più semplici algoritmi di smoothing si basano sulla tecnica "*trasla e moltiplica*" in cui un gruppo di punti adiacenti nei dati originali vengono moltiplicati punto-punto con un insieme di numeri (coefficienti) che definiscono il profilo di addolcimento, i prodotti vengono addizionati e divisi per la somma dei coefficienti, diventando un punto 'smussato', poi l'insieme dei coefficienti viene spostato di un punto lungo i dati originali e si ripete il processo. Il più semplice algoritmo di smoothing è quello *rettangolare* o *slittamento non pesato della media*; esso semplicemente sostituisce ogni punto del segnale con la media degli m punti adiacenti, dove m è un intero positivo detto *ampiezza dello smoothing*. Per esempio, per un smoothing di 3 punti ($m = 3$):

$$S_j = \frac{Y_{j-1} + Y_j + Y_{j+1}}{3}$$

Questo viene calcolato da $j = 2$ a $n-1$, dove S_j è il j^{o} punto nel segnale filtrato, Y_j è il j^{o} punto nel segnale originale e n è il numero totale dei punti nel segnale. La maggior parte dei fogli di calcolo e dei linguaggi di programmazione hanno una funzione per il calcolo della media "mean" o "average" che può eseguire questo lavoro velocemente, quindi $S_j = \text{mean}(y_{j-w/2}:y_{j+w/2})$. Si possono costruire delle operazioni di smoothing simili per qualsiasi ampiezza dello smoothing, m . Solitamente m è un numero dispari. Se il rumore nei dati è un "rumore bianco" (cioè distribuito uniformemente su tutte le frequenze) e la sua deviazione standard è D , allora la deviazione standard del rumore restante nel segnale dopo il primo passaggio di slittamento-della-media non pesata dello smoothing sarà approssimativamente D sulla radice quadrata di m (D/\sqrt{m}), dove m è l'ampiezza dello smoothing. Nonostante la sua semplicità, questo smoothing è in realtà *ottimale* per il problema comune di riduzione del rumore bianco mantenendo *ripida la risposta al gradino* ([cliccare qui per una dimostrazione logica](#)).. La risposta ad un gradino è, infatti, *lineare*, quindi questo filtro ha il vantaggio di rispondere completamente senza alcun effetto residuo entro il suo *tempo di risposta* (che è uguale alla larghezza dello smoothing diviso la frequenza di campionamento). Lo smoothing si può eseguire sia *durante* l'acquisizione dei dati, programmando il digitalizzatore per misurare e mediare più letture e salvarne solo la media, sia *dopo* l'acquisizione ("post-run"), memorizzando tutti i dati acquisiti per poi livellarli. Quest'ultimo richiede più memoria ma è più flessibile.

Lo *smoothing triangolare* è simile a quello rettangolare, visto sopra, eccetto che è implementato con una funzione *pesata* di smoothing. Per uno smoothing di 5 punti ($m = 5$):

$$S_j = \frac{Y_{j-2} + 2Y_{j-1} + 3Y_j + 2Y_{j+1} + Y_{j+2}}{9}$$

per $j = 3$ fino a $n-2$, e allo stesso modo per altre ampiezze di smoothing (si veda lo spreadsheet [UnitGainSmooths.xls](#)). In entrambi i casi, l'intero al denominatore è la *somma dei coefficienti* al numeratore, che si traduce in un “guadagno unitario” dello smoothing che non ha effetto sul segnale quando è un linea retta e che preserva l'area dei picchi.

Spesso è utile eseguire un'operazione di smoothing più di una volta, ovvero, ammorbidire un segnale già fluido, al fine di costruire degli smoothing più lunghi e complicati. Ad esempio, lo smoothing triangolare a 5 punti precedente è equivalente a due passaggi dello smoothing rettangolare a 3 punti. Tre passaggi dello smoothing rettangolare a 3 punti danno come risultato uno smoothing *a pagliaio* da 7 punti, detto anche “p-spline” [“B-spline penalizzata”], per cui i coefficienti sono in rapporto 1:3:6:7:6:3:1. La regola generale è che n passaggi con un'ampiezza w di smoothing hanno come risultato un'ampiezza di smoothing di n^*w-n+1 . Per esempio, 3 passaggi di uno smoothing di 17-punti risultano pari a uno smoothing di 49-punti. Questi 'smussamenti' multipli sono più efficaci nel ridurre il rumore ad alta frequenza nel segnale rispetto ad uno smoothing rettangolare, ma presentano una risposta al gradino più lenta.

In tutti questi smoothing, l'ampiezza m viene solitamente scelta con un valore intero dispari, in modo che i coefficienti siano simmetricamente bilanciati attorno al punto centrale, cosa importante perché *preserva la posizione sull'asse x dei picchi e delle altre caratteristiche*. (Ciò è particolarmente importante in certe applicazioni analitiche e spettroscopiche perché le posizioni dei picchi sono spesso parti importanti delle misure).

Si assume che l'intervallo sull'asse x del segnale sia uniforme, cioè che la differenza tra i valori sull'asse x ed i punti adiacenti sia la stessa per tutto in segnale. Questo è l'assunto in molte delle tecniche di signal processing descritte in questo libro, ed è una caratteristica molto comune (ma non necessaria) dei segnali acquisiti da apparecchiature automatiche e computerizzate.

Algoritmi più avanzati. Lo smoothing di *Savitzky-Golay* (ref 97) è basato sull'approssimazione ai quadrati minimi di polinomi a porzioni di dati. L'algoritmo è descritto su [Wikipedia](#). Rispetto agli smoothing a slittamento della media della stessa ampiezza, il Savitzky-Golay è meno efficace nel ridurre il rumore, ma più efficace nel mantenere il profilo del segnale originale. È in grado di differenziare contemporaneamente allo smoothing. L'algoritmo è più complesso e i tempi di calcolo possono essere maggiori rispetto ai tipi di smoothing discussi, ma con i computer moderni, la differenza raramente significativa. Il codice in vari linguaggi è ampiamente disponibile online.

Vedere pag. 56. La funzione interattiva [iSignal function](#) (pagina 366) ha un'opzione Savitzky-Golay. La funzione *denoise* basata su wavelet (pag. 130) è un algoritmo più sofisticato che tenta di distinguere il segnale dal rumore analizzando la struttura della frequenza del segnale.

Il profilo di qualsiasi algoritmo di smoothing si può determinare applicando quella operazione a una *funzione delta*, un segnale composto da tutti zeri tranne un punto, come mostrato dal semplice script Matlab/Octave [DeltaTest.m](#). Il risultato è la *funzione di risposta all'impulso*.

Riduzione del rumore

Lo smoothing solitamente riduce il rumore in un segnale. Se il rumore è "bianco" (cioè distribuito uniformemente su tutte le frequenze) e la sua deviazione standard è D , allora la deviazione standard del rumore residuo nel segnale dopo un passaggio dello smoothing rettangolare sarà approssimativamente D/\sqrt{m} , dove m è l'ampiezza dello smoothing. Se, invece si usa quello triangolare, il rumore sarà leggermente inferiore, circa $D*0.8/\sqrt{m}$. Le operazioni di smoothing si possono applicare più di una volta: cioè, un segnale già filtrato lo si può ulteriormente filtrare. In qualche caso, ciò può risultare utile se si ha a che fare con una grande quantità di rumore ad *alta*

frequenza nel segnale. Tuttavia, la riduzione del rumore *bianco* è inferiore ad ogni passaggio successivo. Per esempio, *tre* passaggi di uno smoothing rettangolare riduce il rumore bianco di un fattore approssimativamente di $D^*0.7/\sqrt{m}$, solo un leggero miglioramento rispetto a due passaggi. Per uno spreadsheet dimostrativo, si veda [VariableSmoothNoiseReduction.xlsx](#).

Effetto della distribuzione in frequenza del rumore

La distribuzione in frequenza del rumore, designata dal “colore” del rumore (pagina 22), influisce in modo sostanziale sulla capacità di attenuazione del rumore. La funzione Matlab/ Octave [“NoiseColorTest.m”](#) confronta gli effetti di uno smoothing "boxcar" da 20 punti (slittamento della media non pesato) sulla deviazione standard del rumore bianco, rosa, rosso e blu, tutti con una deviazione standard originale non filtrata di 1.0. Dato che lo smoothing è un filtro passa basso, influisce *meno* sul rumore a bassa frequenza (rosa e rosso), e *maggiormente* sul rumore ad alta frequenza (blu e viola), rispetto al rumore bianco.

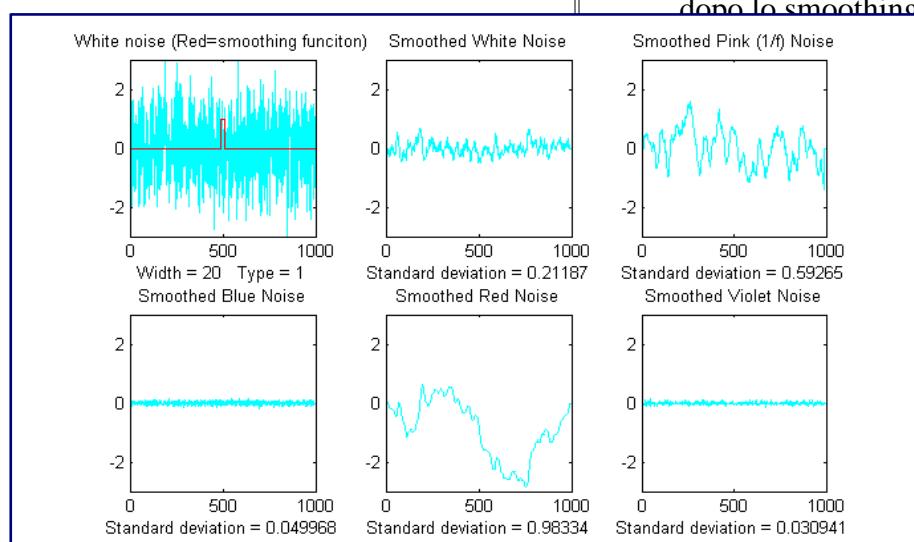
Si noti che il calcolo della deviazione standard è indipendente dall'ordine dei dati e quindi dalla sua distribuzione in frequenza; l'ordinamento di un insieme di dati non cambia la sua deviazione standard. La deviazione standard di un'onda sinusoidale è indipendente dalla sua frequenza. Lo smoothing, tuttavia, modifica sia la distribuzione della frequenza che la deviazione standard di un set di dati.

Effetti agli estremi e il problema dei punti persi

Nelle equazioni precedenti, lo smoothing rettangolare di 3 punti è definito solo da $j = 2$ a $n - 1$. Non ci sono dati sufficienti nel segnale per definire un completo smoothing a 3 punti per il primo punto del segnale ($j = 1$) né per gli ultimi ($j = n$), perché non ci sono punti prima del primo punto né dopo l'ultimo. (Allo stesso

Rumore originale non filtrato	1
Rumore bianco dopo lo smoothing	0.1
Rumore rosa dopo lo smoothing	0.55
Rumore blu dopo lo smoothing	0.01
Rumore rosso (random walk) dopo lo smoothing	0.98

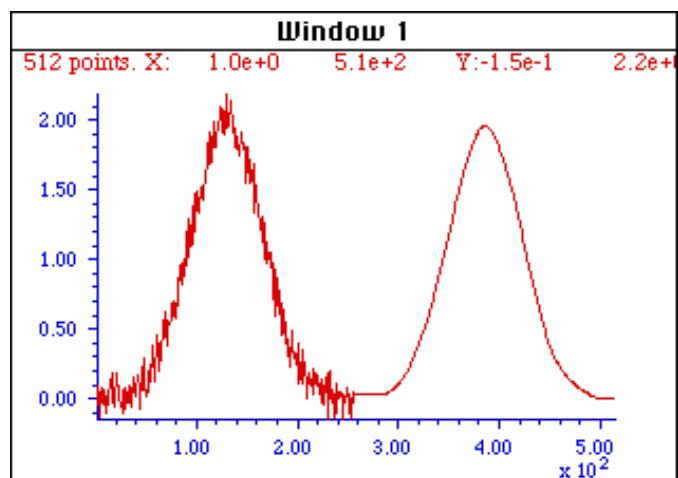
n-1.
punti
mod
o uno



smoothing a 5 punti è definito solo da $j = 3$ fino a $n-2$, e quindi non è possibile calcolare uno smoothing dei primi due punti né per gli ultimi due). In generale, per uno smoothing di ampiezza m , ci saranno $(m-1)/2$ punti all'inizio del segnale e $(m-1)/2$ punti alla fine del segnale per cui non è possibile calcolare uno smoothing largo m come al solito. Cosa fare? Ci sono due approcci. Una è quella di accettare la perdita dei punti e tagliar via tali punti o sostituirli con degli zeri nel segnale

filtrato. (Questo è l'approccio adottato nella maggior parte delle figure in questo documento). L'altro approccio consiste nell'usare *smoothing progressivamente più piccoli* alle estremità del segnale, per esempio usare larghezze di 2, 3, 5, 7... punti per i punti 1, 2, 3 e 4..., e i per punti n, n-1, n-2, n-3... del segnale, rispettivamente. Quest'ultimo è preferibile se le estremità del segnale contengono informazioni importanti, ma aumenta il tempo di esecuzione. La funzione Matlab/Octave *fastsmooth* (pag. 457) può usare entrambi i metodi. Un approccio alternativo consiste nel riempire gli estremi con un'[immagine speculare dei dati stessi](#), operazione comunemente eseguita per eseguire lo smoothing di dati bidimensionali (immagine).

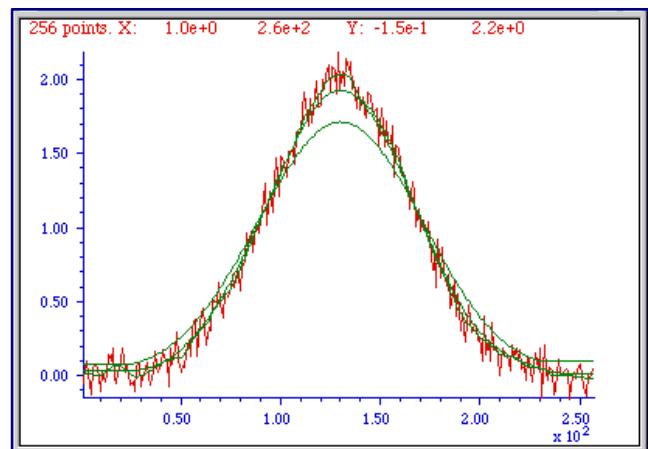
Esempi di smoothing



La figura seguente mostra un semplice esempio di smoothing. La metà sinistra di questo segnale è un picco rumoroso. La metà destra è lo stesso picco dopo l'applicazione di un filtraggio triangolare. Il rumore è notevolmente ridotto mentre il picco è praticamente invariato. La riduzione del rumore consente di misurare con maggiore precisione le caratteristiche del segnale (posizione, altezza, larghezza, area, ecc. del picco) mediante un'ispezione visiva.

La metà sinistra di questo segnale è un picco rumoroso. La metà destra è lo stesso picco dopo l'applicazione di un algoritmo di smoothing. Il rumore è notevolmente ridotto mentre il picco non viene quasi modificato, rendendo più facile misurarne direttamente la posizione, l'altezza e la larghezza mediante una stima grafica o visiva (ma non migliora le misure dei parametri del picco fatte con l'approssimazione dei minimi quadrati; si veda in seguito).

Maggiore è l'ampiezza dello smoothing, maggiore è la riduzione del rumore, ma anche maggiore è la possibilità che il segnale venga *distorto* da tale operazione. La scelta ottimale per l'ampiezza dello smoothing dipende sia dalla larghezza e dalla forma del segnale che dall'intervallo di digitalizzazione. Per i segnali a forma di picco, il fattore critico è il *rapporto di smoothing*, il

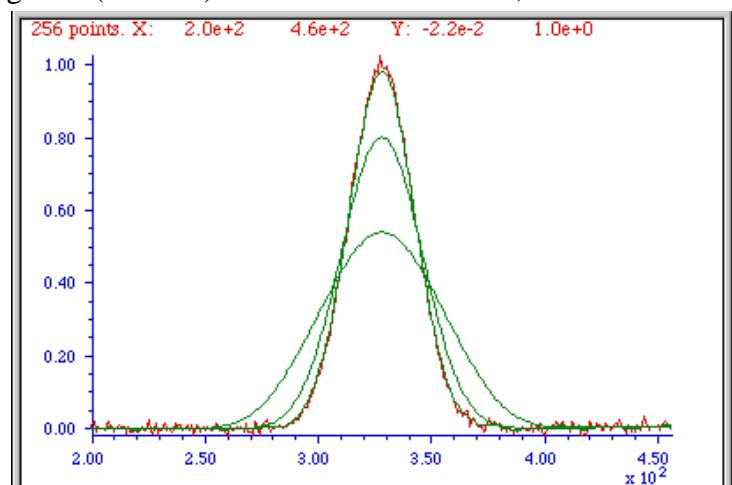


rapporto tra l'ampiezza dello smoothing m ed il numero di punti a metà larghezza del picco. In generale, l'aumento del rapporto di smoothing migliora il rapporto segnale-rumore ma provoca una riduzione nell'ampiezza e un incremento della larghezza del picco. Si tenga presente che l'ampiezza dello smoothing si può esprimere in due modi: (a) come il numero di punti o (b) come l'intervallo sull'asse x (per i dati spettroscopici di solito in nm o in unità di frequenza). I due sono semplicemente correlati:

il numero di punti è semplicemente l'intervallo sull'asse x moltiplicato per l'incremento tra i valori adiacenti sull'asse x. Il *rapporto di smoothing* è lo stesso per entrambi.

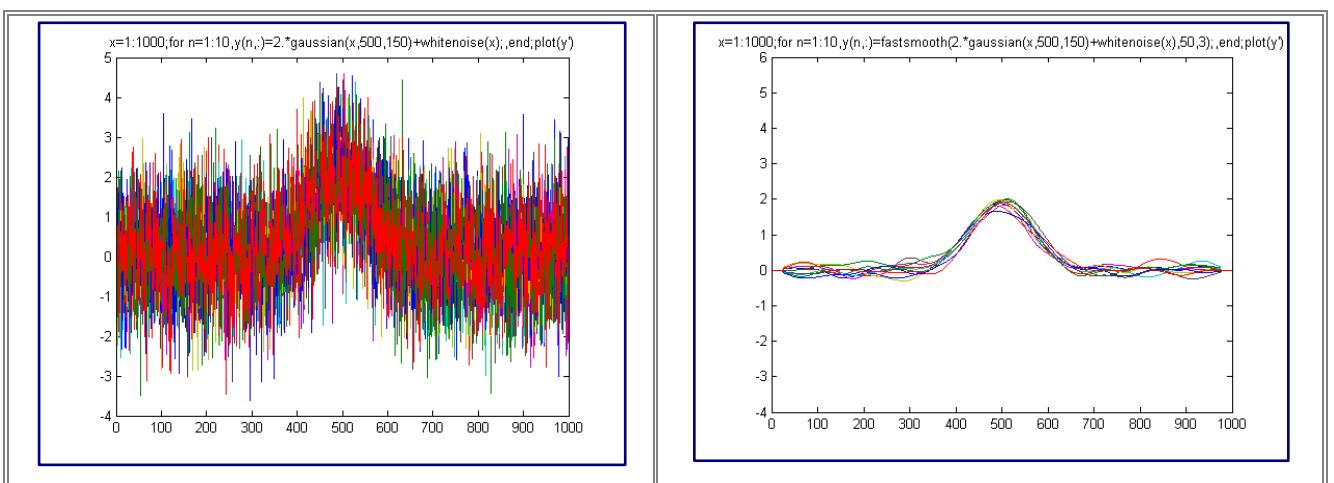
Le figure qui mostrano degli esempi dell'effetto di tre diverse larghezze di smoothing su picchi Gaussiani rumorosi. In una figura, il picco ha un'altezza reale di 2.0 e ci sono 80 punti a metà larghezza del picco. La linea rossa è il picco originale non filtrato. Le tre linee verdi sovrapposte sono i risultati dello smoothing di questo picco con un filtro triangolare di ampiezza (dall'alto in basso) 7, 25, e 51 punti. Dato che la larghezza del picco è di 80 punti, i *rapporti di smoothing* di

questi tre operazioni sono $7/80=0.09$, $25/80= 0.31$ e $51/80=0.64$, rispettivamente. All'aumentare dell'ampiezza dello smoothing, il rumore viene progressivamente ridotto ma viene leggermente ridotta anche l'altezza del picco. Per lo smoothing più largo, la *larghezza* del picco è aumentata notevolmente. Nella figura a destra, il picco originale (in rosso) ha un'altezza reale di 1,0 e una semi-larghezza di 33 punti. (È anche meno rumoroso dell'esempio precedente). Le tre linee verdi sovrapposte sono il risultato degli stessi tre filtri triangolari di ampiezza 7, 25, e 51 punti. Ma dato che la larghezza del picco, in questo caso, è di soli 33 punti, i rapporti di smoothing di questi tre filtri sono maggiori- 0.21, 0.76, e 1.55, rispettivamente. Si può osservare che l'effetto della distorsione del picco (riduzione dell'altezza del picco e aumento della sua larghezza) è maggiore per il picco più stretto perché i rapporti di smoothing sono più alti. I rapporti di smoothing maggiori di 1.0 vengono utilizzati raramente a causa dell'eccessiva distorsione del picco. Da notare che anche nel caso peggiore, le posizioni dei picchi non vengono influenzate (assumendo che i picchi originali fossero simmetrici e non sovrapposti da altri picchi). Se mantenere la forma del picco è più importante dell'ottimizzazione del rapporto segnale-rumore, lo smoothing di Savitzky-Golay risulta migliore di quelli a slittamento della media [sliding-average]. In tutti i casi, il segnale totale resta invariato. Se le larghezze dei picchi variano sostanzialmente, è possibile utilizzare uno smoothing adattivo o segmentato, che consente di variare la larghezza dello smoothing lungo il segnale, (pag, 326). In questo contesto, “segmentato” significa che il segnale è suddiviso in segmenti con un diverso grado di smoothing applicato a ciascun segmento.



Il problema con lo smoothing

Lo smoothing è spesso meno vantaggioso di quanto si pensi. È importante capire che i risultati dello smoothing, come mostrato nelle figure precedenti, potrebbero essere visti come *ingannevoli* perché impiegano un *singolo campione* di un segnale rumoroso che subisce uno smoothing a diversi livelli. Questo fa sì che l'osservatore sottovaluti il contributo del rumore a *bassa frequenza*, che è difficile da stimare visivamente perché ci sono *pochissimi cicli a bassa frequenza* nella registrazione del segnale. Questo problema può essere visualizzato registrando qualche campione indipendente di un segnale rumoroso costituito da un unico picco, come illustrato nelle successive due figure.



Queste figure mostrano dieci grafici sovrapposti con lo stesso picco ma

```
x=1:1000;
for n=1:10,
y(n,:)=2.*gaussian(x,500,150)...
+whitenoise(x);
end
plot(x,y)
```

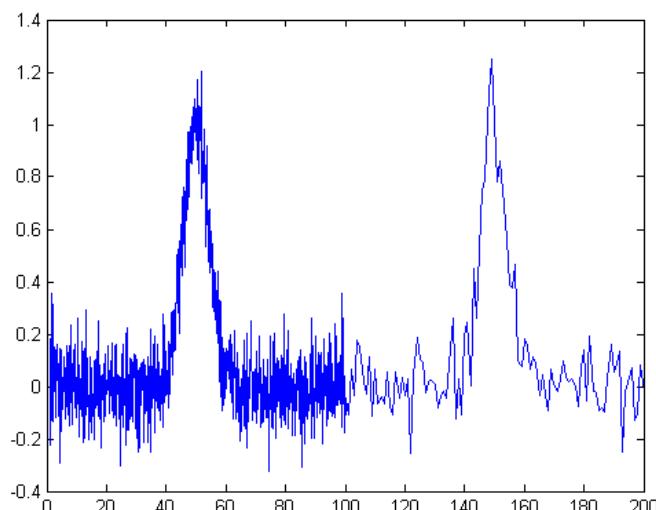
```
x=1:1000;
for n=1:10,
y(n,:)=2.*gaussian(x,500,150)
+whitenoise(x);
y(n,:)=fastsmooth(y(n,:),50,3);
end
plot(x,y)
```

con rumore bianco indipendente, ciascun grafico è disegnato con un colore diverso, senza smoothing a sinistra e con smoothing a destra. Chiaramente, la riduzione del rumore è sostanziale, ma un'attenta ispezione dei diversi segnali con lo smoothing colorati a destra mostra che resta una variazione della posizione, dell'altezza e della larghezza del picco tra i 10 campioni, provocata dal rumore residuo a bassa frequenza. Senza il rumore, ogni picco avrebbe un'altezza di 2, il centro a 500 e una larghezza di 150. *Solo perché un segnale appare omogeneo non significa che non vi sia rumore.* Il rumore a bassa frequenza rimanente nei segnali dopo l'attenuazione può ancora interferire con la misura precisa della posizione, dell'altezza e della larghezza del picco.

(Gli script di generazione sotto ogni figura richiedono il download delle funzioni gaussian.m, whitenoise.m e fastsmooth.m dal link <http://tinyurl.com/cey8rwh>.)

Dovrebbe essere chiaro che lo smoothing raramente elimina *completamente* il rumore, perché la maggior parte del rumore è distribuita su una certa gamma di frequenze e lo smoothing riduce semplicemente il rumore nella *parte* della propria gamma di frequenze. Solo per alcuni tipi di rumore molto specifici (p.es. il rumore sinusoidale a frequenza discreta o gli [spike] puntiformi) si può sperare di approssimare la completa rimozione. Lo smoothing *rende* il segnale più uniforme e *riduce* la deviazione standard del rumore, ma se questo *migliora* o meno la misura, dipende dalla situazione. E non si deve dare per scontato che se un piccolo smoothing risulta buono ulteriori filtraggi siano migliori. Lo smoothing è come l'alcol; a volte ce n'è bisogno - ma non si deve mai esagerare.

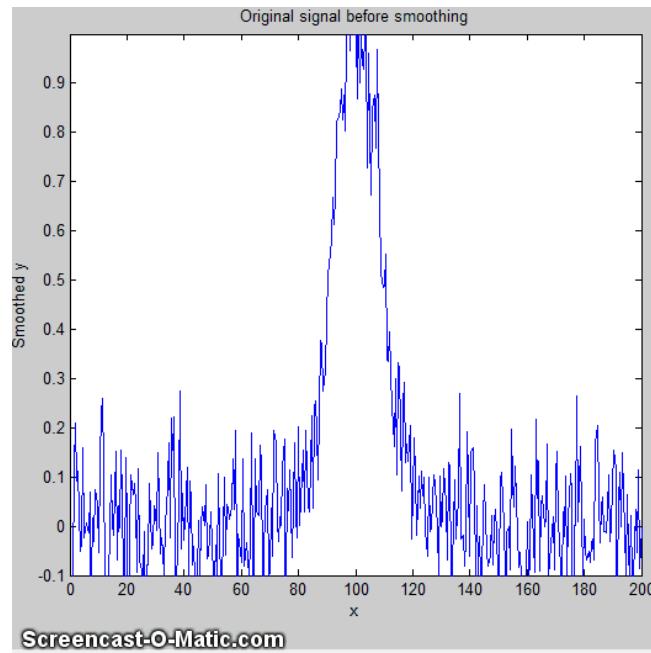
La prossima figura è un altro esempio di segnale che illustra alcuni di questi principi. Il segnale consiste in due picchi Gaussiani, uno posizionato a $x=50$ e l'altro a $x=150$. Entrambi hanno una altezza di 1.0, una semi-larghezza di 10 e con la stessa aggiunta di rumore bianco normalmente distribuito con una deviazione standard di 0.1. L'*intervallo di campionamento sull'asse x*, però, è diverso per i due picchi; è 0.1 per il primo picco da $x=0$ a 100) e 1.0 per il secondo (da $x=100$ a 200). Ciò significa che il primo picco è caratterizzato da *dieci volte più punti* rispetto al secondo picco. Potrebbe



sembrare che il primo picco sia più rumoroso del secondo, ma è solo un'illusione; il rapporto segnale rumore per entrambi è 10. Il secondo picco sembra meno rumoroso solo perché ci sono meno campioni di rumore e *si tende a sottostimare la dispersione dei piccoli campioni*. Il risultato di ciò è che quando il segnale viene attenuato, è molto più probabile che il *secondo picco* venga più distorto dallo smoothing (diventa più basso e più largo) del primo. Il primo picco può tollerare una larghezza di smoothing più ampia e ne risulta un maggior grado di riduzione del rumore. Allo stesso modo, se entrambi i picchi vengono misurati col metodo dell'approssimazione ai minimi quadrati,

trattato in seguito, l'approssimazione del primo picco è più stabile col rumore e i parametri misurati saranno circa *3 volte più accurati* di quelli del secondo picco, perché ci sono 10 volte più punti e la precisione della misura migliora approssimativamente con la radice quadrata del numero di campioni se il rumore è bianco. È possibile scaricare questo file di dati, "udx", in [formato TXT](#) o in Matlab [formato MAT](#).

Ottimizzazione dello smoothing



All'aumentare della larghezza dello smoothing, il suo rapporto aumenta, il rumore si riduce rapidamente all'inizio, poi più lentamente, e anche l'altezza del picco si riduce, prima lentamente poi più velocemente. La *riduzione del rumore* dipende dalla larghezza dello smoothing, dal tipo (p.es. rettangolare, triangolare, ecc.) e dal colore del rumore, ma anche la *riduzione dell'altezza del picco* dipende da essa. Il risultato è che il rapporto segnale-rumore (definito come rapporto tra l'altezza del picco della deviazione standard del rumore) aumenta rapidamente all'inizio per poi raggiungere un massimo. Ciò è illustrato dall'animazione a lato, che mostra il risultato dell'attenuazione di un *picco Gaussiano con in*

più del rumore bianco (prodotto da questo [script Matlab/Octave](#)). Il massimo miglioramento del rapporto segnale rumore dipende dal numero di punti nel picco: più campioni ci sono, maggiori sono le larghezze dello smoothing impiegabili e maggiore è la riduzione del rumore. Questa figura mostra anche che la maggior parte della riduzione del rumore è dovuta a componenti ad *alta frequenza* del rumore, mentre molto del rumore a *bassa frequenza* resta nel segnale anche dopo lo smoothing.

Qual è il miglior rapporto di smoothing? Dipende dallo scopo della misura del picco. Se l'obiettivo finale è quello di misurare l'altezza e la larghezza del picco, allora si devono usare rapporti inferiori a 0.2 ed è preferibile lo smoothing di *Savitzky-Golay* (o il denoise wavelet: cfr. pagina 130). Ma se l'obiettivo è quello di misurare la posizione del picco (valore sull'asse x del picco), si possono usare rapporti maggiori, perché lo smoothing ha poco effetto sulla posizione (a meno che il picco non sia asimmetrico o che l'allargamento del picco non sia così elevato da provocare la sovrapposizione con i picchi adiacenti). Se il picco è in realtà formato da due picchi sovrapposti così da sembrare un unico picco, allora l'unica strada per misurare i parametri è il l'approssimazione o "curve fitting". Sfortunatamente il rapporto segnale-rumore ottimale corrisponde ad un rapporto di smoothing che distorce significativamente il picco, motivo per cui l'approssimazione dei dati originali è il metodo da preferire per la misura della posizione, dell'altezza e della larghezza dei picchi. L'*area* non viene modificata da un'operazione di smoothing ben fatta a meno che non modifichi l'inizio o la fine del picco.

Nelle applicazioni di *analisi chimiche quantitative* basate sulla calibrazione con campioni standard, la riduzione del picco provocata dallo smoothing non è così importante. Se le *stesse* operazioni di signal processing vengono applicate ai campioni e agli standard, la riduzione dell'altezza del picco sarà *la stessa* di quel segnale campione e l'effetto sarà perfettamente *cancellato*. In questi casi, è possibile utilizzare larghezze di smoothing da 0.5 a 1.0 se necessario, per migliorare ulteriormente il

rapporto segnale rumore, come mostrato nella figura precedente (per un semplice smoothing rettangolare a slittamento della media). Nella chimica analitica pratica, raramente è richiesta la misura delle altezze assolute dei picchi; di norma si usa la calibrazione con le soluzioni standard. (Da ricordare: l'obiettivo dell'analisi quantitativa non è la misura di un segnale ma piuttosto la misura di concentrazioni di sostanze ignote). È molto importante, tuttavia, applicare *gli stessi passi* di signal processing ai segnali standard e a quelli da misurare, altrimenti si ottiene un grande errore sistematico.

Per un confronto più dettagliato fra i quattro tipi di smoothing considerati, si veda a pagina 56.

Quando si dovrebbe applicare lo smoothing ad un segnale?

Ci sono quattro motivi per farlo:

- (a) per motivi estetici, per preparare un grafico che appaia migliore e più suggestivo in una presentazione o una pubblicazione, specie soprattutto per enfatizzare comportamenti a *lungo termine* rispetto a quelli a *breve termine*, oppure
- (b) se il segnale contiene principalmente rumore ad *alta frequenza* ("blu"), che può sembrare brutto ma ha meno effetto sulle componenti a bassa frequenza del segnale (p.es. le posizioni, le altezze, le ampiezze e le aree dei picchi) rispetto al rumore bianco, oppure
- (c) se il segnale sarà successivamente analizzato con un metodo che risulterebbe eccessivamente degradato dalla presenza di troppo rumore nel segnale, per esempio, se le altezze dei picchi devono essere determinate *visivamente o graficamente* o con la funzione MAX, o se le larghezze dei picchi vengono misurate con la funzione halfwidth, o se la posizione dei massimi, minimi o dei punti di flesso vengono determinati automaticamente rilevando il passaggio per lo zero della derivata del segnale. L'ottimizzazione della quantità e del tipo di smoothing è importante in questi casi (vedere pagina 41). In genere, se è disponibile un computer per effettuare misure quantitative, è meglio utilizzare metodi ai quadrati minimi sui dati *originali*, anziché fare stime grafiche sui dati filtrati con lo smoothing. Se uno strumento commerciale ha la possibilità di effettuare lo smoothing dei dati, è meglio disabilitare quest'opzione e salvare i dati *originali*; lo si potrà sempre fare in seguito per una presentazione grafica e sarà meglio utilizzare i dati originali per una approssimazione ai quadrati minimi o altri tipi di elaborazioni. Lo smoothing si può usare per *localizzare i picchi*, ma non lo si dovrebbe usare per *misurarli*.
- (d) Il limite formale di rilevamento e il limite di quantificazione di un metodo analitico ([riferimenti 91, 92](#)) possono essere migliorati mediante lo smoothing o la media, a seconda del metodo di misura del segnale, come descritto a pagina 25 e dimostrato dallo script Matlab/Octave [SNRdemo.m](#).

È necessario prestare attenzione alla progettazione di algoritmi che impiegano lo smoothing. Ad esempio, in una popolare tecnica per la ricerca e misura dei picchi discussa in seguito (pag. 227), i picchi vengono individuati rilevando i passaggi per lo zero verso il basso nella *derivata prima* con smoothing, ma posizione, altezza e larghezza di ciascun picco vengono determinate con l'approssimazione dei minimi quadrati (pag. 165) di un segmento dei dati originali *senza smoothing* in prossimità del passaggio per lo zero (pag. 230), anziché prendere semplicemente il massimo dei dati con smoothing. In questo modo, anche se è necessario un pesante smoothing per avere una discriminante affidabile contro i picchi del rumore, i parametri dei picchi estratti dall'approssimazione [curve fitting] non vengono distorti dallo smoothing.

Quando NON si dovrebbe applicare lo smoothing ad un

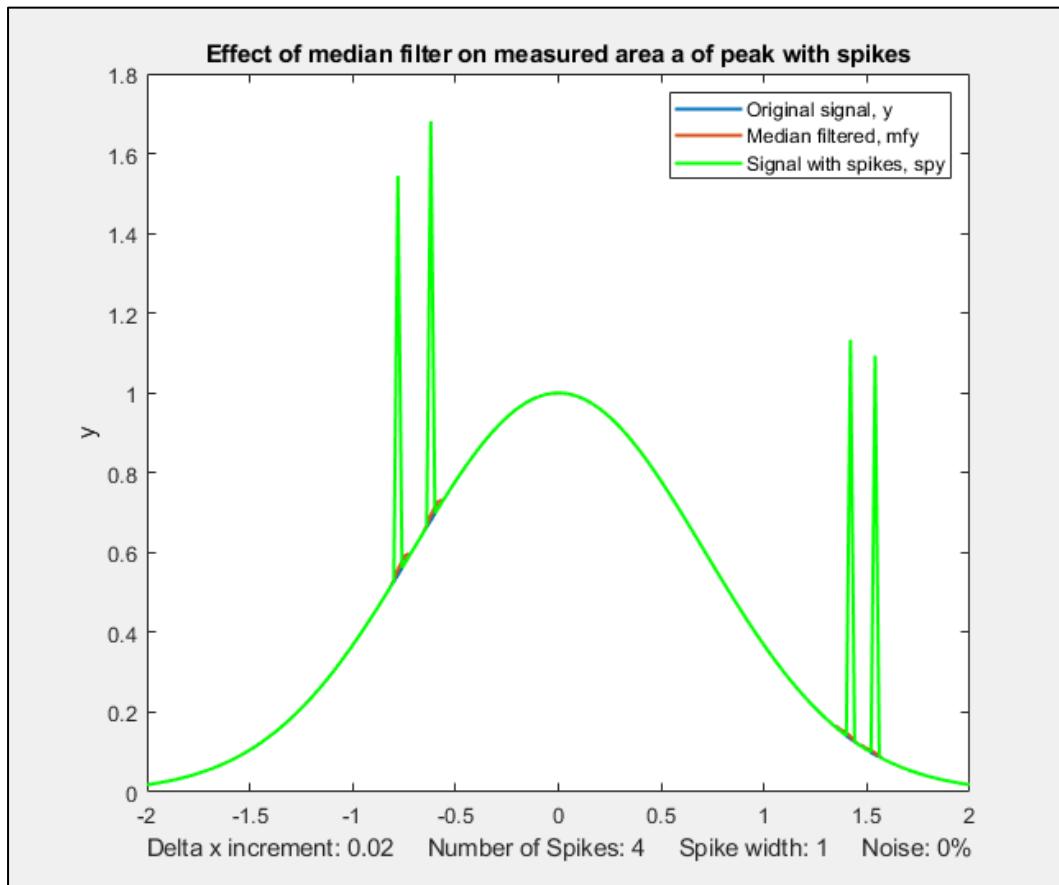
segnale?

Una situazione comune dove solitamente <http://wmbiggs.com/blog/?p=195> non si dovrebbe effettuare lo smoothing, è prima delle procedure di statistica come l'approssimazione ai quadrati minimi. Ci sono diversi motivi (rif. 43).

- (a) Lo smoothing non migliorerà molto la precisione dei parametri misurando con i quadrati minimi tra diversi campioni indipendenti del segnale.
- (b) Tutti gli algoritmi di smoothing sono almeno un po' "lossy", cioè provocano qualche modifica nella forma e nell'ampiezza del segnale.
- (c) È più difficile valutare l'approssimazione ispezionando i residui se i dati subiscono uno smoothing, perché *il rumore potrebbe essere scambiato per l'effettivo segnale*.
- (d) Lo smoothing del segnale sottostimerà seriamente gli errori delle variabili previste dai calcoli algebrici della propagazione dell'errore e dal metodo *bootstrap* (pagina 162). Anche una stima visiva della qualità del segnale viene compromessa dallo smoothing, che fa sembrare il segnale *migliore di quanto non sia in realtà*.

Gestire gli spike e i valori anomali.

A volte i segnali sono contaminati da "spike" strettissimi e molto alti o "valori anomali" che capitano ad intervalli e con ampiezze casuali, ma con larghezze di solo uno o pochi punti. Ad esempio, la spettroscopia ottica che utilizza [rivelatori a tubo fotomoltiplicatore](#) è soggetta a spike provocati dai "raggi cosmici" provenienti dallo spazio esterno che passano attraverso la finestra anteriore del rilevatore, creando un impulso di [radiazione Čerenkov](#). Non solo appaiono brutti, ma sconvolgono anche le ipotesi dei calcoli ai quadrati minimi perché non costituiscono un rumore casuale *distribuito normalmente*. Questo tipo di interferenza è difficile da eliminare con i metodi di smoothing precedenti senza distorcere il segnale. Tuttavia, un filtro "mediano", che sostituisce ogni punto nel segnale con il [mediano](#) (anziché la *media*) di m punti adiacenti, può eliminare gli spike stretti, modificando pochissimo il segnale, se la larghezza degli spike è solo uno o pochi punti e uguale o inferiore a m . Vedere http://en.wikipedia.org/wiki/Median_filter. Lo script "TestSpikefilters.m" mostra il filtro mediano in azione, mentre rimuove l'effetto degli spike:



PercentAreaErrorBefore =4.5%

PercentAreaErrorMedian =0.16%

PercentAreaErrorInterp =0.004%

Per un altro esempio, vedere pag.287.

Un approccio diverso all'eliminazione dei picchi viene utilizzato dalla funzione [killspikes.m](#); essa individua ed elimina i picchi "rappezzandoli" utilizzando l'interpolazione lineare dai punti del segnale immediatamente prima e dopo lo spike. Vedere pagina 54 per i dettagli.

A differenza degli smoothing convenzionali, queste funzioni si possono applicare con profitto *prima* dell'approssimazione ai quadrati minimi. (Tuttavia, se sono gli *stessi spike* a costituire il segnale in esame e le altre componenti interferiscono con la loro misura, si veda pagina 296).

Media dell'Insieme

Un altro modo per ridurre il rumore nei segnali ripetibili, come l'insieme di dieci segnali non filtrati a pagina 44, consiste semplicemente nel calcolarne la media, detta *media di insieme [ensemble averaging]*, che si può ottenere in questo caso semplicemente col codice Matlab/Octave **plot(x, mean(y))**; il risultato mostra una riduzione del rumore bianco di circa $\sqrt{10}=3.2$. Questo migliora il rapporto segnale rumore abbastanza da vedere che c'è un singolo picco con un profilo Gaussiano, che può quindi essere misurato col "curve fitting" (descritto in una prossima sezione, pagina 190) utilizzando il codice Matlab/Octave **peakfit([x; mean(y)],0,0,1)**, col risultato che mostra un'ottima corrispondenza con la posizione (500), l'altezza (2) e la larghezza (150) del picco Gaussiano creato nella terza riga dello script che lo genera (a pagina 44). Un enorme vantaggio della media dell'insieme è che il *rumore in tutte le frequenze viene ridotto*, non solo quello alle *alte* frequenze come nello smoothing. Questo è un grande vantaggio se il segnale o la linea di base si spostano.

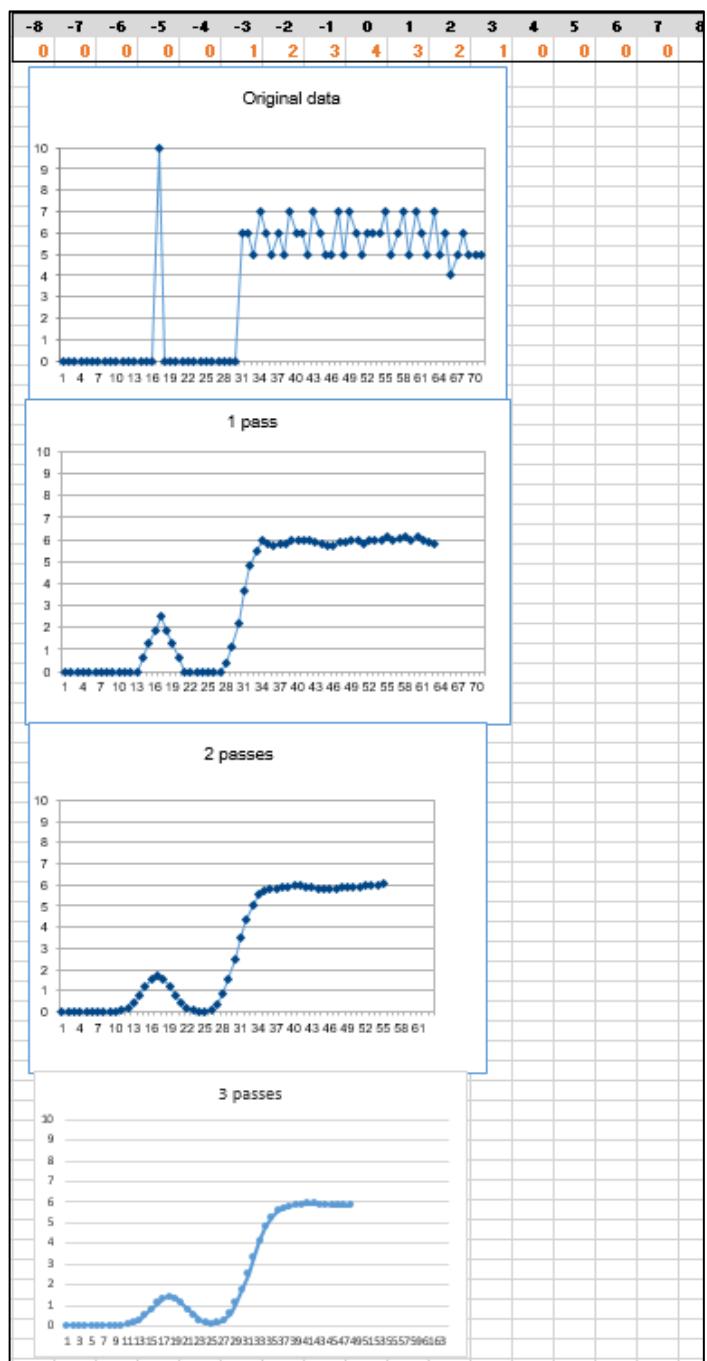
Condensare i segnali sovra-campionati

A volte i segnali vengono registrati più densamente (cioè con una frequenza di campionamento maggiore o con intervalli sull'asse x più piccoli) di quanto sia necessario per catturarne tutte le caratteristiche importanti. Ciò si traduce in dimensioni dei dati maggiori del necessario, che rallentano le procedure di elaborazione del segnale e possono mettere a dura prova la capacità di archiviazione. Per correggere questo problema, le dimensioni dei segnali sovra-campionati possono essere ridotte eliminando dei dati (ad esempio, eliminando un punto ogni due o ogni tre) o sostituendo gruppi di punti adiacenti con le loro *medie*, operazione che è spesso chiamata *raggruppamento*. Il raggruppamento ha il vantaggio di *usare* anziché *scartare* i punti, agisce come uno smoothing fornendo una certa riduzione del rumore. (Se il rumore nel segnale originale è bianco e il segnale viene condensato calcolando la media ogni " n " punti, il rumore viene ridotto nel segnale condensato per la radice quadrata di n , ma senza *alcuna modifica* nella distribuzione in frequenza del rumore residuo). Lo script Matlab/Octave [testcondense.m](#) mostra l'effetto della media con "boxcar" utilizzando la funzione [condense.m](#) per ridurre il rumore senza modificarne il colore. Mostra che il boxcar riduce il rumore misurato, rimuovendo le componenti ad alta frequenza ma non ha alcun effetto sui parametri dei picchi. L'approssimazione ai quadrati minimi dei dati condensati è più veloce e si traduce in un errore approssimazione più basso, ma *nessuna maggiore accuratezza della misura* dei parametri dei picchi. Se ci si ritrova con larghezze di smoothing molto grandi, si prenda in considerazione l'uso della funzione di condensazione.

Dimostrazione video. Questo video di 18 secondi e da tre MByte ([Smooth3.wmv](#)) mostra l'effetto dello smoothing triangolare su un singolo picco Gaussiano con un'altezza di 1.0 e una larghezza di 200. L'ampiezza iniziale del rumore bianco è 0.3, fornendo un rapporto segnale/rumore iniziale di circa 3.3. Un tentativo per misurare l'altezza e la larghezza del picco nel segnale rumoroso, mostrato nella parte inferiore del video, sono inizialmente seriamente imprecisi a causa del rumore. All'aumentare della larghezza dello smoothing, tuttavia, il rapporto segnale-rumore migliora così come la precisione delle misure delle ampiezze e delle larghezze dei picchi. Tuttavia, superando un'ampiezza di smoothing di circa 40 (rapporto di smoothing 0.2), il filtraggio provoca l'abbassamento del picco al di sotto di 1.0 e più largo di 200, *anche se il rapporto segnale-rumore continua a migliorare* all'aumentare dell'ampiezza dello smoothing.

Lo smoothing con gli spreadsheet

Lo smoothing si può fare con i fogli di calcolo utilizzando la tecnica "trasla e moltiplica" descritta in precedenza. Negli spreadsheet dimostrativi [smoothing.ods](#) e [smoothing.xls](#) (schermata) il gruppo dei coefficienti è contenuto nelle formule che calcolano i valori per ciascuna cella dei dati filtrati nelle colonne C ed E. La colonna C esegue uno smoothing *rettangolare* di 7-punti (1 1 1 1 1 1 1).

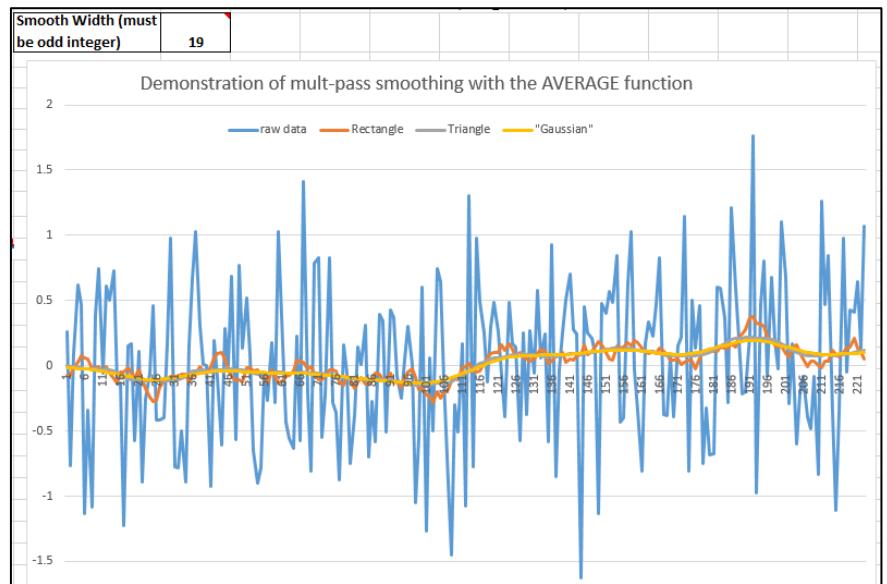


massimo di 17) semplicemente modificando quelle 17 celle. (Per fare degli smoothing più piccoli, basta inserire degli zeri per i coefficienti inutilizzati; in questo esempio, viene definito uno smoothing di 7-punti nelle colonne N - T, i rimanenti coefficienti sono zeri). In questo spreadsheet, viene applicato lo smoothing *tre volte* in sequenza nelle colonne C, E e G, ottenendo un'effettiva larghezza massima di smoothing di $n \cdot w - n + 1 = 49$ punti applicati alla colonna G. Uno svantaggio di questa tecnica per gli spreadsheet è che è complicato espanderli con ampiezze di smoothing molto grandi.

La colonna E esegue uno smoothing *triangolare* a 7-punti (1 2 3 4 3 2 1), applicato ai dati nella colonna A. Si possono inserire (copiare) tutti i dati che si vogliono nella colonna A, e si può estendere lo spreadsheet con colonne più lunghe trascinando l'ultima riga delle colonne A, C ed E verso il basso per quanto serve. Ma per modificare la larghezza dello smoothing, si dovrebbero cambiare le equazioni nelle colonne C ed E copiare le modifiche in basso per l'intera colonna. È pratica comune dividere i risultati per la somma dei coefficienti in modo che il guadagno netto sia unitario e l'area sotto la curva del segnale filtrato con smoothing sia preservata. Gli spreadsheet [UnitGainSmooths.xls](#) e [UnitGainSmooths.ods](#) (schermata) contengono una raccolta di coefficienti di convoluzione a guadagno unitario per gli smoothing rettangolari, triangolari e p-spline con ampiezze da 3 a 29 sia in formato verticale (colonna) che orizzontale (riga) copiare e incollare questi nei fogli di calcolo.

Gli spreadsheet [MultipleSmoothing.xls](#) e [MultipleSmoothing.ods](#) (schermata) mostrano un altro metodo, in cui i coefficienti sono contenuti in un gruppo di 17 celle adiacenti (nella riga 5, colonne da I a Y), facilitando la modifica del *profilo* e della larghezza dello smoothing (fino a un

Una tecnica più flessibile e potente, specialmente per larghezze di smoothing molto ampie, consiste nell'utilizzare la funzione nativa AVERAGE, che di per sé equivale a uno smoothing rettangolare, ma se applicata due o tre volte in sequenza, genera smoothing a forma triangolare e spline. È utilizzato al meglio insieme alla funzione INDIRECT (pagina 345) per controllare un intervallo dinamico di valori. Ciò viene mostrato nello spreadsheet [VariableSmooth.xlsx](#) in cui i dati nella colonna A subiscono uno smoothing di tre applicazioni successive di AVERAGE, nelle colonne B, C e D, ciascuna con una larghezza dello smoothing specificata nella sola cella F3. Se w è la larghezza dello smoothing, che può essere *qualsiasi numero dispari*, lo smoothing risultante nella colonna D ha un'ampiezza totale di $n*w-n+1 = 3*w-2$ punti. La formula della cella delle operazioni di smoothing (=AVERAGE(INDIRECT("A"&ROW(A17)-(\$F\$3-1)/2&":A"&ROW(A17) + (\$F\$3-1)/2))) usa la funzione INDIRECT per applicare la funzione AVERAGE ai dati nelle righe da $w/2$ righe *sopra* a $w/2$ righe *sotto* la riga corrente, dove l'ampiezza dello smoothing w è nella cella F3. Se si copia tale formula nei propri spreadsheet, si devono manualmente *cambiare tutti i riferimenti alla colonna "A"* con la colonna che contiene i dati da filtrare con lo smoothing e cambiare tutti i riferimenti a "\$F\$3" con la posizione dell'ampiezza dello smoothing nello spreadsheet. Infine si trascina in basso, copiando, per coprire tutti i punti dei dati, i riferimenti delle celle si adegueranno automaticamente.



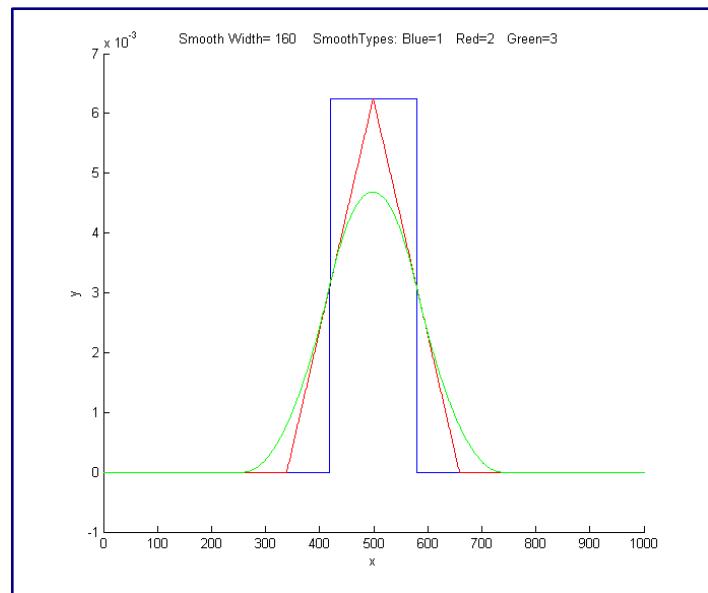
L'esempio nel grafico precedente mostra lo smoothing applicato a segnali DC (corrente continua) con un gradino che si verifica a $x = 111$. Senza lo smoothing (linea blu) il gradino è quasi invisibile. Come esempio di applicazione, il segnale con smoothing potrebbe essere utilizzato per attivare un allarme ogni volta che supera un valore di 0.2, avvertendo che si è verificato qualcosa, mentre il segnale originale non filtrato sarebbe completamente inadatto allo scopo.

Un altro set di fogli di lavoro che utilizza la stessa tecnica AVERAGE(INDIRECT()) è [SegmentedSmoothTemplate.xlsx](#), uno spreadsheet per uno smoothing *segmentato* ad ampiezze multiple può applicare diverse ampiezze di smoothing applicate individualmente alle diverse regioni del segnale. Ciò è particolarmente utile se le larghezze o il livello di rumore dei picchi variano sostanzialmente lungo il segnale. In questa versione, ci sono 20 segmenti. Dei template simili si possono costruire con qualsiasi numero di segmenti.

[SegmentedSmoothExample.xlsx](#) è un esempio con dati ([grafico](#)); si noti che il grafico è opportunamente allineato con le colonne contenenti le ampiezze di smoothing per ciascun segmento. Un foglio correlato, [GradientSmoothTemplate.xlsx](#) o [GradientSmoothExample2.xlsx](#) ([grafico](#)), esegue uno smoothing *gradiente*, linearmente aumentando (o diminuendo) l'ampiezza di smoothing sull'intero segnale, partendo solo dati valori iniziale e finale, e generando automaticamente tanti segmenti e larghezze di smoothing quanti ne sono necessari. (Applica anche il vincolo, nella colonna C, che chiede a ogni larghezza di smoothing di essere un numero dispari, per evitare uno spostamento sull'asse x nei dati filtrati).

Lo smoothing in Matlab e in Octave

La funzione "mean", sia in Matlab che in Python, implementa un singolo smoothing con media scorrevole (pagina 429). La funzione custom di Matlab fastsmooth implementa gli smoothing di tipo trasla-e-moltiplica utilizzando un algoritmo ricorsivo più veloce. È una funzione Matlab della forma `s=fastsmooth(a,w,type,edge)`.



L'argomento "a" è il vettore del segnale di input; "w" è l'ampiezza dello smoothing (un intero positivo); "type" determina il tipo di smoothing: type=1 è uno smoothing rettangolare ('media scorrevole' o 'boxcar'); type=2 è uno smoothing triangolare, equivalente a due passaggi della media scorrevole; type=3 è uno smoothing "p-spline", equivalente a tre passaggi della media scorrevole; questi profili vengono confrontati nella figura a lato. (Vedere pagina 56 per una comparazione di queste modalità di smoothing). L'argomento "edge" controlla come vengono gestite le "estremità" del segnale (i primi e gli ultimi $w/2$ punti). Se edge=0, le estremità sono a zero. (In questo modo il tempo di esecuzione è indipendente dalla larghezza dello smoothing. È quello più veloce). Se edge=1, le estremità subiscono uno smoothing progressivamente meno ampio avvicinandosi alla fine. (In questo modo il tempo di esecuzione aumenta con l'aumentare delle larghezze dello smoothing). Il segnale filtrato viene restituito nel vettore "s". (Si possono omettere gli ultimi due argomenti di input: `fastsmooth(Y,w,type)` filtra con `edge=0` e `fastsmooth(Y,w)` filtra con `type=1` e `edge=0`). Rispetto agli algoritmi di smoothing basati sulla convoluzione, `fastsmooth` usa un semplice algoritmo ricorsivo con tempi di esecuzione tipicamente più rapidi per larghezze di smoothing molto grandi; esegue lo smoothing di un segnale di 1,000,000 di punti con una media scorrevole di 1,000 punti in meno di 0.1 secondi su un PC Windows standard. Ecco un semplice esempio di `fastsmooth` che ne mostra l'effetto sul rumore bianco (grafico).

```
x=1:100;
y=randn(size(x));
plot(x,y,x, fastsmooth(y,5,3,1), 'r')
xlabel('Blue: white noise.      Red: smoothed white noise.')
```

Smoothing segmentato

SegmentedSmooth.m è una versione *segmentata* di `fastsmooth`. La sintassi è la stessa di fastsmooth.m, tranne per il fatto che il secondo argomento di input "smoothwidths" può essere un vettore: `SmoothY = SegmentedSmooth (Y, smoothwidths, type, ends)`. La funzione divide Y in diverse regioni di uguale lunghezza definite dalla lunghezza del vettore 'smoothwidths', poi esegue il filtraggio di ciascuna regione col tipo di smoothing 'type' e con l'ampiezza definita dagli elementi del vettore 'smoothwidths'. Nell'esempio grafico di seguito, `smoothwidths=[31 52 91]`, si suddivide il segnale in tre regioni uguali e filtra la prima regione con uno smoothwidth di 31, la seconda con uno smoothwidth di 51 e l'ultima con uno smoothwidth di 91. Si può usare qualsiasi numero di ampiezze di smoothing e qualsiasi sequenza di ampiezze, proprio come si definisce il vettore "smoothwidths"; non sono necessarie altre modifiche. Digitare "help SegmentedSmooth" per altri esempi.

DemoSegmentedSmooth.m è uno script dimostrativo che mostra il funzionamento con diversi segnali costituiti da picchi rumorosi di larghezza variabile che diventano progressivamente più ampi (figura a lato). Se le larghezze dei picchi aumentano o diminuiscono regolarmente nel segnale, si può calcolare i vettore smoothwidths fornendo solo i numeri di segmenti ("NumSegments"), il primo valore, "startw", e l'ultimo, "endw", in questo modo:

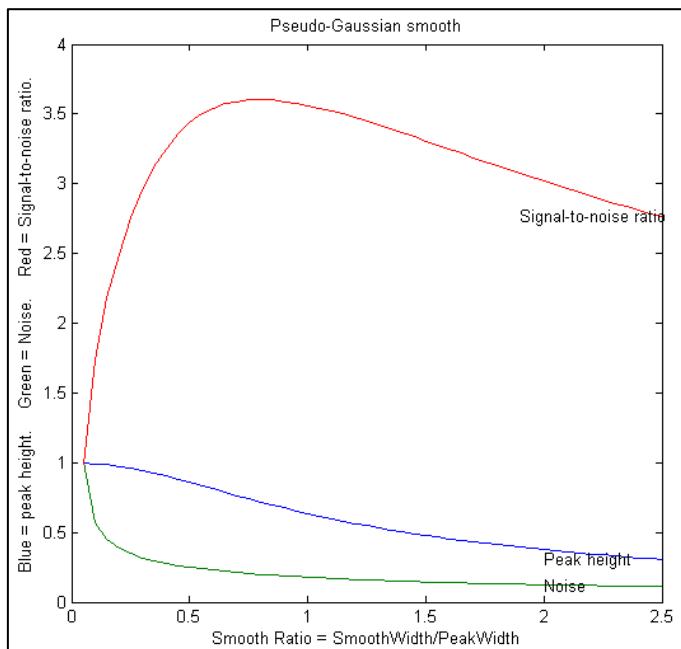
```
wstep=(endw-startw)/NumSegments;
smoothwidths=startw:wstep:endw;
```

Altre funzioni di smoothing.

Diederick ha pubblicato una funzione di smoothing Savitzky-Golay in Matlab, scaricabile dal Matlab File Exchange. È incluso nella funzione iSignal (pagina 366).

Greg Pittam ha pubblicato una modifica della funzione fastsmooth che tollera i NaN ("Not a Number") nel file dei dati (nanfastsmooth(Y,w,type,tol)) ed un'altra versione per lo smoothing di dati "angolo" che si ripetono ogni 360° o ogni 2π radianti (nanfastsmoothAngle(Y,w,type,tol)).

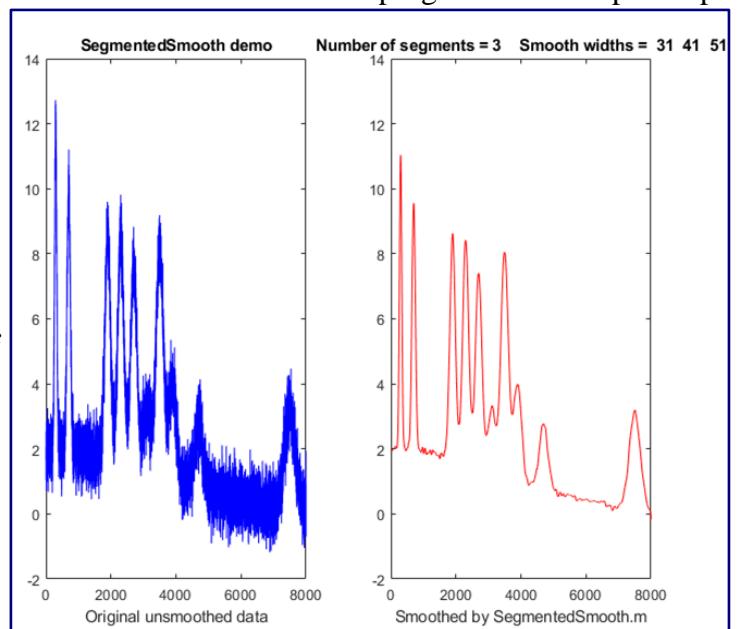
SmoothWidthTest.m è uno script dimostrativo sull'uso della funzione fastsmooth per mostrare l'effetto dello smoothing sull'altezza, sul rumore e sul rapporto segnale-rumore di un picco. Il



profilo del picco si può cambiare alla riga 7, il tipo di smoothing alla riga 8 e il rumore alla riga 9. A lato è mostrato un tipico risultato per un picco Gaussiano con rumore bianco e smoothing con una p-spline (pseudo-gaussiana). In questo caso, come per la maggior parte dei profili dei picchi, il rapporto segnale-rumore ottimale si ottiene con un rapporto di smoothing di circa 0.8. Tuttavia, tale valore ottimale corrisponde a una *significativa riduzione dell'altezza del picco*, che potrebbe essere un problema. Una larghezza dello smoothing di circa la metà della larghezza originale del picco lo distorce meno ma consente comunque di ottenere una buona riduzione del rumore.

SmoothVsCurvefit.m è uno script simile ma confronta anche l'approssimazione della curva come metodo alternativo per misurare l'altezza del picco *senza lo smoothing*.

Questo effetto viene esplorato più completamente nel codice seguente, che mostra un esperimento in Matlab o in Octave che crea un picco Gaussiano, lo filtra con lo smoothing, e confronta la versione filtrata con quella originale, poi usa le funzioni max(), halfwidth() e trapz() per stampare l'altezza, la semi-larghezza e l'area del picco. ("max" e "trapz" sono entrambe funzioni di Matlab e Octave, ma si deve eseguire il download di halfwidth.m. Per saperne di più su queste funzioni, si digita "help" seguito dal nome della funzione).



```

x=[0:.1:10]';
y=exp(-(x-5).^2);
plot(x,y)
ysmoothed=fastsmooth(y,11,3,1);
plot(x,y,x, ysmoothed, 'r')
disp([max(y) halfwidth(x,y,5) trapz(x,y)])
disp([max(ysmoothed) halfwidth(x,ysmoothed,5) trapz(x, ysmoothed)])
```

max	halfwidth	Area
1	1.6662	1.7725
0.78442	2.1327	1.7725

Questi risultati mostrano che lo smoothing *riduce* l'altezza del picco (da 1 a 0.784) e ne *aumenta* la larghezza (da 1.66 a 2.13), ma *non ha un effetto sservabile* sull'area del picco se si misura l'*area totale* del picco espanso. Lo smoothing è utile se altezza, posizione o larghezza del picco vengono misurati con metodi semplici, ma *non è necessario eseguire lo smoothing dei dati* se il rumore è bianco e i parametri del picco vengono misurati con i metodi dei quadrati minimi, perché i risultati di tali metodi ottenuti sui dati non filtrati saranno più accurati del segnale distorto con lo smoothing (vedere pagina 226).

Altre funzioni per la riduzione del rumore.

La funzione utente Matlab/Octave condense.m, condense(y,n), restituisce una versione condensata di y in cui ciascun gruppo di n punti viene sostituito dalla sua media, riducendo la lunghezza di y del fattore n. (Per i set di dati x,y, utilizzare questa funzione su **sia** la variabile indipendente x **che** la variabile dipendente y in modo che le caratteristiche di y appaiano agli stessi valori di x). Il rumore bianco casuale nel segnale viene ridotto di \sqrt{n} ma il colore del rumore rimane invariato.

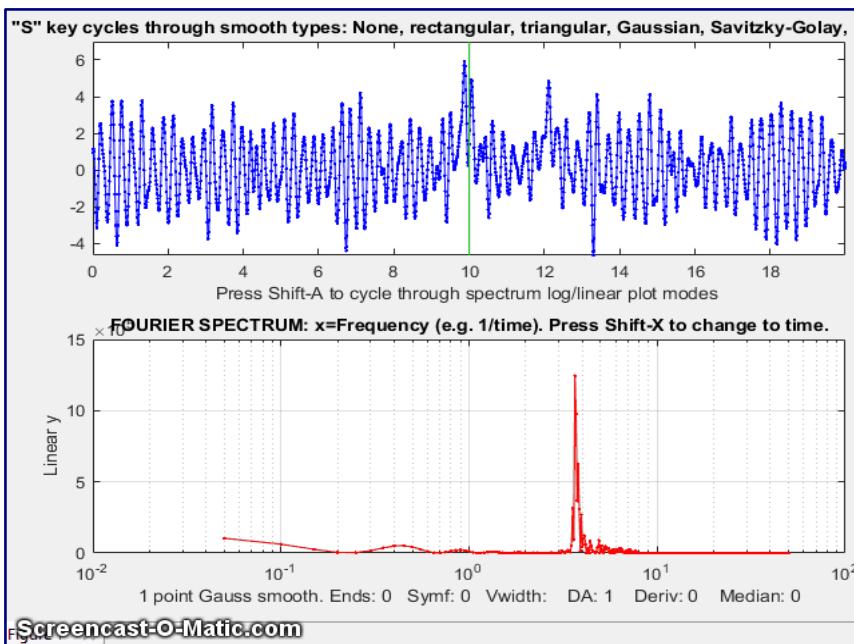
La funzione utente Matlab/Octave medianfilter.m, medianfilter(y,w), esegue un'operazione di filtraggio basato sulla mediana che sostituisce ciascun valore di y con la mediana di w punti adiacenti (che dev'essere un intero positivo). killspikes.m è un filtro basato sulla soglia [threshold] per eliminare gli spike. La sintassi è fy = killspikes(x, y, threshold, width). Ogni volta che trova un salto positivo o negativo nei dati tra y(n) e y(n+1) che eccede "threshold", sostituisce i successivi "width" punti con un segmento interpolato linearmente che si estende da x(n) a x(n+width+1). Lo script TestSpikefilters confronta entrambi i filtri di spike su una Gaussiana con spike e mostrando l'accuratezza con cui si recupera l'area originale del picco.

ProcessSignal è una funzione Matlab/Octave a riga di comando che esegue lo smoothing e la derivazione su una serie temporale x,y (vettori di colonna o righe). Può impiegare tutti i tipi di smoothing descritti. Digitare "help ProcessSignal" sulla riga di comando. Questa funzione restituisce il segnale processato come vettore che ha lo stesso profilo di x, indipendentemente da quello di y. La sintassi è

```
Processed = ProcessSignal(x, y,
DerivativeMode, w, type, ends,
Sharpen, factor1, factor2, Symize,
Symfactor, SlewRate,
MedianWidth).
```

Lo smoothing in tempo reale in Matlab viene trattato a pagina 339.
Lo smoothing in Python è descritto a pag. 429.

iSignal (pag. 366) è una funzione interattiva da tastiera per Matlab



che include lo smoothing per segnali di serie temporali utilizzando *tutti gli algoritmi discusso in precedenza*, incluso lo smoothing di Savitzky-Golay, quello segmentato, un filtro mediano e una funzione di condensazione. Con semplici sequenze di tasti si regolano tutti i parametri dello smoothing in modo continuo osservandone immediatamente gli effetti sul segnale, facilitando l'osservazione di come i diversi tipi e quantità di smoothing influiscono sul rumore e sul segnale, come le altezze, le larghezze e le aree dei picchi. Altre funzionalità di iSignal comprendono la derivazione, lo "sharpening", l'interpolazione, la misura del picco ai quadrati minimi e la modalità a spettro di frequenze che mostra come lo smoothing e le altre funzioni modifichino lo spettro delle frequenze nei segnali. Il semplice script “[iSignalDeltaTest](#)” mostra la risposta in frequenza delle funzioni di smoothing di iSignal applicandole ad uno spike puntiforme, consentendo la modifica del tipo e dell'ampiezza dello smoothing per osservare il cambiamento della risposta in frequenza. (Visualizzare il codice [qui](#) o scaricarne il [file ZIP](#) con i dati di esempio per il test). La versione Octave è [isignal octave.m](#), che ha dei tasti diversi per il pan e lo zoom.

Da provare: Ecco un esperimento da provare utilizzando *iSignal*. Si utilizza un esempio pre-registrato di un segnale molto rumoroso con molto rumore ad alta frequenza (blu) *che oscura completamente un picco ottimo* centrato in $x=150$, altezza= $1e-4$; SNR=90. Per prima cosa, si effettua il download di [iSignal.m](#) e [NoisySignal.mat](#) nel path di ricerca di Matlab, poi si eseguono questi comandi:

```
>> load NoisySignal
>> isignal(x,y);
```

Si usano i tasti **A** e **Z** per aumentare o diminuire l'ampiezza dello smoothing, e il tasto **S** per scorrere i diversi tipi di smoothing. Suggerimento: usare lo smoothing “p-spline” e continuare ad aumentarne l'ampiezza fino a far apparire il picco. (Sfortunatamente, iSignal al momento non funziona in *Octave*, ma funziona in un browser Web con *Matlab Online*. Cfr. <https://www.mathworks.com/products/matlab-online.html>).

Nota: Se si sta leggendo online, si può fare un click-destro su qualsiasi link dei file .m su questo sito e poi si seleziona **Save Link As...** per eseguire il download sul proprio computer ed usarli in Matlab.

Live Script per lo smoothing

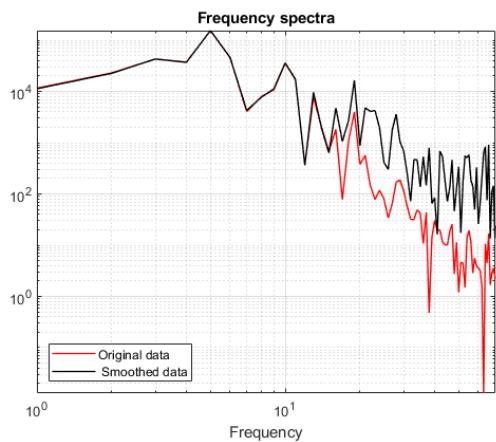
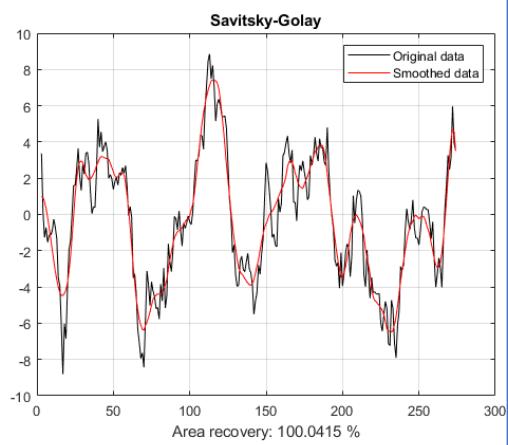
Ecco un *Live Script Matlab* interattivo per eseguire diversi tipi di smoothing applicati ai dati sperimentali archiviati su disco (pagina 360; link per il download: [DataSmoothing mlx](#)). Può eseguire la rimozione dei picchi, lo smoothing della media mobile con un massimo di 5 passaggi, il filtraggio passa-basso di Savitsky-Golay e Fourier (pagina 122) e la rimozione del rumore wavelet (pagina 126, che richiede [Matlab Wavelet Toolkit](#)). Sclickando sul pulsante **Open data file** nella riga 1 si apre un browser di file, che consente di navigare fino al file di dati (in formato .csv o .xlsx; lo script presuppone che i dati x,y siano nelle prime due colonne). Tutte le variabili e le impostazioni appaiono come al solito nell'area di lavoro di Matlab; i dati finiti con smoothing si trovano nel vettore “sy”.

Data Smoothing

with median filter, recursive sliding average, Savitsky-Golay, Fourier low-pass, or wavelet denoise (which requires the Matlab Wavelet Toolbox).

Tom O'Haver 4/18/2023

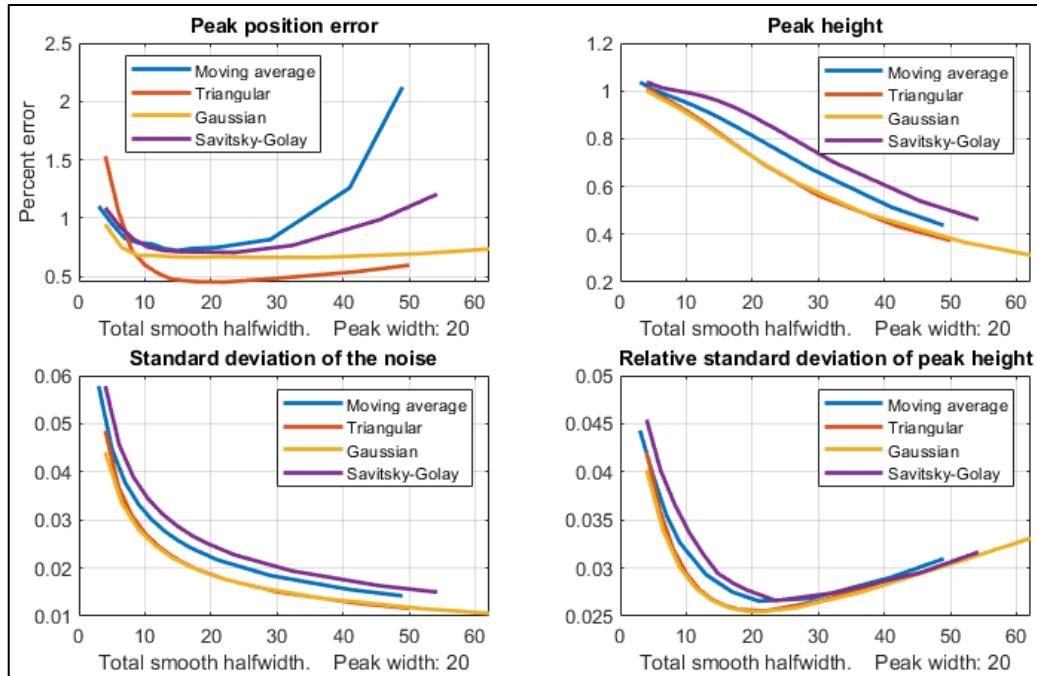
```
1 Open data file file= uigetfile('.csv;*.xlsx');% Click this button to load data  
2 mydata=xlsread(file); %  
  
3 PlotBeforeAndAfter=  ; % Check this box to display smoothed signal as well as ori  
4 FrequencySpectra= ; % Check this box to display frequency spectra  
5  
6 % Set the x-axis scale expansion beginning and end points (0 - 100%)  
7 startpc= 1  ; % Percentage of data points to start data selec  
8 endpc= 97.6  ; % Percentage of data points to end data selectic  
9  
10 RemoveSpikes= ; % Check this box to use median filter to remove spikes  
11 SpikeWidth= 6  ; % Estimated spike width, in data points  
12  
13 SmoothType = Savitsky-Golay  ; % Select smoothing algrithm  
14  
15 % For sliding average:  
16 SmoothWidth= 41  ; % Smooth width for sliding average  
17 NumPasses= 4  ; % Number of sliding average passes (1-5)  
18 SmoothEnds=  ; % Click this box to smooth the ends of the data record.  
19  
20 % For Fourier low-pass filter:  
21 FrequencyCutoff= 3.5  ; % Fourier filter cutoff frequency  
22 CutOffRate= 4  ; % Fourier filter cutoff rate  
23  
24 % For Savitsky-Golay smooth only  
25 SGSmoothWidth= 31  ; % Frame length for Savitsky-Golay  
26 PolynomialOrder= 5  ; % Polynomial order must be less than the  
27  
28 % For wavelet denoising only (requires Matlab Wavelet Toolox be installed)  
29 WaveletType=bior5.5  ;  
30 Level= 4  ; % The higher the level, the lower is the frequency  
31  
32 NumPoints=length(mydata);
```



Lo script ha diversi controlli interattivi. Gli slider **startpc** e **endpc** nelle righe 7 e 8 consentono di selezionare quale porzione dell'intervallo di dati elaborare, dallo 0% al 100% dell'intervallo totale del file di dati. La casella di controllo **RemoveSpikes** applica un filtro mediano (pagina 47) per rimuovere picchi stretti e netti. Il menu a discesa **SmoothType** nella riga 13 seleziona l'algoritmo di smoothing; ognuno ha uno o più controlli specifici per il tipo di smoothing nelle righe da 16 a 30. La prima scelta è l'algoritmo della media mobile ricorsiva ([fastsmooth.m](#)) (pagina 39). La larghezza dello smoothing e il numero di passaggi sono controllati dai cursori nelle linee 16 e 17. Gli altri controlli vengono spiegati nelle righe di commento indicate (in verde). Il filtraggio di Fourier, Savitsky-Golay e il denoising wavelet sono argomenti che verranno spiegati nelle prossime sezioni. La casella **PlotBeforeAndAfter** nella riga 3, dà la possibilità di disegnare il segnale originale (in nero) insieme al segnale elaborato (in rosso). La casella **FrequencySpectra** nella riga 4 consente di visualizzare lo *spettro della frequenza* dei segnali originali e/o elaborati (pagina 87). **Nota:** per visualizzare i tracciati grafici a destra del codice, come mostrato sopra, cliccare col tasto destro del mouse sullo spazio vuoto a destra e selezionare "Disable synchronous scrolling". Nota: con un doppio-click su qualsiasi slider se ne possono modificare i range se quelli iniziali sono insufficienti.

Confronto delle prestazioni dello smoothing

La funzione Matlab/Octave "[MultiPeakOptimization.m](#)" è una funzione autonoma che confronta le prestazioni di quattro tipi di operazioni di smoothing lineare: (1) lo slittamento della media rettangolare, (2) triangolare, (3) p-spline (equivalente a tre passi dello slittamento della media), e (4) il Savitzky-Golay. Questi sono i quattro tipi di smoothing trattati, corrispondenti ai quattro valori



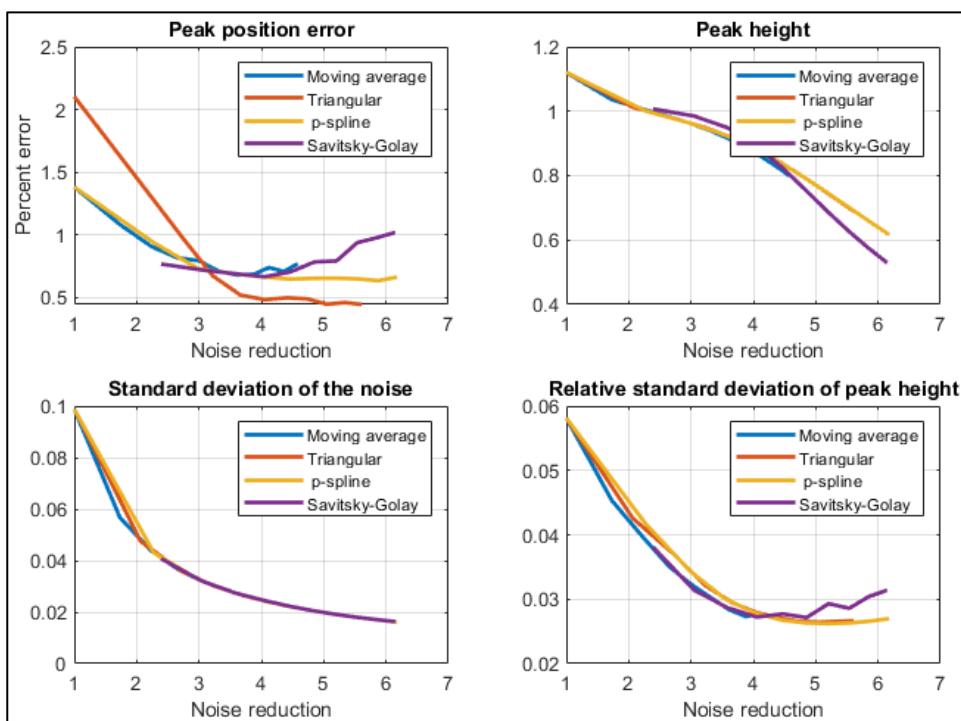
dell'argomento "SmoothMode" di input delle funzioni [ProcessSignal](#) e delle funzioni iterative [iSignal](#). Queste quattro operazioni di smoothing vengono applicate ad un segnale di 18000 punti consistente in 181 picchi Gaussiani tutti con un'altezza di 1.0 e una FWHM ([full-width at half-maximum \[larghezza intera a metà del massimo\]](#)) di 20 punti ("wid", riga 10), che sono tutti separati da un valore x di 160.01 (riga 16), con in più del rumore bianco casuale distribuito normalmente con una media pari a zero e una deviazione standard di "Noise" (riga 20). La posizione del picco sull'asse x e l'altezza sull'asse y di ciascun picco con smoothing vengono determinate dall'altezza e dalla posizione del singolo punto massimo del segnale per ciascun picco. La deviazione standard relativa delle altezze misurate dei picchi viene registrata in funzione della "larghezza totale di smoothing", *tsw*, che è definita come la semi-larghezza della risposta all'impulso di ciascuno smoothing. I risultati sono mostrati nella figura sotto per una semi-larghezza di 20 e una deviazione standard del rumore di 0.2 (cioè il 20% dell'altezza del picco).

I quattro quadranti del grafico sono: (in alto a sinistra) l'errore di posizione del picco espresso come percentuale della separazione del picco; (in alto a destra), l'altezza media dei picchi con smoothing; (in basso a sinistra), la deviazione standard del rumore con smoothing; (in basso a destra) la deviazione standard relativa delle altezze misurate dei picchi. Le diverse tipologie di smoothing sono indicate dal colore: blu - scorrimento della media; rosso - triangolare; giallo - p-spline e viola - Savitzky-Golay.

Si ha che i risultati di questi diversi tipi di smoothing sono abbastanza simili, ma che quello di Savitzky-Golay offre la minore riduzione dell'altezza del picco ma anche la più piccola riduzione dell'ampiezza del rumore, rispetto agli altri metodi. Tutti questi metodi di smoothing determinano miglioramenti simili nella deviazione standard dell'altezza del picco (pannello in basso a destra) e nell'errore di posizione (pannello in alto a sinistra). Inoltre, in tutti i casi, la prestazione ottimale si

ottiene quando la larghezza totale dello smoothing è approssimativamente uguale alla semi-larghezza del picco. Le conclusioni sono le stesse per un picco Lorentziano, come dimostrato da una funzione simile, "[MultiPeakOptimizationLorentzian.m](#)", [grafico](#), con la differenza che la riduzione dell'altezza del picco è maggiore per il Lorentziano. Per le applicazioni in cui la forma del segnale deve essere preservata il più possibile, il metodo da scegliere è il Savitzky-Golay. Nelle applicazioni di rilevamento dei picchi (pag. 64), invece, dove lo scopo dello smoothing è quello di ridurre il rumore nel segnale derivato, il mantenimento della forma della derivata è meno importante perché i parametri dei picchi vengono determinati dall'approssimazione dei minimi quadrati. Pertanto, lo smoothing triangolare o p-spline è adatto a questo scopo e può essere più rapido per larghezze di smoothing molto grandi.

Le differenze tra questi metodi sono ancora minori quando le ascisse nei grafici precedenti vengono cambiate dalla *larghezza di banda dell'intero smoothing al fattore di riduzione del rumore bianco*, definito come la radice quadrata del reciproco della somma del quadrato della funzione di risposta all'impulso, come mostrato di seguito.

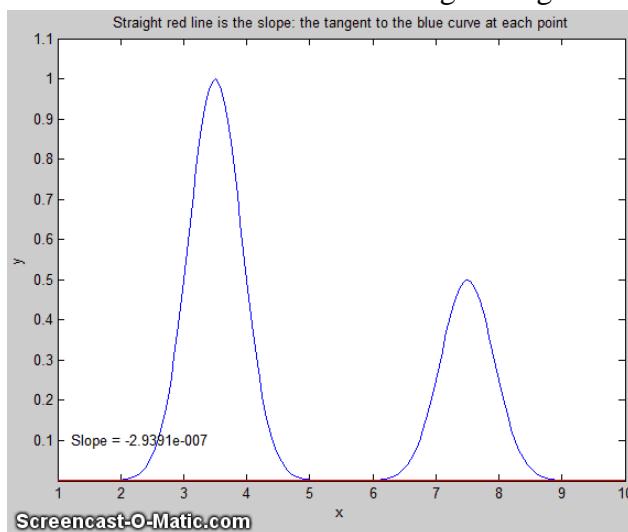


Un dettaglio importante è che questi risultati si applicano solo se il rumore nel segnale è *bianco* (pag. 28). Se si esegue lo smoothing della derivata di un segnale, ad esempio la derivata seconda di un picco Gaussiano con rumore bianco ([grafico](#)), il contenuto alle alte frequenze sia del segnale che del rumore viene notevolmente migliorato, e questi risultati saranno diversi, mostrando prestazioni relative molto inferiori per la media mobile semplice ([grafico](#)). Anche in questo caso lo smoothing di Savitzky-Golay risulta superiore.

Un metodo più sofisticato di riduzione del rumore, detto *denoising wavelet*, verrà presentato a pagina 126.

Differenziazione

La differenziazione simbolica delle funzioni è un argomento introdotto in tutti i corsi elementari di Calcolo. La differenziazione di segnali digitalizzati è un'applicazione di questo concetto che ha



molti utilizzi nell'elaborazione analitica dei segnali. La derivata prima di un segnale è la velocità di variazione di y rispetto a x , ovvero, dy/dx , che si interpreta come la pendenza della tangente al segnale in ciascun punto, come mostrato nell'animazione a lato da questo [script](#). (e l'animazione non viene visualizzata, cliccare su [questo link](#)). L'algoritmo più semplice per il calcolo diretto della derivata prima è chiamato metodo delle "differenze finite":

$$Y'_j = \frac{Y_{j+1} - Y_j}{X_{j+1} - X_j} = \frac{Y_{j+1} - Y_j}{\Delta X} \quad X'_j = \frac{X_{j+1} + X_j}{2} \quad (\text{per } 1 < j < n-1).$$

dove X'_j e Y'_j sono i valori X e Y del j^{esimo} punto della derivata, n = numero dei punti nel segnale e ΔX è la differenza tra i valori di X dei punti adiacenti. Una variante comunemente usata di questo algoritmo calcola la pendenza *media* fra tre punti adiacenti:

$$Y'_j = \frac{Y_{j+1} - Y_{j-1}}{2\Delta X} \quad X'_j = X_j \quad (\text{per } 2 < j < n-1).$$

Questo è detto metodo a *differenza centrale*; ha il vantaggio che non provoca uno spostamento della posizione sull'asse x della derivata. È possibile anche calcolare derivate '*gap-segment*' in cui l'intervallo sull'asse x tra i punti nell'espressione precedente è maggiore di uno; per esempio, Y_{j-2} e Y_{j+2} , o Y_{j-3} e Y_{j+3} , ecc. Ciò equivale ad applicare uno smoothing a media mobile (rettangolare) (pagina 38) oltre alla derivata.

La *derivata seconda* è la derivata della derivata: è la misura della *curvatura* del segnale, ovvero, la velocità di variazione della pendenza del segnale. Si può calcolare applicando il calcolo della derivata prima due volte di seguito. L'algoritmo più semplice per il calcolo diretto della derivata seconda in un unico passaggio è:

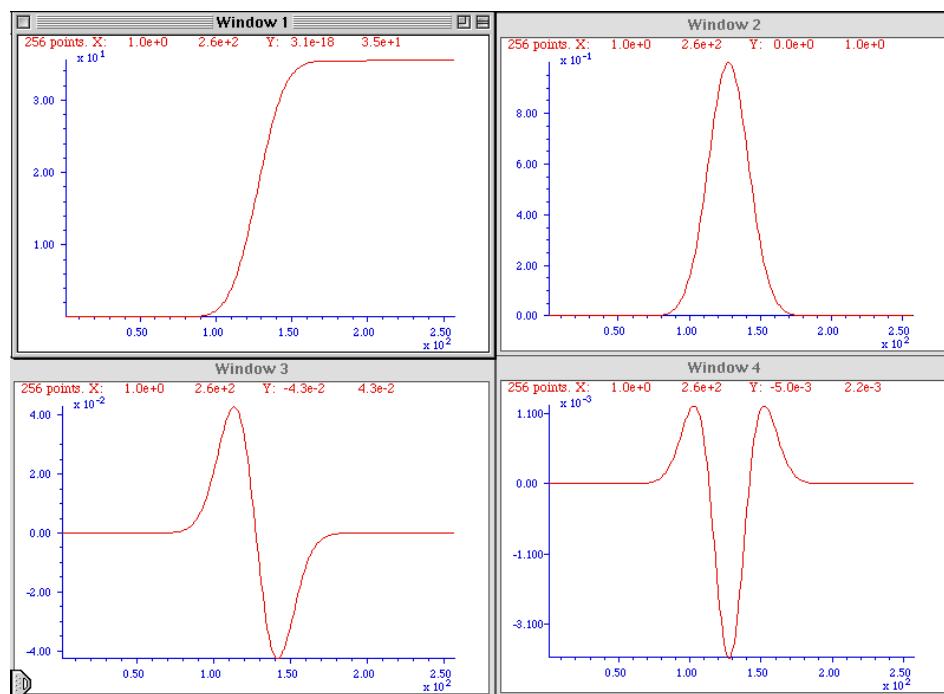
$$Y''_j = \frac{Y_{j+1} - 2Y_j + Y_{j-1}}{\Delta X^2} \quad X'_j = X_j \quad (\text{per } 2 < j < n-1).$$

Allo stesso modo, le derivate di ordine superiore si possono calcolare utilizzando la giusta sequenza di coefficienti: per esempio, +1, -2, +2, -1 per la derivata terza e +1, -4, +6, -4, +1 per la derivata 4^a, sebbene queste derivate si possano anche calcolare semplicemente applicando ripetutamente le derivate di ordine più basso. La derivata prima viene interpretata come la *pendenza* del segnale in ciascun punto e la derivata seconda come la *curvatura*. Dove la curvatura del segnale è concava verso il *basso*, la derivata seconda è *negativa* e dove il segnale è concavo verso l'*alto*, la derivata seconda è *positiva*. Per le derivate di ordine superiore, non c'è una nomenclatura; ogni derivata è solo il tasso di cambiamento della precedente.

Lo smoothing di Savitzky-Golay (pagina 38) si può usare anche come algoritmo di differenziazione scegliendo opportunamente gli argomenti di input; combina perfettamente differenziazione e smoothing in un unico algoritmo.

L'*accuratezza* della differenziazione numerica è mostrata dallo script Matlab/Octave [GaussianDerivatives.m](#) (link al [grafico](#)), che confronta le esatte espressioni *analitiche* per la derivata di una Gaussiana ([subito disponibili da Wolfram Alpha](#)) con i valori *numerici* ottenuti dalle espressioni precedenti, dimostrando che il profilo e l'ampiezza delle derivate coincidono perfettamente a patto che l'intervallo di campionamento non sia troppo grossolano. Esso dimostra anche che è possibile ottenere la derivata n -esima applicando n derivate prime ripetutamente. In definitiva, la limitazione della precisione numerica del computer potrebbe essere una limitazione, ma solo in alcuni casi estremi (come dimostrato a pagina 332). (Un metodo di differenziazione alternativo basato sulla *Trasformata di Fourier*, pag. 87, può calcolare qualsiasi ordine di derivazione ma, in pratica, non è stato molto utilizzato. Vedere il riferimento 88).

Proprietà Fondamentali delle Derivate dei Segnali

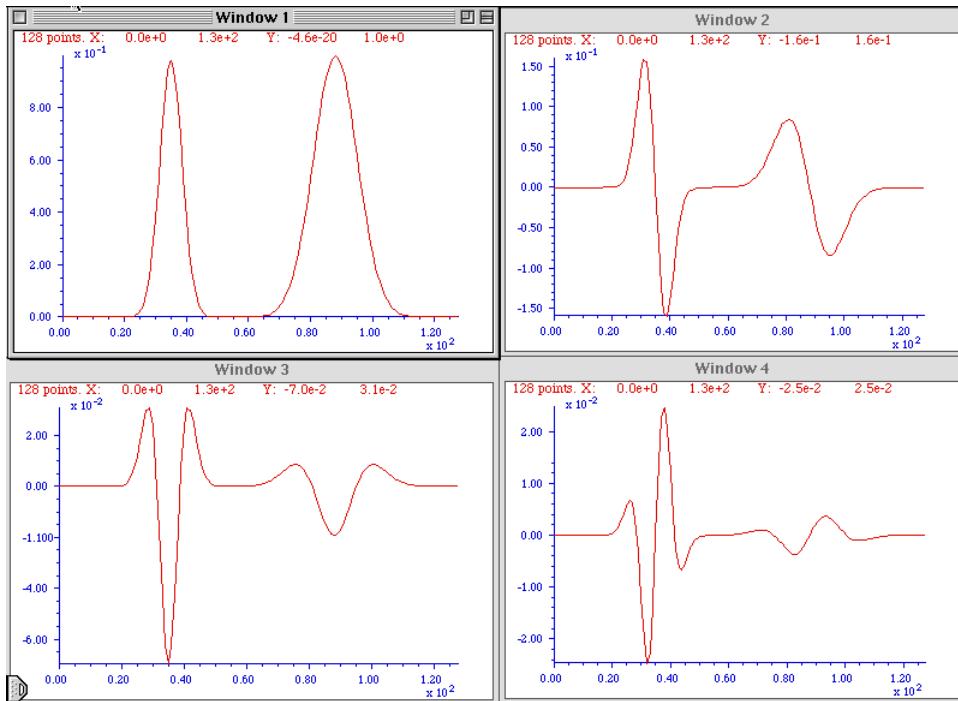


ogni segnale ricordando che la derivata è semplicemente la pendenza del segnale di partenza: dove la pendenza di un segnale sale, la derivata è positiva; mentre in un segnale con una pendenza discendente, la derivata è negativa; e dove un segnale ha una pendenza nulla, la derivata è zero. ([Il codice Matlab/Octave per questa figura](#).)

Il segnale sigmoidale della Window 1 ha un *punto di flesso* (il punto dove la pendenza è massima) al centro dell'intervallo dell'asse x. Ciò corrisponde al *massimo* della sua derivata prima (Window 2) ed al *passaggio per lo zero* (il punto dove il segnale attraversa l'asse x passando da positivo a negativo o viceversa) della derivata seconda nella Window 3. Questo comportamento può essere utile per localizzare con precisione il punto di flesso in un segnale sigmoideo, calcolando la posizione del passaggio per lo zero della sua derivata seconda. Allo stesso modo, la posizione del massimo in un segnale a forma di picco si può calcolare precisamente calcolando la posizione del passaggio per lo zero della sua derivata prima. Picchi con profili diversi hanno derivate di forme diverse: la funzione Matlab/Octave [DerivativeShapeDemo.m](#) mostra il profilo della derivata prima di 16 diversi modelli di picchi (graFICO a pagina 411). Qualsiasi profilo regolare di un picco con un singolo massimo ha derivate sequenziali che mostrano una serie di *massimi e minimi alternati*, *il cui numero totale è uno in più rispetto all'ordine di derivazione*. Le derivate di ordine pari hanno un

La figura a lato mostra i risultati di derivate successive di un picco Gaussiano generato al computer. Il segnale in ciascuna delle quattro finestre è la derivata prima della precedente; cioè, la Window 2 è la derivata prima della Window 1, la Window 3 è la derivata prima della Window 2, la Window 3 è la derivata *seconda* della Window 1, e così via. È possibile prevedere la forma di

massimo o un minimo al centro del picco, e le derivate di ordine dispari hanno un passaggio per lo zero al centro del picco ([codice Matlab/Octave](#)). Qui si può anche vedere che l'ampiezza numerica delle derivate (valori sull'asse y) è molto inferiore del segnale originale perché le derivate sono le *differenze* tra i punti adiacenti dei valori y, diviso per l'incremento della variabile indipendente. (È lo stesso motivo per cui il contachilometri delle auto mostra solitamente un numero molto più grande del tachimetro (a meno di non avere un'auto nuovissima e guidare molto velocemente). Il tachimetro è essenzialmente la derivata prima del contachilometri).



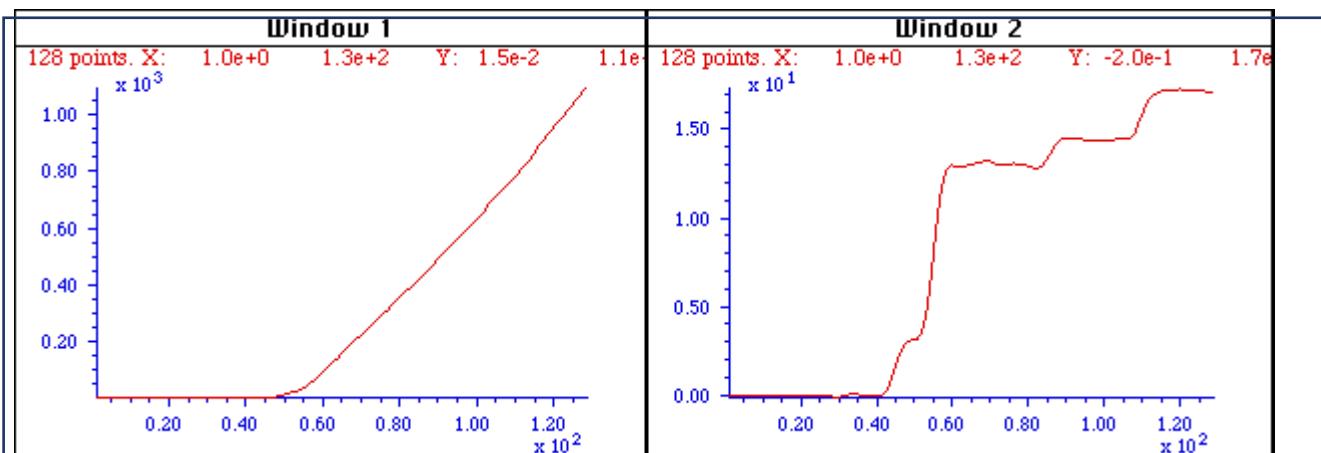
Una proprietà importante della differenziazione dei segnali a forma di picco è l'effetto della *larghezza* del picco sull'altezza delle derivate. La figura a lato mostra i risultati delle derivate successive di due bande Gaussiane generate al computer. Le due bande hanno la stessa amplificazione (altezza del picco) ma una è esattamente il

doppio dell'altra in larghezza. Come si vede, il picco *più largo* ha una derivata di altezza *più piccola* e l'effetto diventa più evidente con le derivate di ordine superiore. In generale, l'altezza della derivata n^a di un picco è inversamente proporzionale alla potenza n^a della sua larghezza, a parità di forma e altezza del segnale. Pertanto, la differenziazione in effetti discrimina i picchi più ampi e maggiore è l'ordine della derivata maggiore è la discriminazione. Questo comportamento può essere utile nelle applicazioni analitiche quantitative per rilevare i picchi sovrapposti e oscurati da picchi di fondo più forti ma più larghi. (Il codice Matlab/Octave per questa figura). L'altezza della derivata di un picco dipende anche dalla *forma* del picco ed è direttamente proporzionale all'*altezza* del suo picco. Profili Gaussiani e Lorentziani hanno forme e ampiezze delle derivate prime e seconde leggermente diverse. L'altezza della derivata n^a di un picco Gaussiano di altezza H e larghezza W si può stimare con [l'equazione empirica](#) $H^*(10^{(0.027*n^2+n*0.45-0.31)}.*W^{(-n)})$, dove W è l'intera larghezza a metà del massimo (FWHM) misurata nel numero di punti di x,y.

Sebbene la differenziazione modifichi completamente il profilo del *tipo di picco* dei segnali, un *segnalet periodico* come un'onda sinusoidale si comporta diversamente. La derivata di un'onda sinusoidale di frequenza f è un'onda sinusoidale *sfasata in fase*, o coseno, della *stessa frequenza* e con un'ampiezza che è proporzionale a f , come si può dimostrare in [Wolfram Alpha](#). La derivata di un segnale periodico contenente diverse componenti sinusoidali di frequenza diversa *conterrà ancora le stesse frequenze*, ma con ampiezze e fasi alterate. Per questo motivo, quando si prende la derivata di un segnale musicale o vocale, la musica o il parlato sono ancora completamente riconoscibili, ma con le alte frequenze aumentate in ampiezza rispetto alle basse frequenze e ne risulta un suono "acuto" e "metallico". Cfr. pagina 383 per un esempio.

Applicazioni della Differenziazione

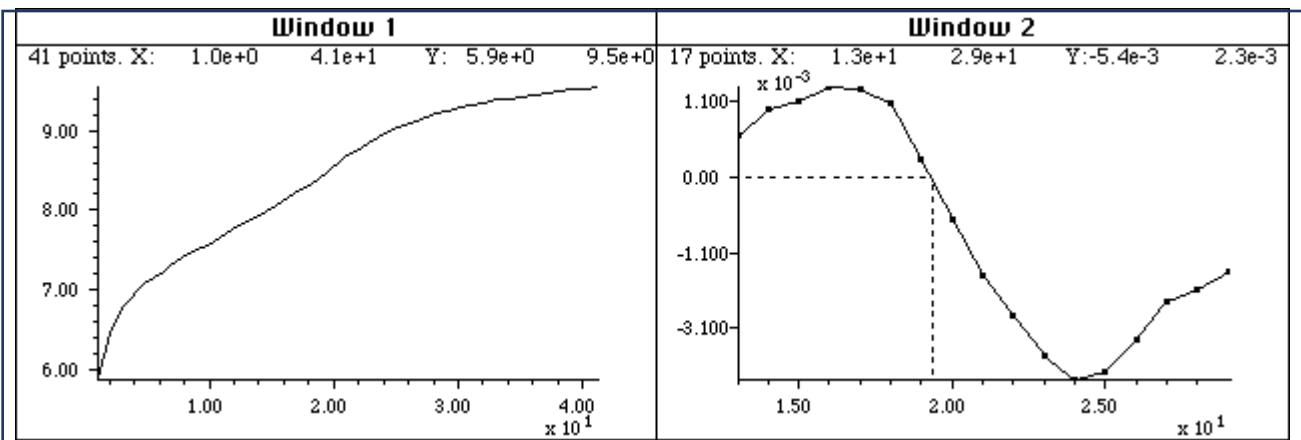
Un semplice esempio di applicazione della differenziazione di segnali sperimentali è mostrato nella figura seguente. Questo è il tipico segnale registrato nelle titolazioni amperometriche e incerti tipi di analisi termiche e in esperimenti cinetici: una serie di segmenti rettilinei con diverse pendenze. L'obiettivo consiste nel determinare quanti segmenti ci sono, dove si separano e la pendenza di ciascun segmento. Questo risulta difficile da fare partendo dai dati grezzi, perché le differenze delle pendenze sono piccole e la risoluzione dello schermo del computer è limitata. Il compito si semplifica calcolando la derivata prima (pendenza) del segnale (a destra). Ogni segmento è ora chiaramente visto come un passo diverso la cui altezza (valore sull'asse y) è la pendenza. L'asse y ora ha come unità dy/dx . Si noti che nell'esempio i passi nella derivata del non sono totalmente piatti, indicando che i segmenti nel segnale originale non erano perfettamente rettilinei. Ciò molto probabilmente è dovuto a del rumore casuale nel segnale originale. Sebbene tale rumore non sia particolarmente visibile, è molto più evidente nella derivata.



*Il segnale a sinistra sembra essere più o meno una linea retta, ma la sua **derivata** (dx/dy) calcolata numericamente, disegnata sulla destra, mostra che la linea in effetti ha diversi segmenti approssimativamente rettilinei con diverse pendenze e con delle separazioni ben definite tra ciascun segmento.*

Solitamente si osserva che la differenziazione degrada il rapporto segnale rumore, a meno che l'algoritmo di differenziazione non includa uno smoothing (pag. 38) accuratamente ottimizzato per ciascuna applicazione. Gli algoritmi numerici per la differenziazione sono numerosi così come quelli per lo smoothing e si devono scegliere con cura per controllare la degradazione del rapporto segnale/rumore (pag. 68).

Un classico uso della derivata seconda in analisi chimica è la localizzazione dei "punti di equivalenza" [endpoint] nella titolazione potenziometrica. Nella maggior parte delle titolazioni, la curva di titolazione ha la forma sigmoidea e il punto di flesso, il punto in cui la pendenza è massima e la curvatura è zero, indica l'endpoint. La derivata prima della curva di titolazione esporrà, quindi, un *massimo* nel punto di flesso, e la derivata seconda mostrerà un *passaggio per lo zero* in quel punto. I massimi e i passaggi per lo zero si localizzano più facilmente con precisione rispetto ai punti di flesso.



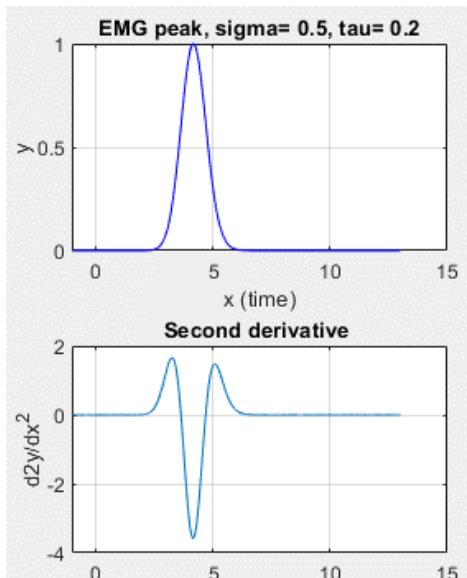
Il segnale a sinistra è la [curva della titolazione del pH](#) di un acido molto debole con una base forte, col volume in mL sull'asse X e il pH sull'asse Y. Il "punto di equivalenza" è quello in cui si ha la maggior pendenza; è anche un punto di flesso, dove la curvatura del segnale è zero. Con un acido debole come questo, è difficile localizzare precisamente questo punto partendo dalla curva originale della titolazione. Il "punto di equivalenza" si localizza più facilmente nella [derivata seconda](#), mostrata a destra, nel passaggio per lo zero.

La figura mostra una titolazione della curva del pH di un acido debolissimo con una base forte, col volume in mL sull'asse X e il pH sull'asse Y. Il punto di equivalenza volumetrica (l'endpoint "teorico") è a 20 mL. Il "punto di equivalenza" è quello in cui si ha la maggior pendenza; è anche un punto di flesso, dove la curvatura del segnale è zero. Con un acido debole come questo, è difficile localizzare precisamente questo punto partendo dalla curva originale della titolazione. La derivata seconda della curva è mostrata a destra nella Window 2. Il passaggio per lo zero della

derivata seconda corrisponde all'endpoint e misurabile con una precisione maggiore. Si noti che nel grafico della derivata seconda, sono state espanso le scale sia dell'asse x che dell'asse y per mostrare più chiaramente il punto di passaggio per lo zero. Le linee punteggiate mostrano il passaggio per lo zero cade a circa 19.4 mL, prossimo al valore teorico di 20 mL.

Le derivate si possono usare anche per rilevare asimmetrie inaspettate in picchi altrimenti simmetrici. Ad esempio, i picchi Gaussiani puri sono simmetrici, ma se subiscono un ampliamento esponenziale (pagina 126), diventano asimmetrici. Se il grado di allargamento è piccolo, può essere difficile da rilevare visivamente, ed è qui che la differenziazione può aiutare. Lo script Matlab/Octave [DerivativeEMGDemo.m \(grafico\)](#) mostra dalla derivata 1^a alla

5^a di una Gaussiana leggermente esponenzialmente allargata (EMG); di queste derivate, la seconda mostra chiaramente picchi positivi diversi ma che dovrebbero essere uguali per un picco completamente simmetrico. Le derivate di ordine maggiore non offrono alcun chiaro vantaggio e sono più suscettibili al rumore bianco nel segnale. Per un altro esempio, se un picco Gaussiano è fortemente sovrapposto da un picco più piccolo, il risultato è solitamente asimmetrico. Lo script [DerivativePeakOverlapDemo \(grafico\)](#) mostra le derivate dalla 1^a alla 5^a di due Gaussiane sovrapposte mentre il secondo picco è così piccolo e così vicino che è impossibile distinguerlo ad occhio, ma la derivata seconda mostra chiaramente l'asimmetria confrontando le altezze dei due picchi positivi. [DerivativePeakOverlap.m](#) rileva l'estensione minima della sovrapposizione dei

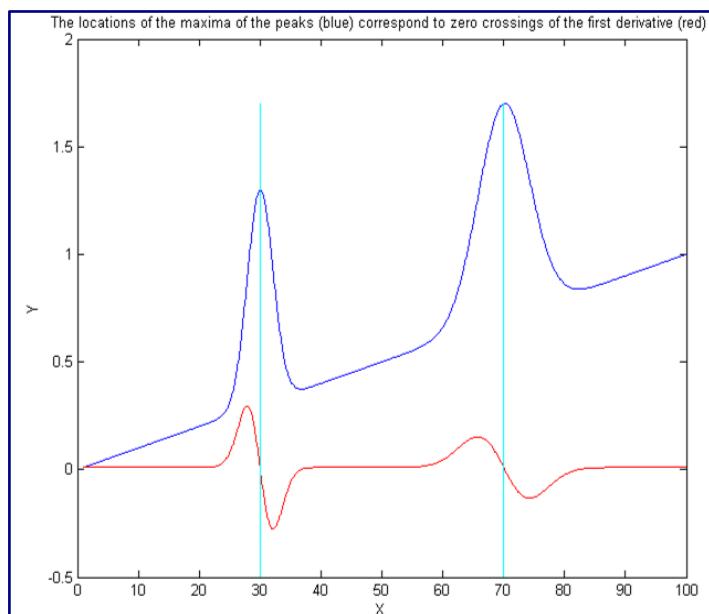


picchi tra la derivata prima e la seconda, cercando il punto in cui sono visibili due picchi; per ogni separazione di prova, stampa la separazione, la risoluzione e il numero di picchi rilevati nelle derivate prima e seconda. Ecco un altro [esempio](#) di derivata seconda.

Le derivate possono essere utilizzate anche per *correggere* l'asimmetria del picco, aggiungendo una porzione ponderata della derivata prima al picco originale, come descritto nella sezione sull'asimmetria del picco a pagina 77.

Rilevamento dei picchi

Un uso comune della differenziazione è nel rilevamento dei picchi in un segnale, soprattutto quando non se ne conosce il numero o la loro posizione. È chiaro dalle proprietà fondamentali descritte nella sezione precedente che la derivata prima di un picco ha un passaggio per lo zero verso il basso nel massimo del picco, utilizzabile per localizzare il valore x del picco, come mostrato a lato ([script](#)). Se *non c'è rumore* nel segnale, allora tutti i punti con dei valori più bassi da entrambi i lati saranno massimi di picchi. Ma c'è sempre almeno un po' di rumore nei segnali reali in esame, e questo provocherà dei falsi passaggi per lo zero semplicemente per il rumore. Per evitare questo problema, una tecnica popolare esegue lo smoothing del segnale prima di ricavarne la derivata prima, prima di cercarne i passaggi per lo zero discendenti, e poi prendere solo quei passaggi la cui pendenza superi un certo valore minimo (detta "soglia della pendenza") nel punto in cui l'ampiezza del segnale originale supera un certo minimo (detta "soglia dell'ampiezza"). Regolando attentamente la larghezza dello smoothing, la soglia della pendenza e quella dell'ampiezza, è possibile rilevare solo i picchi desiderati su un'ampia gamma di larghezze di picchi ed ignorare quelli troppo piccoli, troppo larghi o troppo stretti. Inoltre, poiché lo smoothing può distorcere il segnale dei picchi, riducendone le altezze ed aumentandone la larghezza (pag. 38), tale tecnica si può estendere per misurare la posizione, l'altezza e la larghezza di ciascun picco con l'approssimazione ai quadrati minimi della curva di un segmento del *segnale originale non filtrato in prossimità dell'apice del picco* (dove il rapporto segnale rumore è solitamente migliore). Pertanto, anche se è necessario uno smoothing forte per avere una discriminazione affidabile contro il rumore, i parametri estratti dall'approssimazione della curva non vengono



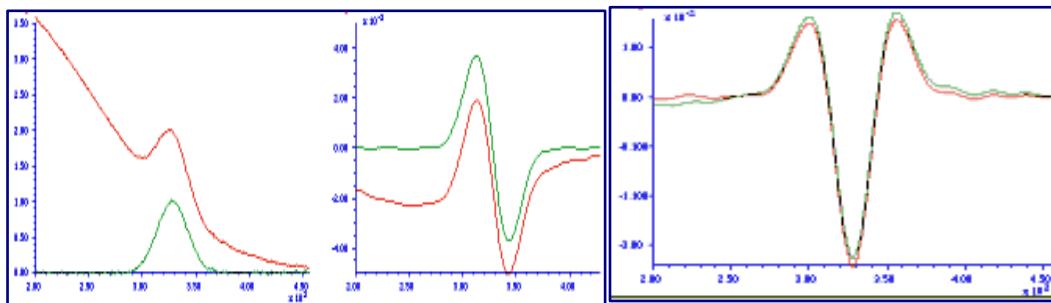
distorti e l'effetto del rumore casuale viene ridotto dall'approssimazione della curva su più punti del picco. Questa tecnica è stata implementata in Matlab/Octave (pag. 71) e in spreadsheet (pag. 70). Algoritmi di rilevamento dei picchi come questo sono ampiamente applicati nella spettroscopia, nella ricerca biomedica, nel monitoraggio ambientale, nell'analisi finanziaria, nell'elaborazione delle immagini, nelle neuroscienze, nella fisica e nella scienza dei materiali, nella chimica e nella cromatografia, nel parlato e nell'elaborazione audio (riferimento 100).

Spettroscopia Derivativa (o Differenziale)

In spettroscopia, la differenziazione degli spettri è una tecnica ampiamente utilizzata, in particolare nell'infrarosso, nell'assorbimento degli u.v.-visibili, nella fluorescenza e nella spettrofotometria di riflettanza, indicata come spettroscopia derivativa. I metodi derivativi sono stati utilizzati nella spettroscopia analitica per tre scopi principali:

- (a) discriminazione spettrale, come tecnica di identificazione qualitativa per esaltare le piccole differenze tra spettri quasi identici;
- (b) miglioramento della risoluzione spettrale (peak sharpening), come tecnica per aumentare la risoluzione apparente delle bande spettrali sovrapposte al fine di determinare più facilmente il numero di bande e le loro lunghezze d'onda;
- (c) analisi quantitativa, come tecnica per la correzione dell'assorbimento di fondo irrilevante e un modo per facilitare l'analisi multicomponente. (Dato che la differenziazione è una tecnica lineare, l'ampiezza di una derivata è proporzionale all'ampiezza del segnale originale, il che consente applicazioni di analisi quantitativa che impiegano una qualsiasi delle tecniche standard di calibrazione (pagina 439). La maggior parte degli spettrofotometri commerciali sono ora dotati di funzioni derivative. Alcuni strumenti sono progettati per misurare le derivate spettrali otticamente, ovvero tramite configurazioni a doppia lunghezza d'onda o a modulazione della lunghezza d'onda.

Dato che l'ampiezza della derivata n^a di un segnale a forma di picco è inversamente proporzionale alla n^a potenza della larghezza del picco, la differenziazione è utilizzabile come metodo generale per discriminare dalle caratteristiche spettrali più larghe quelle componenti più strette. Questo è alla base dell'applicazione della differenziazione come metodo di correzione del background dei segnali nell'analisi quantitativa spettrofotometrica. Spesso nelle applicazioni pratiche di spettrofotometria all'analisi di campioni complessi, le bande spettrali dell'analita (cioè il composto da misurare) sono sovrapposte ad un background ampio, gradualmente curvato. I segnali di background di questo tipo si possono ridurre con la differenziazione.



L'idea è illustrata dalla figura a lato, che mostra uno spettro UV simulato (assorbanza in funzione della

lunghezza d'onda in nm), con la curva verde che rappresenta lo spettro dell'analita puro e la linea rossa che rappresenta lo spettro di una miscela contenente l'analita più altri composti che danno luogo all'ampio background inclinato dell'assorbimento. La derivata prima di questi due segnali appare al centro; si può vedere che la differenza tra lo spettro dell'analita puro (verde) e lo spettro della miscela (rosso) è ridotta. Tale effetto è notevolmente più evidenziato nella derivata seconda, mostrata a destra. In questo caso gli spettri dell'analita puro e della miscela sono praticamente identici. Affinché tale tecnica funzioni, è necessario che l'assorbimento in background sia più largo (cioè, abbia una curvatura più bassa) del picco dello spettro dell'analita, ma questo capita spesso. A causa della maggiore discriminazione dal background, si usano spesso le derivate di secondo grado (e talvolta anche di ordine superiore). Vedere [DerivativeDemo.m](#) per una dimostrazione Matlab/Octave.

Talvolta si dice (erroneamente) che la differenziazione "aumenta la sensibilità" dell'analisi. Si può vedere come si sarebbe tentati di dire qualcosa di simile osservando le tre figure precedenti; sembra che l'ampiezza del segnale della derivata sia maggiore (almeno graficamente) di quello del segnale dell'analita originale. Tuttavia, non è corretto confrontare le ampiezze dei segnali con le derivate perché hanno *unità di misura diverse*. Le unità sull'asse x dello spettro originale sono le *assorbanze*;

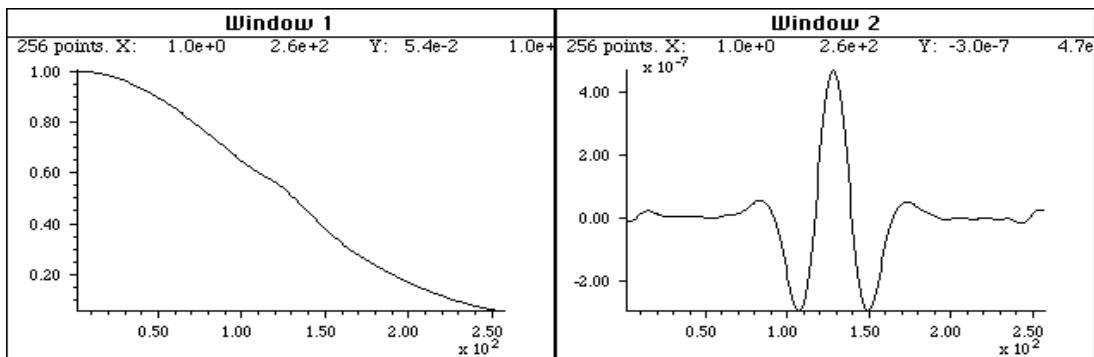
le unità della derivata prima sono *assorbanze per nm* e le unità per la derivata seconda sono *assorbanze per nm²*. Non si può confrontare l'assorbanza con l'assorbanza per nm così come non sono comparabili le miglia con le miglia orarie. (Non ha senso, per esempio, dire che 30 miglia all'ora sono maggiori di 20 miglia). È possibile, tuttavia, confrontare il *rapporto segnale-background* e il *rapporto segnale-rumore*. Per esempio, nell'esempio precedente, sarebbe corretto dire che il rapporto segnale-background è migliore (più alto) nelle derivate.

In parole povere, [l'opposto della derivazione è l'integrazione](#), quindi avendo la derivata di un segnale, ci si potrebbe aspettare di poter rigenerare il segnale originale (derivata zero-esima) per l'integrazione. Ma c'è un inghippo; il termine costante nel segnale originale (come una linea di fondo piatta) si perde completamente con la differenziazione; l'integrazione non può ripristinarla. Quindi a rigor di termini, la differenziazione rappresenta una *perdita* netta di informazione, e quindi la differenziazione si dovrebbe usare solo in situazioni in cui non interessa il termine costante del segnale originale.

Esistono diversi modi per misurare l'ampiezza della derivata di uno spettro per l'analisi chimica quantitativa: il valore assoluto della derivata ad una specifica lunghezza d'onda, in valore di una specifica caratteristica (come un massimo) o la differenza tra un massimo e un minimo. Un'altra tecnica ampiamente utilizzata è la misurazione del passaggio per lo zero, che prende le letture delle ampiezze della derivata alla lunghezza d'onda in cui un picco di interferenza attraversa lo zero sull'asse y (ampiezza). In tutti questi casi, è importante misurare i riferimenti standard e i campioni ignoti esattamente allo stesso modo. Inoltre, poiché l'ampiezza della derivata di un picco dipende fortemente dalla sua larghezza, è importante controllare i fattori ambientali che potrebbero leggermente modificare la larghezza del picco spettrale, come la temperatura.

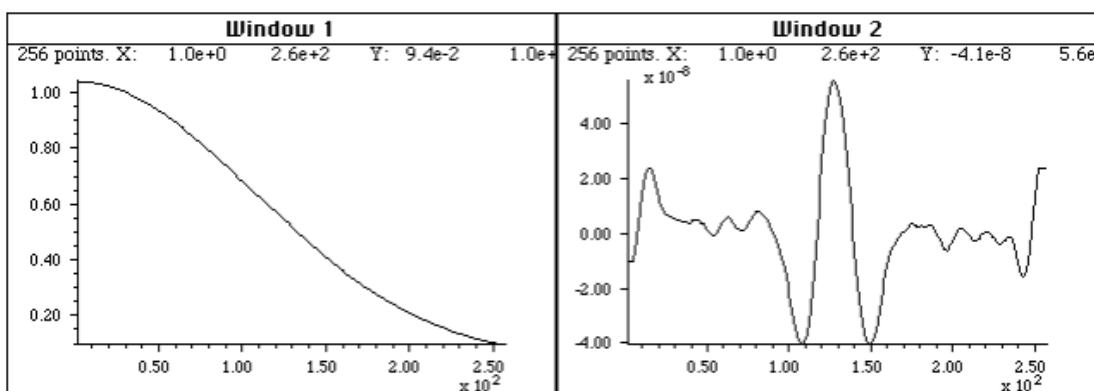
Analisi delle Tracce

Uno dei più diffusi usi della tecnica derivativa nell'elaborazione del segnale nel lavoro pratico di analisi è la misura di piccole quantità di sostanze (“tracce”) in presenza di una grande quantità di materiale potenzialmente interferente. In tali applicazioni è frequente che i segnali analitici siano deboli, rumorosi e sovrapposti ad ampi segnali in background. La precisione della misurazione è spesso degradata da variazioni della linea di base da campione a campione a causa di un assorbimento a larga banda non specifico, posizionamento non riproducibile della cuvetta (cella campione), sporco o impronte sulle pareti della cuvetta, non perfetta corrispondenza della trasmissione della cella e torbidità della soluzione. Gli scostamenti della linea di base da queste sorgenti sono solitamente o indipendenti dalla lunghezza d'onda (occlusione della luce dovuta a bolle o grosse particelle sospese) o mostrano una graduale dipendenza dalla lunghezza d'onda (torbidità da piccole particelle). Quindi la differenziazione è utile per aiutare a discriminare gli assorbimenti di interesse da queste fonti che spostano la linea di base. Un beneficio evidente della soppressione di un ampio background tramite la differenziazione è che si riducono anche le *variazioni*, da campione a campione, dell'ampiezza del background. Ciò può comportare in molti casi un miglioramento della precisione della misura, specialmente quando il segnale dell'analita è piccolo rispetto al background o se c'è un'ampia variabilità incontrollata del background. Un esempio di miglioramento della capacità di rilevare tracce di un componente in presenza di un forte background è illustrato nella figura.



Lo spettro di assorbimento a sinistra mostra una debole variazione in prossimità del centro dovuta a una piccola concentrazione della sostanza da misurare (p.es. l'ingrediente attivo di una preparazione farmaceutica). Il picco è oscurato dal forte fondo causato da altre sostanze presenti nel campione. Sulla destra è mostrata la derivata quarta di questo spettro. Il background è stato quasi totalmente soppresso e il picco dell'analita ora è evidente, facilitandone la misura.

Lo spettro a sinistra mostra una debole protuberanza vicino al centro (a $x=130$) a causa dell'analita. Il rapporto segnale/rumore in questo spettro è ottimo, ma nonostante ciò l'ampio background inclinato oscura il picco e ne rende molto difficoltosa la misura quantitativa. Sulla destra è mostrata la derivata quarta di questo spettro. Il background è stato quasi totalmente soppresso e il picco dell'analita ora è evidente, facilitandone la misura. Un caso peggiore è mostrato di seguito. Questo è essenzialmente lo stesso spettro di prima, solo che la concentrazione dell'analita è dieci volte più bassa. La domanda è: c'è una quantità rilevabile di analita in questo spettro? È quasi impossibile dirlo dal normale spettro, ma l'ispezione della derivata quarta (a destra) mostra che la risposta è sì. Qui è visivamente evidente un po' di rumore, tuttavia il rapporto segnale/rumore è sufficientemente buono per una ragionevole misura quantitativa.



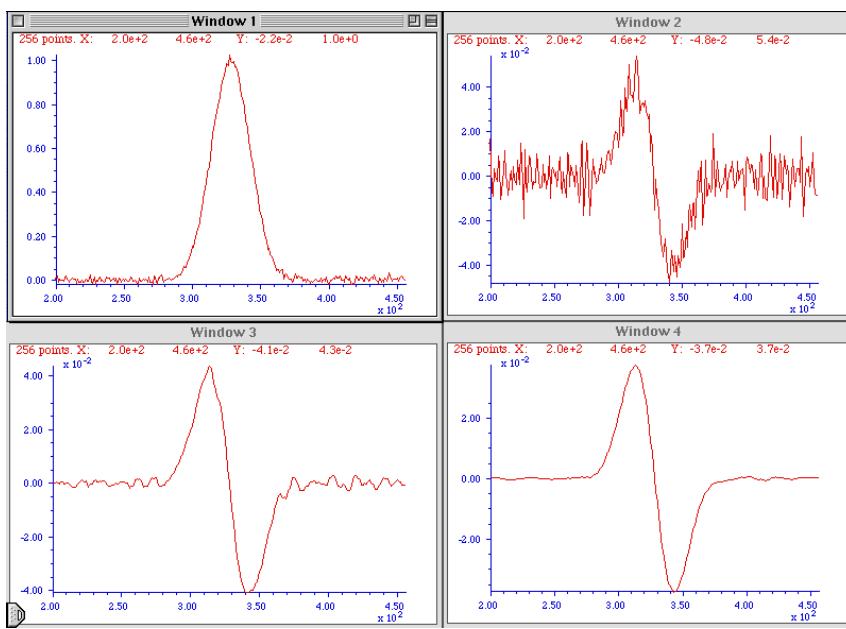
Come la figura precedente, ma in questo caso il picco è 10 volte più basso, tanto da non poter nemmeno essere visto nello spettro a sinistra. La derivata quarta (a destra) mostra che un picco c'è, ma molto ridotto in ampiezza (si noti la scala più piccola dell'asse y) e con un rapporto segnale-rumore peggiore.

La differenziazione del segnale è diventata diffusissima nella [spettroscopia quantitativa](#), in particolare per il controllo qualitativo nell'[industria farmaceutica](#). In tale applicazione l'analita potrebbe tipicamente essere l'ingrediente attivo di una preparazione farmaceutica e l'interferenza del background potrebbe derivare dalla presenza di riempitivi, emulsionanti, agenti aromatici o coloranti, tamponi, stabilizzanti o altri eccipienti. Naturalmente, nelle applicazioni di analisi delle tracce, occorre prestare attenzione ad ottimizzare il più possibile il rapporto segnale/rumore dello strumento.

Sebbene alla fine verrà dimostrato che tecniche più avanzate come il curve fitting possono eseguire abbastanza bene anche molte di queste misure (pagina 291), le tecniche derivative hanno il vantaggio concettuale della semplicità matematica e di un modo grafico di presentazione dei dati

facilmente comprensibile.

Derivate e Rumore: L'importanza dello Smoothing



Spesso si dice anche che “la differenziazione aumenta il rumore”. Questo è vero, ma non è il problema principale. Infatti, calcolando la derivata prima di un insieme di numeri casuali *se ne aumenta la deviazione standard solo per la radice quadrata di 2*, ciò è dovuto semplicemente alla solita propagazione degli errori nella somma o nella differenza di due numeri. Ad esempio, la deviazione standard (std) dei numeri

generati dalla funzione Matlab/Octave randn è 1.0 e la deviazione standard della sua derivata prima, `std(deriv1(randn(size(1:10000))))`, è pari a 1.4. Ma anche un po' di smoothing (pag. 38) applicato alla derivata ridurrà ampiamente tale deviazione standard, p.es. uno smoothing di 2-punti applicato dalla funzione fastsmooth,

`std(fastsmooth(deriv1(randn(size(1:10000))), 2, 3))`, è pari a circa 0.4. Più importante è il fatto che il *rapporto segnale/rumore* di una derivata *non filtrata* è quasi sempre molto più basso (più povero) di quello nel segnale originale, soprattutto perché *l'ampiezza numerica della derivata è solitamente molto più piccola* (come si può vedere in tutti questi esempi). Ma lo smoothing è *sempre* utilizzato in applicazioni pratiche per controllare questo problema; con uno smoothing *ottimale*, il rapporto segnale rumore di una derivata può effettivamente essere *maggior*e dell'originale non filtrato. Per l'applicazione efficace della differenziazione nell'analisi quantitativa, è essenziale utilizzare la differenziazione in combinazione con uno smoothing sufficiente, per ottimizzare il rapporto segnale-rumore. Questo è illustrato nella figura a lato. ([Codice Matlab](#) per questa figura).

La Window 1 mostra una banda Gaussiana con una piccola quantità di rumore bianco. Le "Finestre" 2, 3, e 4, mostrano la derivata prima del segnale ma con una crescente larghezza di smoothing. Come si vede, *senza uno smoothing sufficiente, i rapporto segnale/rumore della derivata può risultare sostanzialmente peggiore di quello del segnale originale*. Tuttavia, con una quantità adeguata di smoothing, il rapporto segnale/rumore della derivata filtrata è molto migliore e può essere anche visibilmente migliore di quello del segnale originale non filtrato.

Questo effetto delle derivate con smoothing è ancora più evidente nella derivata *seconda*, come mostrato a lato (il codice Matlab/Octave per questa

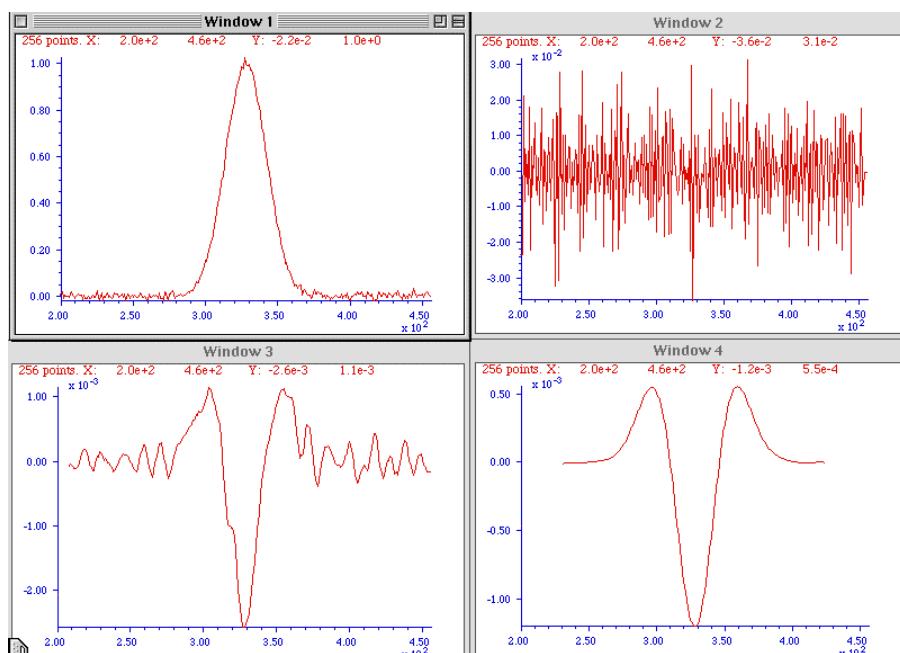


figura). In questo caso, il rapporto segnale/rumore della derivata seconda non filtrata (Window 2) è talmente pessimo che non è possibile nemmeno visivamente percepire il segnale, ma con lo smoothing la derivata seconda sembra a posto. *La differenziazione non aggiunge effettivamente rumore al segnale*; se non ci fosse rumore nel segnale originale, anche le derivate non avrebbero rumore (eccezione: cfr. pagina 332).

Ciò che è particolarmente interessante del rumore nelle derivate dei segnali, tuttavia, è il "colore". Questo rumore non è *bianco*; ma è *blu* - cioè ha molta più potenza alle *alte* frequenze del rumore bianco. La conseguenza di ciò è che il rumore nel segnale differenziato viene facilmente e notevolmente ridotto mediante lo *smoothing*, come mostrato.

Dato che lo smoothing a slittamento della media e la differenziazione sono entrambe operazioni lineari, non fa differenza se l'operazione di smoothing viene applicata prima o dopo la differenziazione. Ciò che però è importante è la natura dello smoothing, il suo rapporto di smoothing (rapporto tra la larghezza dello smoothing e la larghezza del picco originale), ed il numero di volte cui si applica lo smoothing al segnale. Il valore ottimale del rapporto di smoothing per i segnali derivati vanno approssimativamente da 0.5 a 1.0. Per una derivata prima, bastano *due* applicazioni di un semplice smoothing rettangolare (o una applicazione di quello triangolare). Per una derivata seconda, sono sufficienti *tre* applicazioni dello smoothing rettangolare o due applicazioni di quello triangolare. La regola generale è: per la derivata n^a , usare uno smoothing che sia l'equivalente almeno a $n+1$ applicazioni di uno rettangolare. Il [metodo Savitzky-Golay](#) è ideale per il calcolo di derivate con smoothing perché combina la differenziazione con il giusto tipo di smoothing. Il programma Matlab signal processing [*iSignal*](#), discusso a pagina 366, usa questo approccio.

Se le larghezze dei picchi variano molto nella registrazione del segnale - per esempio, se i picchi si allargano regolarmente all'aumentare delle x - può essere utile utilizzare uno smoothing adattativo segmentato (pag. 326), che varia la larghezza dello smoothing nel segnale.

Lo smoothing delle derivate dei segnali solitamente si traduce in una sostanziale attenuazione dell'amplificazione della derivata; a destra nella figura precedente, l'ampiezza della derivata più filtrata (Window 4) è molto inferiore di quella meno filtrata (Window 3). Tuttavia, questo non sarà un problema nelle applicazioni di *analisi quantitativa*, a condizione che la curva standard (analitica) venga preparata utilizzando la stessa identica procedura di derivazione, smoothing e misura del campione ignoto. Dato che la differenziazione e lo smoothing sono entrambe [tecniche lineari](#), l'ampiezza di una derivata filtrata con smoothing è esattamente proporzionale all'ampiezza del segnale originale, il che consente alle applicazioni di analisi quantitative di impiegare tutte le tecniche di calibrazione standard (pagina 439). A patto di applicare le *stesse* tecniche di elaborazione del segnale agli standard e ai campioni, tutto funziona.

A causa dei diversi tipi e gradi di smoothing che si possono includere nel calcolo della differenziazione digitale di segnali sperimentali, è difficile confrontare i risultati dei diversi strumenti ed esperimenti a meno che non siano noti i dettagli di tali calcoli. Nella strumentazione commerciale e nei pacchetti software, tali dettagli possono essere nascosti. Tuttavia, se si possono ottenere sia il segnale originale (derivata zero-esima), che la derivata e/o la versione filtrata con smoothing del segnale dallo stesso strumento o dal pacchetto software, allora si potrà usare la tecnica della deconvoluzione di Fourier, che verrà discussa in seguito, per scoprire e duplicare i calcoli ignoti.

È interessante notare che il fallimento dello smoothing di una derivata ha provocato alla fine l'incidente della primo veicolo spaziale del [programma Mariner della NASA](#) il 22 luglio del 1962, riportato tra "gli 11 peggiori bug software" da InfoWorld. Nel suo libro del 1968 "[The Promise of](#)

Space", Arthur C. Clarke ha descritto la missione come "distrutta dal trattino più costoso della storia". Il "trattino" era in effetti la barretta in apice, sopra il simbolo della velocità (la derivata prima della posizione), scritto a mano in un taccuino. La barretta superiore indica una funzione di *media* o uno *smoothing*, per cui nella formula si sarebbe dovuto calcolare lo *smoothing* della derivata della posizione rispetto al tempo. Senza la funzione di smoothing, anche delle piccolissime variazioni renderebbero molto rumorosa la derivata attivando prematuramente i booster per la correzione, rendendo instabile il volo del razzo. Ad essere onesti, nei confronti di quegli ingegneri pionieristici, era abbastanza presto nella storia dell'esplorazione spaziale.

Dimostrazioni Video

Il primo video di 13 secondi, 1.5 MByte ([SmoothDerivative2.wmv](#)) mostra gli enormi miglioramenti del rapporto segnale-rumore che sono possibili quando si applica lo smoothing alle derivate dei segnali, in questo caso, una derivata quarta.

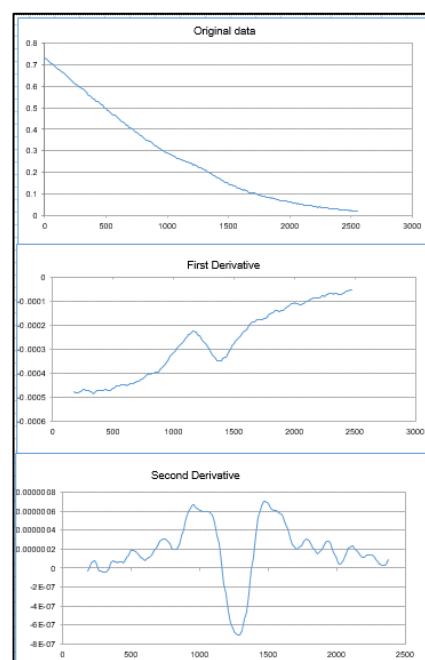
Il secondo video di 17 secondi, 1.1 MByte, ([DerivativeBackground2.wmv](#)) mostra la misura di un debole picco sepolto in un background molto inclinato. All'inizio di questo breve video, l'altezza (Amp) del picco viene regolata tra 0 e 0.14, ma il background è così forte che i cambiamenti di amplificazione del picco, localizzato a $x = 500$, sono difficili da vedere. Viene poi calcolata la derivata quarta (Order=4) e viene espansa la scala (Scale), con un'ampiezza dello smoothing (Smooth) di 88. Infine, l'ampiezza (Amp) del picco viene nuovamente variata sulla stessa gamma, ma ora i cambiamenti nel segnale sono abbastanza evidenti e facilmente misurabili.

La differenziazione dei segnali analogici si può eseguire con un semplice [circuito di amplificatore operazionale](#); si possono collegare in cascata due o più di questi circuiti per ottenere la derivata seconda o quelle di ordine superiore. Gli stessi problemi di rumore sopra descritti si applicano anche alla differenziazione analogica, richiedendo l'uso di circuiti di filtraggio passa-basso analoghi allo smoothing.

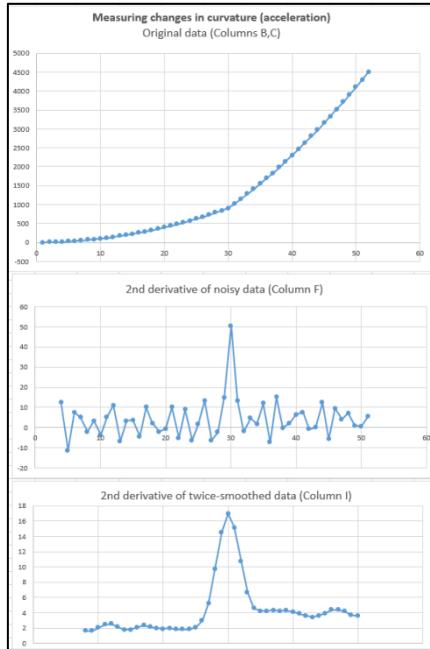
Differenziazione nei fogli di calcolo

Le operazioni di differenziazione come quelle descritte si possono facilmente eseguire nei fogli di calcolo come Excel o Calc di OpenOffice. Sia le derivate che le operazioni di smoothing si possono eseguire col metodo trasla-e-moltiplica descritto nel capitolo sullo smoothing (pagina 38). In linea di principio, è possibile combinare qualsiasi grado di differenziazione e di smoothing in un set di coefficienti per il trasla-e-moltiplica (come illustrato qui), ma è più flessibile e la regolazione è più facile se si calcolano separatamente la derivata e lo smoothing in colonne successive. Ciò è illustrato da [DerivativeSmoothing.ods](#) per OpenOffice Calc e da [DerivativeSmoothing.xls](#) per Excel, che eseguono lo smoothing dei dati e calcolano la derivata prima di Y (colonna **B**) rispetto a X (colonna **A**), poi applicano lo smoothing e la differenziazione successivamente per calcolare la derivata seconda e la terza filtrate.

Gli stessi coefficienti per lo smoothing (nella riga 5, colonne da **K** ad **AA**) vengono applicati successivamente per ciascuno stadio della differenziazione; qui si può inserire qualsiasi set di numeri (preferibilmente simmetrico rispetto al numero centrale nella colonna **S**). I dati si possono digitare o incollare nelle colonne **A** e **B** (X e Y), e nelle righe dalla 8 alla 263.



[DerivativeSmoothingWithNoise.xlsx](#) (a destra) mostra l'effetto dello smoothing del rapporto segnale-rumore delle derivate di un debole picco situato a $x = 1200$ su una linea di pbase inclinata.



Utilizza gli stessi dati di [DerivativeSmoothing.xls](#) ma aggiunge del rumore bianco simulato ai dati Y. È possibile controllare la quantità di rumore aggiunto (cella D5).

Un altro esempio di applicazione della derivata è nello spreadsheet [SecondDerivativeXY2.xlsx](#) (a sinistra), che mostra come localizzare e misurare i cambiamenti nella derivata seconda (una misura della curvatura o accelerazione) di un segnale nel tempo. Questo spreadsheet mostra l'apparente aumento del rumore causato dalla differenziazione e la misura in cui il rumore può essere ridotto con lo smoothing (in questo caso da due passaggi di uno smoothing triangolare di 5 punti). La derivata seconda filtrata con lo smoothing mostra un ampio picco nel punto in cui cambia l'accelerazione (in $x=30$), ed è chiaro che la linea di base su entrambi i lati del picco è distintamente diversa, mostrando la variazione dell'accelerazione

prima e dopo il picco ($y=2$ e 4 , rispettivamente). Altri esempi di differenziazione e filtraggio tramite convoluzione "shift-and-multiply" includono [MultipleConvolutionFirstDerivativeDemo.xls](#) e [MultipleConvolution4thDerivativeDemo.xls](#)

Differenziazione in Matlab e in Python

Le funzioni di differenziazione per differenze finite come quelle descritte si possono facilmente creare in Matlab o in Octave. Alcune semplici funzioni per le derivate di dati temporali equidistanti: [deriv](#), una derivata prima col metodo della differenza centrale di 2-punti, [deriv1](#), una derivata prima senza smoothing che usa le differenze tra punti adiacenti, [deriv2](#), una derivata seconda col metodo della differenza centrale di 3-punti, una derivata terza [deriv3](#) che usa la formula con 4-punti e [deriv4](#), per le derivate quarte con la formula a 5-punti. Ognuna è una semplice funzione Matlab del tipo **d=deriv(y)**; l'argomento di input è il vettore del segnale "**y**", e il segnale differenziato viene restituito nel vettore "**d**". Per i dati *non* equidistanti della variabile indipendente sull'asse (x), ci sono versioni delle funzioni per la derivata prima e seconda, [derivxy](#) e [seccderivxy](#), che prendono due argomenti (x,y), dove **x** e **y** sono i vettori contenenti le variabili indipendenti e dipendenti.

[SmoothDerivative.m](#) combina la differenziazione con lo smoothing. La sintassi è **SmoothedDeriv = SmoothedDerivative(x,y,DerivativeOrder,w,type,ends)** dove 'DerivativeOrder' determina l'ordine della derivata (da 0 a 5), 'w' è l'ampiezza dello smoothing, 'type' è il tipo di smoothing:

Se type=0, il segnale non subisce lo smoothing

Se type=1, è rettangolare (slittamento della media o boxcar)

Se type=2, è triangolare (2 passaggi della boxcar)

Se type=3, p-spline (3 passaggi della boxcar)

Se type=4, si applica lo smoothing di Savitzky-Golay

'ends' controlla come vengono gestite le "estremità" del segnale (i primi $w/2$ punti e gli ultimi $w/2$ punti): Se ends=0, le estremità vengono azzerate: Se ends=1, le estremità subiscono uno smoothing progressivamente più piccolo avvicinandosi alla fine. Digitare "help SmoothDerivative" per degli

esempi ([grafico](#)). Un metodo di differenziazione alternativo basato sulla Trasformata di Fourier (pagina 87) può calcolare derivate di qualsiasi ordine e include anche lo smoothing (riferimento 88).

Rilevamento dei picchi. Il codice più semplice per trovare i picchi in un insieme di dati x,y cerca semplicemente ogni valore y che abbia valori adiacenti y inferiori su entrambi i lati ([allpeaks.m](#)). Un approccio alternativo consiste nell'usare la derivata prima per trovare tutti i massimi individuando i punti di passaggio per lo zero, ovvero, i punti in cui la derivata prima "d" (calcolata da [derivxy.m](#)) passa da positivo a negativo. In questo esempio, la funzione "sign" è una funzione nativa che restituisce 1 se l'elemento è maggiore di zero, 0 se è uguale a zero e -1 se è minore di zero. La routine stampa il valore di x e di y ad ogni attraversamento dello zero:

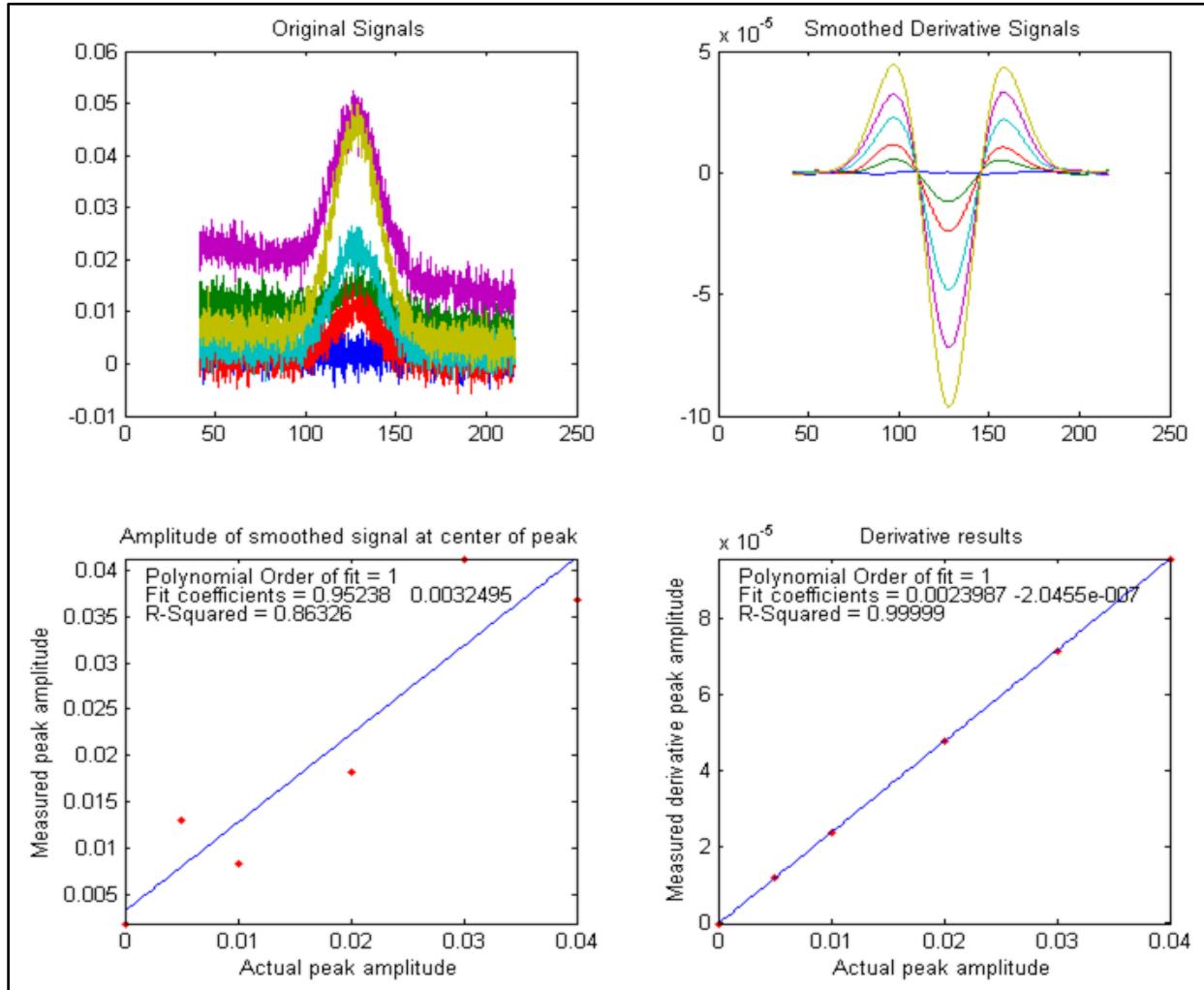
```
d=derivxy(x,y);  
for j=1:length(x)-1  
if sign(d(j))>sign(d(j+1))  
disp([x(j) y(j)])  
end  
end
```

Se i dati sono rumorosi, ci saranno molti attraversamenti fasulli, che però verranno ridotti con lo smoothing dei dati. Se i dati sono scarsamente campionati, è possibile ottenere un valore più accurato della posizione del picco (valore sull'asse x nell'attraversamento dello zero) interpolando tra il punto precedente e quello successivo l'attraversamento dello zero, utilizzando la funzione Matlab/Octave "interp1" o "spline":

```
interp1([d(j) d(j+1)], [x(j) x(j+1)], 0)
```

In Python, si può importare una [funzione derivativa](#) con "`from scipy.misc import derivative`".

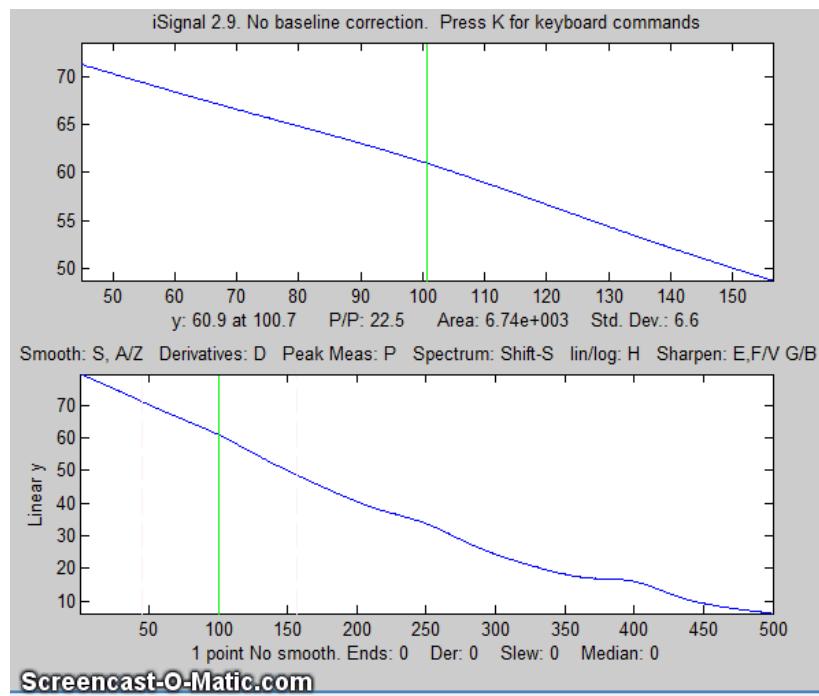
ProcessSignal.m è una funzione Matlab/Octave a riga di comando che esegue lo smoothing e la derivazione su una serie temporale x,y (vettori di colonna o righe). Digitare "help ProcessSignal". Restituisce il segnale processato come vettore che ha la stessa forma di x, indipendentemente dalla forma di y. La sintassi è **Processed = ProcessSignal(x, y, DerivativeMode, w, type, ends, Sharpen, factor1, factor2, Symsize, Symfactor, SlewRate, MedianWidth)**



DerivativeDemo.m (sotto) è una funzione demo Matlab/Octave autonoma che utilizza

[ProcessSignal.m](#) e [plotit.m](#) per dimostrare un'applicazione della differenziazione al metodo dell'analisi quantitativa di un picco sepolto in un background instabile (ad esempio come in varie forme di spettroscopia). Lo scopo è quello di ricavare la misura dell'altezza del picco che varia linearmente con quella effettiva del picco ed è minimamente influenzata dal background e dal rumore. Per avviarlo, basta digitare **DerivativeDemo** al prompt dei comandi. Si possono modificare diverse variabili interne (p.es. Noise, BackgroundAmplitude) per rendere la misura più facile o più ardua. Si noti che, nonostante il fatto che l'ampiezza della derivata sembri essere numericamente inferiore al segnale originale (perché ha unità diverse), il rapporto segnale/rumore della derivata è migliore di quello dei segnali originali ed è molto meno influenzato dall'instabilità del background.

iSignal.m (pag. 366), mostrato a lato) è una funzione interattiva per Matlab che esegue molte delle



operazioni di signal-processing descritte, compresa la differenziazione e lo smoothing per segnali temporali, fino alla derivata 5^a, includendo automaticamente il tipo di smoothing richiesto. Delle semplici sequenze di tasti consentono di regolare i parametri dello smoothing (il tipo, l'ampiezza e la gestione delle estremità) mentre se ne osserva dinamicamente l'effetto sul segnale. Nella [GIF animata di esempio](#) mostrata qui, una serie di tre picchi $x=100, 250$ e 400 , con altezze nel rapporto 1:2:3, sono

sepolti in un forte background curvo; vengono calcolate le derivate seconda e quarta con smoothing per sopprimere il background. Il codice è visualizzabile qui e si può scaricare il [file ZIP](#) con i dati dell'esempio. L'interattività dei tasti funziona anche se si esegue [Matlab in un browser web](#), ma non su [Matlab Mobile](#) né in Octave. (Nota: figure come quella precedente che mostrano la scritta "Screencast-O-Matic" in basso a sinistra sono grafici *animati* visualizzabili in un browser web o in [Microsoft Word 365](#) ma non vengono animate nei visualizzatori PDF testati).

Come esempio di smoothing in iSignal, le istruzioni seguenti generano la derivata 4^a di un picco Gaussiano rumoroso e lo mostrano in **iSignal**. Sarà prima necessario effettuare il download di [isignal.m](#), [gaussian.m](#) e [deriv4.m](#).

```
>> x=[1:.1:300]';  
>> y=deriv4(100000.*gaussian(x,150,50)+.1*randn(size(x)));  
>> isignal(x,y);
```

Il segnale è soprattutto rumore blu (a causa della differenziazione del rumore bianco) a meno di non attenuarlo in modo considerevole. Si usano i tasti **A** e **Z** per aumentare e diminuire l'ampiezza dello smoothing e il tasto **S** per scorrere tra i diversi tipi di smoothing. Suggerimento: usare lo smoothing P-spline ed aumentare l'ampiezza dello smoothing.

La *differenziazione in tempo reale* in Matlab è trattata a pagina 339.

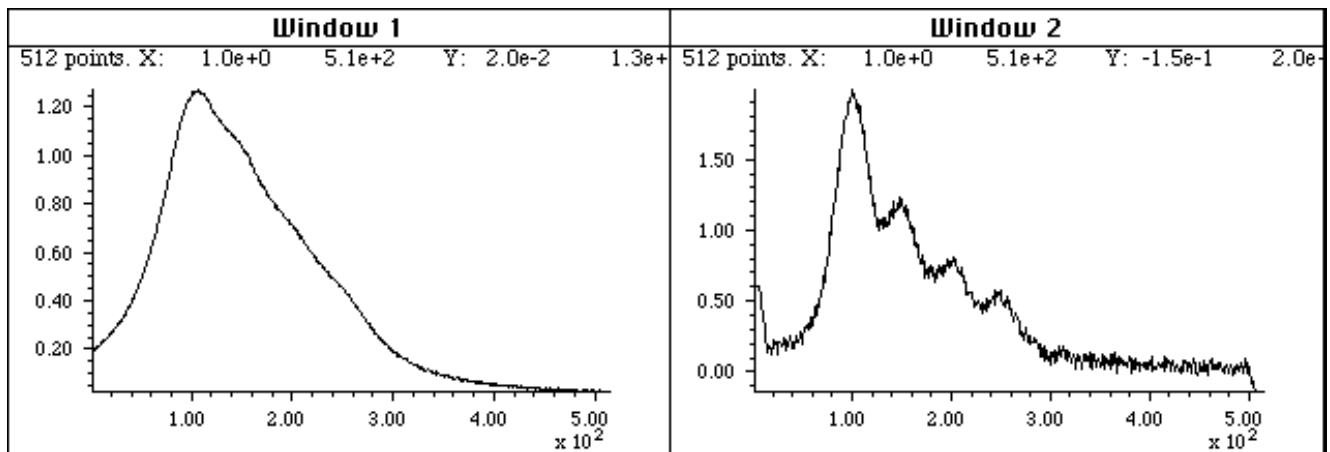
Live Script per la Differenziazione

[DataDifferentiationmlx](#) ([grafico](#)) è un Live Script per la differenziazione e lo smoothing applicato ai dati sperimentali archiviati su disco. È simile a [DataSmoothingmlx](#) discusso a pagina 55, con l'aggiunta di uno slider (riga 9) per selezionare l'ordine di derivazione (fino a 10). Nota: Se la casella di controllo "PlotBeforeAndAfter" è selezionata, la derivata (curva rossa) verrà scalata per corrispondere al massimo del segnale originale (curva nera). Se questa casella non è selezionata, la derivata verrà visualizzata da sola con la sua ampiezza *effettiva*. (Ciò avviene perché le ampiezze

numeriche delle derivate sono spesso di diversi ordini di grandezza rispetto ai segnali originali).

Peak Sharpening

La figura seguente mostra uno spettro a sinistra costituito da diverse bande o picchi scarsamente risolti (cioè parzialmente sovrapposti). L'ampia sovrapposizione delle bande rende impossibile la misura accurata delle loro intensità e posizioni dei picchi, anche se il rapporto segnale/rumore è ottimo. Sarebbe più facile misurare accuratamente la posizione dei picchi se fossero risolti in modo più completo, cioè se i picchi fossero più stretti.



Un algoritmo di "peak sharpening" applicato al segnale a sinistra migliora artificialmente la risoluzione apparente dei picchi. Nel segnale risultante, a destra, si possono misurare più accuratamente le intensità e le posizioni dei picchi, ma a costo di una diminuzione del rapporto segnale/rumore.

Sharpening con derivata pari

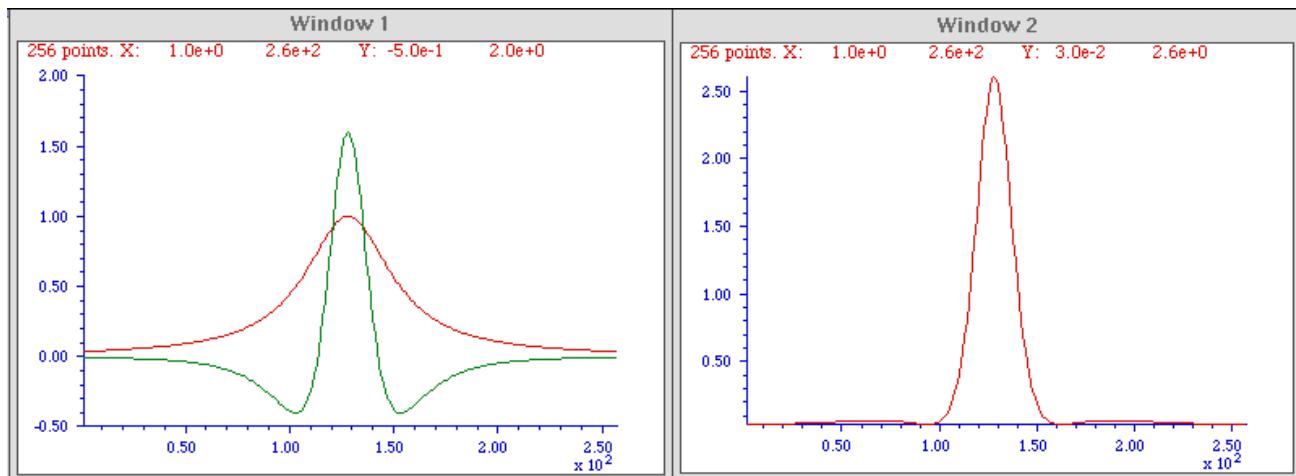
La tecnica utilizzata qui, chiamata "*peak sharpening*" o *incremento della risoluzione*, usa degli algoritmi per migliorare artificialmente la risoluzione apparente dei picchi. Uno dei più semplici algoritmi di questo tipo calcola la somma ponderata del segnale originale e il negativo della sua derivata seconda:

$$R_j = Y_j - k_2 Y''$$

dove R_j è il segnale con risoluzione migliorata, Y è il segnale originale, Y'' è la derivata seconda di Y e k_2 è un fattore di ponderazione, selezionato dall'utente, per la derivata 2^{a} . Spetta all'utente la selezione di un fattore di ponderazione k_2 che offra il miglior compromesso tra la nitidezza (sharpening), la degradazione del segnale/rumore e l'appiattimento della linea di base. La scelta migliore dipende dalla larghezza, forma e dall'intervallo di digitalizzazione del segnale.

Inevitabilmente, il rapporto segnale/rumore si degrada, ma questo si può moderare con lo smoothing (pagina 38), ma a scapito della riduzione dello [sharpening]. Tuttavia, questa tecnica sarà *utile solo se la sovrapposizione dei picchi è un fattore limitante rispetto al rapporto segnale/rumore*.

Ecco come funziona. La figura mostra, in Window 1, un picco generato al computer (con un profilo Lorentziano) in rosso, sovrapposto al *negativo* della sua derivata seconda in verde).



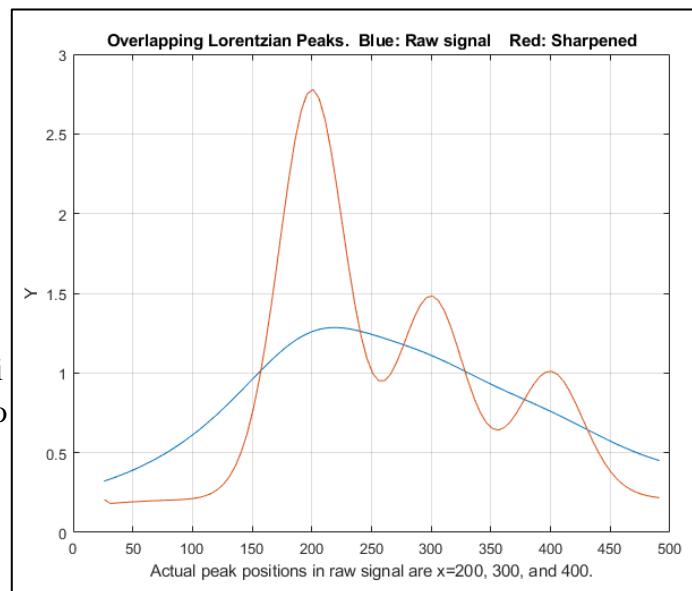
La derivata seconda è amplificata (moltiplicandola per una costante modificabile) in modo che i bordi negativi della derivata seconda invertita (all'incirca da $X = 0$ a 100 e da $X = 150$ a 250) siano un'immagine speculare dei lati del picco originale in tali regioni. In questo modo, quando il picco originale viene aggiunto alla derivata seconda invertita, i due segnali *quasi* si annulleranno nelle due regioni laterali ma si rafforzeranno reciprocamente in quella centrale (da $X = 100$ a 150). Il risultato, mostrato nella Window 2, è una sostanziale riduzione (50% circa) della larghezza, ed un corrispondente aumento dell'altezza, del picco. Questo effetto è ancora più evidente con picchi di forma Lorentziana; con i picchi Gaussiani, l'aumento della risoluzione è meno accentuato (solo il 20 - 30% circa).

Le ridotte larghezze dei picchi con sharpening rendono più facile distinguere i picchi sovrapposti. Nell'esempio a destra, il segnale originale sintetizzato al computer (riga blu) è la somma di tre picchi Lorentziani sovrapposti a $x=200$, 300 e 400. I picchi sono molto larghi; le loro semi-larghezze sono 200, che è maggiore della loro separazione. Il risultato è che i picchi si sovrappongono molto nei dati originali, formando quello che *sembra un unico grande picco asimmetrico* (riga blu) con un massimo a $x=220$. Tuttavia, il risultato dell'algoritmo di "sharpening con derivata pari" (linea rossa) mostra i picchi

sottostanti *nelle loro posizioni corrette*. La linea di base, tuttavia, che originariamente era zero lontano dal centro del picco, è stata spostata, come si può vedere da $x=25$ a 100.

Si noti che la riga di base in entrambi i lati del picco migliorato, non è molto piatto, specie per un picco Lorentziano, perché la cancellazione tra il picco originale e la derivata seconda invertita è solo approssimativa; il fattore di ponderazione k viene regolato per minimizzare questo effetto. Lo sharpening avrà un effetto minimo, se non nessuno, sulla linea di base, perché se questa è lineare, la sua derivata sarà zero e se è gradualmente curva, la sua derivata seconda sarà molto piccola rispetto a quella del picco.

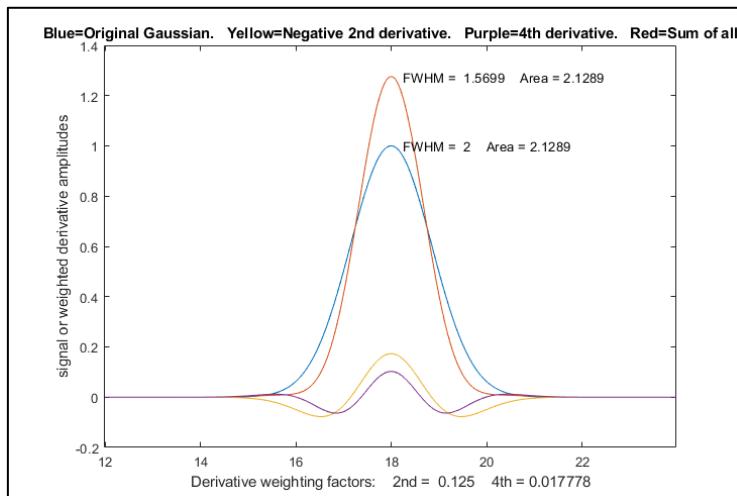
Questa tecnica è stata utilizzata in varie forme di spettroscopia e cromatografia per molti anni (riferimenti 74-76) e, in qualche caso, utilizzando l'elettronica analogica. Matematicamente, questa tecnica è una versione semplificata di un'espansione convergente delle serie di Taylor, in cui vengono presi solo i termini delle derivate pari nell'espansione e per i quali i coefficienti sono di segno alterno. L'esempio precedente è la versione più semplice possibile che include solo i primi due termini - il picco originale e la sua derivata seconda negativa. Risultati leggermente migliori si



possono ottenere aggiungendo un quarto termine derivato, con due fattori regolabili k2 e k4:

$$Rj = Yj - k2*Y'' + k4*Y'''$$

dove Y'' e Y''' sono le derivate 2^a e 4^a di Y . Il risultato è una riduzione del 21% in larghezza per un picco Gaussiano, come mostrato in figura ([script Matlab/Octave](#)) e una riduzione del 60% per un picco Lorentziano ([script](#)). Questo algoritmo è stato utilizzato nell'esempio precedente del picco sovrapposto. (È possibile aggiungere un sesto termine derivativo, ma la serie converge rapidamente e i risultati migliorano di poco, a costo della maggiore complessità per avere tre fattori da regolare).



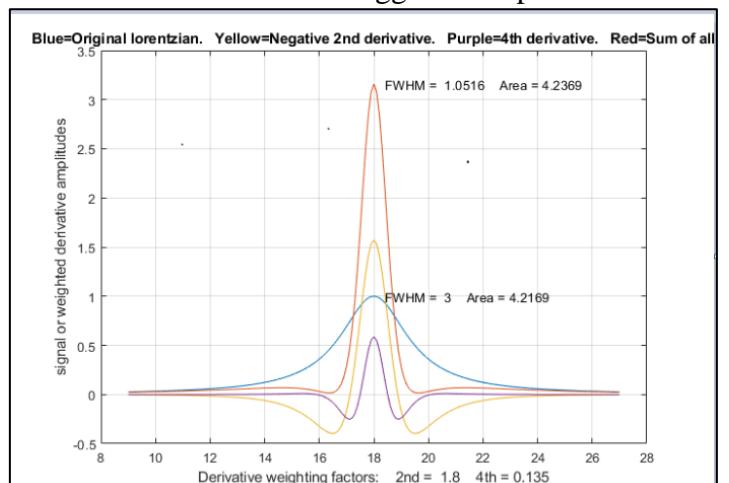
Non esiste un valore ottimale universale per i fattori di ponderazione delle derivate; dipende da quello che si considera il miglior compromesso tra lo sharpening e l'appiattimento della linea di base. Tuttavia, un buon punto di partenza per un picco Gaussiano è $k2 = W^2/32$ e $k4 = W^4/900$, dove W è la semi-larghezza del picco in unità x (p.es. il tempo in cromatografia). Con questi fattori, un picco Gaussiano verrà ridotto in larghezza del 21%, la linea di base

sarà ancora visivamente piatta e il picco risultante si approssimerà al modello Gaussiano con un errore di approssimazione inferiore allo 0.3% e un R^2 di 0.9999. Valori maggiori di k porteranno a picchi più stretti, ma la linea di base non sarà così piatta. Per un picco originale Lorentziano (a destra), con $k2=W^3/3$ e $k4 = W^4/600$, l'ampiezza del picco viene ridotta di un fattore 3, ma il picco risultante si approssima a un modello Gaussiano con un errore di adattamento percentuale maggiore dell'1.15% e un R^2 di 0.9966. Si noti che i fattori k per le derivate seconda e quarta variano con la larghezza W elevata alla 2^a e alla 4^a potenza rispettivamente, quindi possono variare su un'ampia gamma

numerica per picchi di larghezze diverse. Per questo motivo, se le larghezze dei picchi variano in modo sostanziale nel segnale, è utile utilizzare le versioni *segmentate* e *gradiente* di questo metodo, come fatto in precedenza per lo smoothing, (pag. 52), in modo che la nitidezza [sharpening] possa essere ottimizzata per ciascuna regione del segnale (si veda di seguito). “Segmentato” significa che ogni segmento di segnale è definito in modo indipendente; “gradiente” indica un aumento o una diminuzione graduale tra i valori iniziali e finali specificati.

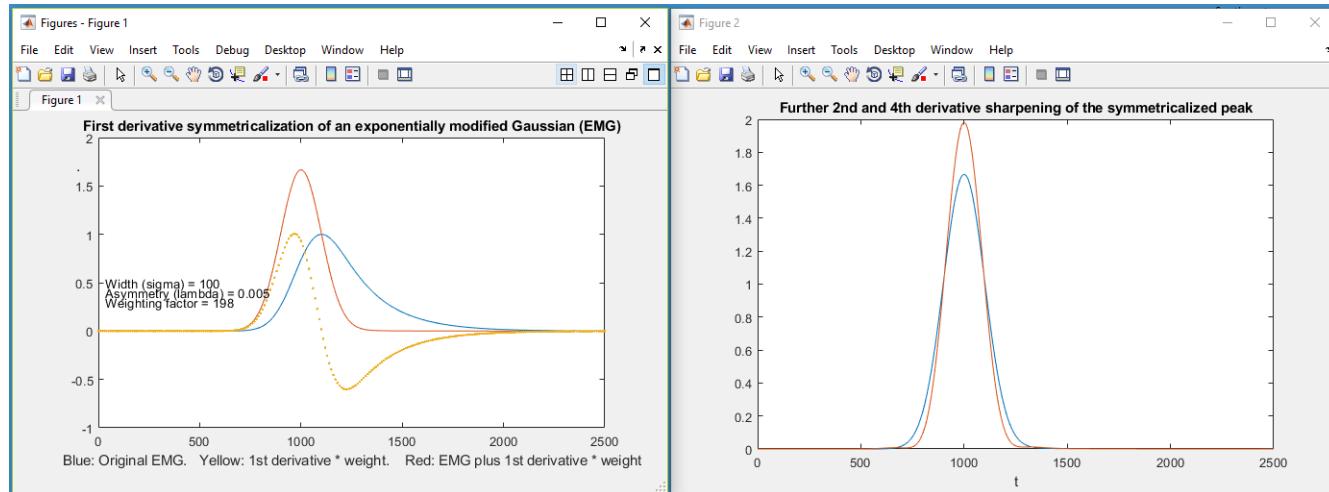
Simmetrizzazione della derivata prima con area costante (“detailing”)

Se il picco è *asimmetrico* - cioè, scende più rapidamente su un lato rispetto all'altro - allora l'addizione ponderata (o la sottrazione) di una *derivata prima*, Y' , può essere utile, perché la derivata prima del picco è *anti-simmetrica* (positiva da un lato e negativa dall'altro). Nell'esempio grafico precedente, il picco asimmetrico (in blu) si allunga a destra e la sua derivata prima, Y' , (punteggiata



in giallo) ha un lobo positivo a sinistra e uno più largo ma più piccolo a destra. Quando al picco si aggiunge la derivata ponderata, il *lolo positivo della derivata rinforza il bordo anteriore* e il *lolo negativo sopprime il lato esteso*, con conseguente miglioramento della simmetria. (Se la EMG (Gaussiana Modificata Esponenzialmente) fosse inclinata a *sinistra*, verrebbe aggiunto il *negativo* della sua derivata). Anche questa è una vecchia tecnica, essendo stata utilizzata in cromatografia almeno dal 1965 (riferimento 75, 76), dove è stata chiamata “de-tailing”.

$$S_j = Y_j + k_1 Y'$$



In effetti, è possibile dimostrare che questa semplice tecnica funziona perfettamente per *picchi ampliati esponenzialmente di qualsiasi forma*, come la "[Gaussiana Modificata Esponenzialmente](#)" ([EMG](#)) mostrata qui (riferimento 73).

Col giusto fattore di ponderazione della derivata prima, k_1 , il risultato è una Gaussiana asimmetrica con una [semi-larghezza](#) sostanzialmente inferiore a quella dell'originale (linea arancione); infatti, è esattamente la Gaussiana sottostante alla quale è stata applicata la convoluzione esponenziale (Riferimenti 70, 71).

Il fattore di ponderazione della derivata prima k_1 è indipendente dall'altezza e dalla larghezza del picco ed è semplicemente uguale alla costante di tempo esponenziale τ ($1/\lambda$, in alcune formulazioni della EMG). Funziona perfettamente se il τ del picco è lo stesso. In pratica, k_1 deve essere determinato sperimentalmente, il che è più facile da fare per l'ultimo picco in un gruppo di picchi ([grafico](#), [animazione](#)). In parole povere, se si ha k_1 troppo alto, il risultato scenderà al di sotto della linea di base dopo il picco.

È facile determinare sperimentalmente il valore ottimale di k_1 ; basta aumentarlo fino a quando il segnale elaborato S_j scende al di sotto della linea di base dopo il picco, quindi ridurlo finché la linea di base non si appiattisce, come mostrato [nell'animazione GIF a questo link](#). Naturalmente, nell'applicazione reale il segnale conterrà rumore, ne seguirà che il risultato simmetrizzato sarà più rumoroso del segnale originale. Il fattore di ponderazione della derivata prima, k_1 , dovrà essere stimato a occhio ed è quindi soggetto a qualche incertezza.

Se uno stadio dell'addizione derivativa non risolve il problema, provare una delle routine doppio-esponenziali descritte di seguito. Inoltre, questo sembra essere un comportamento generale e funziona allo stesso modo per qualsiasi altra forma di picco allargata dalla convoluzione esponenziale, come una Lorentziana e funziona anche per i picchi che sono già stati ampliati da una precedente convoluzione esponenziale (cioè un doppio esponenziale), gestibile con due fasi successive dell'addizione delle derivate con diversi τ .

Il picco simmetrizzato S_j risultante dalla procedura dell'addizione con la derivata prima può ulteriormente subire uno sharpening con le tecniche delle derivate pari descritte, supponendo che il rapporto segnale/rumore dell'originale sia abbastanza buono.

Un'utile proprietà di tutti questi algoritmi di addizione derivativa è che non cambiano l'area *totale* dei picchi perché l'area *totale* della curva di *ogni* derivata di un segnale con picchi che torna alla linea di base è essenzialmente *zero* (l'area sotto i lobi negativi cancella l'area sotto i lobi positivi). Pertanto, queste tecniche possono risultare utili nella misura delle aree di picchi sovrapposti (pagina 126).

Tuttavia, un altro problema è che la linea di base su entrambi i lati del picco evidenziato con lo sharpening può *non essere perfettamente piatta*, lasciando delle interferenze dai picchi vicini, anche se si ottiene la risoluzione della linea di base dei picchi adiacenti. Per la tecnica delle derivate pari applicata ad un picco Gaussiano, circa il 99.7% ([link al grafico](#)) dell'area del picco è contenuta nel massimo centrale e per un picco Lorentziano, circa l'80% dell'area del picco ([link al grafico](#)) è contenuta nel massimo centrale.

Poiché la differenziazione e lo smoothing sono entrambe [tecniche lineari](#), si applica il [principio di sovrapposizione](#) e l'ampiezza di un segnale simmetrizzato o con sharpening è direttamente proporzionale all'ampiezza del segnale originale, consentendo l'applicazione di analisi quantitative impiegando qualsiasi delle tecniche di calibrazione standard (pagina 434). Ma è *essenziale* applicare le *stesse tecniche di elaborazione del segnale sia agli standard che ai campioni* e misurare i segnali allo stesso modo.

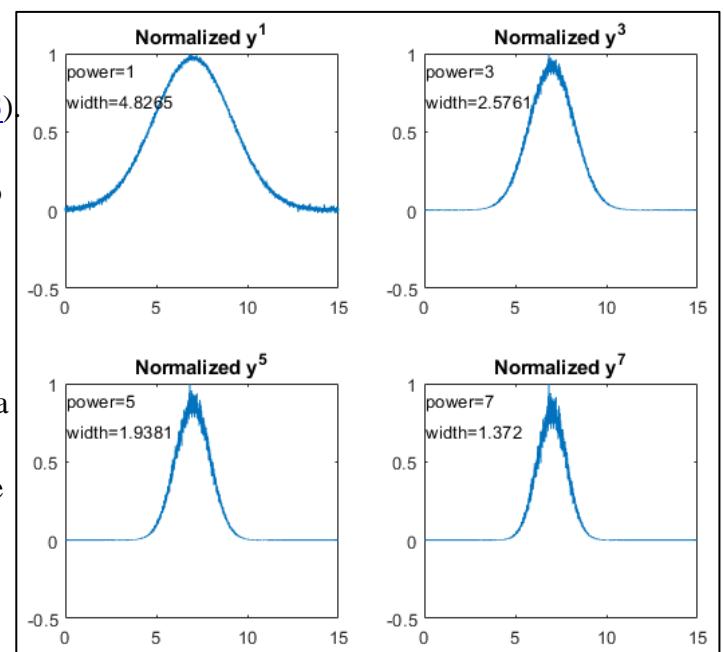
Lo sharpening dei picchi è utile nel rilevamento e nella misura automatica dei picchi (pagina 227) per aumentare la capacità di rilevare i deboli picchi sovrapposti che appaiono solo come sporgenze nel segnale originale. Se si sta leggendo online, click per un [esempio animato](#). Lo sharpening dei picchi è utile anche prima di [misurare le aree](#) (pag. 138) di picchi sovrapposti, perché è più facile e più preciso misurare le aree di picchi completamente separati.

Il metodo della Legge della Potenza

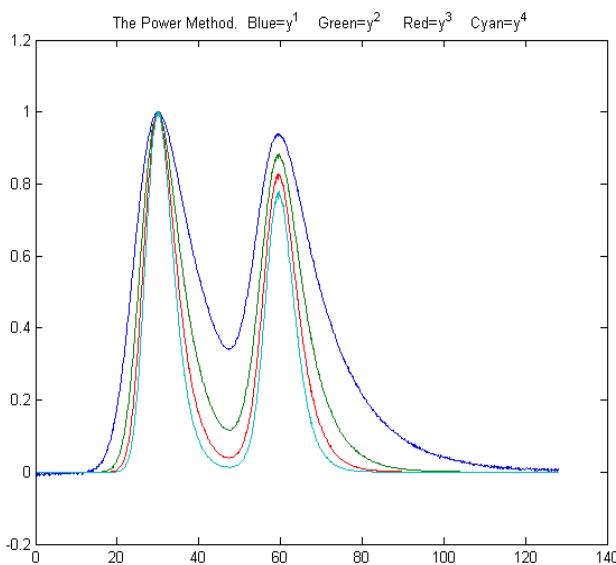
Un metodo semplicissimo per lo sharpening consiste nell'elevare ogni punto a una potenza *n* maggiore di 1. ([riferimenti 61, 63](#))

L'effetto di ciò è quello di modificare le forme dei picchi, essenzialmente allungando la regione centrale più alta del picco ad ampiezze maggiori e ponendo più peso sui punti prossimi al picco, ottenendo una *larghezza del picco più piccola*. Per i picchi Gaussiani in particolare, il risultato è un'altra Gaussiana con una larghezza ridotta della radice quadrata della potenza *n*. La tecnica è dimostrata dallo script Matlab/Octave [PowerLawDemo.m](#), nella figura lato, che disegna Gaussiane rumorose elevate alla potenza da p=1 a 7, le altezze dei picchi

normalizzate a 1.0, mostrano che all'aumento della potenza, la larghezza decresce e si riduce il rumore sulla linea di base ma aumenta sul massimo del picco. Poiché questo processo non sposta le posizioni dei picchi, la risoluzione del picco (definita come il rapporto tra la separazione del picco e la larghezza della sua base) aumenta. Per i picchi Gaussiani, l'area del picco originale si può calcolare dall'area sotto la curva normalizzata e trattata con lo sharpening con l'elevamento a



potenza ([riferimento 63](#)).



Nella figura, la linea blu mostra due picchi Gaussiani Modificati Esponenzialmente (EMG) leggermente sovrapposti. Le altre linee sono il risultato dell'innalzamento dei dati alla potenza di $n = 2, 3$ e 4 ciascuna normalizzata ad un'altezza di 1.00 . I risultati sono molto simili a profili Gaussiani (solo perché la maggior parte dei profili dei picchi sono localmente Gaussiani in prossimità del massimo), e le larghezze dei picchi, misurate con la funzione [halfwidth.m](#), sono ridotte: $19.2, 12.4, 9.9$ e 8.4 unità per le potenze da 1 a 4 , rispettivamente. Questo metodo è indipendente e può essere utilizzato insieme agli altri metodi di sharpening discussi in precedenza.

Tuttavia, per un segnale di due Gaussiane sovrapposte, il risultato dell'elevamento a potenza del segnale non è lo stesso dell'addizione di due Gaussiane a potenza ridotta: semplicemente, $a^n + b^n$ non è lo stesso di $(a+b)^n$ per $n > 1$. Ciò è dimostrato graficamente dallo script [PowerPeaks.m \(grafico\)](#), che approssima un modello a due Gaussiane alla somma di due Gaussiane sovrapposte elevate a potenza; all'aumentare della potenza n , i picchi si restringono e gli avvallamenti tra di loro diventano più profondi, ma il segnale risultante non è più la somma di due Gaussiane a meno che la risoluzione sia sufficientemente alta che i due picchi non si sovrappongono in modo significativo.

Alcune delle limitazioni al metodo della legge di potenza sono:

- (a) Funziona solo se i picchi in esame creano un massimo distinto (non è efficace per i picchi laterali che sono così piccoli da formare solo delle *sorgenze*; ci deve essere un avvallamento tra i picchi);
- (b) La linea di base deve essere zero per ottenere i risultati migliori;
- (c) Per i segnali rumorosi c'è una diminuzione del rapporto segnale-rumore perché la larghezza minore significa che meno punti dati stanno contribuendo alla misura (lo smoothing, pagina 38, può aiutare).

Compensazione per la non linearità. Naturalmente, il metodo della potenza introduce una grave non linearità nel segnale, modificando i rapporti tra le altezze dei picchi (come è evidente nella figura precedente) e complicando l'ulteriore elaborazione, specie la calibrazione della misura quantitativa. Ma c'è un modo semplice per compensare questo: dopo che i dati originali sono stati elevati alla potenza n e le altezze dei picchi e/o le aree sono state misurate, i picchi si possono semplicemente elevare alla potenza $1/n$, ripristinando la linearità originale (ma, in particolare, non la pendenza) della curva di calibrazione utilizzata nella misura analitica quantitativa. (Questo funziona perché l'area del picco è proporzionale all'altezza per la larghezza e l'altezza dei picchi elevati in potenza è proporzionale alla potenza ennesima dell'altezza originale, ma l'ampiezza del picco non è una funzione dell'altezza del picco alla costante n , quindi l'area dei picchi trasformati resta proporzionale alla potenza ennesima dell'altezza originaria). La tecnica è dimostrata quantitativamente, per due picchi variabili sovrapposti, dallo script Matlab/Octave [PowerLawCalibration-Demo.m \(grafico\)](#), che prende la potenza n^a del segnale di picchi sovrapposti, misura le aree di quelli trasformati e poi calcola la potenza $1/n$ delle aree misurate, costruendo e utilizzando una curva di calibrazione per convertire le aree in concentrazioni. Le aree dei picchi vengono misurate mediante tagli verticali, utilizzando il punto a metà percorso per contrassegnare il confine tra i picchi. Lo script simula un segnale misto con concentrazioni definibili nelle righe 15 e

16. È possibile modificare la potenza e qualsiasi parametro nelle righe 14-22. I risultati mostrano che il metodo della potenza migliora l'accuratezza delle misure fintanto che la risoluzione 4-sigma (il rapporto tra la separazione dei picchi a 4 volte la sigma delle Gaussiane) è all'incirca superiore a 0.4. Si ha una maggiore accuratezza quando i picchi sono approssimativamente uguali in larghezza e quando il rapporto tra le due concentrazioni non è molto diverso dal rapporto negli standard da cui viene costruita la curva di calibrazione. Si noti che, anche quando il rumore casuale simulato (nella riga 22) è zero, i risultati non sono perfetti a per effetto della sovrapposizione dei picchi sulla misura dell'area, che varia a seconda del rapporto tra le due componenti nella miscela.

La funzione autonoma [PowerMethodDemo.m](#) mostra il metodo dell'elevamento a potenza per misurare l'area del piccolo picco sporgente parzialmente sovrapposto da un picco molto più forte che interferisce ([grafico](#)). Mostra l'effetto del rumore casuale, dello smoothing e di qualsiasi background sotto i picchi.

Combinare i metodi di sharpening. Il metodo dell'elevamento a potenza è indipendente da, e può essere utilizzato con, i metodi derivativi discussi. Tuttavia, poiché il metodo della potenza non è lineare, l'ordine *con cui le operazioni sono eseguite* è importante. Il *primo* passaggio dovrebbe essere la simmetrizzazione con la derivata prima se il segnale è allargato esponenzialmente, il *secondo* passo dovrebbe essere lo sharpening derivativo e il metodo dell'elevamento a potenza dovrebbe essere l'*ultimo*. La ragione di questo ordine è che il metodo della potenza dipende dall'esistenza di una valle tra i picchi *ma non può creare una*, mentre i metodi derivativi *possono* essere in grado di creare un avvallamento tra i picchi se la sovrapposizione non è troppo eccessiva. Inoltre, quando utilizzato per ultimo, il metodo della potenza riduce la severità delle oscillazioni della linea di base che sono un residuo dello sharpening con le derivate pari (particolarmente evidente nel picco Lorentziano). Gli script Matlab/Octave [SharpenedGaussianDemo2.m](#) ([Grafico](#)) e [SharpenedLorentzianDemo2.m](#) ([Grafico](#)) espongono questo punto per picchi Gaussiani e Lorentziani rispettivamente, confrontando il risultato del solo sharpening con le derivate pari, con quello dello sharpening con le derivate pari seguito dal metodo dell'elevamento a potenza (ed eseguendo il metodo della potenza in due modi, prendendo il quadrato del picco con sharpening o moltiplicandolo per il picco originale). Sia per le forme Gaussiane che per le Lorentziane, i risultati finali dello sharpening si approssimano ai modelli Gaussiani per mostrare i cambiamenti dei parametri dei picchi. Il risultato è che la combinazione dei metodi produce (a) il picco finale più stretto e (b) una migliore approssimazione al profilo finale Gaussiano. Naturalmente, restano i problemi con la linearità del metodo della potenza, ma si possono compensare come in precedenza.

Deconvoluzione. Un'altra tecnica di elaborazione del segnale che può aumentare la risoluzione dei picchi sovrapposti è la *deconvoluzione* (che verrà trattata a pagina 108). È applicabile nelle situazioni in cui il profilo originale dei picchi è stato espanso e/o reso asimmetrico con qualche processo o funzione di espansione. Se il processo di espansione è descrivibile matematicamente o misurato a parte, allora la deconvoluzione dai picchi espansi osservati è in linea di principio in grado di estrarre il profilo del picco sottostante.

Peak Sharpening per fogli di calcolo Excel e Calc

Il metodo di sharpening con derivate pari con due termini derivativi (2^a e 4^a) è disponibile per Excel e Calc sotto forma di un template vuoto ([PeakSharpeningDeriv.xlsx](#) e [.ods](#)) o con la presenza di dati di esempio ([PeakSharpeningDerivWithData.xlsx](#) e [.ods](#)). È possibile digitare i valori dei fattori di ponderazione delle derivate K1 e K2 direttamente nelle celle **J3** e **J4**, oppure è possibile immettere l'ampiezza stimata del picco (FWHM in numero di punti) nella cella **H4** e il foglio di calcolo

calcolerà K1 e K2. Esiste anche una versione dimostrativa con picchi simulati regolabili con cui sperimentare ([PeakSharpeningDemo.xlsx](#) e [PeakSharpeningDemo.ods](#)).

Click buttons to change K1 and K2						
K1=	8761.4	-- K1	- K1	+ K1	++ K1	
K2=	1.02E+07	-- K2	- K2	+ K2	++ K2	
Rs=	0.625					

Esistono anche [versioni che hanno pulsanti](#) cliccabili (i dettagli a lato) per una comoda regolazione interattiva dei fattori K1 e K2 dell'1% o del 10% per ogni click. Si possono inserire direttamente le prime stime di K1 e K2 nelle celle J4 e J5 e poi usare i pulsanti per affinare i valori.

Se il segnale è rumoroso, regolare il livellamento utilizzando i 17 coefficienti nella riga 5 delle colonne da **K** ad **AA**, proprio come con i fogli di calcolo per lo smoothing (pag. 49). (Nota: Sfortunatamente, questi pulsanti ActiveX non funzionano nella versione iPad di Excel).

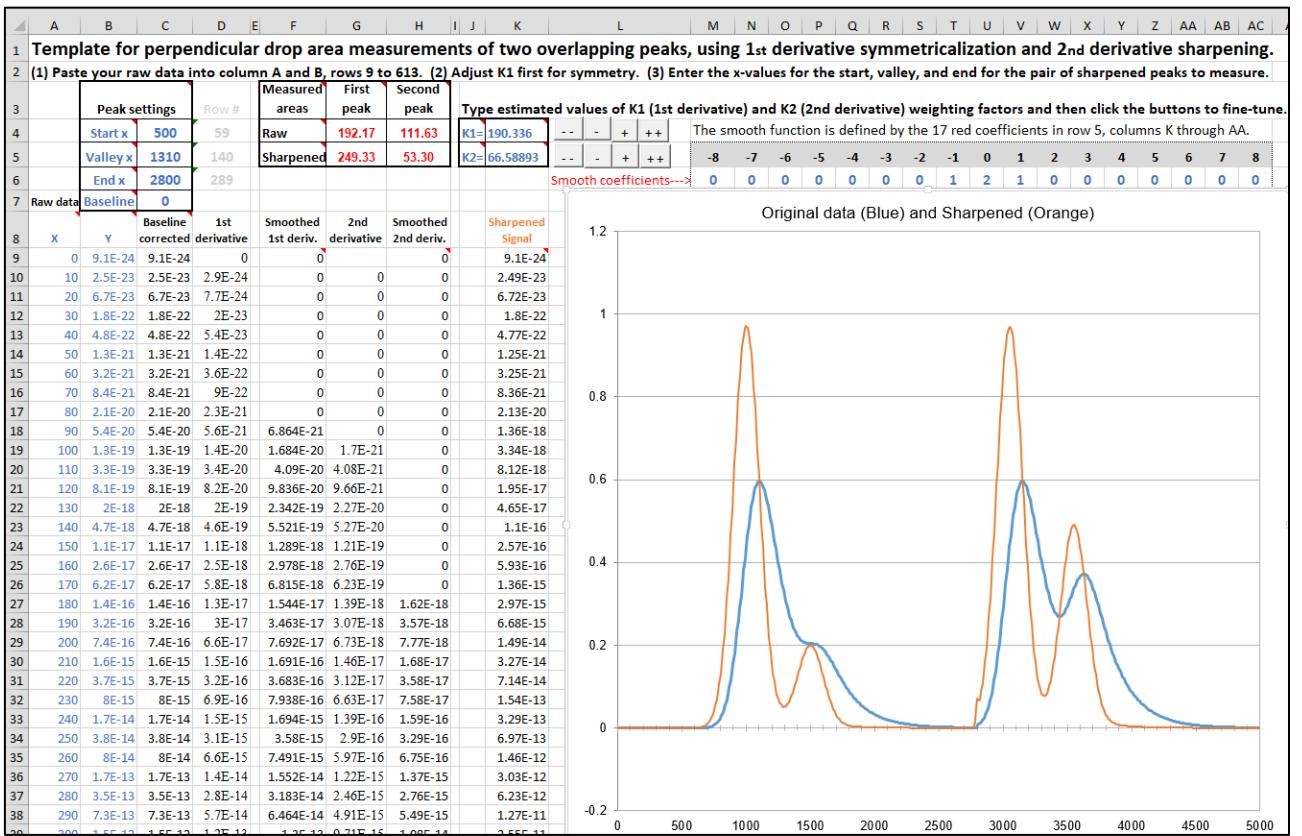
Esiste anche un versione “segmentata” del template in cui è possibile specificare le costanti per lo sharpening per ognuno dei 20 segmenti del segnale ([SegmentedPeakSharpeningDeriv.xlsx](#)). Per quelle applicazioni in cui le larghezze dei picchi aumentano (o diminuiscono) gradualmente nel tempo, c'è anche un template per lo sharpening del picco con *gradiente* in cui basta impostare la larghezza del picco iniziale e quella del picco finale e il foglio di calcolo applicherà il fattori di sharpening richiesti K1 e K2. ([GradientPeakSharpeningDeriv.xlsx](#)) e un esempio con dati già inseriti [GradientPeakSharpeningDerivExample.xlsx](#));

Il template [PeakSymmetricalizationTemplate.xlsxm](#) (schermata di seguito) esegue la simmetrizzazione delle Gaussiane modificate esponenzialmente (EMG) mediante l'aggiunta ponderata della derivata prima. [PeakSymmetricalizationExample.xlsxm](#) è un'applicazione con dei dati di esempio già inseriti. Mostrata in seguito.

C'è anche una versione demo che consente di determinare l'accuratezza della tecnica sintetizzando picchi sovrapposti con risoluzione, asimmetria, altezza relativa del picco, rumore e linea di base specificati: [PeakSharpeningAreaMeasurementEMGDemo2.xlsxm](#) ([grafico](#)). Questi fogli di calcolo consentono anche un ulteriore sharpening con la derivata seconda del picco simmetrico risultante.

[PeakDoubleSymmetrizationExample.xlsxm](#) esegue la simmetrizzazione di un picco doppiamente allargato esponenzialmente. Dispone di pulsanti per regolare interattivamente i due pesi della derivata prima. Due varianti ([1](#), [2](#)) includono i dati per due picchi sovrapposti, per i quali le aree vengono misurate con un taglio perpendicolare.

[EffectOfNoiseAndBaselineNormalVsPower.xlsx](#) mostra l'effetto del metodo dell'elevamento a potenza sulle misure delle aree di picchi Gaussiani allargati esponenzialmente, inclusi i diversi effetti che hanno il rumore casuale e una linea di base non nulla sullo sharpening.



Il template “PeakSymmetrizationTemplate.xlsx” mostra la misura delle aree della prima e della seconda coppia di picchi asimmetrici sovrapposti dopo aver applicato la simmetrizzazione con la derivata prima.

Sharpening dei picchi con Matlab e Octave

La funzione utente per Matlab/Octave [sharpen](#) ha la forma **SharpenedSignal = sharpen (signal, k1, k2, SmoothWidth)**, dove "signal" è il vettore del segnale originale, gli argomenti k2 e k4 sono i fattori di ponderazione della derivata 2^a e 4^a e SmoothWidth è la larghezza dello smoothing. Il segnale migliorato in risoluzione viene restituito nel vettore "SharpenedSignal". Se si sta leggendo online, si può cliccare sul link sopra per ispezionare il codice o fare click-distro del mouse per scaricarlo per utilizzarlo all'interno di Matlab o di Octave. I valori **k** determinano il compromesso tra la nitidezza [sharpness] del picco e l'appiattimento della linea di base; i valori variano in base alla forma e alla larghezza del picco e devono essere regolati in base alle proprie necessità. Per i picchi con profilo Gaussiano, un valore ragionevole per k₂ è PeakWidth²/32 e per k₄ è PeakWidth⁴/900 (o PeakWidth²/8 e PeakWidth⁴/700 per picchi Lorentziani), dove PeakWidth è la larghezza totale a metà massimo dei picchi in unità x. Poiché i metodi di sharpening sono tipicamente sensibili al rumore casuale nel segnale, è solitamente necessario applicare un'operazione di smoothing: la funzione Matlab/Octave [ProcessSignal.m](#) consente di applicare sia lo sharpening che lo smoothing in un'unica funzione.

Ecco un semplice esempio Matlab/Octave che crea un segnale composto da quattro picchi Gaussiani, parzialmente sovrapposti, di uguale altezza e larghezza, applica sia il metodo di sharpening derivativo che quello dell'elevamento a potenza e confronta un grafico (mostrato di seguito) del segnale originale (in blu) con la versione a risoluzione migliorata (in rosso).

```

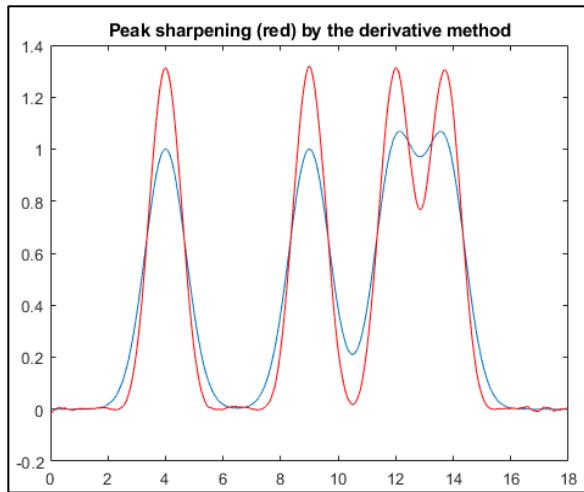
x=0:.01:18;
y=exp(-(x-4).^2)+exp(-(x-9).^2)+exp(-(x-12).^2)+exp(-(x-13.7).^2);
y=y+.001.*randn(size(x));
k1=1212;k2=1147420;
SharpenedSignal=ProcessSignal(x,y,0,35,3,0,1,k1,k2,0,0,0,0,0);

```

```

figure(1)
plot(x,y,x,SharpenedSignal,'r')
title('Peak sharpening (red) by the derivative method')
figure(2)
plot(x,y,x,y.^6,'r')
title('Peak sharpening (red) by the power method')

```



Quattro picchi Gaussiani sovrapposti di uguale altezza e larghezza.

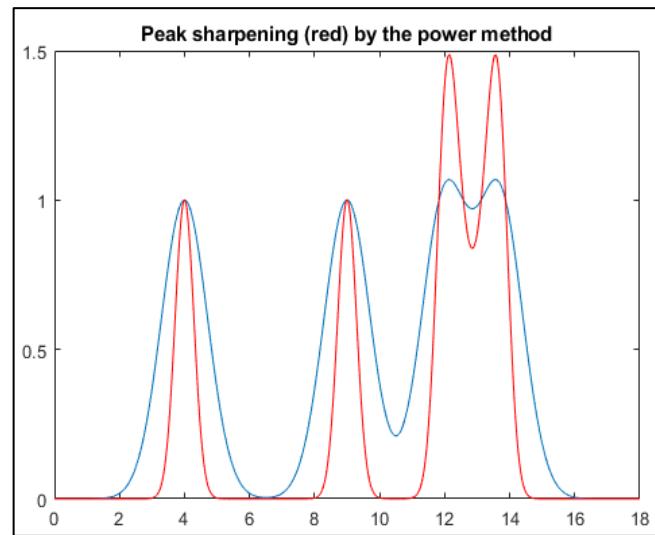
In blu: l'originale. In rosso: dopo lo sharpening col metodo delle derivate pari.

[SharpenedOverlapDemo.m](#) è uno script che tenta di *determinare automaticamente il grado ottimale di sharpening con le derivate pari* che minimizza gli errori nella misura delle aree dei picchi di due Gaussiane sovrapposte col metodo del taglio perpendicolare utilizzando la funzione autopeaks.m. Lo fa applicando diversi gradi di sharpening e disegnando gli errori delle aree (differenza percentuale tra gli errori veri e quelli misurati) rispetto al fattore di sharpening. Mostra anche l'altezza della valle tra i picchi con una linea gialla. Questo dimostra che:

- (1) il fattore ottimale di sharpening dipende dalla larghezza e dalla separazione dei due picchi, e dal rapporto delle loro altezze;
- (2) il grado di sharpening non è eccessivamente critico, mostrando spesso un'ampia regione ottimale;
- (3) l'ottimo per i due picchi non è necessariamente lo stesso; e
- (4) l'ottimo per la misura dell'area potrebbe non avversi nel punto in cui l'avvallamento è zero.

(Per eseguire questo script si devono avere gaussian.m, derivxy.m, autopeaks.m, val2ind.m e halfwidth.m nel path di ricerca di Matlab. Il download si effettua da

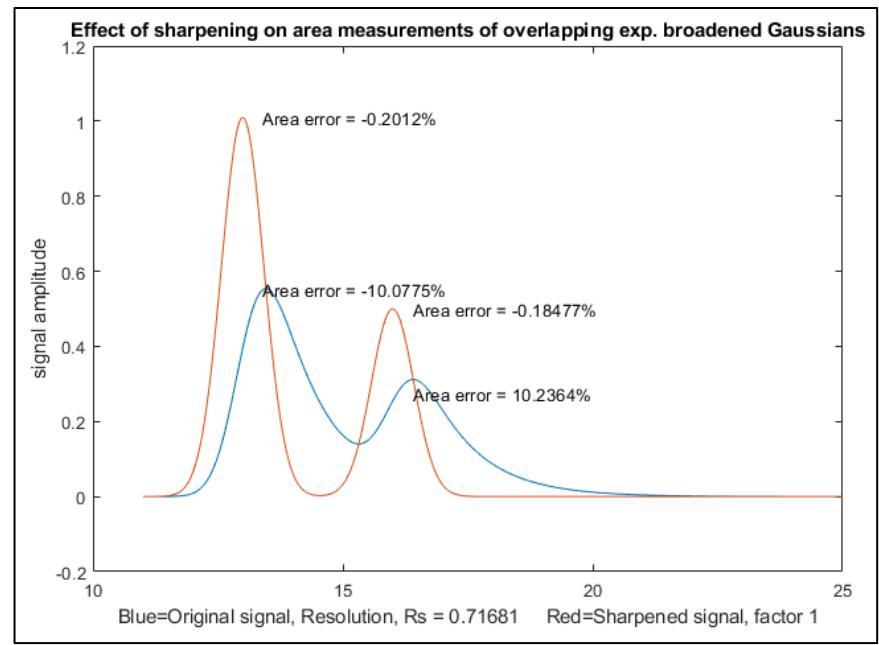
<https://terpconnect.umd.edu/~toh/spectrum/>.



Il metodo dell'elevamento a potenza è efficace se c'è un avvallamento tra i picchi sovrapposti, ma questo introduce una non-linearietà che dev'essere corretta in seguito, mentre il metodo delle derivate preserva le aree originali dei picchi e il rapporto tra le altezze dei picchi. [PowerLawCalibrationDemo](#) mostra la linearizzazione delle curve di calibrazione trasformate in potenza per due picchi

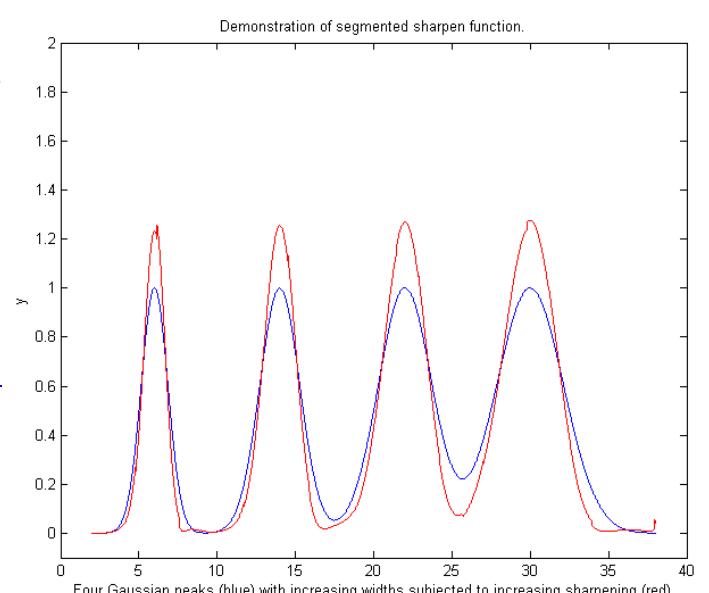
sovraposti prendendo la potenza ennesima dei dati, localizzando l'avallamento tra di essi, misurando le aree col metodo del taglio perpendicolare (pagina 135), e quindi prendendo la potenza 1/n delle aree misurate ([grafico](#)).

La simmetrizzazione ad area costante (de-tailing) di picchi asimmetrici mediante l'aggiunta ponderata della derivata prima viene eseguita dalla funzione `ySym = symmetrize(t, y, factor, smoothwidth, type, ends)`; "t" e "y" sono i vettori della variabile indipendente e di quella dipendente, "factor" è il fattore di ponderazione della derivata prima, "smoothwidth", "type" e "ends" sono i parametri [SegmentedSmooth](#) per lo smoothing derivativo interno. Per eseguire una simmetrizzazione segmentata, "factor" e "smoothwidth" possono essere dei vettori. Nella versione 2, `symmetrize.m` esegue solo lo smoothing della derivata, non dell'intero segnale. [SymmetrizeDemo.m](#) esegue tutti e cinque gli esempi nel file di help `symmetrize.m`, ciascuno in una finestra separata. Se tau *non* è noto, lo si può determinare *per un singolo picco isolato* utilizzando [AutoSymmetrize\(t, y, SmoothWidth, plots\)](#), che cerca il valore di tau che produce il picco più simmetrico, confrontando la pendenza delle tangenti sui lati di entrata e di uscita. Nell'esempio precedente, il picco originale (linea blu) è una Gaussiana modificata esponenzialmente calcolata matematicamente con un valore di tau di 100 e la linea rossa è l'output generato da `AutoSymmetrize`, che stima il tau con un'accuratezza dell'1%. Digitare "help AutoSymmetrize". Le aree dei due sono uguali entro lo 0.01%. [SymmetrizedOverlapDemo.m](#) mostra l'ottimizzazione della simmetrizzazione con la derivata prima per la misura dell'area di due Gaussiane sovrapposte allargate esponenzialmente.



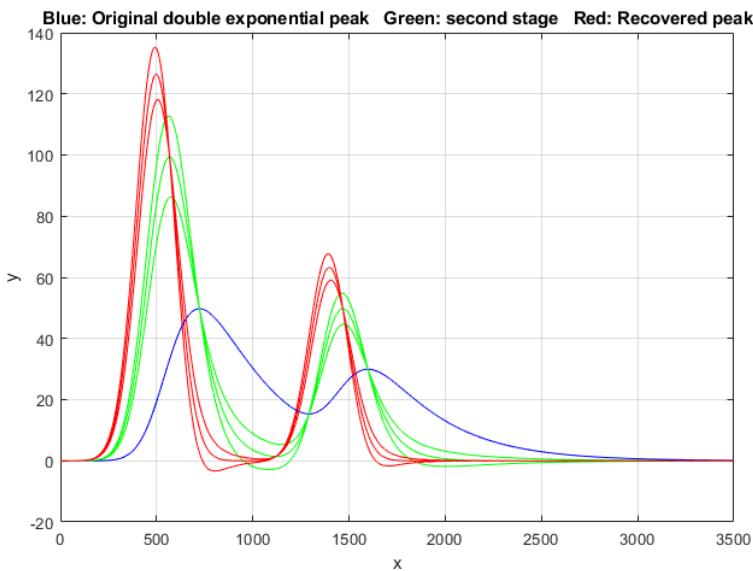
Sharpening segmentato dei picchi con derivate pari. Se la larghezza dei picchi o la varianza del rumore cambia sostanzialmente lungo il segnale, si può usare la *versione* segmentata [SegmentedSharpen.m](#), per cui gli argomenti `factor1`, `factor2` e `SmoothWidth` sono *vettori*. Lo script [Demo-SegmentedSharpen.m](#), mostrato a lato, usa questa funzione per eseguire lo sharpening di quattro picchi Gaussiani con larghezze dei picchi gradualmente crescenti da sinistra a destra così come il grado di sharpening, mostrando che la larghezza del picco viene [ridotta dal 20% al 22%](#) dall'originale.

[DemoSegmentedSharpen2.m](#) mostra quattro picchi con la *stessa* larghezza cui si applica un grado crescente di sharpening.



La simmetrizzazione esponenziale doppia in Matlab/Octave viene eseguita dalla funzione [DEMSymm.m](#) che applica due successive addizioni ponderate della derivata prima, con fattori di ponderazione idealmente uguali ai due tau. L'obiettivo è rendere i picchi più simmetrici e più stretti preservandone l'area. Una tecnica di raggruppamento (bracketing) a tre livelli (uno in più e uno in meno) aiuta a determinare i valori migliori per i due fattori di ponderazione.

La tecnica viene mostrata dallo script [DemoDEMSymm.m](#) e sono due varianti ([1](#), [2](#)), che creano due picchi sovrapposti doppiamente esponenziali dagli originali Gaussiani, poi chiama la funzione DEMSymm.m per eseguire la simmetrizzazione. Nell'esempio a lato, la linea centrale è il valore ottimale. In sintesi, se si tenta di simmetrizzare un picco asimmetrico mediante l'aggiunta ponderata della derivata prima e il risultato è ancora asimmetrico, è possi-

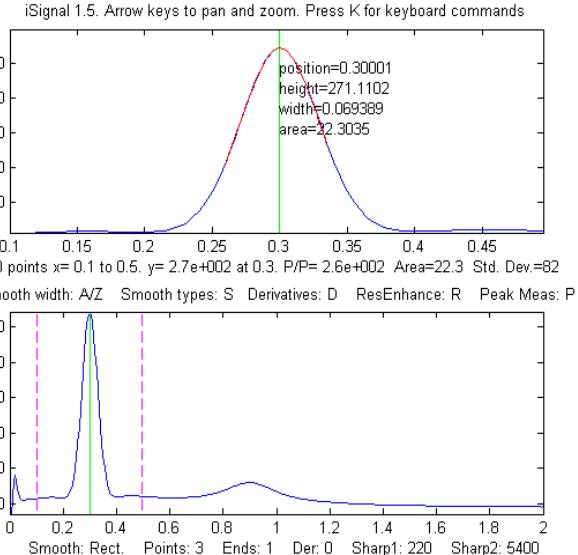
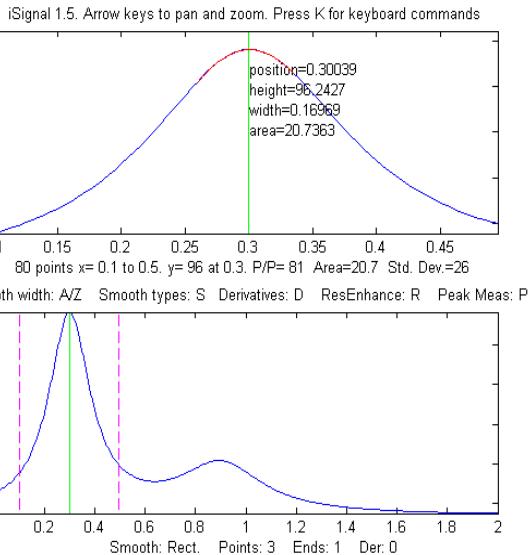


bile che l'asimmetria residua sia dovuta a un altro stadio di ampliamento esponenziale con un diverso *tau*, quindi in tal caso, l'applicazione di DEMSymm.m produrrà probabilmente un risultato finale più simmetrico.

[ProcessSignal](#), è una funzione Matlab/Octave a riga di comando che esegue smoothing, differenziazione e sharpening del picco su una serie temporale x,y (vettori colonne o righe). Digitare "help ProcessSignal". Restituisce il segnale processato come vettore che ha lo stesso profilo di x, indipendentemente da quello di y.

```
Processed=ProcessSignal(x, y, DerivativeMode, w, type, ends, Sharpen,
factor1, factor2, Symsize, Symfactor, SlewRate, MedianWidth)
```

[iSignal](#) (Versione 8.3, pagina 366) è una funzione interattiva Matlab multi-uso che include lo sharpening per segnali temporali, utilizzando sia il metodo delle derivate pari (funzione *sharpen*) che quello della simmetrizzazione con la derivata prima, consente di regolare da tastiera i fattori di ponderazione delle derivate e dello smoothing in modo continuo osservando dinamicamente l'effetto sul segnale. Il tasto **E** attiva e disattiva la funzione di sharpening dei picchi. Il codice è visualizzabile [qui](#) o si può scaricare il [file ZIP](#) con i dati dell'esempio per il test. *iSignal* calcola le impostazioni di sharpening e smoothing per picchi Gaussiani e Lorentziani utilizzando rispettivamente i tasti **Y** e **U**, e usando l'espressione precedente. Basta isolare un solo picco tipico nella finestra superiore con i tasti pan e zoom, e premere **P** per attivare la modalità di misura del picco e poi premere **Y** per pic-

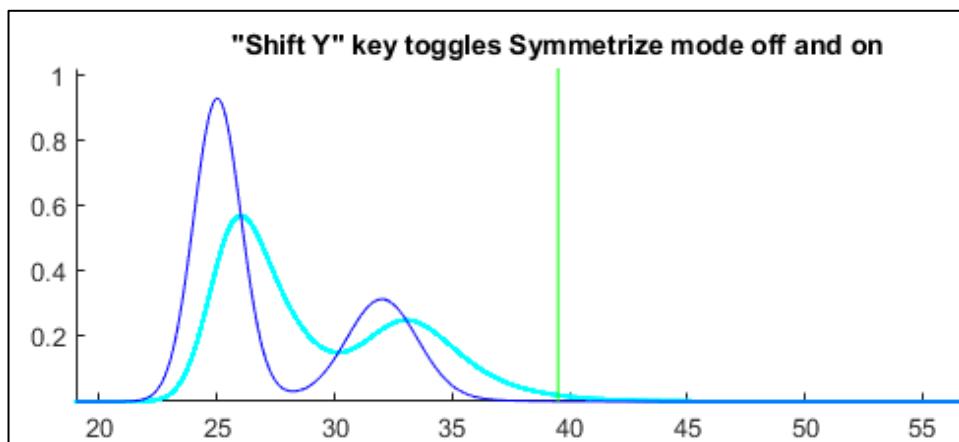


chi Gaussiani o **U** per quelli Lorentziani. È possibile ottimizzare lo sharpening con i tasti **F/V** e **G/B** e lo smoothing con i tasti **A/Z**. (Se il segnale ha picchi di larghezze molto diverse, un unico settaggio non sarà ottimale per tutti i picchi. In questi casi si può usare la funzione di sharpening segmentato, [SegmentedSharpen.m](#)).

Prima dello Sharpening in iSignal

Dopo lo Sharpening in iSignal

In *iSignal*, e in *iPeak*, il tasto **Shift-Y** attiva la tecnica di simmetrizzazione della derivata prima ed usa i tasti **1**, **Shift-1**, **2** e **Shift-2** per regolare il fattore di ponderazione del 10% o dell'1% ad ogni pressione del tasto. L'idea è quella di aumentare il fattore fino a quando la linea di base dopo il picco diventa negativa, quindi aumentarla leggermente in modo che sia *il più bassa possibile ma non negativa*.



iSignal può anche utilizzare il metodo di trasformazione con la potenza (si preme il tasto \wedge , si inserisce la potenza, n (qualsiasi numero positivo maggiore di 1.00) e si preme **Enter**. Per invertirlo, basta elevare alla potenza $1/n$. [iPeak](#), (pag. 245), è un programma Matlab interattivo per il rilevamento e la misura dei picchi, ha una modalità di sharpening basata sulla tecnica delle derivate pari, nonché quella di simmetrizzazione con la derivata prima utilizzando gli stessi tasti di *iSignal*. Cfr. `ipeakdemo5` a pagina 261. L'animazione GIF lo mostra in azione.

Lo sharpening del picco, sia mediante la simmetrizzazione con derivata pari che con i metodi di autodeconvoluzione di Fourier, è incluso come parte del tool interattivo di rilevamento dei picchi [PeakDetection mlx](#) discusso a pagina 245.

Lo sharpening in tempo-reale del picco in Matlab è trattato a pagina 339.

Analisi delle armoniche e la Trasformata di Fourier.

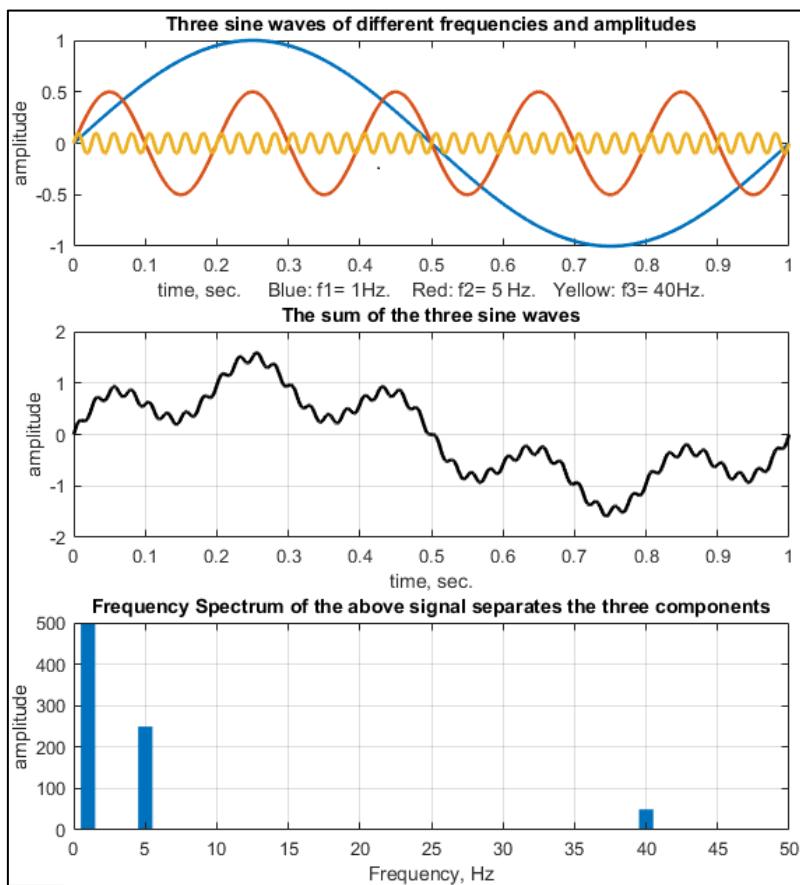
Alcuni segnali mostrano componenti periodiche che si ripetono a intervalli fissi per tutto il segnale, come un'onda sinusoidale. È spesso utile descrivere esattamente l'ampiezza e la frequenza di tali componenti periodiche. Infatti, è possibile analizzare *qualsiasi* insieme di dati arbitrari nelle sue componenti periodiche, sia che i dati appaiano periodici o meno. [L'analisi armonica](#) è convenzionalmente basata sulla [trasformata di Fourier](#), che è un modo per esprimere un segnale

come somma di [seni e coseni](#). Si può dimostrare che qualsiasi segnale arbitrario campionato discretamente può essere descritto completamente dalla somma di un numero finito di componenti seno e coseno le cui frequenze sono 0, 1, 2, 3 ... n/2 volte la frequenza $f=1/n\Delta x$, dove Δx è l'intervallo tra i valori adiacenti di x e n è il numero totale di punti. La trasformata di Fourier è semplicemente l'insieme delle ampiezze di quelle componenti seno e coseno (o, che è equivalente matematicamente, [la frequenza e la fase delle componenti seno](#)). Quei coefficienti si potrebbero calcolare semplicemente ma laboriosamente moltiplicando il segnale punto per punto con ciascuna di quelle componenti seno e coseno e sommando i prodotti. La famosa “[Fast Fourier Transform](#)” (FFT) risale al 1965 ed è un algoritmo più veloce ed efficiente che utilizza la simmetria delle funzioni seno e coseno e altre scorciatoie matematiche per ottenere lo stesso risultato *molto* più rapidamente. La trasformata *inversa* di Fourier (IFT) è un algoritmo simile che converte una trasformata di Fourier nel segnale originale. Per comodità matematica, le trasformate di Fourier sono solitamente espresse in termini di “[numeri complessi](#)”, perché le parti “reali” e “immaginarie” consentono di combinare le informazioni di seno e coseno (o ampiezza e fase) per ciascuna frequenza in un unico numero complesso, utilizzando l'identità $\exp(i2\pi ft) = \cos(2\pi ft) + i\sin(2\pi ft)$. Anche per i dati che non sono complessi, usare la “exp” anziché “sin+cos” è più *compatto ed elegante* e molti linguaggi per computer possono gestire automaticamente aritmetiche complesse quando le quantità sono complesse. Ma questa terminologia può essere fuorviante: le parti seno e coseno sono *ugualmente importanti*; solo perché le due parti sono chiamate “reale” e “immaginaria” in matematica *non implica che la prima sia meno significativa* della seconda. (Per una spiegazione matematica rigorosa, vedere [Fourier Transforms](#) di Gary Knott).

Il concetto di trasformata di Fourier è coinvolto in due moderni metodi strumentali molto importanti nelle analisi chimiche. Nella [Spettroscopia infrarossa in trasformata di Fourier \(FTIR\)](#), la trasformata di Fourier dello spettro viene misurata direttamente dallo strumento, come l'interferogramma formato tracciando il segnale del rivelatore rispetto allo spostamento dello specchio in un interferometro di Michelson a scansione. Nella [spettroscopia di risonanza magnetica nucleare in trasformata di Fourier \(FTNMR\)](#), l'eccitazione del campione da parte di un intenso e breve impulso di energia a radiofrequenza produce un segnale di decadimento di induzione libera che è la trasformata di Fourier dello spettro della risonanza. In entrambi i casi, viene usato *un computer per acquisire lo spettro* mediante trasformata inversa di Fourier del segnale misurato (interferogramma o decadimento di induzione libera).

Lo [spettro della potenza](#) o [spettro delle frequenze](#) è un semplice modo per mostrare l'ampiezza totale in ciascuna di queste frequenze. Viene calcolato come la radice quadrata della somma dei quadrati dei coefficienti delle componenti seno e coseno. Lo spettro della potenza conserva l'informazione sulla *frequenza* ma scarta quella sulla *fase*, in modo che lo spettro della potenza di un seno sarebbe uguale a quello di un coseno della stessa frequenza, anche se le trasformate di Fourier complete di seno e coseno sono diverse in fase. Nelle rare situazioni in cui le *componenti della fase* di un segnale sono la principale fonte di rumore (p.es. slittamenti casuali della posizione del segnale sull'asse x), può essere vantaggioso basare la misura sullo spettro della potenza, che scarta le informazioni della fase, mediante la media di insieme (pag. 25) dello spettro della potenza di segnali ripetuti: questo viene mostrato dagli script Matlab/Octave [EnsembleAverageFFT.m](#) e [EnsembleAverageFFTGaussian.m](#).

Un segnale di serie temporale con n punti fornisce uno spettro della potenza con soli $(n/2)+1$ punti. Il primo punto è la componente della frequenza zero (la costante), corrispondente alla componente DC ("corrente continua") del segnale: appare come una linea retta e piatta. Il secondo punto

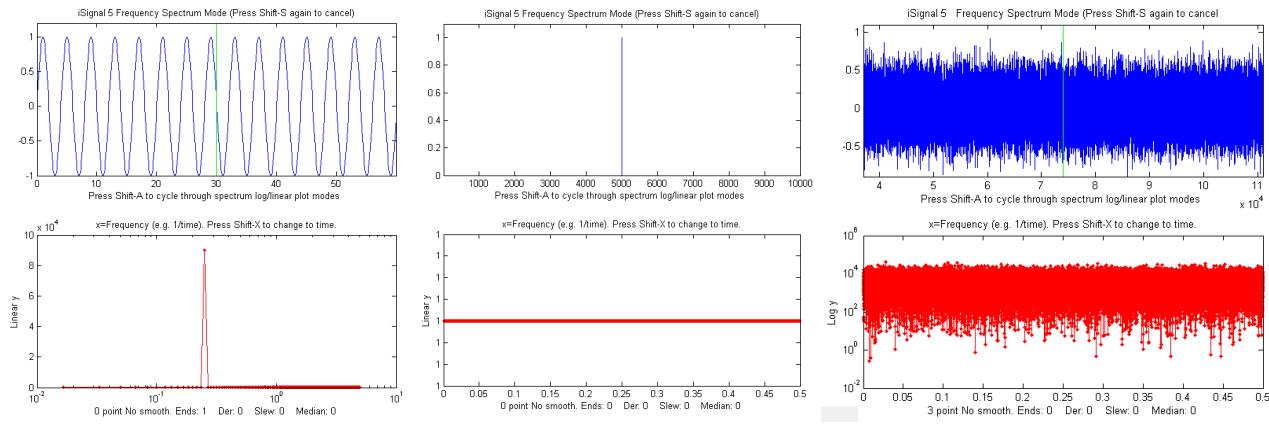


corrisponde alla frequenza $1/n\Delta x$ (il cui periodo è esattamente pari alla durata dei dati), il punto successivo a $2/n\Delta x$, poi a $3/n\Delta x$, ecc., dove Δx è l'intervallo tra i punti adiacenti sull'asse x e n è il numero totale di punti. L'ultimo punto (la frequenza più alta) nello spettro della potenza è $(n/2)/n\Delta x=1/2\Delta x$, che è la metà della frequenza di campionamento. La figura a sinistra mostra un segnale simulato di un secondo, 1000 punti con una frequenza di campionamento di 1000 Hz (pannello centrale). Questo segnale contiene solo tre sinusoidi (mostrate separatamente in diversi colori nel pannello superiore), tutte chiaramente distinguibili sommate nel segnale stesso

(pannello centrale). Si possono anche contare i cicli delle componenti sinusoidali per confermarne le frequenze. Le frequenze si presentano tutte nei punti e con le ampiezze relative previste nello spettro delle ampiezze di Fourier, che sono state disegnate qui come un grafico a barre (pannello inferiore), mostrando solo le frequenze fino a 50 Hz, su un massimo di 500 Hz. Funziona allo stesso modo anche con i coseni, che differiscono dai seni solo per la fase (spostamento sull'asse x).

I limiti del campionamento. La frequenza più alta che sia rappresentabile in una forma d'onda campionata in modo discreto è la metà della frequenza di campionamento, detta [frequenza di Nyquist](#). Nel segnale sopra, la frequenza Nyquist è $1/2 \times 1000 = 500$ Hz. I tentativi di digitalizzare segnali analogici con frequenze più alte vengono "riplegati" a frequenze più basse [Frequency Folding], distorcendo gravemente il segnale. Questo si chiama *aliasing*. La risoluzione in frequenza, cioè la differenza tra le frequenze di punti adiacenti nello spettro di frequenze calcolato, è semplicemente il reciproco della durata temporale del segnale, 1 Hz.

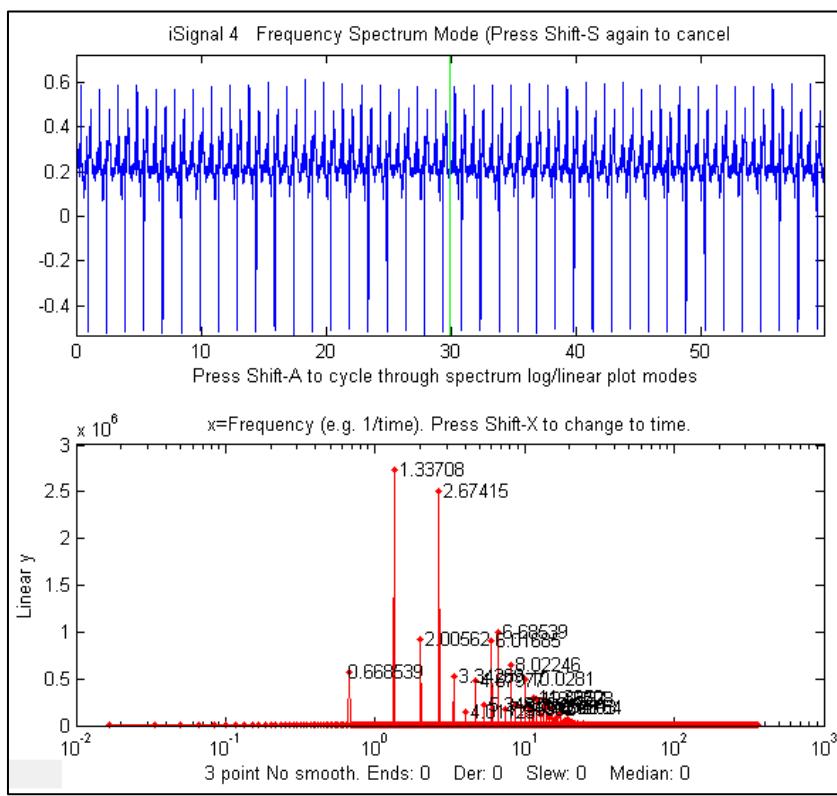
Lo script Matlab autonomo [AliasingDemo.m \(grafico\)](#) dimostra il fenomeno dell'aliasing e del Frequency Folding. Crea un'onda sinusoidale di frequenza fissa (100 Hz), quindi la campiona ripetutamente a frequenze di campionamento gradualmente decrescenti, iniziando a 600 Hz, ben al di sopra della frequenza di Nyquist (200 Hz) e terminando a 130 Hz, ben al di sotto della frequenza di Nyquist. Il grafico in esecuzione mostra che la distorsione causata dal campionamento inizialmente è piccola ma aumenta drasticamente quando la frequenza di campionamento si avvicina a 200 Hz, al di sotto della quale la frequenza apparente, indicata dal numero di picchi contati, diminuisce.



Un'onda sinusoidale o coseno pura che ha un numero intero esatto di cicli all'interno del segnale registrato ha una singola componente di Fourier diversa da zero corrispondente alla sua frequenza (in alto a sinistra). Al contrario, un segnale costituito da zeri ovunque tranne che in un singolo punto, chiamato funzione delta, ha componenti di Fourier uguali a tutte le frequenze (pagina precedente, al centro). Il rumore casuale ha anche uno spettro di potenza distribuito su un'ampia gamma di frequenze. La distribuzione dell'ampiezza del rumore dipende dal colore del rumore (pagina 29) con il rumore rosa che ha più potenza alle basse frequenze, il rumore blu che ha più potenza alle alte frequenze e il rumore bianco che ha più o meno la stessa potenza a tutte le frequenze (in alto a destra).

Esempio di segnali reali.

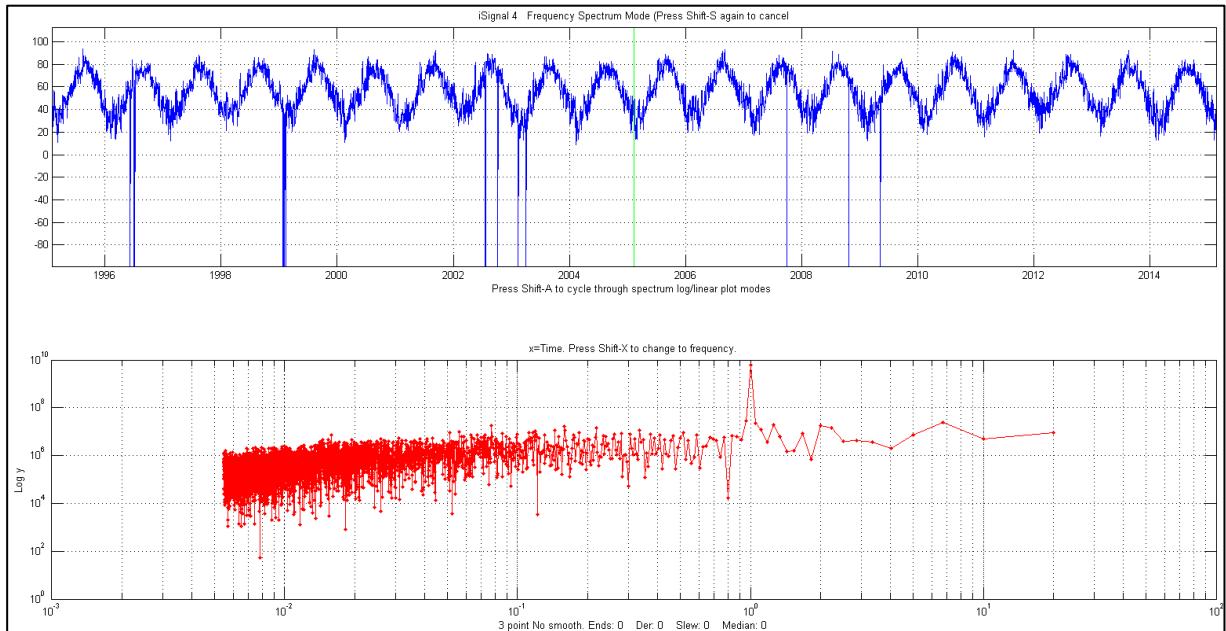
La figura seguente mostra una registrazione di 60 secondi di dati reali di un battito cardiaco, chiamata [elettrocardiogramma](#) (ECG), che è un esempio di una forma d'onda periodica che si ripete nel tempo. La figura mostra la forma d'onda in blu nel pannello in alto e il suo spettro in frequenza in red nel pannello in basso. L'unità più piccola che si ripete è detto *periodo*, e il suo reciproco è la [frequenza fondamentale](#). Le forme d'onda periodiche non sinusoidali, come questa, mostrano una serie di componenti di frequenze che sono *multiple della frequenza fondamentale*, e sono dette



"armoniche". Questo spettro mostra una frequenza fondamentale di 0.6685 Hz (pari a 40.1 battiti al minuto, un po' lenta per una normale frequenza cardiaca umana in stato di veglia), con molteplici armoniche alle frequenze $\times 2, \times 3, \times 4, \dots$, ecc., volte la frequenza fondamentale. La frequenza più bassa nello spettro è 0.067 Hz (il reciproco della durata della registrazione) e quella più alta è 400 Hz (la metà della frequenza di campionamento). La fondamentale e le armoniche sono *picchi stretti* e su questo grafico sono etichettate con le loro frequenze. Lo spettro è qualitativamente simile a quello dei picchi identici perfettamente regolari ([grafico](#)). I suoni vocali registrati, specie le vocali, hanno anche questo tipo di forma d'onda periodica con armoniche ([grafico](#)). La nitidezza [sharpness]

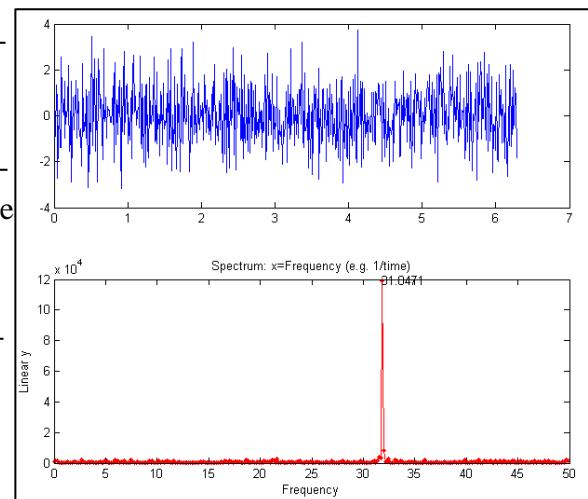
dei picchi nell'elettrocardiogramma mostra che l'ampiezza e la frequenza sono molto costanti nell'intervallo di registrazione di 60 secondi in questo esempio (che è un comportamento normale per un cuore sano). I cambiamenti nell'ampiezza o nella frequenza durante l'intervallo di registrazione produrranno *cluster* o *bande* di componenti di Fourier anziché picchi netti, come nell'esempio a pagina 295.

Un altro esempio familiare di oscillazione periodica stabile è la variazione stagionale della temperatura, ad esempio la [temperatura media giornaliera misurata a New York tra il 1995 e il 2015](#), mostrata nella figura seguente. Questo segnale mostra un'ovvia periodicità, ad eccezione dei forti picchi negativi (dovuti a punti mancanti, forse locali interruzioni di corrente). *Da notare la scala logaritmica* sull'asse y dello spettro nel pannello inferiore; questo spettro copre una gamma di ampiezze molto grande.



In questo esempio, lo spettro nel pannello inferiore, in rosso, è disegnato con il *tempo* (il reciproco della frequenza) sull'asse x. Questo viene chiamato [periodogramma](#). Nonostante il notevole rumore casuale dovuto alle variazioni meteorologiche locali e ai dati mancanti, è evidente il picco previsto esattamente a 1 anno; questo picco è *oltre 100 volte più forte* del rumore di fondo ed è molto *stretto* perché la periodicità è estremamente precisa (infatti, è letteralmente *astronomicamente* precisa). Al contrario, il rumore casuale è *non* periodico ma è piuttosto distribuito più o meno equamente sull'intero periodogramma.

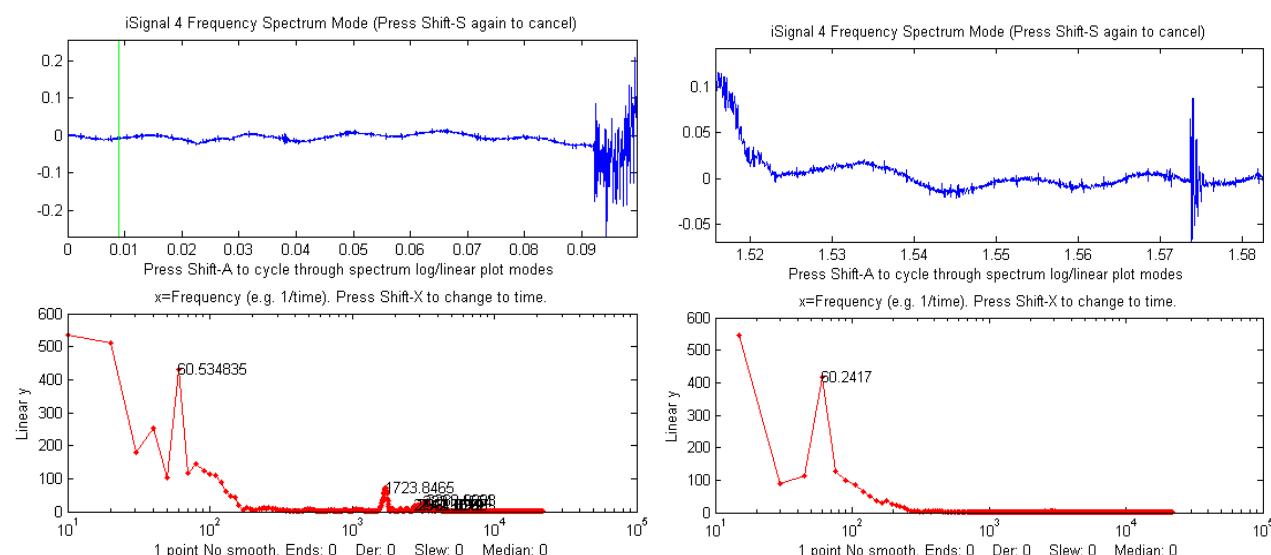
La figura a lato mostra alcuni dati simulati per illustrare quanto sia difficile vedere una componente periodica in presenza di rumore casuale e tuttavia quanto sia facile individuarla nello spettro delle frequenze. In questo esempio, il segnale (pannello superiore) contiene una *miscela uguale* di rumore bianco casuale e una singola onda sinusoidale; l'onda sinusoidale è quasi completamente oscurata dal rumore casuale. Lo spettro delle frequenze (create con la funzione Matlab/Octave "[PlotFrequencySpectrum](#)") è mostrato nel pannello in inferiore. Lo spettro delle frequenze del rumore bianco è distribuito uniformemente su tutto lo spettro, mentre l'onda sinusoidale è concentrata in un *singolo* elemento spettrale, dove risalta nettamente. Ecco il codice Matlab/Octave



che ha generato quella figura; si ne può fare il Copia e Incolla in Matlab/Octave:

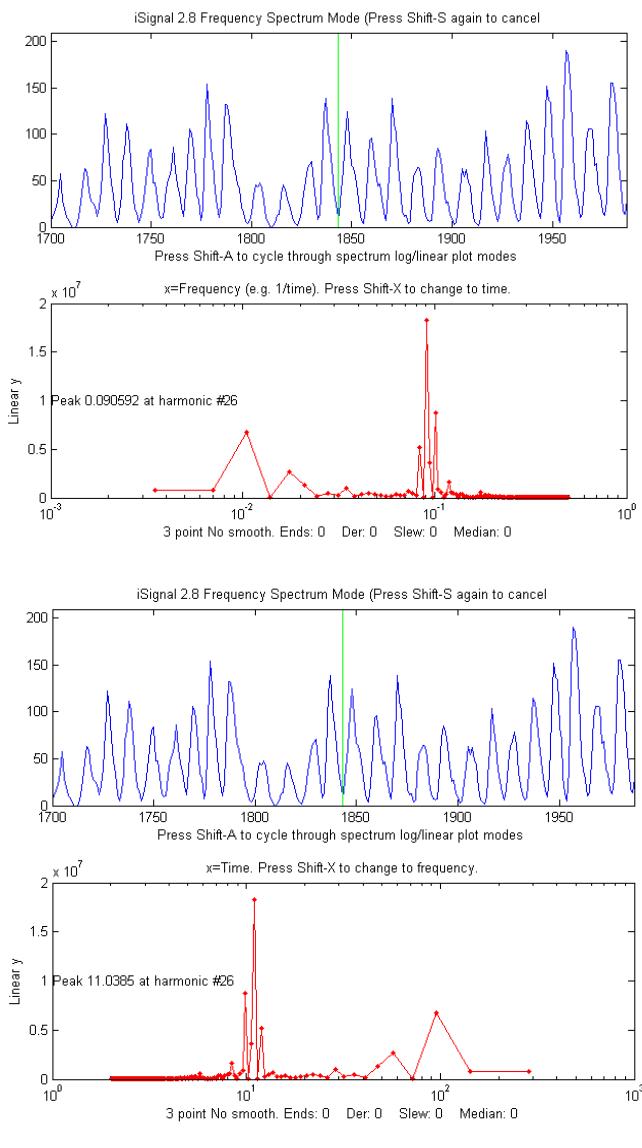
```
x=[0:.01:2*pi]';
y=sin(200*x)+randn(size(x));
subplot(2,1,1);
plot(x,y);
subplot(2,1,2);
PowerSpectrum=PlotFrequencySpectrum(x,y,1,0,1);
```

Un'applicazione pratica comune è l'uso dello spettro della potenza come strumento diagnostico per distinguere le componenti del segnale e quelle del rumore. Un esempio è l'interferenza di una linea elettrica AC rappresentata nella prossima figura, che ha una frequenza fondamentale a 60 Hz negli USA ([perché questa frequenza?](#)) o 50 Hz in molte altre nazioni. Anche in questo caso la nitidezza dei picchi nello spettro mostra che l'ampiezza e la frequenza sono molto stabili; le aziende elettriche si preoccupano di mantenere la frequenza della corrente alternata molto costante per evitare problemi tra le diverse sezioni della rete elettrica. Altri esempi di segnali e dei loro spettri in frequenza sono [mostrati in seguito](#).

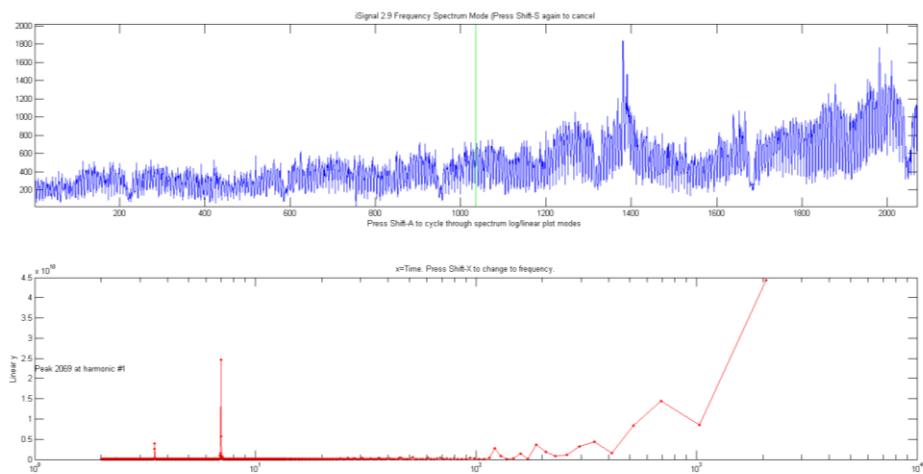


iSignal, che mostra i dati di una registrazione audio, ingrandita nel periodo di "quiete" immediatamente precedente (a sinistra) e successivo (a destra) del suono effettivo. Ciò dimostra che in quei periodi c'è un'oscillazione sinusoidale residua ($x = \text{tempo in secondi}$). Nel pannello inferiore, lo spettro di potenza di ciascun segnale ($x = \text{frequenza in Hz}$) mostra un forte e stretto picco prossimo ai 60 Hz, suggerendo che l'oscillazione è causata dall'interferenza di una linea elettrica a 60 Hz (dato che la registrazione è stata effettuata negli USA; se fosse stata fatta in Europa sarebbero stati 50 Hz). Una schermatura e una messa a terra migliori dell'apparecchiatura potrebbero ridurre l'interferenza. Il "prima" dello spettro, a sinistra, ha una risoluzione della frequenza di soli 10 Hz (il reciproco della durata della registrazione è all'incirca 0.1 secondi) e comprende soltanto 6 cicli della frequenza a 60 Hz (motivo per cui quel picco nello spettro è il 6° punto); per ottenere una risoluzione migliore si sarebbe dovuto iniziare prima la registrazione, per ottenere una registrazione più lunga. Il "dopo" dello spettro, a destra, ha un tempo di registrazione ancora più breve e quindi una risoluzione della frequenza inferiore.

Segnali a forma di picco hanno gli spettri della potenza concentrati nella gamma delle basse frequenze, mentre il rumore casuale spesso si diffonde su una gamma di frequenze più ampia. Questo è il motivo per cui lo smoothing (filtraggio passa-basso) può rendere un segnale rumoroso *apparentemente* migliore, ma è anche il motivo per cui lo smoothing solitamente non aiuta nelle misure quantitative, perché la maggior parte delle informazioni dei picchi si trovano alle basse frequenze, dove restano inalterate le basse frequenze del rumore dopo lo smoothing (Cfr. pagina 41).



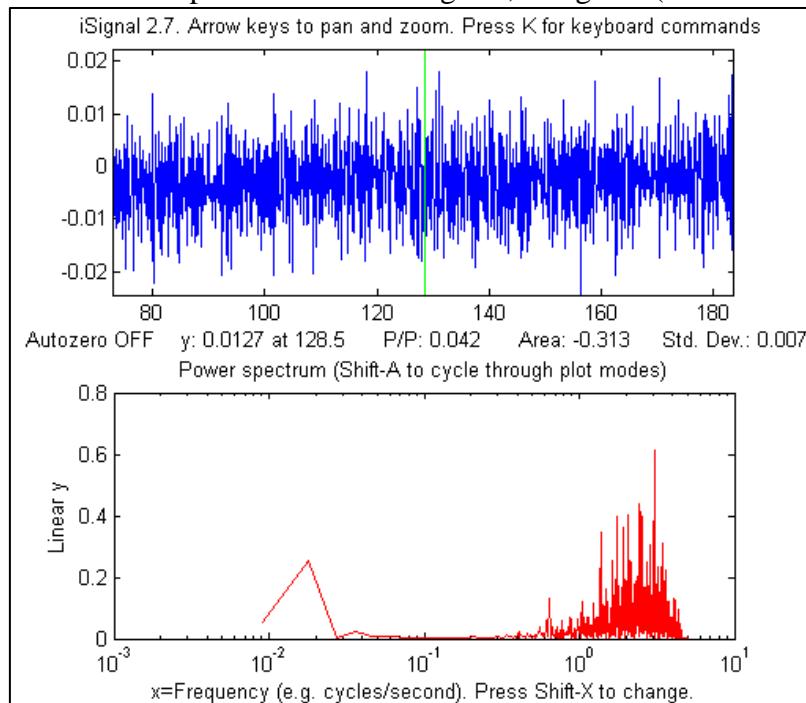
Le figure precedenti mostrano un classico esempio di analisi armonica; viene mostrata la variazione annuale del numero di macchie solari osservate, registrate annualmente fin dal 1700! In questo caso, l'asse del tempo è in *anni* (finestra superiore). Un grafico dello spettro di potenza (finestra in basso a sinistra) mostra un picco forte a 0.09 cicli/anno e il periodogramma (a destra) mostra un picco al ben noto ciclo di 11 anni, più qualche accenno di un ciclo più debole con un periodo di circa 100 anni. ([Questi dati](#) si possono scaricare, così come gli aggiornamenti dei [dati annuali sulle macchie solari dal NOAA](#). Questi spettri di frequenza vengono tracciati utilizzando la funzione Matlab [iSignal](#) (pagina 366). In questo caso, i picchi nello spettro *non* sono dei picchi singoli e stretti, ma piuttosto formano un *agglomerato* di componenti di Fourier, perché *l'ampiezza e la frequenza non sono costanti* nell'arco dei quasi 300 anni dei dati, come è ovvio esaminando i dati nel dominio del tempo. I [forti brillamenti solari osservati nel 2024](#) cadono sul prossimo [massimo delle macchie solari](#) previsto e anche su un massimo delle [emissioni solari in radiofrequenza](#), che [ha effetti sulle comunicazioni e sulle reti elettriche](#).



Un esempio di serie temporale con periodicità multiple complesse sono le [visualizzazioni giornaliere delle pagine](#) ($x=\text{giorni}$,

y=visualizzazioni) per [questo sito web](#) su un periodo di 2070 giorni (circa 5.5 anni). [Nel grafico del periodogramma](#) (mostrato qui) si possono chiaramente vedere i picchi stretti a 7 e 3.5 giorni, corrispondenti alla prima e seconda armonica del previsto ciclo giorno feriale/weekend. Sono visibili anche picchi più piccoli a 365 (corrispondenti a un forte calo, ogni anno, delle vacanze invernali) e a 182 giorni (circa un semestre), probabilmente causato da un maggiore utilizzo nel ciclo semestrale biennale nelle università. I valori elevati nei tempi più lunghi sono causati dal graduale aumento dell'utilizzo in quel periodo di tempo, che può essere considerato come una componente a frequenza molto bassa il cui periodo è molto più lungo dell'intera registrazione dei dati.

Un altro esempio è mostrato di seguito; il segnale (nella finestra in alto) non contiene componenti

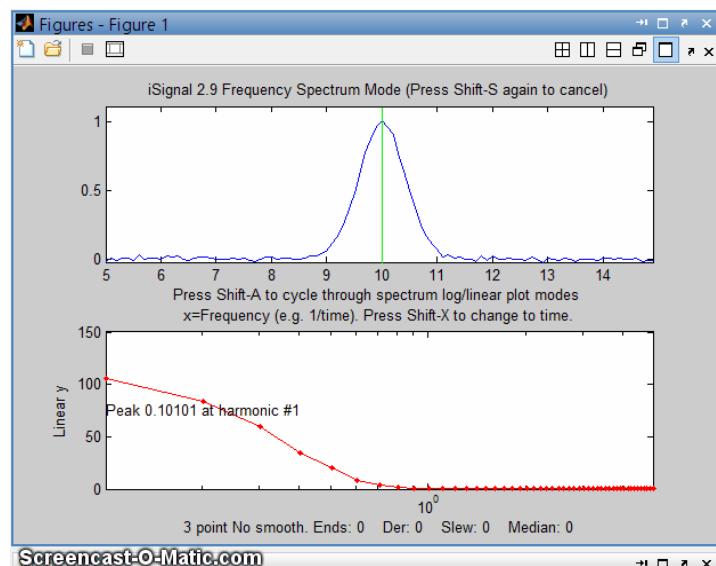
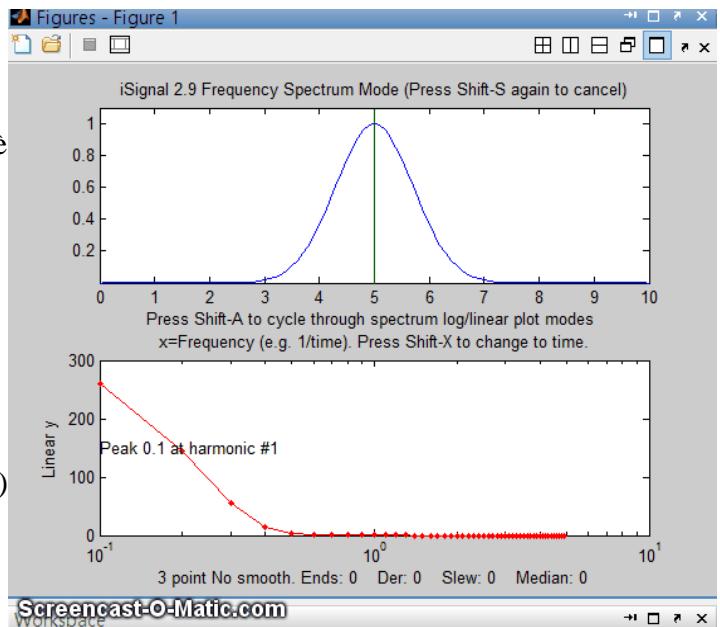


periodiche visivamente evidenti; *sembra* essere solo del rumore casuale. Tuttavia, lo spettro delle frequenze (nella finestra in basso) mostra che c'è molto di più, in questo segnale, di quanto sembri. Ci sono due componenti di frequenza principali: una a bassa frequenza intorno a 0.02 e l'altra ad alta frequenza tra 0.5 e 5. (Se le unità dell'asse x del grafico del segnale fossero state *secondi*, le unità del grafico dello spettro di frequenza sarebbero *Hz*; si noti che l'asse x è logaritmico). In questo caso, la componente di frequenza *più bassa* è in effetti il *segnale*

l'altra componente è [rumore blu](#), residuo delle precedenti operazioni di signal processing. Le due componenti sono fortunatamente ben separate sull'asse delle frequenze, suggerendo che il filtraggio passa-basso (ad esempio, lo smoothing, pagina 38) sarà in grado di rimuovere il rumore senza distorcere il segnale.

In tutti gli esempi mostrati sopra, i segnali sono segnali di serie temporali con la *frequenza* (o il *tempo*) come variabile indipendente. Più in generale, è anche possibile calcolare la trasformata di Fourier e lo spettro di potenza di *qualsiasi* segnale, come ad esempio uno *spettro ottico*, dove la variabile indipendente potrebbe essere la lunghezza d'onda o numero d'onda, o un *segnale elettrochimico*, dove la variabile indipendente potrebbe essere in volt, oppure un *segnale spaziale*, dove la variabile indipendente potrebbe essere in unità di lunghezza. In questi casi, le unità dell'asse x dello spettro di potenza sono semplicemente il reciproco delle unità dell'asse x del segnale originale (p.es. nm^{-1} per un segnale il cui asse x è in nm).

L'analisi degli spettri di frequenza dei segnali fornisce un altro modo per comprendere il rapporto segnale-rumore, il filtraggio, lo smoothing e la differenziazione. Lo smoothing è una forma di filtraggio *passa basso*, riducendo le componenti ad alta frequenza di un segnale. Se il segnale ha fattezze graduali, come i picchi Gaussiani, allora lo spettro sarà concentrato soprattutto alle *basse* frequenze. Maggiore è la larghezza del picco, più lo spettro sarà concentrato alle basse frequenze. (Se la figura seguente risulta animata, cliccare su questo [link](#)) Con un segnale con rumore bianco (distribuito uniformemente su tutte le frequenze), lo smoothing migliorerà il segnale, perché riduce le componenti ad alta frequenza del rumore. Tuttavia, il rumore a bassa frequenza rimarrà nel segnale dopo lo smoothing, continuando ad interferire con la misura di parametri come altezze, posizioni, larghezze e aree dei picchi. Questo si può dimostrare con una misura dei quadrati minimi.



negativo come un'onda sinusoidale e lo spettro in frequenza si sposta progressivamente verso le frequenze più alte. Questo comportamento è tipico di qualsiasi segnale con smoothing. Quindi l'intervallo ottimale per le informazioni di un *segnaletto differenziato* è limitato ad un intervallo relativamente ristretto, con poche informazioni utili al di sopra e al di sotto di tale intervallo.

Il fatto che il rumore bianco (pag. 28) sia distribuito nel dominio della frequenza più o meno equamente su tutte le frequenze ha un sottile vantaggio rispetto ad altri colori del rumore quando il segnale e il rumore non possono essere separati in modo netto nel dominio del tempo; è possibile stimare più facilmente l'intensità del rumore osservandolo nelle regioni di frequenza in cui il segnale non interferisce, dato che la maggior parte dei segnali non occupa tutta la gamma di frequenze dello spettro. Questa idea verrà utilizzata in seguito come metodo per stimare gli errori delle misure basate sull'approssimazione della curva con i minimi quadrati di dati rumorosi (pag. 162).

La tecnica dello peak sharpening (pag. 74) enfatizza anche le componenti ad alta frequenza aggiungendo una porzione delle derivate seconda e quarta al segnale originale. Lo si può vedere chiaramente nello script Matlab/Octave [PeakSharpeningFrequencySpectrum.m](#), che mostra lo

Al contrario, la differenziazione è una forma di filtro *passa alto*, riducendo le componenti in *bassa* frequenza di un segnale ed enfatizzando tutte le componenti in *alta* frequenza presenti nel segnale. Un semplice picco Gaussiano generato al computer mostrato sopra; (cliccare per l'[animazione GIF](#)) ha la maggior parte della sua potenza concentrata in alcune frequenze basse, ma applicando ordini successivi di differenziazione (cerchi gialli), la forma d'onda della derivata oscilla da positiva a

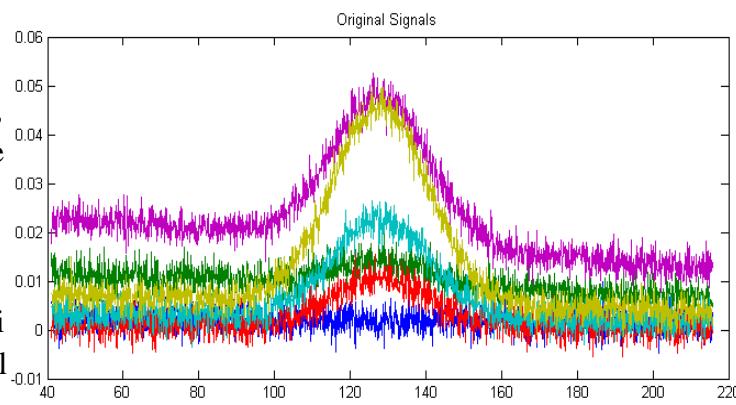
spettro di frequenza della versione originale e di quella con sharpening di un segnale costituito da diversi picchi ([grafico](#)).

[SineToDelta.m](#). Un'animazione dimostrativa (click per il [grafico animato](#)) che mostra la forma d'onda e lo spettro di potenza di un'onda sinusoidale attivata da un impulso rettangolare di durata variabile (il cui spettro di potenza è una funzione "sinc") che cambia continuamente da un'onda sinusoidale pura da un estremo (dove il suo spettro di potenza è una funzione delta) fino ad arrivare a un impulso a punto singolo all'altro estremo (dove il suo spettro di potenza è una linea piatta).

[GaussianSineToDelta.m](#) è simile, tranne per il fatto che mostra un'onda sinusoidale attivata da una *Gaussiana* pulsante, il cui spettro di potenza è una funzione Gaussiana, ma che ha lo stesso comportamento ai due estremi della durata dell'impulso ([grafico animato](#)).

I segnali sperimentali reali sono spesso contaminati da deriva e spostamento della linea di base, che sono essenzialmente effetti a *bassa frequenza* e rumore casuale, solitamente distribuito su *tutte le frequenze*. Per questi motivi, la

differenziazione viene sempre utilizzata insieme allo smoothing. Lavorando assieme, smoothing e differenziazione agiscono come una sorta di filtro *passa banda selettivo* che fa passare in modo ottimale la banda di frequenze contenente le informazioni del segnale differenziato ma riduce sia gli effetti delle *frequenze più basse*, come la deriva e il background, e sia il rumore alle *alte*



frequenze. Un esempio si può vedere nel [DerivativeDemo.m](#) descritto in un paragrafo precedente (pag. 71). Nell'insieme di sei segnali originali, mostrato sopra in diversi colori, il rumore casuale si verifica principalmente alle alte frequenze, con *molti cicli* lungo l'asse x, e il fenomeno dello slittamento della linea di base si verifica a frequenze inferiori, con una sola *piccola frazione di ciclo* in tale intervallo. Al contrario, il picco in esame, al centro delle x, occupa una gamma di frequenze *intermedie*, con *alcuni cicli* in tale intervallo. Pertanto, si potrebbe prevedere che una misura quantitativa basata sulla differenziazione e lo smoothing potrebbe funzionare bene, perché si enfatizzano le frequenze intermedie.

Smoothing e differenziazione cambiano le *ampiezze* delle varie componenti delle frequenze del segnale, ma non cambiano, né spostano le frequenze stesse. Un esperimento descritto in seguito (pag. 383) illustra quest'idea applicando lo smoothing e la differenziazione ad una breve registrazione del linguaggio umano. È interessante notare che diversi gradi di smoothing e differenziazione cambiano il [timbro](#) della voce ma hanno uno *scarsa effetto sull'intelligibilità*; dato che la sequenza delle intonazioni, 'pitch', non si sposta né in tonalità né nel tempo, ma semplicemente cambia l'ampiezza a causa dello smoothing e della differenziazione. Per questo motivo, il parlato registrato può sopravvivere alla digitalizzazione, alla trasmissione su lunghe distanze, agli algoritmi di compressione e alla riproduzione tramite minuscoli altoparlanti e cuffie senza una significativa perdita di intelligibilità. La musica, d'altra parte, subisce una perdita maggiore in tali circostanze, come si può vedere ascoltando la [musichetta di attesa telefonica](#), che risulta spesso terribile *anche se il parlato sulla stessa linea è completamente comprensibile*, perché la musica ha una struttura di frequenza diversa dal parlato. Gli impianti cocleari per non udenti hanno la stessa limitazione (come drammatizzato nel film "Sound of Metal" del 2019).

Dettagli Software

In uno spreadsheet o in un linguaggio per computer, un'onda sinusoidale si può descrivere con la funzione 'sin' $y=\sin(2\pi f x + p)$ o $y=\sin(2\pi(1/t)x + p)$, dove π è 3.14159..., f è la *frequenza* della forma

d'onda, t è il *periodo* della forma d'onda, p è la *fase* e x è la variabile indipendente (solitamente il tempo).

Ci sono diversi [siti Web](#) che calcolano le trasformate di Fourier interattivamente (p.es. [WolframAlpha](#)). Microsoft Excel ha una funzione aggiuntiva (add-in) che consente di eseguire trasformate di Fourier in modo relativamente semplice: (Cliccare su **Tools > Add-Ins... > Analysis Toolpak > Fourier Analysis**). Si veda "[Excel e Fourier](#)" per i dettagli. Vedere "*Excellaneous*" (<http://www.bowdoin.edu/~rdelevie/excellaneous/>) per una vasta ed eccellente raccolta di funzioni aggiuntive e macro per Excel, del Dr. Robert deLevie del Bowdoin College. Ci sono diversi programmi dedicati all'analisi spettrale FFT, compreso **ScopeDSP** (<https://iowegian.com/scopedsp/>) e **Audacity** (<http://sourceforge.net/projects/audacity/>). Se si sta leggendo questo online, si può fare **Ctrl-Click** su questi link per aprire automaticamente questi siti

Matlab e Octave

Matlab e Octave hanno funzioni native per il calcolo della trasformata di Fourier ([fft](#) e [ifft](#)). Queste funzioni esprimono i loro risultati come numeri complessi. Ad esempio, se si calcola la trasformata di Fourier di un semplice vettore a 3 elementi, si ottiene un risultato a 3 elementi di numeri complessi:

```
y=[0 1 0];  
fft(y)  
ans = 1.0000 -0.5000-0.8660i -0.5000+0.8660i
```

dove la "i" indica la parte "immaginaria". Il primo elemento della fft è solo la somma degli elementi in y . La fft inversa, [ifft\(\[1.0000 -0.5000-0.8660i -0.5000+0.8660i\]\)](#), restituisce il vettore originale [\[0 1 0\]](#).

Per un altro esempio, la fft di $[0 \ 1 \ 0 \ 1]$ è $[2 \ 0 \ -2 \ 0]$. In generale, la fft di un vettore di n elementi di numeri reali restituisce un vettore di n elementi di numeri reali o complessi, ma solo i primi $n/2+1$ elementi sono unici; il resto è un'immagine speculare del primo. Le operazioni sui singoli elementi della fft, come nel [filtraggio di Fourier](#), devono tener conto di questa struttura.

Lo spettro della frequenza "s" di un vettore "y" di un segnale si può calcolare come [real\(sqrt\(fft\(s\).*conj\(fft\(s\)\)\)\)](#). Ecco un semplice esempio di cui si conosce anticipatamente la risposta, almeno qualitativamente: un vettore di 8 elementi di interi che disegnano un *singolo ciclo di un'onda sinusoidale*:

```
y=[0 7 10 7 0 -7 -10 -7];  
s=real(sqrt(fft(y).*conj(fft(y))))
```

Lo spettro di frequenza in questo caso è $[0 \ 39.9 \ 0 \ 0.201 \ 0 \ 0.201 \ 0 \ 39.9]$.

In **Python**, la sintassi è simile: [y=np.array\(\[0, 7, 10, 7, 0, -7, -10,-7\]\)](#) [s=np.real\(np.sqrt\(fft.fft\(y\)*np.conj\(fft.fft\(y\)\)\)\)](#). Ancora una volta, il primo elemento è la media (che è zero) e gli elementi da 2 a 4 sono l'immagine speculare degli ultimi 4. Gli elementi unici sono i primi quattro, che sono le ampiezze delle componenti dell'onda sinusoidale le cui frequenze sono 0, 1, 2, 3 volte la frequenza di un'onda sinusoidale che approssimerebbe un singolo ciclo del periodo del segnale. In questo caso, il *secondo* elemento (39.8) è di gran lunga il più grande, che è proprio quello che ci aspetteremmo per un segnale che si avvicina a un singolo ciclo di un *seno* (piuttosto che un'onda *coseno*). Se il segnale fosse stato *due* cicli di un'onda sinusoidale, $s = [0 \ 10 \ 0 \ -10 \ 0 \ 10 \ 0 \ -10]$, il *terzo* elemento sarebbe stato il più forte (da provare). La fre-

quenza più alta rappresentabile da un vettore di 8 elementi è quella che ha un periodo pari a 2 elementi. Sono necessari almeno 4 punti per mostrare un ciclo, ad es. [0 +1 0 -1].

Se si sta leggendo online, cliccare [qui](#) per uno script Matlab che crea e disegna un'onda sinusoidale e poi usa la funzione fft per calcolarne e disegnarne lo spettro della frequenza. Da provare con diverse frequenze (terza riga). Vedere cosa succede quando la frequenza si avvicina a 50. Suggerimento: la frequenza di Nyquist è $1/(2*\text{Deltat}) = 1/0.02=50$. Da vedere, inoltre, cosa succede cambiando Deltat (prima riga), che determina quanto finemente viene campionata l'onda sinusoidale.

La funzione [FrequencySpectrum.m](#) (sintassi `fs=FrequencySpectrum(x,y)`) restituisce la parte reale dello spettro della potenza di Fourier di x,y come matrice. [PlotFrequencySpectrum.m](#) disegna spettri di frequenze e periodogrammi in coordinate lineari o logaritmiche. Digitare "help PlotFrequencySpectrum" o provare questo esempio:

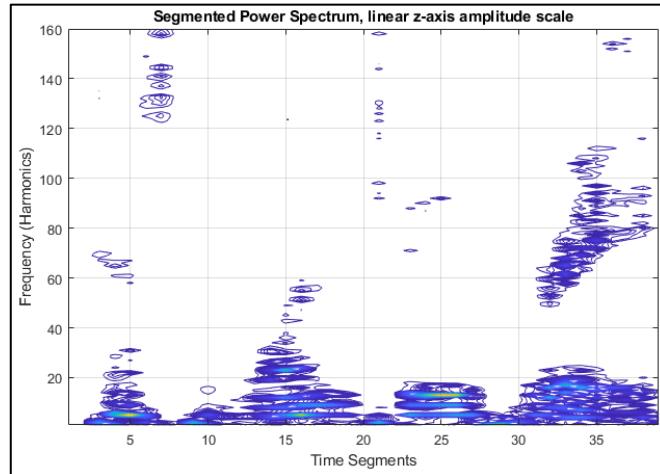
```
x=[0:.01:2*pi]';
f=25; % Frequency
y=sin(2*pi*f*x)+randn(size(x));
subplot(2,1,1);
plot(x,y);
subplot(2,1,2);
FS=PlotFrequencySpectrum(x,y,1,0,1);
```

Il grafico dello spettro delle frequenze FS (`plotit(FS);` [grafico](#)) mostra un unico fortissimo picco a 25. La frequenza del picco più forte in FS è data da `FS(val2ind(FS(:,2),
max(FS(:,2))), 1)`.

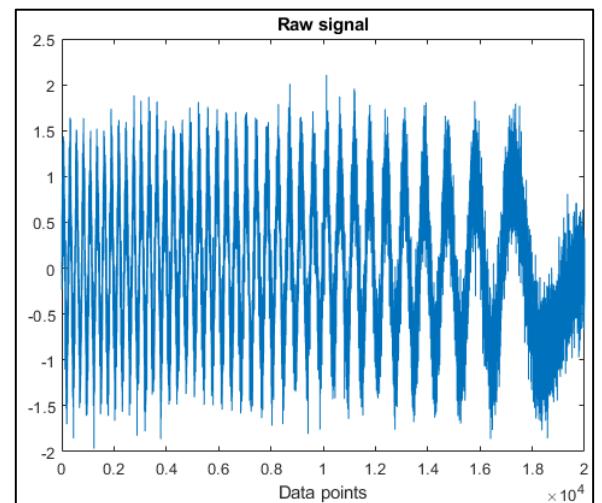
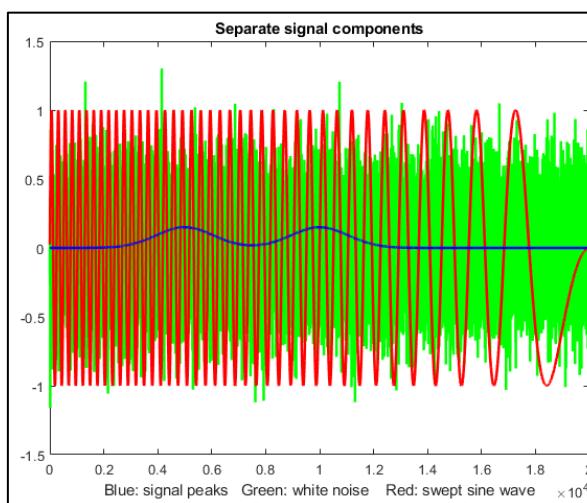
Per qualche altro esempio sull'uso della FFT, si vedano [questi esempi](#). Una funzione per la "[Trasformata Lenta di Fourier](#)" è stata [pubblicata](#); è da *3000 a 7000 volte più lenta* con un vettore di 10,000 punti, come si può vedere da questo pezzo di codice che si può copiare/incollare nella riga di comando di Matlab: `y=cos(.1:.01:100); tic; fft(y); ffttime=toc; tic; sft(y); sftime=toc; TimeRatio=sftime/ffttime.`

Spettro di Fourier della potenza segmentato nel tempo.

La funzione [PlotSegFreqSpect.m](#), sintassi `PSM = PlotSegFreqSpect(x, y, NumSegments, MaxHarmonic, logmode)`, crea e visualizza un spettro di potenza di Fourier *segmentato nel tempo*. Suddivide il segnale in segmenti di uguale lunghezza "NumSegments", moltiplica ciascuno per una finestra di Hanning apodizzante, calcola lo spettro di potenza di ciascun segmento e traccia la grandezza delle prime componenti di Fourier "MaxHarmonic" rispetto al numero del segmento come [disegno del contorno](#). La funzione restituisce la matrice dello spettro di potenza (tempo-frequenza-ampiezza) di dimensione NumSegments x MaxHarmonic. Se logmode=1, calcola e disegna il logaritmo in base 10 delle ampiezze come curve di livello con diversi colori per rappresentare le altezze (blu=basso; giallo=alto). Altri esempi pratici nel file di help comprendono lo spettro del [clacson di un'auto di passaggio](#), mostrando l'effetto Doppler e un [campione di parlato umano](#) (sopra a sinistra).



L'esempio seguente ([script](#)) mostra un segnale complesso costituito da tre componenti sommate insieme (in basso a sinistra): due deboli picchi Gaussiani a $x=5000$ e 10000 (blu) con altezza=0.15,



una forte interferenza sinusoidale variabile in frequenza (rosso) e del rumore bianco (verde). Quando si sommano tutte queste tre componenti, i picchi Gaussiani restano completamente sepolti e sono invisibili nel segnale originale, in alto a destra. (Lo si chiamerà segnale dei "picchi sepolti", e lo si userà ancora più avanti, pagina 128). L'ispezione della funzione PlotSegFreqSpect (a sinistra) rivela la forte striscia gialla diagonale della 'spazzola' dell'onda sinusoidale e il blu e bianco del background del rumore casuale, ma sono anche visibili due macchie gialle nei segmenti temporali 2 e 4 in basso nella parte inferiore dello spettro segmentato (a sinistra). Ciò che si vede è che c'è qualcosa intorno al 2° e al 4° segmento temporale a frequenze più alte rispetto all'area circostante. Una volta constatato, si può limitare l'intervallo per ulteriori osservazioni e si possono evidenziare i picchi con lo [smoothing per ridurre le alte frequenze](#) o con l'[approssimazione dei dati originali](#) (introdotta a pagina 165). Infatti, i risultati dell'approssimazione della curva mostrati di seguito forniscono valori buoni ($\pm 10\%$ o migliori) per le posizioni del picco (valori reali =5000 e 10000), altezze (0,15), e larghezze (2500), nonostante l'invisibilità dei picchi nei dati originali.

	Peak#	Position	Height	Width	Area
1	5031.4	0.15749	2280.6	382.33	
2	10036	0.16136	2407.2	413.46	

Ma i picchi si devono *vedere* per sapere come provarlo. In base ai soli dati originali, si potrebbe non riuscire a provarlo.

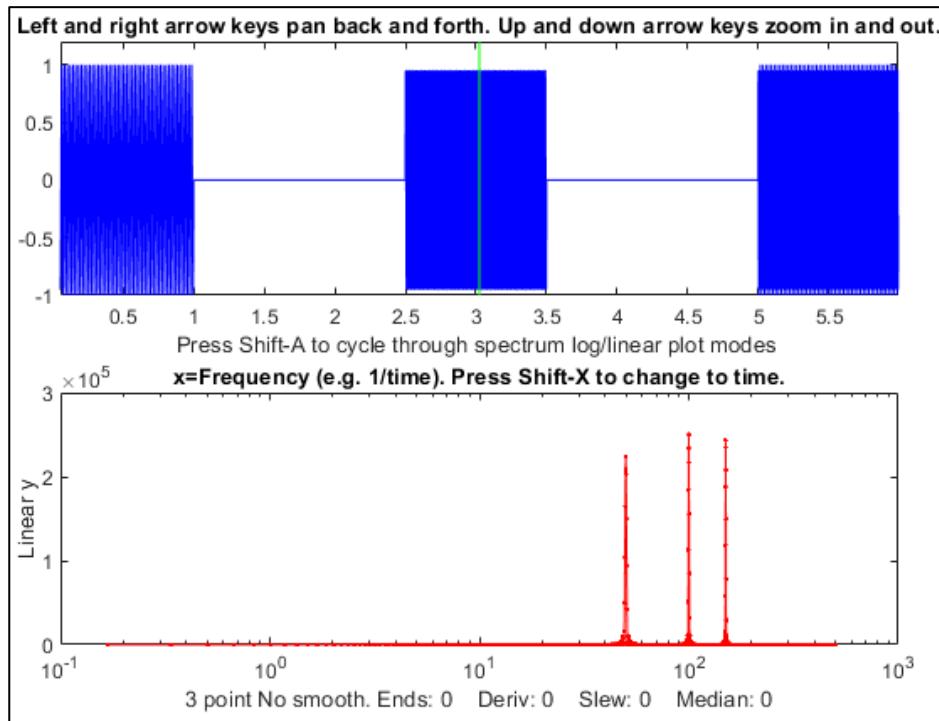
Osservazione dello spettro di Frequenza con [iSignal](#)

iSignal (pagina 366) è uno strumento interattivo multiuso per il signal processing che ha una [modalità di Spettro di Frequenza](#), si attiva/disattiva col tasto **Shift-S**; calcola lo spettro di frequenza del segmento del segnale visualizzato nella finestra superiore mostrandolo in quella inferiore (in rosso). Si possono usare i tasti pan e zoom per regolare la regione del segnale da visualizzare o premere **Ctrl-A** per selezionare l'intero segnale. Si preme ancora **Shift-S** per tornare nella modalità normale. Nella modalità dello spettro di frequenza, si può premere **Shift-A** per attraversare le quattro modalità di disegno (lineare, semilog X, semilog Y o log-log). A causa

dell'ampia gamma di ampiezze e frequenze di alcuni segnali, le modalità di disegno logaritmico producono spesso un grafico più chiaro rispetto alla modalità lineare. Si può premere anche **Shift-X** per passare sull'asse x la *frequenza* o il *tempo*. Dettagli e istruzioni sono a pagina 366. Si può scaricare un [file ZIP](#) che contiene la versione 8 di iSignal.m e qualche demo e dei campioni per il test.

Visualizzazione della frequenza.

Cosa succede se il contenuto in frequenza cambia nel tempo? Si consideri, ad esempio, il segnale mostrato nella figura seguente. Il segnale (scaricabile da [SineBursts.mat](#)) è costituito da tre raffiche di onde sinusoidali a tre diverse frequenze, separate con degli zeri.

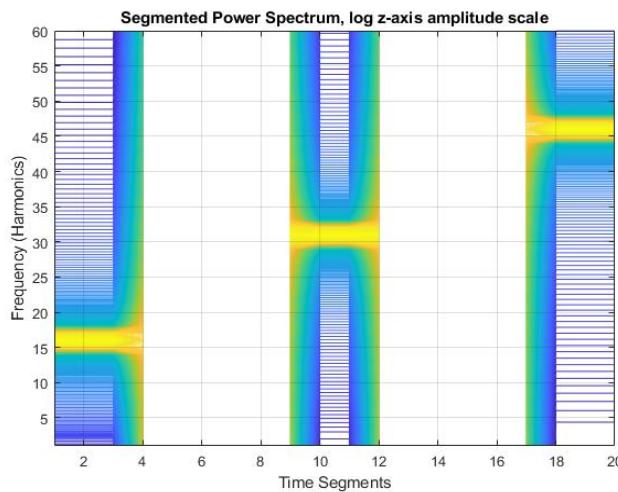
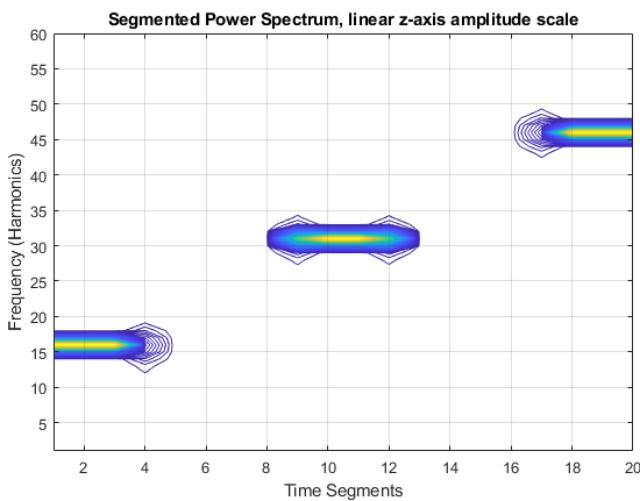


La funzione Matlab iSignal.m mostra un segnale (pannello superiore) e il suo spettro di frequenza (pannello inferiore).

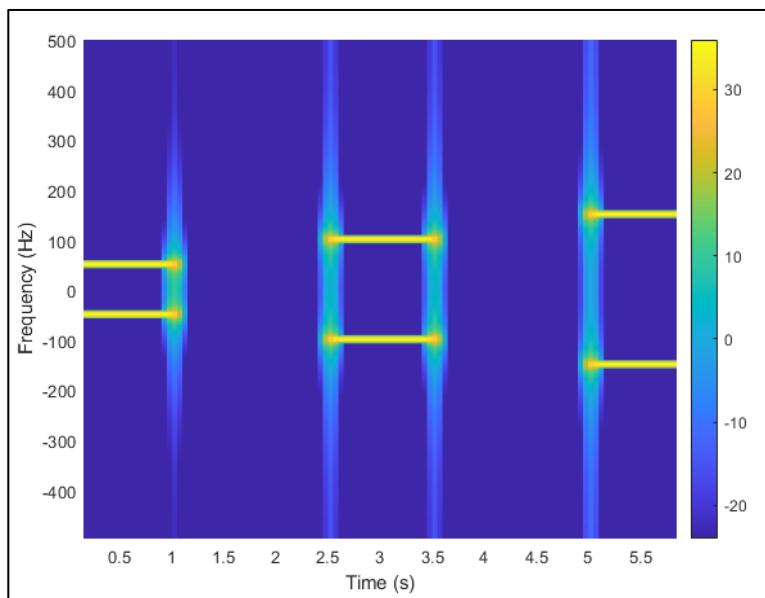
Qui il segnale viene mostrato nel pannello superiore nella funzione [iSignal.m](#) (pagina 366) digitando:

```
load SineBursts
isignal(x,y);
```

sulla riga di comando di Matlab. Premendo **Shift-S**, lo spettro delle frequenze appare nel pannello inferiore, mostrando le tre componenti discrete della frequenza. Ma chi appartiene a chi? La normale trasformata di Fourier di per sé non offre alcun indizio, ma iSignal consente di eseguire il pan e lo zoom nel segnale, utilizzando i tasti freccia, in modo da poterne isolare uno per volta. Alternativamente, la funzione [PlotSegFreqSpec.m](#), appena descritta nella sezione precedente (figure seguenti) è un altro modo per visualizzare il segnale in un *unico grafico statico che mostra chiaramente la variazione nel tempo della frequenza del segnale*.



È possibile eseguire una visualizzazione simile con la funzione nativa di Matlab “Short Time

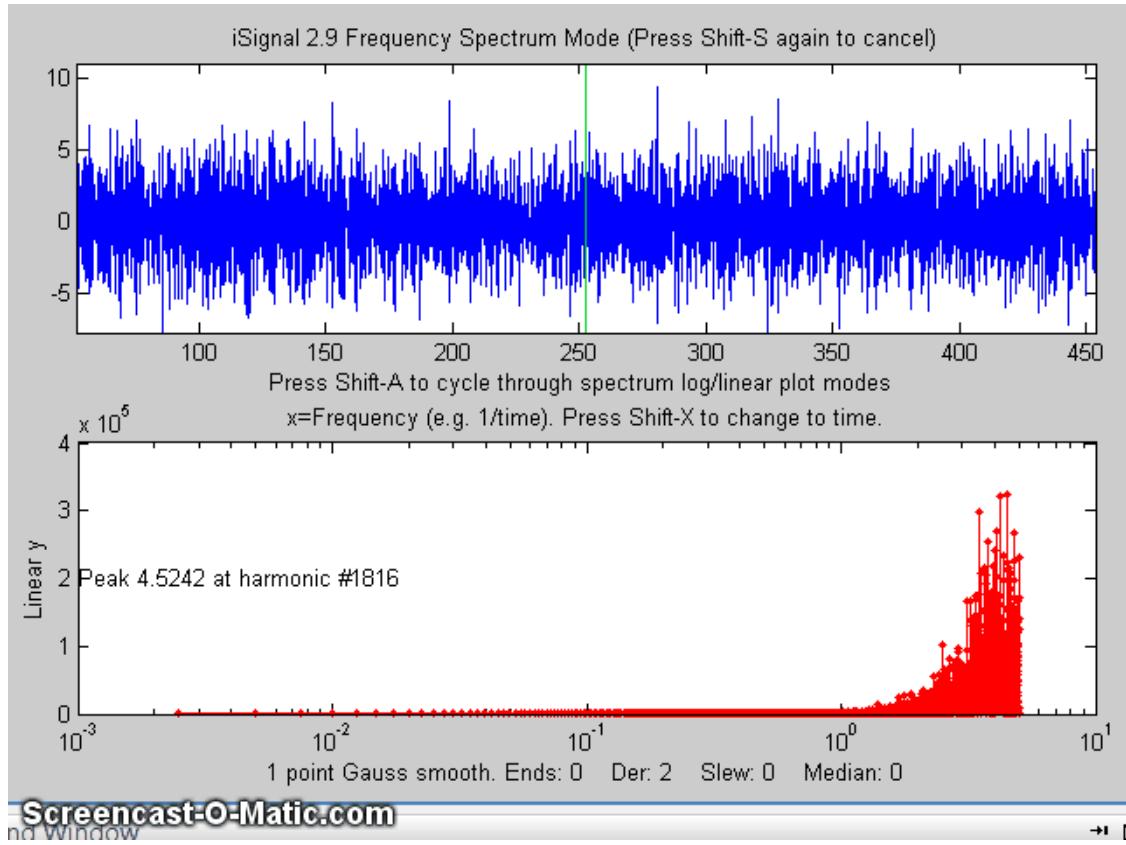


Fourier Transform” stft.m (sotto), che mostra sia le frequenze positive che quelle negative.

Miglioramento del segnale

Una caratteristica molto importante di iSignal è che *tutte le funzioni di signal processing restano attive nella modalità a spettro di frequenza* (smoothing, derivata, ecc.), quindi si può osservare immediatamente l'effetto di tali funzioni sullo spettro di frequenze del segnale. Alcune operazioni per l'elaborazione dei segnali possono avere l'effetto collaterale di aumentare l'influenza del rumore casuale o di distorcere il segnale. Il vantaggio di iSignal è che si può osservare direttamente il

compromesso tra l'effetto desiderato e quelli collaterali regolando interattivamente le variabili per l'elaborazione del segnale. La figura sulla prossima pagina mostra un esempio. Viene illustrato l'effetto dell'aumento dell'ampiezza dello smoothing sulla derivata [2^a](#) di un segnale contenente tre deboli picchi rumorosi. Senza smoothing, il segnale sembra essere tutto rumore casuale; con un sufficiente smoothing, i tre deboli picchi diventano chiaramente visibili (in forma derivativa) e misurabili.



L'[animazione sopra](#) mostra la modalità spettro di frequenza di *iSignal.m*, mentre l'ampiezza dello smoothing si cambia con i tasti **A** e **Z**. Questo mostra drammaticamente come il segnale (pannello superiore) e lo spettro di frequenza (sotto) sono entrambi influenzati dalla larghezza dello smoothing. (Se si sta leggendo online, cliccare per l'[animazione GIF](#).)

Lo script “[iSignalDeltaTest](#)” mostra la risposta in frequenza delle funzioni di smoothing e di differenziazione di *iSignal* applicandole ad una [funzione delta](#). Modificando il tipo di smoothing, l'ampiezza e l'ordine di derivazione si vedrà come cambia lo spettro della potenza.

Dimostrazione che lo spettro di Fourier di una Gaussiana è anche una Gaussiana

Una cosa speciale (per matematici) riguardo al segnale con profilo *Gaussiano* rispetto a tutte le altre forme è che lo spettro di frequenza di Fourier di una Gaussiana è *anch'esso* una Gaussiana. Lo si può dimostrare numericamente scaricando le funzioni [gaussian.m](#) e [isignal.m](#) ed eseguendo le seguenti istruzioni:

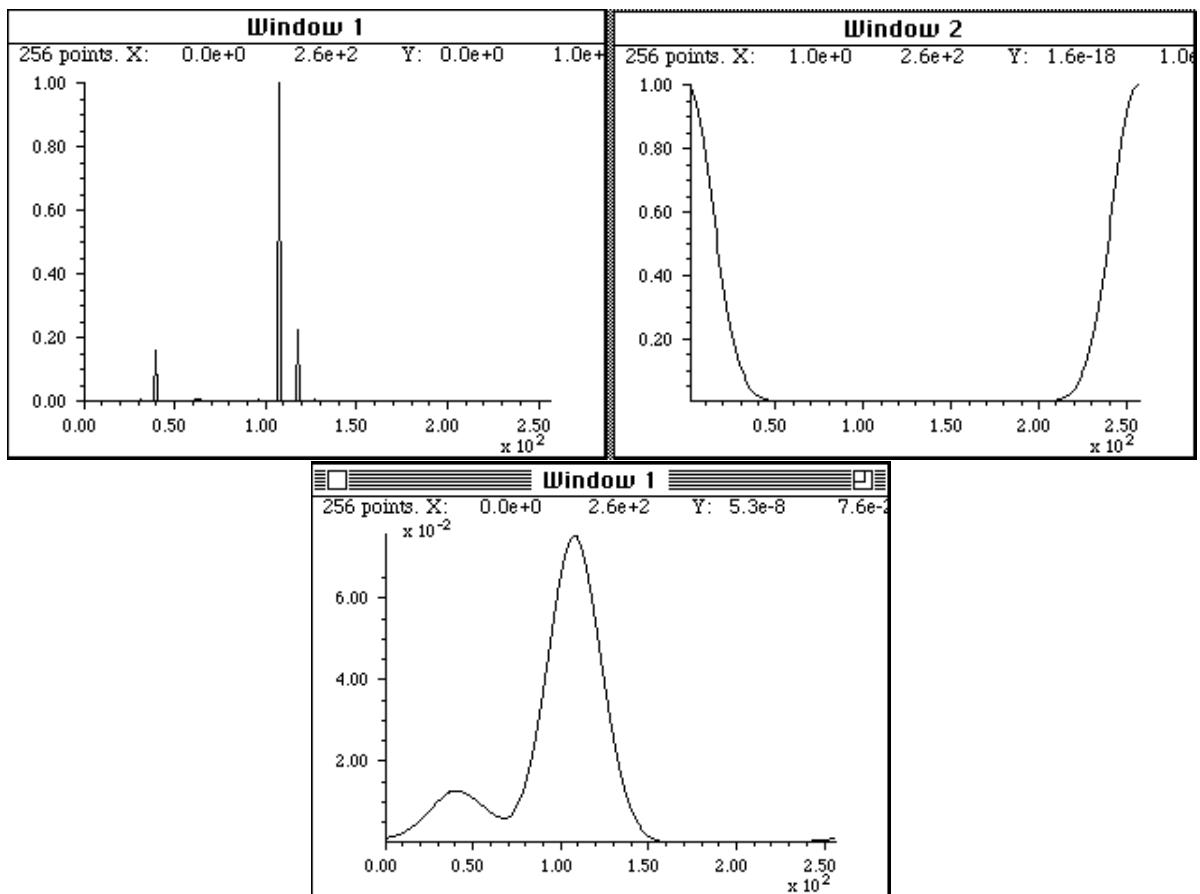
```
x=-100:.2:100;
width=2; y=gaussian(x,0,width);
isignal([x;y],0,400,0,3,0,0,0,10,1000,0,0,1);
```

Cliccare sulla finestra della figura, premere **Shift-T** per trasferire lo spettro di frequenza nel pannello superiore, poi premere **Shift-F**, premere **Enter** tre volte e cliccare sul picco nella finestra superiore. Il programma calcola un'approssimazione ai quadrati minimi di un modello Gaussiano allo spettro di frequenza ora nel pannello superiore. L'approssimazione è essenzialmente perfetta. Ripetendo questo con Gaussiane di *diverse larghezze* (p.es. width=1 o 4), si troverà che la larghezza

del picco nello spettro di frequenza è *inversamente proporzionale* alla larghezza del picco del segnale. Al limite di un picco infinitamente *stretto*, la Gaussiana diventa una *funzione delta* e il suo spettro di frequenza è piatto. Al limite di un picco infinitamente *largo*, la Gaussiana diventa una linea piatta e lo spettro di frequenza è diverso da zero solo alla frequenza zero.

Convoluzione di Fourier

La convoluzione è un'operazione eseguita su due segnali che comporta la moltiplicazione di un segnale per una versione ritardata o traslata di un altro segnale, integrando o mediando il prodotto e ripetendo il processo per diversi ritardi. La convoluzione è un processo utile perché descrive accuratamente alcuni effetti che si verificano ampiamente nelle misurazioni scientifiche, come l'influenza di un [filtro in frequenza di un segnale elettrico](#) o della [passa banda spettrale di uno spettrometro](#) sulla forma di un segnale ottico registrato, che provoca la diffusione del segnale nel tempo e la riduzione dell'ampiezza del picco.



La convoluzione di Fourier qui viene usata per determinare come apparirà lo spettro ottico in Window 1 (in alto a sinistra) scansionato con uno spettrometro la cui funzione di feritoia (risoluzione spettrale) è descritta dalla funzione Gaussiana in Window 2 (in alto a destra). La funzione Gaussiana è già stata ruotata in modo che il suo massimo cada in $x=0$. La convoluzione risultante dello spettro ottico (in basso al centro) mostra che le due linee vicino a $x=110$ e 120 non saranno risolte, ma la linea in $x=40$ lo sarà parzialmente. La convoluzione di Fourier viene usata in questo modo per correggere la non linearità della curva analitica provocata dalla risoluzione dello spettrometro, nella [spettroscopia di assorbimento](#) in iperlineare (Pagina 268).

In pratica, spesso si esegue il calcolo moltiplicando punto per punto i due segnali nel dominio di Fourier. Innanzitutto, si ottiene la trasformata di Fourier di ciascun segnale. Poi le due trasformate di Fourier vengono moltiplicate punto per punto con le regole per la moltiplicazione complessa e il risultato elaborato poi con la trasformata inversa di Fourier. Le trasformate di Fourier sono

solitamente espresse in termini di "[numeri complessi](#)", con e parti reali e immaginarie; se la trasformata di Fourier del primo segnale è $a + ib$, e quella del secondo segnale è $c + id$, allora il prodotto delle due trasformate di Fourier è $(a + ib)(c + id) = (ac - bd) + i(bc + ad)$. Sebbene questo sembri essere un metodo di arrotondamento, risulta essere più veloce dell'algoritmo trasla-e-moltiplica quando il numero di punti nel segnale è grande. La convoluzione si può usare come algoritmo potente e generale per lo smoothing e la differenziazione. Molti linguaggi per computer eseguiranno automaticamente questa operazione quando le due quantità da dividere sono numeri complessi. Nei testi matematici, la convoluzione viene spesso indicata col simbolo * ([Riferimento](#)).

La convoluzione di Fourier viene utilizzata come algoritmo molto generale per lo smoothing e la differenziazione dei segnali digitali, eseguendo la convoluzione del segnale con un insieme di numeri (solitamente) piccolo che rappresenta il vettore di convoluzione. Lo smoothing si esegue per convoluzione con un set di numeri positivi, p. es. [1 1 1] per una "boxcar" a 3 punti. La convoluzione con [-1 1] calcola una derivata prima; [1 -2 1] calcola una derivata seconda. Le convoluzioni successive di Conv1 e poi di Conv2 equivalgono ad una convoluzione con la convoluzione di Conv1 e Conv2. La derivata prima con lo smoothing viene fatta usando un vettore di convoluzione in cui nella prima metà i coefficienti sono negativi e nella seconda metà sono positivi (p. es. [-1 -2 0 2 1]).

Semplici vettori di convoluzione a numeri interi

Vettori per le differenziazioni:

[-1 1] Derivata prima

[1 -2 1] Derivata seconda

[1 -3 3 -1] Derivata terza

[1 -4 6 -4 1] Derivata quarta

Risultati di convoluzioni successive di due vettori Conv1 e Conv2: (* sta per convoluzione)

Conv1	Conv2	Risultato	Descrizione
[1 1 1]	*	[1 1 1]	= [1 2 3 2 1] Smoothing triangolare
[1 2 1]	*	[1 2 1]	= [1 4 6 4 1] Smoothing P-spline
[-1 1]	*	[-1 1]	= [1 -2 1] Derivata seconda
[-1 1]	*	[1 -2 1]	= [1 -3 3 -1] Derivata terza
[1 -2 1]	*	[1 -2 1]	= [1 -4 6 -4 1] Derivata quarta
[-1 1]	*	[1 1 1]	= [1 0 0 -1] [gap-segment] della derivata prima
[-1 1]	*	[1 2 1]	= [1 1 -1 -1] Derivata prima con smoothing
[1 1 -1 -1]	*	[1 2 1]	= [1 3 2 -2 -3 -1] Lo stesso con più smoothing
[1 -2 1]	*	[1 2 1]	= [1 0 -2 0 1] Derivata seconda [gap-segment]
[1 1 1 1]	*	[1 1 1 1]	= [1 2 3 4 3 2 1] 2 passaggi dello slittamento-della-media a 4 punti

Rettangolo * rettangolo = triangolo o trapezio, a seconda delle larghezze relative.

Gaussiana * Gaussiana = Gaussiana di larghezza maggiore.

Gaussiana * Lorentziana = Profilo Voigt (cioè qualcosa tra Gaussiana e Lorentziana, a seconda delle larghezze relative. Simile (ma non identica) alla somma pesata di una Gaussiano o Lorentziano).

Dettagli software per la convoluzione

Gli **spreadsheet** si possono usare per eseguire la convoluzione "trasla e moltiplica" per set di dati campionati digitalmente (ad esempio, [MultipleConvolution.xlsx](#), [MultipleConvolutionFirstDerivativeDemo.xls \(schermata\)](#) e [MultipleConvolution4thDerivativeDemo.xls \(schermata\)](#) per Excel e [MultipleConvolutionOO.ods](#) per Calc), ma per insiemi di dati più grandi le prestazioni sono più lente della convoluzione di Fourier (che è più semplice da eseguire in Matlab o Octave che nei fogli di calcolo). I fogli di calcolo, tuttavia, mostrano l'operazione di ""trasla e moltiplica"" cella per cella in modo più chiaro ed esplicito.

Matlab e **Octave** hanno una funzione nativa per la convoluzione di due vettori: **conv**. Questa funzione è utilizzabile per creare filtri e funzioni di smoothing, come lo smoothing a [slittamento della media](#) e quello [triangolare](#). Per esempio,

```
ysmoothed=conv(y,[1 1 1 1 1], 'same') ./5;
```

esegue lo smoothing del vettore y con uno slittamento non pesato della media a 5 punti (boxcar), e

```
ysmoothed=conv(y,[1 2 3 2 1], 'same') ./9;
```

esegue lo smoothing triangolare del vettore y con 5 punti. L'argomento opzionale 'same' restituisce la parte centrale della convoluzione che ha la stessa dimensione di y. Se l'argomento opzionale è "full", la lunghezza del risultato è di uno inferiore alla somma delle lunghezze dei due vettori.

La differenziazione viene effettuata con lo smoothing utilizzando un vettore di convoluzione in cui la prima metà dei coefficienti è negativa e la seconda metà è positiva (p.es. [-1 0 1], [-2 -1 0 1 2], o [-3 -2 -1 0 1 2 3]) per calcolare una derivata prima con una quantità crescente di smoothing.

La funzione **conv** in Matlab/Octave si può usare facilmente per combinare operazioni successive di convoluzione, per esempio, una derivata seconda seguita da uno smoothing triangolare di 3 punti:

```
>> conv([1 -2 1],[1 2 1])
ans = 1 0 -2 0 1
```

Il prossimo esempio crea una funzione di trasferimento a trascinamento esponenziale [exponential trailing] (c), che ha un effetto simile al semplice filtro passa basso RC e lo applica a y.

```
c=exp(-(1:length(y))./30);
yc=conv(y,c,'full')./sum(c);
```

In ciascuno dei tre esempi precedenti, il risultato della convoluzione viene diviso per la somma della funzione di trasferimento della convoluzione, per garantire che la convoluzione abbia un guadagno netto di 1.000 e quindi non influenzi l'area sotto la curva del segnale. Ciò rende l'operazione matematica più vicina alle convoluzioni fisiche che diffondono il segnale nel tempo e riducono l'ampiezza del picco ma conservano l'energia totale nel segnale, che per un segnale di tipo picco è proporzionale all'area sotto la curva.

In alternativa, si può eseguire la convoluzione *senza* utilizzare la funzione nativa di Matlab/Octave "conv", moltiplicando le trasformazioni di Fourier di *y* e di *c* utilizzando la funzione "fft.m" e poi la trasformazione inversa del risultato con la funzione "ifft.m". I risultati sono essenzialmente gli stessi e il tempo trascorso è leggermente più veloce rispetto all'utilizzo della funzione conv. Tuttavia, c'è dev'essere completato con degli zeri per coincidere con la dimensione di *yc* perché la moltiplicazione, o la divisione, punto-per-punto di due vettori richiede che abbiano la stessa lunghezza. La funzione "conv" esegue automaticamente qualsiasi riempimento a zero richiesto.

```
yc=ifft(fft(y).*fft(c));
```

Quando si utilizza la convoluzione ai fini dello smoothing, è auspicabile che l'area sotto la curva *y* rimanga la stessa dopo lo smoothing. Ciò è facilmente garantito dividendo per la somma dei membri di *c*:

```
yc=ifft(fft(y).*fft(c))./sum(c);
```

[**GaussConvDemo.m**](#) mostra che una Gaussiana di altezza unitaria convoluta con una Gaussiana della stessa larghezza è una Gaussiana con un'altezza di $1/\sqrt{2}$ e una larghezza di $\sqrt{2}$ e con la stessa area della Gaussiana originale. (La Window 2 mostra un tentativo di recuperare la "y" originale dalla "yc" convoluta con la funzione deconvgauss). Si può facoltativamente aggiungere del rumore nella riga 9 per vedere come la convoluzione filtra il rumore e come la deconvoluzione lo ripristina. Richiede gaussian.m, peakfit.m e deconvgauss.m nel path di Matlab.

[**iSignal**](#) (pagina 366) ha un tasto **Shift-V** che mostra il menù delle operazioni di convoluzione/deconvoluzione che consentono di eseguire la convoluzione di una funzione Gaussiana o esponenziale col segnale e richiede la larghezza Gaussiana o la costante di tempo (in unità X).

Fourier convolution/deconvolution menu

1. Convolution

2. Deconvolution

Select mode 1 or 2: 1

Shape of convolution/deconvolution function:

1. Gaussian

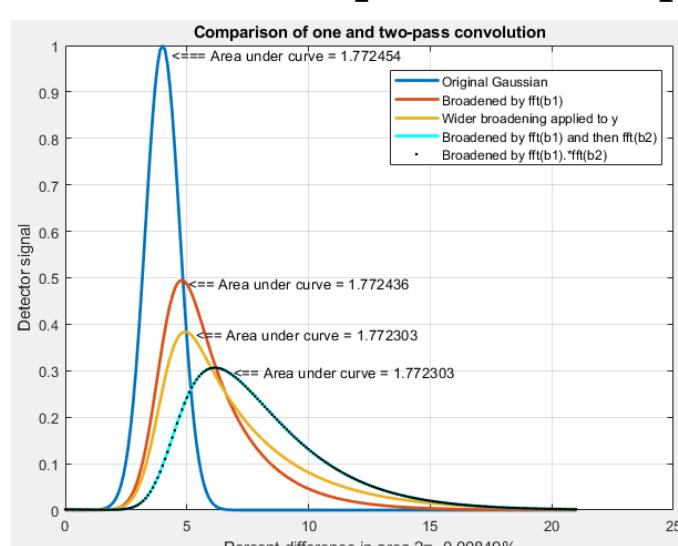
2. Exponential

Select shape 1 or 2: 2

Enter the exponential time constant:

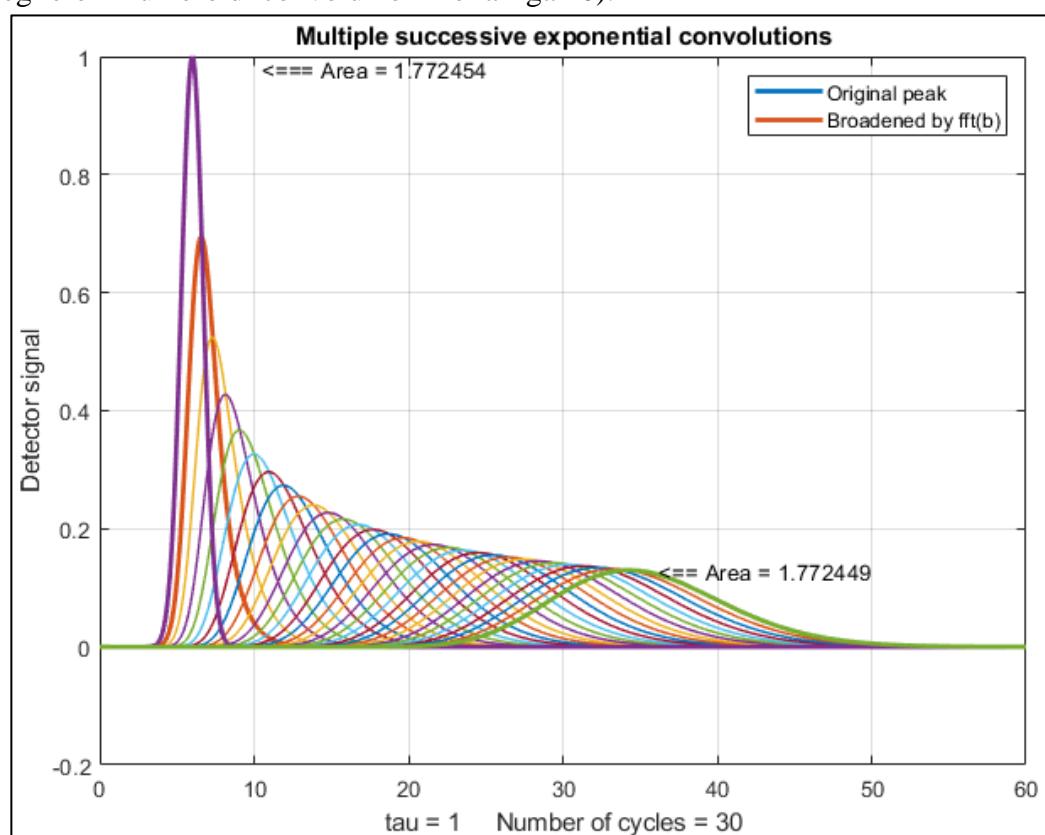
Infine si inserisce la costante di tempo (in unità x) e si preme **Enter**.

Convoluzione sequenziale multipla



Nel mondo reale, i meccanismi di ampliamento del segnale non sono sempre riducibili a una singola convoluzione. A volte possono essere in gioco due o più convoluzioni contemporaneamente. Un buon esempio di ciò si verifica nel processo di separazione del riciclaggio a *doppia colonna* ([TCRSP](#)), una nuova tecnica cromatografica in cui il campione iniettato viene riciclato in due colonne per ottenere una risoluzione sempre migliore, consentendo ai cromatografi di risolvere problemi di separazione complicati causati da coefficienti

di ripartizione delle componenti troppo simili e/o da una troppo bassa efficienza delle colonne [rif. 90]. Nella TCRSP, dopo che il campione è stato separato dalla prima colonna, fluisce nella seconda identica colonna e, dopo la separazione, le valvole di commutazione lo ricollegano alla prima colonna. Il ciclo si ripete tante volte quanto è necessario. Ogni passaggio attraverso una colonna aumenta leggermente la separazione tra le componenti, in modo che con un numero di cicli sufficientemente elevato si possano separare sostanze molto simili. Le separazioni cromatografiche spesso comportano allargamenti asimmetrici dei picchi (pagina 133), solitamente modellabili con una Gaussiana modificata esponenzialmente (EMG). Qualsiasi espansione che avviene nel primo passaggio si verificherà ripetutamente in quelli successivi. Il risultato netto sarà una forma di picco finale *non descrivibile con una singola convoluzione*. Il successo della tecnica TCRSP dipende dal fatto che la separazione tra i componenti aumenta più rapidamente dell'aumento dell'ampliamento causato dalle successive convoluzioni dei meccanismi di allargamento. Ma più convoluzioni in sequenza producono risultati che differiscono da un'unica grande convoluzione. Ciò è dimostrato dal semplice esempio di due convoluzioni esponenziali in sequenza applicate a una Gaussiana, come mostrato nella figura della pagina precedente, generata da uno [script Matlab](#). La curva blu è la Gaussiana originale. La curva rossa è il risultato di una singola convoluzione da parte di una funzione esponenziale la cui costante di tempo τ è 2. La curva ciano è il risultato di due successive convoluzioni con lo stesso τ . La curva arancione è un tentativo di duplicare quella con un'unica convoluzione più ampia con τ pari a 3. Quel tentativo fallisce; il risultato è una scarsa corrispondenza con la curva ciano. Infatti, gli esperimenti mostrano che *nessuna singola convoluzione esponenziale più ampia può eguagliare il risultato di due (o più) convoluzioni successive*; la forma è fondamentalmente diversa. Più convoluzioni esponenziali producono un picco meno asimmetrico, più spostato su valori maggiori di x . D'altra parte funziona una *singola convoluzione* di una funzione, che è il *prodotto* delle trasformate di Fourier delle due funzioni separate (punti neri). Con un numero maggiore di convoluzioni successive, i picchi diventano più simmetrici e più Gaussiani, come dimostrato da questo [grafico](#), generato da questo [script Matlab](#). (Si può seguire il numero di convoluzioni nella riga 20).



Deconvoluzione di Fourier

La [deconvoluzione](#) di Fourier è l'inversa della [convoluzione](#) di Fourier così come la divisione è l'inversa della moltiplicazione. Sapendo che m per x è uguale a n , dove m e n sono noti ma x è ignoto, allora x è uguale a n diviso m . Allo stesso modo, sapendo che il vettore M convoluto col vettore X è uguale al vettore N , dove M e N sono noti ma X è ignoto, allora X è uguale a M *deconvoluto* da N .

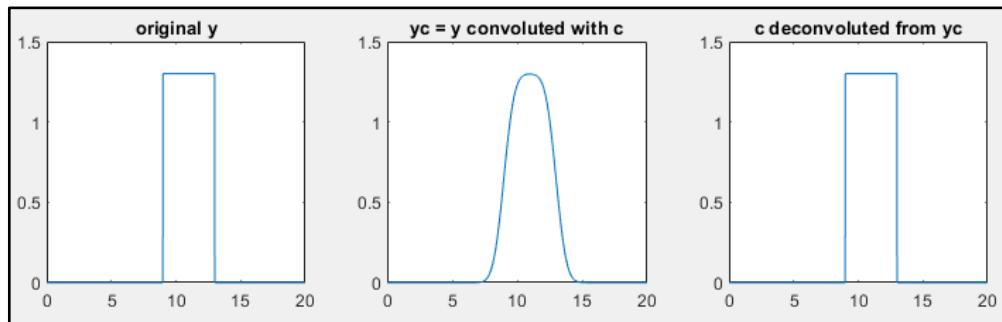
In pratica, la deconvoluzione di un segnale da un altro viene solitamente eseguita con una *divisione* punto-per-punto dei due segnali nel dominio di Fourier, ovvero, dividendo la trasformata di Fourier dei due segnali punto-per-punto e poi eseguendo la trasformazione inversa del risultato. Le trasformate di Fourier sono solitamente espresse in termini di numeri complessi, con le parti "reali" e "immaginarie" che rappresentano le parti seno e coseno. Se la trasformata di Fourier del primo segnale è $a + ib$, e quella del secondo segnale è $c + id$, allora il *rapporto* delle due trasformate di Fourier, per le regole della [divisione dei numeri complessi](#), è

$$\frac{a + ib}{c + id} = \frac{ac + bd}{c^2 + d^2} + i \frac{bc - ad}{c^2 + d^2}$$

La maggior arte dei linguaggi scientifici per computer (come il Fortran, Matlab e Python) eseguiranno automaticamente questa operazione quando si devono dividere due numeri complessi.

Nota: È importante rendersi conto che la parola "[deconvoluzione](#)" può avere *due diversi significati* nella letteratura scientifica, il che può creare confusione. Il dizionario Oxford definisce il termine come "Un processo di risoluzione di qualcosa nei suoi elementi costitutivi o la rimozione di complicazioni per chiarirlo", che in un certo senso si applica alla deconvoluzione di Fourier. Tuttavia, la stessa parola è talvolta usata anche per il processo di risoluzione o decomposizione di un insieme di segnali sovrapposti nei loro componenti additivi separati mediante la tecnica dell'[approssimazione iterativa ai quadrati minimi](#) (pagina 190) di un modello proposto del segnale dell'insieme dei dati. Tale processo, è però concettualmente distinto dalla deconvoluzione di Fourier, perché in tale deconvoluzione, il profilo del picco in esame è ignoto ma si presume che la funzione di ampliamento sia nota; mentre nell'approssimazione iterativa ai quadrati minimi, è esattamente il contrario: il profilo del picco si presume essere noto ma la larghezza del processo di ampliamento, che determinano la larghezza e la forma dei picchi nei dati registrati, è solitamente ignota. Quindi il termine "deconvoluzione spettrale" è *ambiguo*: potrebbe significare la deconvoluzione di Fourier di una funzione di risposta di uno spettro, o potrebbe significare la decomposizione di uno spettro nelle sue diverse componenti. Questi sono processi diversi; da non confondere.

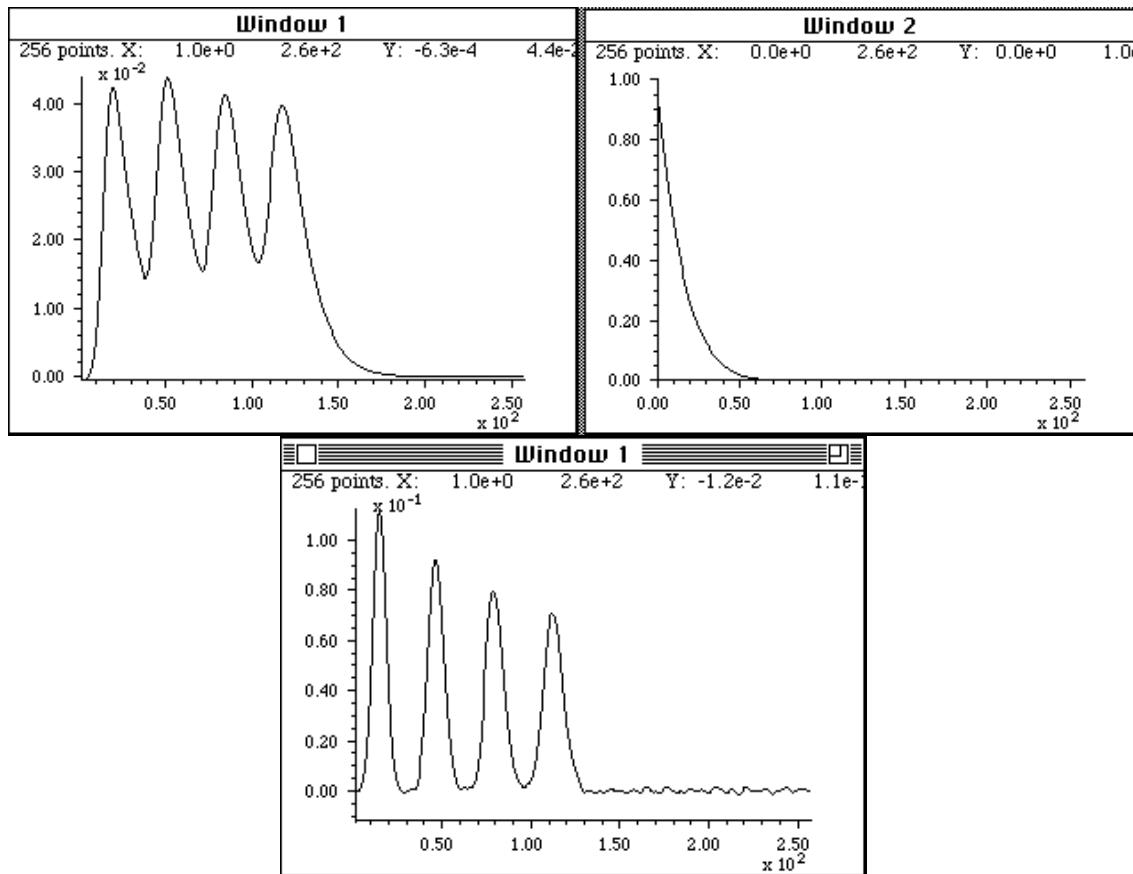
Il significato pratico della deconvoluzione di Fourier nel signal processing è quello di un metodo



computazionale utilizzato per invertire il risultato di una convoluzione che si verifica nel dominio fisico, ad esempio, per invertire l'effetto

della distorsione del segnale di un filtro elettrico o della risoluzione finita di uno spettrometro. In alcuni casi, la convoluzione fisica può essere misurata sperimentalmente applicando la funzione di un singolo impulso ("delta") all'ingresso del sistema, i dati, poi, possono essere utilizzati come vettore di deconvoluzione. In tale applicazione, la deconvoluzione funziona perfettamente solo quando i segnali non contengono rumore e quando la funzione di convoluzione originale è

perfettamente nota, come nel caso mostrato in figura dove viene eseguita la convoluzione di un impulso quadrato con una funzione Gaussiana, c.



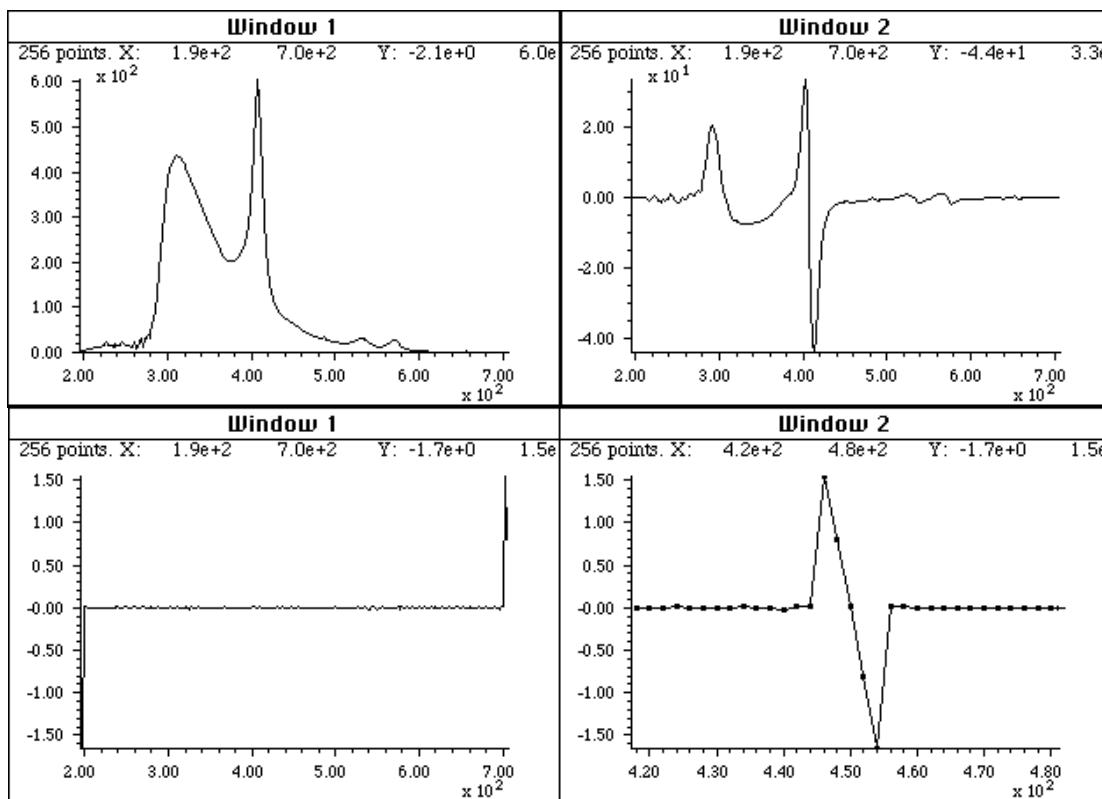
La deconvoluzione di Fourier viene qui usata per rimuovere l'influenza della distorsione di una funzione di risposta a coda esponenziale [exponential tailing response] da un segnale registrato (Window 1, in alto a sinistra) che è il risultato di un filtro passa-basso integrato nell'elettronica per ridurre il rumore. La funzione di risposta (Window 2, in alto a destra) dev'essere nota e solitamente viene calcolata basandosi su un modello teorico oppure viene misurata sperimentalmente come segnale di output prodotto applicando una funzione impulso (delta) all'ingresso del sistema. La funzione di risposta, col suo massimo a $x=0$, viene deconvoluta dal segnale originale. Il risultato (in basso, al centro) mostra un'approssimazione più vicina al profilo reale dei picchi; tuttavia, il rapporto segnale/rumore è inevitabilmente degradato rispetto al segnale registrato, perché l'operazione di deconvoluzione di Fourier sta semplicemente recuperando il segnale originale prima che passasse per il filtro passa-basso, il rumore e il resto.

(Se si sta leggendo online, cliccare per lo [script Matlab/Octave](#).)

Si noti che questo processo ha un effetto visivamente simile allo sharpening con derivata del picco (pag. 74) sebbene quest'ultimo non richieda una conoscenza specifica della funzione di ampliamento che ha causato la sovrapposizione dei picchi.

Anche se non è nota alcuna convoluzione fisica che abbia ampliato il segnale, è possibile utilizzare la deconvoluzione come metodo di sharpening del picco mediante deconvoluzione di un modello che sia un profilo del picco nel segnale; è detta "auto-deconvoluzione", perché il profilo della funzione di deconvoluzione è lo stesso di quello dei picchi nel segnale. L'autodeconvoluzione è un metodo comune di sharpening dei picchi applicabile a un segnale costituito da uno o più picchi con una forma di picco prevedibile. L'idea è che un modello silenzioso della forma del picco venga deconvoluto dal segnale e l'ampiezza di quel picco del modello venga aggiustata per fornire il grado di sharpening desiderato.

La deconvoluzione può essere utilizzata anche per determinare la forma di un'operazione di convoluzione sconosciuta che è stata precedentemente applicata a un segnale, eseguendo la deconvoluzione del segnale originale e di quello convoluto, come mostrato nella pagina seguente.



Una diversa applicazione della deconvoluzione di Fourier consiste nel rivelare la natura di una funzione di trasformazione dei dati sconosciuta che è stata applicata a un insieme di dati dallo strumento di misura stesso. In questo esempio, la figura in alto a sinistra è uno spettro di assorbimento ultravioletto-visibile registrato da uno spettrometro commerciale ad array di fotodiodi (asse-X: nanometri; asse-Y: milliassorbanza). La figura in alto a destra è la [derivata prima](#) di quello spettro prodotta da un algoritmo (ignoto) nel software fornito con lo spettrometro. L'obiettivo qui è capire la natura dell'[algoritmo di differenziazione/smoothing](#) utilizzato dal software interno dello strumento. Il segnale in basso a sinistra è il risultato sorprendentemente semplice della deconvoluzione della derivata dello spettro (in alto a destra) dallo spettro originale (in alto a sinistra). Questa, quindi, deve essere la funzione di convoluzione utilizzata dall'algoritmo di differenziazione nel software dello spettrometro o un equivalente. Ruotandolo ed espandendolo sull'asse x si visualizza meglio la funzione (in basso a destra). Espressa in termini di numeri interi più piccoli, la serie di convoluzione è semplicemente +2, +1, 0, -1, -2, che è una combinazione di differenziazione e smoothing (pagina 104). Questo esempio elementare di “[reverse engineering](#)” facilita il confronto dei risultati con altri strumenti o la duplicazione di questi risultati su altre apparecchiature.

Quando si applica la deconvoluzione di Fourier ai dati sperimentali, ad esempio, per rimuovere l'effetto di un operatore noto di filtro passa-basso o di ampliamento causato dal sistema sperimentale, ci sono *tre problemi seri* che limitano l'utilità del metodo:

- (1) Una convoluzione matematica potrebbe non essere un modello accurato per la convoluzione che si verifica nel dominio fisico;
- (2) L'ampiezza della convoluzione, ad esempio la costante di tempo di un operatore di un filtro passa-basso o la forma e la larghezza di una funzione di fenditura dello spettrometro, deve essere nota o almeno regolata dall'utente per ottenere i migliori risultati.
- (3) Si verifica solitamente una seria degradazione del rapporto segnale/rumore; qualsiasi rumore aggiunto al segnale dal sistema *dopo* la convoluzione dell'operatore di ampliamento o di quello del filtro passa-basso verrà notevolmente amplificato quando la trasformata di

Fourier viene divisa per quella dell'operatore di ampliamento, perché le componenti ad alta frequenza dell'operatore di ampliamento (il *denominatore* nella divisione delle trasformate di Fourier) sono solitamente molto piccole, alcune delle quali dell'ordine di 10^{-12} o 10^{-15} , ne risulta un'enorme amplificazione di quelle particolari frequenze nel segnale deconvoluto. Questo può essere controllato in una certa misura con lo smoothing o filtrando per ridurre l'ampiezza dei componenti a frequenza più alta.

Si può vedere l'amplificazione del rumore ad alta frequenza che si verifica nel primo esempio grafico sopra nelle pagine precedenti. D'altra parte, questo effetto *non* viene osservato nel secondo esempio, perché in quel caso il rumore era presente nel segnale originale, *prima* della convoluzione eseguita dall'algoritmo della derivata dello spettrometro. Le componenti ad alta frequenza al denominatore nella divisione delle trasformate di Fourier sono tipicamente molto *più grandi* rispetto all'esempio precedente, evitando l'amplificazione del rumore e gli errori di divisione per zero, e l'unico rumore post-convoluzione deriva da errori numerici di arrotondamento nei calcoli matematici eseguiti dall'operazione di derivata e smoothing, che è sempre molto più piccolo del rumore nel segnale sperimentale originale.

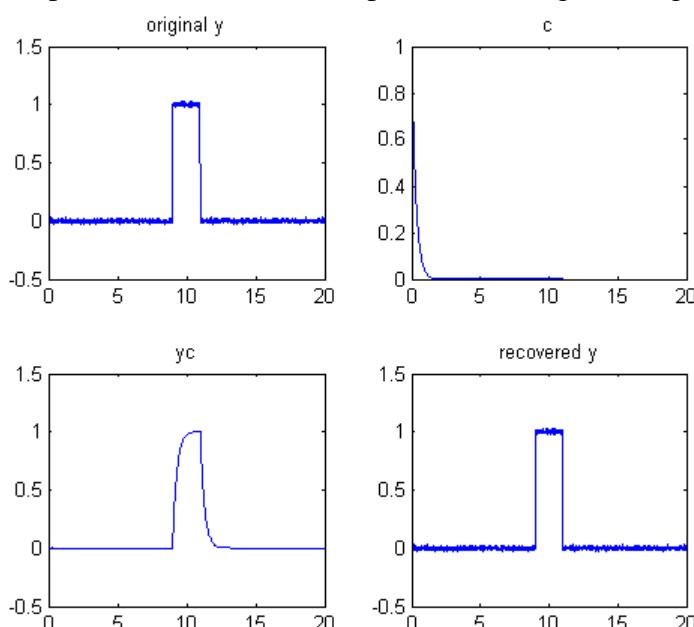
In molti casi, la larghezza della convoluzione fisica non è nota esattamente, quindi la deconvoluzione deve essere regolata empiricamente per ottenere i risultati migliori. Allo stesso modo, anche la larghezza dell'operazione finale di smoothing deve essere regolata per ottenere i migliori risultati. Raramente il risultato sarà perfetto, specialmente se il segnale originale è rumoroso, ma spesso è una approssimazione migliore del segnale reale in esame rispetto ai dati registrati senza deconvoluzione.

Come metodo di *sharpening* del picco, la deconvoluzione può essere paragonata al [metodo di sharpening derivativo descritto in precedenza](#) o al [metodo della potenza](#), in cui il segnale originale viene semplicemente elevato a una potenza positiva n .

Software per la deconvoluzione

Matlab e Octave

Matlab e Octave hanno una funzione nativa per la deconvoluzione di Fourier: [*deconv*](#). Un esempio della sua applicazione è mostrato nella prossima figura: il vettore *yc* (riga 6) rappresenta un impulso rettangolare rumoroso (*y*) convoluto con una funzione di trasferimento *c* prima di essere misurato. Nella riga 7, *c* viene deconvoluto da *yc*, per recuperare l'originale *y*. Ciò richiede che la funzione di trasferimento *c* sia nota. L'impulso del segnale rettangolare viene recuperato in basso a destra (*ydc*), completo del rumore che era presente nel segnale originale. La deconvoluzione di Fourier inverte



non solo l'effetto della distorsione del segnale da parte della convoluzione con la funzione esponenziale, ma anche l'effetto del passa-basso per il filtraggio del rumore. Come spiegato, c'è una significativa amplificazione di qualsiasi rumore aggiunto *dopo* la convoluzione dalla funzione di trasferimento (riga 5). Questo script è utilizzabile per mostrare che c'è una grande differenza tra il rumore aggiunto *prima* della convoluzione (riga 3), che viene recuperato senza modifiche dalla deconvoluzione di Fourier assieme al

segnale, e il rumore aggiunto *dopo* la convoluzione (riga 6), che viene amplificato rispetto a quello del segnale originale. [Scaricare questo script](#). Si noti che il termine “`sum(c)`” nella riga 7 è inserito semplicemente per ridimensionare l'ampiezza del risultato (in particolare l'area sotto la curva) in modo che corrisponda alla `y` originale.

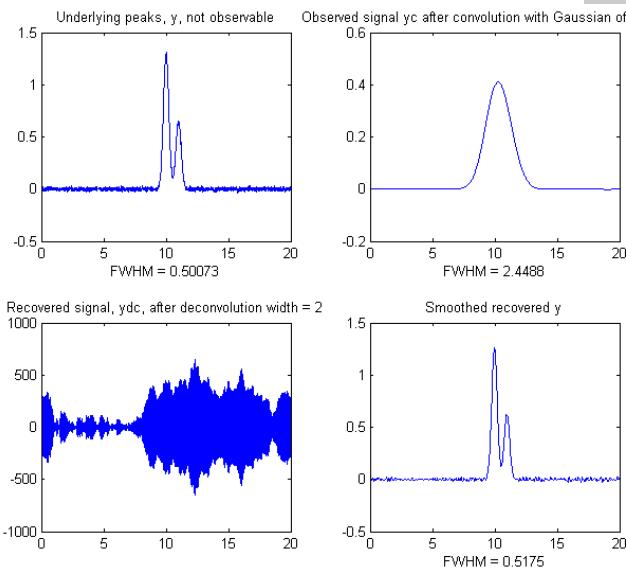
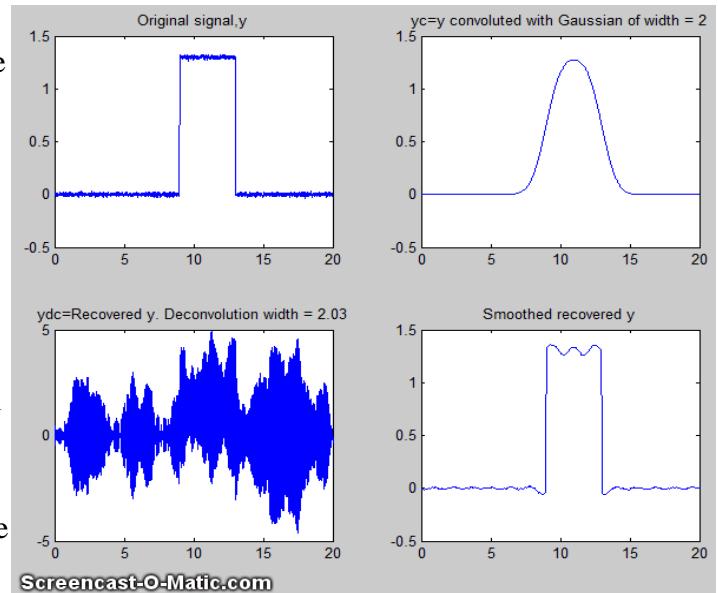
```
x=0:.01:20;y=zeros(size(x)); % 2000 point signal with 200-point
y(900:1100)=1; % rectangle in center, y
y=y+.01.*randn(size(y)); % Noise added before the convolution
c=exp(-(1:length(y))./30); % exponential convolution function, c
yc=conv(y,c,'full')./sum(c); % Create exponential trailing function, yc
% yc=yc+.01.*randn(size(yc)); % Noise added after the convolution
ydc=deconv(yc,c).*sum(c); % Recover y by deconvoluting c from yc
% Plot all the steps
subplot(2,2,1); plot(x,y); title('original y'); subplot(2,2,2);
plot(x,c);title('c'); subplot(2,2,3); plot(x,yc(1:2001)); title('yc');
subplot(2,2,4); plot(x,ydc);title('recovered y')
```

In alternativa, si può eseguire in proprio la deconvoluzione di Fourier *senza* utilizzare la funzione Matlab/Octave nativa "deconv" dividendo le trasformate di Fourier di `yc` e `c` utilizzando la funzione Matlab/Octave nativa "fft.m" e trasformando inversamente il risultato con la funzione Matlab/Octave nativa "ifft.m". Si noti che `c` dev'essere [riempita con zeri](#) per corrispondere alla dimensione di `yc`. I risultati sono essenzialmente gli stessi (tranne che per la precisione numerica in virgola mobile del computer, che di solito è trascurabile), ed è *più veloce* rispetto all'utilizzo della funzione `deconv`:

```
ydc=ifft(fft(yc)./fft([c zeros(1,2000)]).*sum(c);
```

Se si sta leggendo online, [cliccare qui](#) per un semplice esempio esplicito di convoluzione e deconvoluzione di Fourier per un piccolo vettore di 9 elementi, con i vettori stampati in ciascuna fase.

Lo script [DeconvDemo3.m](#) è simile all'esempio precedente, tranne per il fatto che mostra la convoluzione e la deconvoluzione *Gaussiana* di Fourier sullo stesso impulso rettangolare, utilizzando la formula fft/ifft appena descritta. Il grafico animato della schermata a lato (Se si sta leggendo online, si può cliccare sul [link per l'animazione](#)) mostra l'effetto della modifica della larghezza della deconvoluzione. Il segnale originale deconvoluto in questo esempio (quadrante in basso a sinistra) è estremamente rumoroso, ma quel rumore è soprattutto [rumore "blu" \(alta frequenza\)](#) che si può facilmente ridurre con un po' di [smoothing](#) (pagina 38). Come si vede in entrambi gli esempi animati qui, la deconvoluzione funziona *al meglio* quando l'ampiezza della deconvoluzione corrisponde esattamente alla larghezza della convoluzione a cui è stato sottoposto il segnale osservato; più ci si allontana, peggiori saranno le oscillazioni e gli altri artefatti del segnale. In pratica, si devono provare diverse larghezze di deconvoluzione per trovare quella che si traduce nelle *oscillazioni più piccole*, che ovviamente diventano più difficili da vedere se il segnale è molto rumoroso. Da notare che in



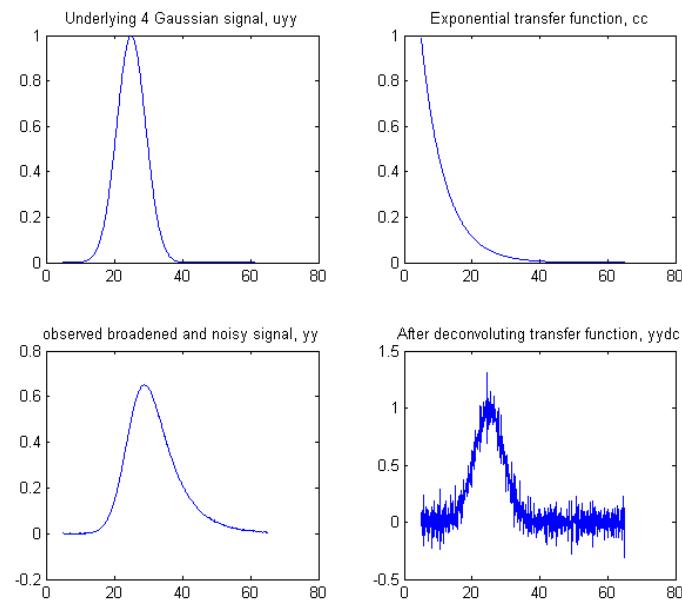
questo esempio l'ampiezza della deconvoluzione deve stare entro l'1% dell'ampiezza della convoluzione. In generale, quanto più ampia è l'ampiezza della convoluzione fisica rispetto al segnale, tanto più accuratamente la larghezza della deconvoluzione deve corrispondere all'ampiezza della convoluzione fisica.

[DeconvDemo5.m](#) (a lato) mostra un esempio con *due* picchi ravvicinati di pari larghezza che sono *completamente irrisolti* nel segnale osservato, ma vengono recuperati con il loro rapporto di altezza 2:1 intatto nel risultato con la deconvoluzione e lo smoothing. Questo è un esempio di “auto-deconvoluzione” Gaussiana.

[DeconvDemo6.m](#) è lo stesso eccetto tranne che i picchi sono [Lorentziani](#). Da notare che tutti questi script richiedono funzioni [scaricabili](#) da <http://tinyurl.com/cey8rwh>). In tutte le simulazioni precedenti, il metodo di deconvoluzione funziona sempre bene perché il rapporto segnale/rumore del “segnale osservato” (quadrante in alto a destra) è abbastanza buono; il rumore non è nemmeno visibile sulla scala qui presentata. In assenza di qualsiasi informazione sull'ampiezza della funzione di deconvoluzione, trovare quella corretta dipende dalla minimizzazione sperimentale delle oscillazioni che appaiono quando l'ampiezza della deconvoluzione non è corretta e uno scarso rapporto segnale/rumore è un forte impedimento. Ovviamente, lo smoothing può ridurre il rumore, specie quello ad alta frequenza (blu), ma aumenta anche leggermente la larghezza dei picchi, il che funziona *contro* il punto di deconvoluzione, quindi non se ne deve abusare. L'immagine a lato mostra le larghezze dei picchi (come larghezza piena a metà del massimo); le larghezze dei picchi deconvoluti (quadrante in basso a destra) sono solo leggermente maggiori rispetto ai picchi sottostanti (non osservati) (quadrante in alto a sinistra) sia a causa della imperfetta deconvoluzione sia per gli effetti dell'ampliamento dello smoothing necessario per ridurre il rumore ad alta

frequenza. Come regola approssimativa ma pratica, se c'è del rumore *visibile* nel segnale osservato, è probabile che sia il risultato dell'auto-deconvoluzione, del tipo mostrato in [DeconvDemo5.m](#), e sarà troppo rumoroso per essere utile.

Nell'esempio mostrato a lato. ([Effettuare il download di questo script](#)), il segnale (*uyy*) è una *Gaussiana*, ma nel segnale osservato (*yy*) il picco è *ampliato esponenzialmente* risultando un picco *traslato, più basso e più largo*. Supponendo che la costante di tempo dell'ampliamento esponenziale ('*tc*') sia nota, possa essere indovinata o misurata (pag. 77), la deconvoluzione di Fourier di *cc* da *yy* rimuove con successo l'ampliamento di *yydc*, ripristina altezza, posizione e larghezza originali della Gaussiana, ma a scapito di un notevole aumento del rumore. Il rumore è causato dal fatto che è stato aggiunto un po' di rumore bianco costante *dopo* la convoluzione di ampliamento (*cc*), per rendere la simulazione più realistica. Tuttavia, il rumore residuo nel segnale deconvoluto è "**blu**" (ad alta frequenza, cfr. pag. 28) e quindi viene facilmente ridotto con lo **smoothing** (pag. 41) ed ha un effetto minore sull'approssimazione con i quadrati minimi rispetto al rumore bianco. (Per un'impresa più ardua, si provi più rumore nella riga 6 o una cattiva stima della costante di tempo ('*tc*') nella riga 7). Per disegnare il segnale recuperato sovrapposto a quello originale : **plot(xx,uyy,xx,yydc)**. Per disegnare il segnale osservato con quello originale : **plot(xx,uyy,xx,yy)**. Per approssimare il segnale recuperato ad una Gaussiana per determinare i parametri del picco:



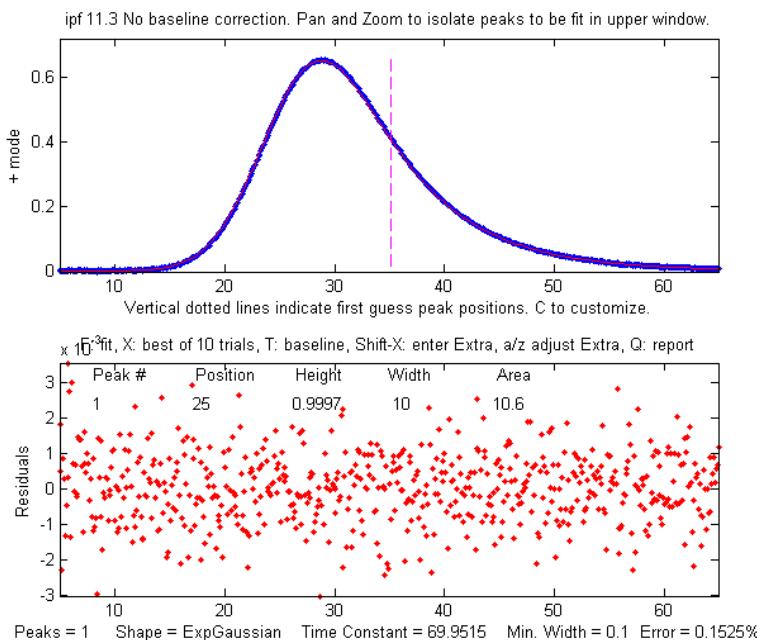
[FitResults,FitError]=peakfit([xx;yydc],26,42,1,1,0,10), che fornisce valori eccellenti per le posizioni, le altezze e le larghezze dei picchi originali. Ci si può rendere conto che con *dieci volte* il livello di rumore precedente (Noise=.01 nella riga 6), i valori dei parametri del picco determinati con l'approssimazione (curve fitting) restano ancora abbastanza buoni, e anche con *100x rumore in più* (Noise=.1 nella riga 6) i parametri del picco sono *più accurati di quanto ci si aspetterebbe* per quella quantità di rumore (perché quel rumore è **blu**). Da ricordare che non è necessario effettuare lo smoothing dei risultati della deconvoluzione di Fourier prima del [curve fitting], [come visto a pagina 46](#).

Un'alternativa all'approccio della deconvoluzione di cui sopra, consiste nell'usare il [curve fitting iterativo](#) (pag. 190) per approssimare direttamente il segnale osservato ad un modello di [Gaussiana esponenzialmente espansa](#) (profilo numero 5) :

```
>> [FitResults,FitError] = peakfit([xx;yy], 26, 50, 1, 5, 70, 10)
```

Entrambi i metodi forniscono dei buoni valori per i parametri del picco, ma la deconvoluzione di Fourier è più veloce perché l'approssimazione del segnale deconvoluto con un semplice modello Gaussiano è più veloce dell'approssimazione iterativa della curva al segnale osservato con il più complicato modello di Gaussiana ampliata esponenzialmente.

Se il fattore esponenziale "tc" non è noto, lo si può determinare con l'approssimazione iterativa utilizzando ipf.m (pag. 405), regolando manualmente il fattore esponenziale ('extra') interattivamente con i tasti A e Z per ottenere l'approssimazione migliore:



argomento di input ("start") , con i valori entro un fattore di due o giù di lì tra quelli corretti:

```
>> [FitResults, FitError]=peakfit([xx;yy],0,0,1,31,70,10, [20 10 50])
Peak# Position Height Width Area tc
1 25.006 0.99828 10.013 10.599 69.83
GoodnessOfFit =
0.15575 0.99998
```

Il valore del fattore esponenziale determinato da questo metodo è 69.8, sempre prossimo a 70. Tuttavia, se il segnale è molto rumoroso, ci sarà un po' di incertezza nel valore del fattore esponenziale determinato in questo modo - ad esempio, il valore varierà leggermente se per la misura vengono selezionate regioni leggermente diverse del segnale (p.es. col pan e lo zoom in ipf.m o cambiando gli argomenti del centro e della finestra in peakfit.m). Si veda a pagina 299 per un altro esempio con quattro Gaussiane sovrapposte.

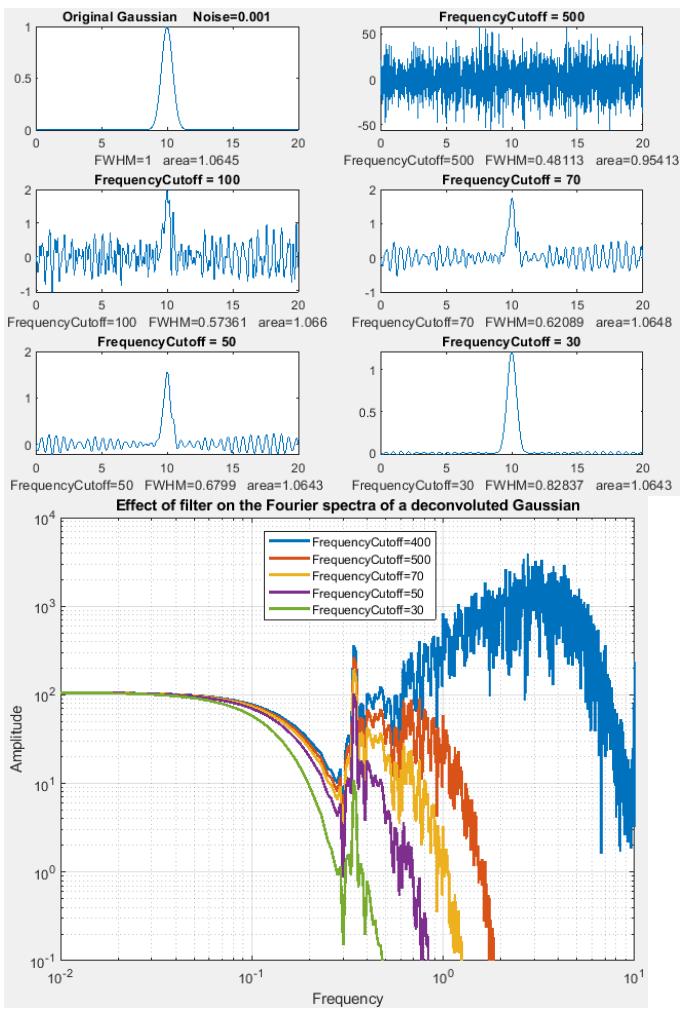
Riduzione del rumore nei segnali deconvoluti

Il modo più comune per controllare l'amplificazione del rumore ad alta frequenza risultante dalla deconvoluzione, come descritto sopra, è il filtraggio passa-basso, mediante una qualche forma di filtro a media mobile o un filtro di Fourier. La limitazione fondamentale di tali filtri, sfortunatamente, è che sono limitati nella loro capacità di gestire situazioni in cui valori prossimi allo zero nella fft della funzione di deconvoluzione al denominatore risultano in valori astronomicamente grandi nel segnale deconvoluto. Per apprezzare questo problema, si considerino le figure seguenti, create dallo script Matlab [DenomAdditionDemo.m](#), che mostra l'applicazione dell'auto-deconvoluzione per rendere più nitido [sharpen] un singolo picco Gaussiano isolato deconvolvendo una stretta funzione Gaussiana centrata sullo zero (0.8 volte la larghezza del picco originale), utilizzando solo un filtro di Fourier (pagina 123) per ridurre il rumore e il ringing. (Doppio-click per visualizzare le figure ingrandite)

>>ipf([xx;yy]);

che in questo caso si adatta meglio quando il fattore esponenziale "tc" è regolato a circa 69.9 (prossimo al valore corretto di 70 in questa simulazione).

In alternativa, si può usare [peakfit.m](#) con la Gaussiana esponenzialmente allargata *variabile non vincolata* (profilo 31), che troverà automaticamente il valore migliore di "tc", ma in questo caso i risultati migliori si otterranno se gli si fornisce una prima suggerimento come ottavo



La figura della window 1 (a sinistra) mostra il picco originale (in alto a sinistra) e i restanti sub-plot mostrano i picchi deconvoluti dopo lo smoothing mediante un filtro di Fourier con 5 diversi tagli di frequenza per ridurre il ringing. Gli spettri di frequenza corrispondenti di questi cinque segnali deconvoluti sono mostrati nella figura della window 2 (a destra), che mostra due distinte regioni di frequenza:

- (a) Il lato sinistro (bassa frequenza) è una regione curva e liscia [smooth]. Questa è la regione di frequenza dominata dai picchi che sono stati accentuati [sharpened] dalla deconvoluzione. Quanto più stretto è il picco, tanto più gradualmente la curva ricade nelle frequenze più alte.
- (b) L'estremità destra (alta frequenza) dello spettro è dominata dal rumore e dal ringing nel segnale deconvoluto. Vogliamo ridurre il più possibile questa regione senza ridurre troppo le frequenze più basse (che allargherebbe i picchi).

L'osservazione di questi spettri può essere una guida utile per regolare i parametri della deconvoluzione. Il ringing nei picchi del segnale nella Figura della window 1 corrisponde allo *spike pronunciato prossimo al centro dello spettro*. Quando la frequenza di taglio diminuisce, le componenti ad alta frequenza vengono ridotte, come previsto, ma *quello spike rimane* anche al taglio più basso (linea verde), a quel punto l'ampiezza del picco deconvoluto è stato ampliato dal filtro, contrariamente all'intento originale di rendere più nitido.

Riduzione del rumore in eccesso mediante aggiunta al denominatore

Questo problema può essere risolto con un metodo indipendente di riduzione del rumore introdotto dall'autore e da Farooq Wahab nel 2023 (riferimento 98). Ciò comporta semplicemente l'aggiunta di una piccola costante positiva diversa da zero o di una funzione di distribuzione al denominatore nel

calcolo della deconvoluzione, che impedisce numeri eccessivamente piccoli nel denominatore. La quantità aggiunta è piccola, tipicamente dall'1% al 5% dell'ampiezza del denominatore. In Matlab, la semplice operazione di deconvoluzione è codificata in questo modo:

```
ydc=ifft(fft(y)./(fft(df))).*sum(df)
```

Ecco il codice per il caso in cui l'aggiunta al denominatore sia una costante:

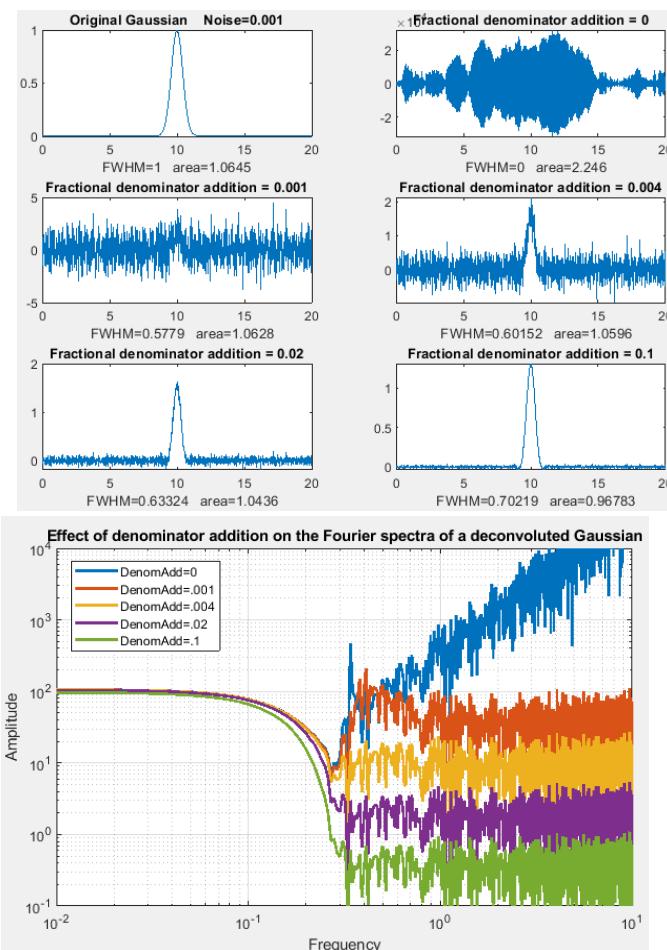
```
ydc=ifft(fft(y)./(fft(df)+FDA.*max(fft(df)))).*sum(df)
```

dove y è il segnale originale, df è la funzione di deconvoluzione e FDA è l'aggiunta al denominatore frazionario. L'addizione viene scalata al valore massimo della fft della funzione di deconvoluzione df in modo che la quantità aggiunta si adatti alle variazioni di ampiezza dei diversi segnali sperimentali.

Un'alternativa è quella di aggiungere una costante *solo a quei membri del denominatore al di sotto di una soglia specificata*, ad es. utilizzando la funzione “no lower than” [non inferiore a] [nlt\(a,b\)](#):

```
ydcDA=ifft(fft(y)./(nlt(fft(df),DA.*0.01.*max(fft(df)))).*sum(df);
```

(Il metodo dell'aggiunta al denominatore è incluso nell'esempio Matlab Live Script a pagina 360).



Le figure delle window 3 e 4 (sopra) mostrano l'effetto della variazione dell'addizione del denominatore frazionario *senza* filtro della frequenza. La subplot in alto a sinistra mostra il picco originale e le restanti cinque subplot mostrano i risultati dell'aggiunta di importi crescenti al denominatore, da zero a 0.1. Gli spettri della frequenza di questi cinque segnali deconvoluti appaiono nella figura della window 4 (a destra), mostrando che l'effetto dell'addizione del denominatore è quello di ridurre l'ampiezza complessiva della metà destra rumorosa dello spettro di frequenza, *senza* modificando la distribuzione della frequenza. Con l'aggiunta di zero (blu), il grande picco vicino al centro si vede come prima, ma anche l'addizione più piccola (rosso) lo

elimina. Con l'aggiunta massima del 10% (verde), il rumore è notevolmente ridotto ma il picco deconvoluto è più nitido (FWHM=0,7) rispetto alla larghezza del picco con il solo filtro applicato (FWHM=0,82) e ha lo stesso SNR. Nota: eseguendo nuovamente lo script [DenomAdditionDemo.m](#), verrà generato ogni volta un campione di rumore diverso.

I due spettri di frequenza nelle figure sopra mostrano che il filtraggio passa-basso e l'addizione del denominatore sono operazioni ortogonali nel dominio della frequenza. Il filtraggio opera "orizzontalmente" lungo l'asse x (frequenza), mentre l'addizione al denominatore opera "verticalmente" lungo l'asse y (ampiezza) e riduce l'ampiezza di tutte le frequenze che sono maggiormente amplificate dalla deconvoluzione. Entrambi i metodi riducono il rumore, ma funzionano in modi diversi e possono essere utilizzati insieme.

Un buon modo per esplorare l'interazione tra i valori delle numerose variabili è utilizzare Live Script di Matlab [DenomAdditionDemo mlx \(grafico\)](#), che ha *cursori* e un *menù a discesa* per regolare i parametri in modo interattivo. Vedere pag. 360.

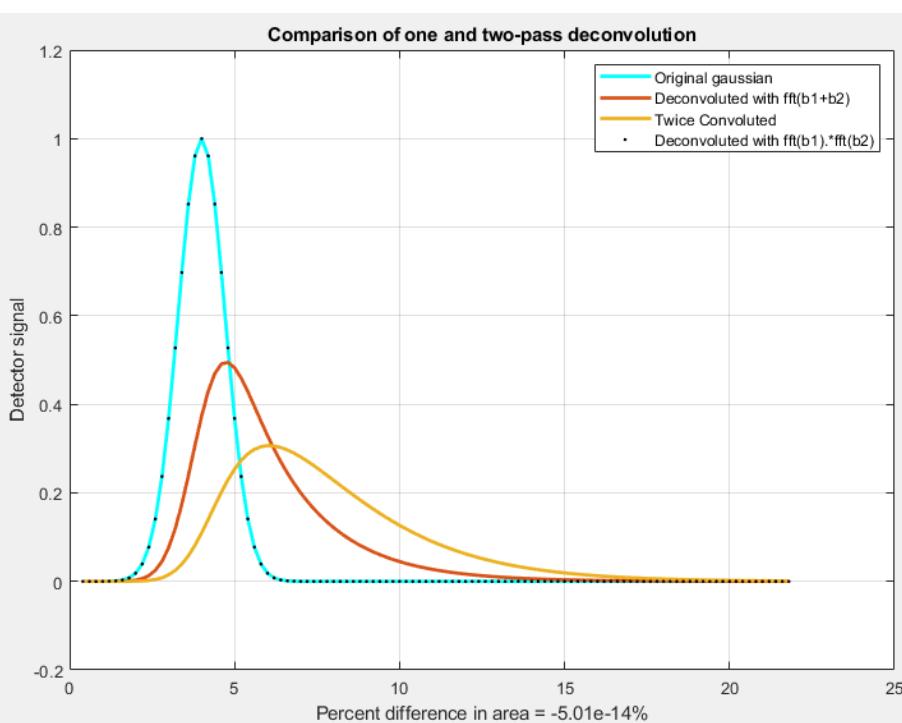
Il riferimento 98 esplora questo metodo in maggiore dettaglio, inclusa l'aggiunta di distribuzioni anziché una costante e mostrando diversi esempi di applicazioni a segnali sperimentali.

Deconvoluzione per la misura dell'area dei picchi

La misura delle aree sotto i picchi è un requisito comune nell'analisi quantitativa (pag. 138), ma funziona solo se c'è una separazione sufficiente tra i picchi. Poiché la deconvoluzione rende più stretti i picchi ma non modifica l'area sottostante, può essere utilizzata per migliorare la misura delle aree dei picchi sovrapposti. Il Live Script Matlab [DenomAdditionDemo mlx \(grafico\)](#) utilizza l'autodeconvoluzione di Fourier per rendere più nitidi i picchi per migliorare la precisione del metodo del taglio verticale per una coppia di picchi sovrapposti (selezionare la casella a destra di "PeakAreaMeasurements" nella riga 3. Il formato del Live Script consente controlli interattivi per esplorare le impostazioni per la generazione del segnale e la misura dell'area del picco). Vedere pagina 120.

Nello script Matlab [GLSDPerpDropDemo16.m](#), le aree di un gruppo di tre picchi parzialmente sovrapposti vengono misurate, mediante il metodo del taglio verticale, prima e dopo lo sharpening del picco mediante autodeconvoluzione di Fourier. Le misure vengono ripetute con altezze di picco casuali, per verificare come la sovrapposizione dei picchi interferisce con la misura precisa dell'area. Dopo sedici prove con altezze casuali dei picchi, le aree reali dei picchi vengono tracciate rispetto alle aree misurate e i valori R^2 per ciascun caso vengono confrontati prima e dopo la

deconvoluzione. I risultati sono riepilogati in [questo file PDF](#). Conclusione: in ogni caso, dal più "facile" al più impegnativo, le aree dei picchi con sharpening dalla deconvoluzione sono le più precise.



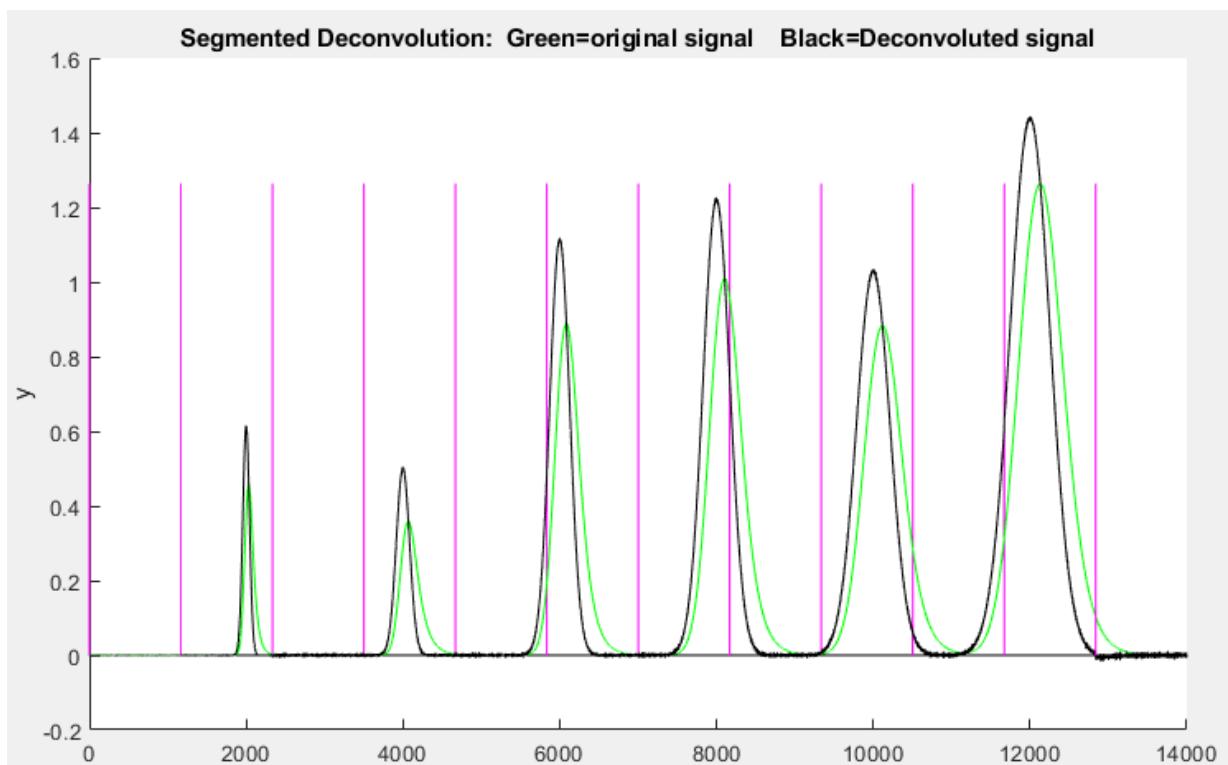
Deconvoluzione sequenziale multipla

Nei casi in cui il segnale originale sia stato soggetto a due o più convolu-

zioni distorte sequenziali (pagina 106), l'inversione di tali convoluzioni richiede più deconvoluzioni sequenziali e *non può essere annullata accuratamente da una singola deconvoluzione maggiore*. Come semplice esempio di tale situazione, lo script Matlab/Octave [DeconvoluteTwiceBroadenedPeak.m](#), mostra il tentativo di deconvoluzione di due ampliamenti esponenziali, rappresentati dai vettori di $b1$ e $b2$, che sono stati applicati ad un oggetto originariamente un picco Gaussiano. Nel grafico risultante a lato, la curva azzurra è il segnale sottostante originale ignoto, la curva gialla è il segnale osservato dopo che quello originale è stato ampliato due volte in modo esponenziale e la curva rossa è un tentativo di deconvoluzione di una singola funzione esponenziale più ampia con una costante di tempo maggiore, utilizzando il metodo "de-tailing" descritto a pagina 77. Il tentativo è ovviamente infruttuoso; infatti, nessuna singola deconvoluzione può rimuovere gli effetti di due o più convoluzioni. La linea tratteggiata nera è il risultato di una deconvoluzione con il prodotto $\text{fft}(b1) * \text{fft}(b2)$, che è la trasformata di Fourier della *convoluzione* di $b1$ e $b2$. Il tentativo ha successo: i punti neri si sovrappongono esattamente alla Gaussiana originale, in blu.

Deconvoluzione segmentata

Se le larghezze dei picchi e della code [tailing] variano in modo sostanziale nel segnale, si può usare una deconvoluzione *segmentata*, che consente al vettore di deconvoluzione di adattarsi alle condizioni locali nelle diverse regioni del segnale. [SegExpDeconv\(x,y,tc\)](#) divide x,y in diversi segmenti di uguale lunghezza definiti dalla lunghezza del vettore "tc", poi ciascun segmento viene deconvoluto con un decadimento esponenziale della forma $\exp(-x./t)$ dove "t" è l'elemento corrispondente del vettore "tc". Si può usare qualsiasi numero e sequenza dei valori di t . [SegExpDeconvPlot.m](#) è lo stesso, tranne per il fatto che disegna i segnali originali e quelli deconvoluti e mostra le divisioni tra i segmenti con linee magenta verticali per facilitare la regolazione del numero e dei valori dei segmenti. Questo viene mostrato dallo script [SegExpDeconvPlotExample.m](#) mostrato nella figura seguente). L'inevitabile aumento del rumore può essere attenuato mediante l'aggiunta al denominatore (pagina 116) o mediante lo smoothing segmentato (pagina 326).



[SegGaussDeconv.m](#) e [SegGaussDeconvPlot.m](#) sono uguali tranne per il fatto che eseguono la deconvoluzione Gaussiana simmetrica (centrata sullo zero). [SegDoubleExpDeconv.m](#) e

[SegDoubleExpDeconvPlot.m](#) eseguono una deconvoluzione esponenziale simmetrica (centrata sullo zero). Se le larghezze dei picchi aumentano gradualmente nel segnale, è possibile calcolare un valore iniziale ragionevole per il vettore “tc” fornendo solo il numero di segmenti (“NumSegments”), il primo valore, “start” e l’ultimo “endt”:

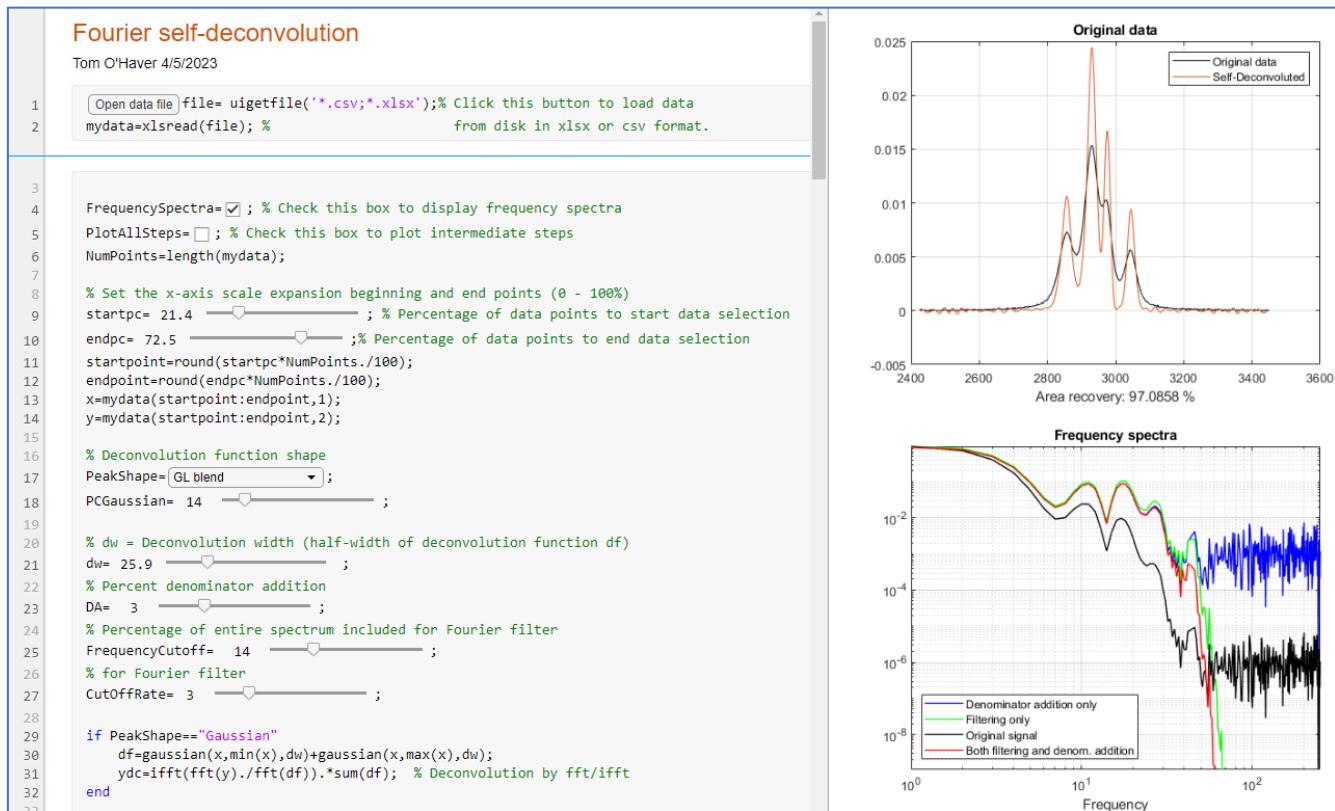
```
tstep=(endt-startt)/NumSegments;  
tc=startt:tstep:endt;
```

Live script del tool per l'Auto-deconvoluzione

Il Live Script [DeconvoluteData mlx](#) può eseguire l'autodeconvoluzione di Fourier sui dati archiviati nel disco. Cliccando sul pulsante "Open data file" nella riga 1 si apre un browser di file, che consente di navigare fino al file di dati (in formato .csv o .xlsx; lo script presuppone che i dati x,y siano nelle prime due colonne; lo si può cambiare nelle righe 13 e 14). Nel caso mostrato qui, il file di dati è '[HepteneTestData.csv](#)', mostrato come variabile 'file' nel workspace di Matlab. (Per visualizzare le figure a destra come mostrato di seguito, cliccare col pulsante destro del mouse sul pannello di destra e selezionare "Disable synchronous scrolling").

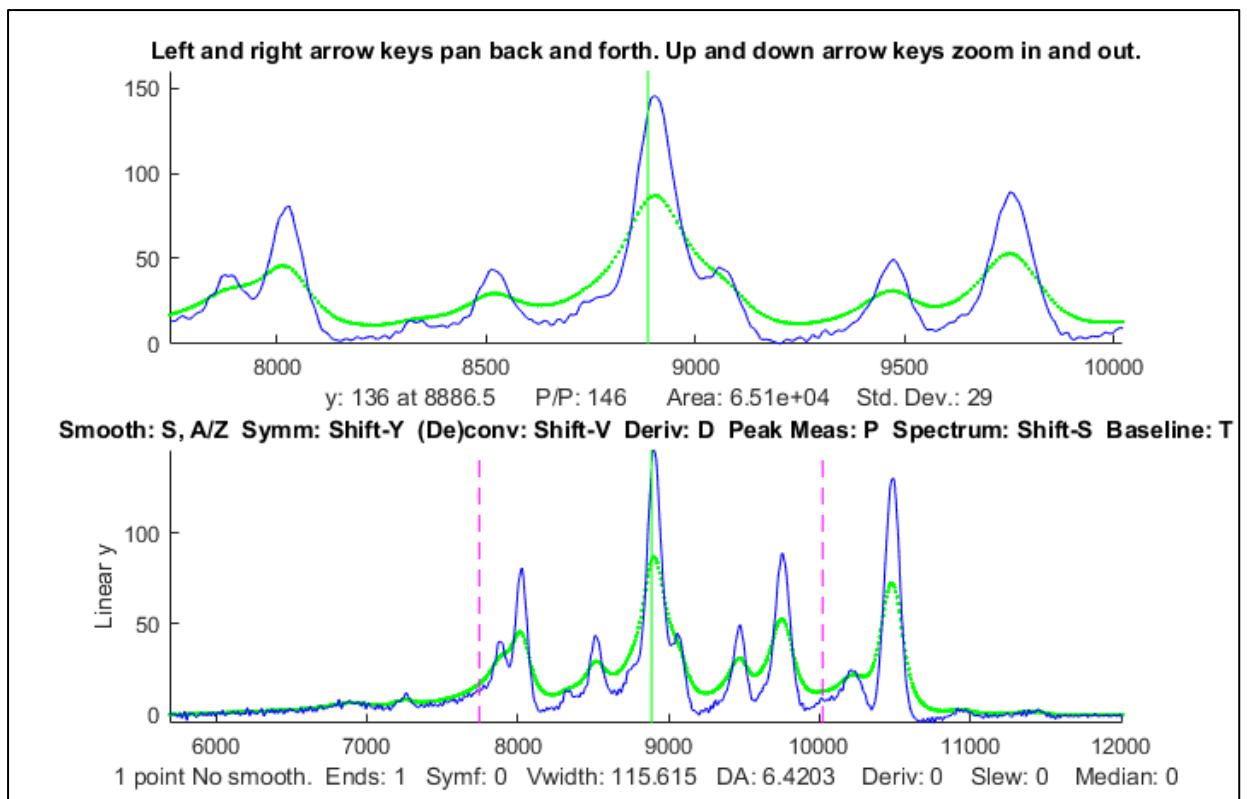
Gli slider **startpc** e **endpc** nelle righe 9 e 10 consentono di selezionare quale porzione dell'intervallo dei dati elaborare, dallo 0% al 100% dell'intervallo totale dei dati. Il menù a discesa **PeakShape** nella riga 17 seleziona il profilo della funzione di convoluzione (in questo caso, un mix Gaussiana-Lorentziana) e il cursore **PCGaussian** nella riga successiva consente selezione della percentuale Gaussiana di quel profilo. Lo slider **dw** alla riga 21 controlla la metà dell'ampiezza della deconvoluzione, lo slider **DA** alla riga 23 controlla l'aggiunta percentuale al denominatore. Lo smoothing, mediante il filtro di Fourier, è regolato da **FrequencyCutoff** e **CutOffRate** nelle righe 25 e 27. Tutte le variabili sono accessibili nel workspace di Matlab; il segnale finale è 'syDA'. Nota: con un doppio clic su uno qualsiasi degli slider si possono modificare i relativi intervalli se quello iniziale è insufficiente.

Cliccare sulla casella di controllo **FrequencySpectra** nella riga 4 per visualizzare gli spettri di frequenza. Cliccare sulla casella di controllo **PlotAllSteps** nella riga 5 per visualizzare tutti i passaggi che portano al risultato finale.



Deconvoluzione interattiva con iSignal

In [iSignal versione 8.3](#) e successive (pagina 366), si può premere **Shift-V** per mostrare il [menu per le operazioni di convoluzione e deconvoluzione di Fourier](#) che operano su una funzione Gaussiana, Lorentziana o esponenziale. Verrà richiesta la larghezza iniziale o la costante di tempo della funzione di deconvoluzione (in unità X), poi si possono usare i tasti **3** e **4** per diminuire o aumentare la larghezza del 10% (o **Shift-3** e **Shift-4** per regolarla dell'1%). Questa versione di iSignal include un ulteriore modo per ridurre il ringing e il rumore nel segnale deconvoluto, aggiungendo una costante al denominatore (riferimento 85) e regolandola con i tasti **5** e **6** per diminuire o aumentare la costante del 10% (o **Shift-5** e **Shift-6** per regolare dell'1%).

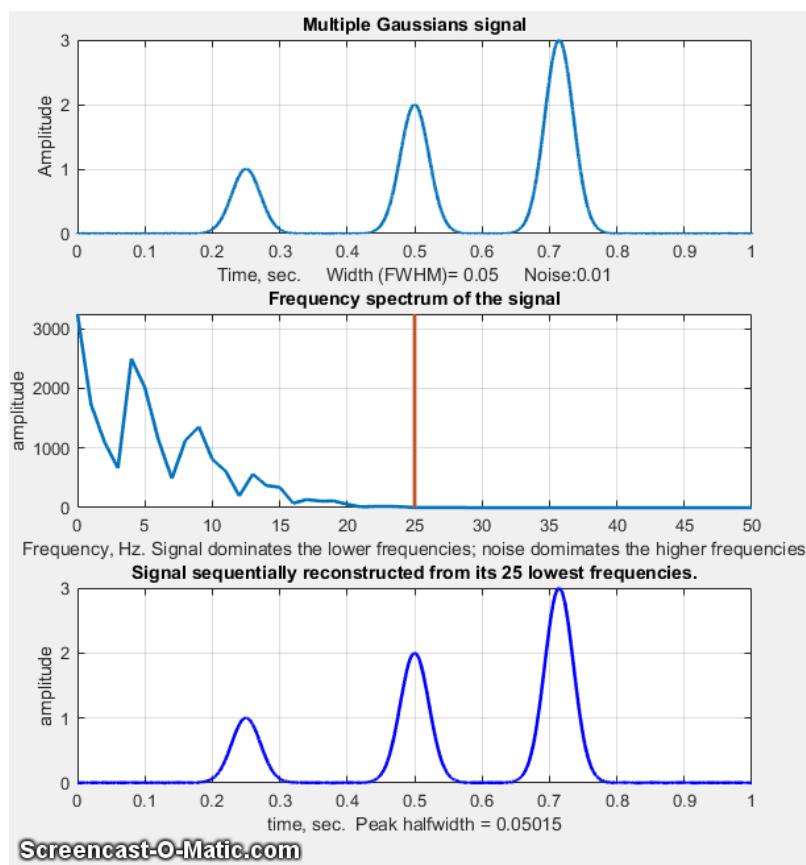


In questo esempio, il segnale originale appare come una linea verde tratteggiata e il risultato della deconvoluzione con una funzione di deconvoluzione *Lorentziana* appare come una linea blu. La larghezza della deconvoluzione è stata regolata quanto più larga possibile senza provocare significativi cali negativi tra i picchi, che per molti tipi di dati sperimentali, non sarebbero possibili. (Da ricordare che la matematica dell'operazione di deconvoluzione è strutturata in modo che l'*area dei picchi resti invariata*, pur aumentando le larghezze e riducendo le altezze). Il primo piano ingrandito del pannello superiore mostra che diversi picchi con sporgenze vengono risolte come picchi distinti, consentendo la misura delle loro posizioni in modo più accurato. Fortunatamente, l'ampiezza di quei picchi scoperti è maggiore della piccola quantità di rumore residuo nel segnale (grazie al buon rapporto segnale-rumore del segnale originale).

Filtro di Fourier

Un filtro di Fourier è un tipo di funzione di filtraggio basato sulla diretta manipolazione delle [componenti di frequenza](#) di un segnale. Funziona prendendo la [trasformata di Fourier](#) del segnale, poi attenuando o amplificando delle frequenze specifiche e infine trasformando inversamente il risultato.

In molte misure scientifiche, come nella spettroscopia e nella cromatografia, i segnali sono forme relativamente regolari che si possono rappresentare con un numero sorprendentemente piccolo di componenti di Fourier. Ad esempio, la figura seguente ([script](#)) mostra nel pannello superiore un



frequenze comprese tra 1 e 25. Il segnale ricostruito inizia come una grande massa informe con solo poche frequenze incluse. I picchi emergono e si restringono man mano che vengono aggiunte più frequenze, e la linea di base tra i picchi diventa più piatta, fino a quando il risultato è visivamente indistinguibile dal segnale originale quando vengono incluse 26 frequenze. Ma si noti come appare il segnale ricostruito quando arriva solo a 16 frequenze. A quel punto, l'ampiezza delle frequenze è già scesa di molto e c'è relativamente poca ampiezza nelle frequenze restanti, quindi i tre picchi

segnale simulato con tre picchi filtrati con smoothing, con altezze di picco pari a 1, 2 e 3, dove l'asse x è il tempo in secondi. Il pannello centrale mostra le prime 50 frequenze del suo spettro di Fourier, dove l'asse x è la frequenza in Hz. L'ampiezza delle componenti di Fourier è più forte alle basse frequenze e scende quasi a zero a 25 Hz.

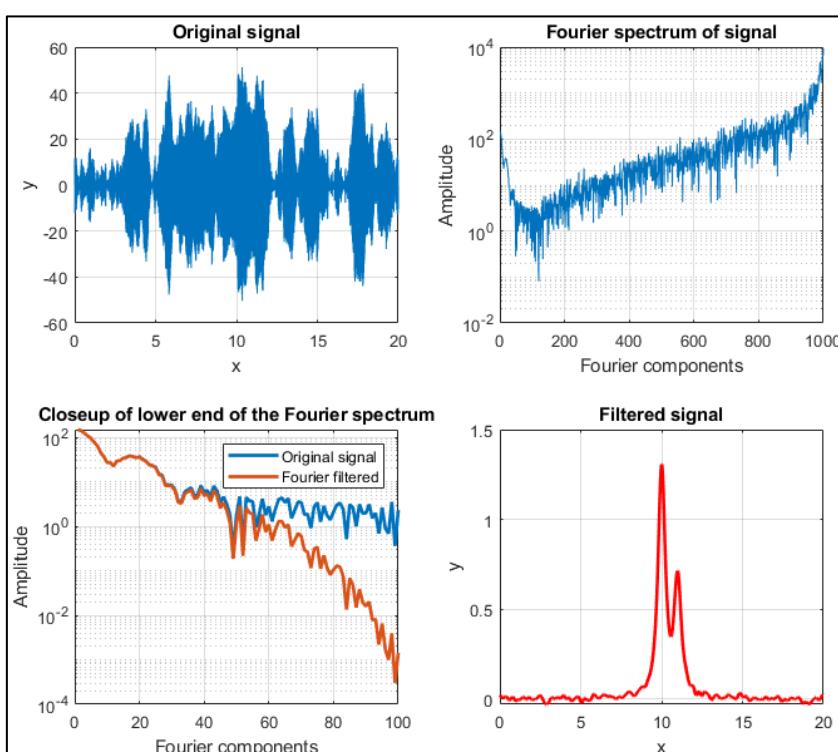
Il pannello inferiore mostra il segnale ricostruito sommando le prime "n" componenti di Fourier, dove $n=1, 2, 3, \dots$. Un'animazione GIF di questo processo ([visibile nella versione per Microsoft Word 365 o cliccare per visualizzarla in un browser web](#)) mostra il risultato della progressiva inclusione delle

sono evidenti. La linea di base, però, ha un'ondulazione piccola ma distinta, causata dal taglio brusco delle frequenze oltre quel punto. Ciò può essere evitato includendo più frequenze o utilizzando un *filtro con una forma regolabile* che consenta di controllare la frequenza di taglio.

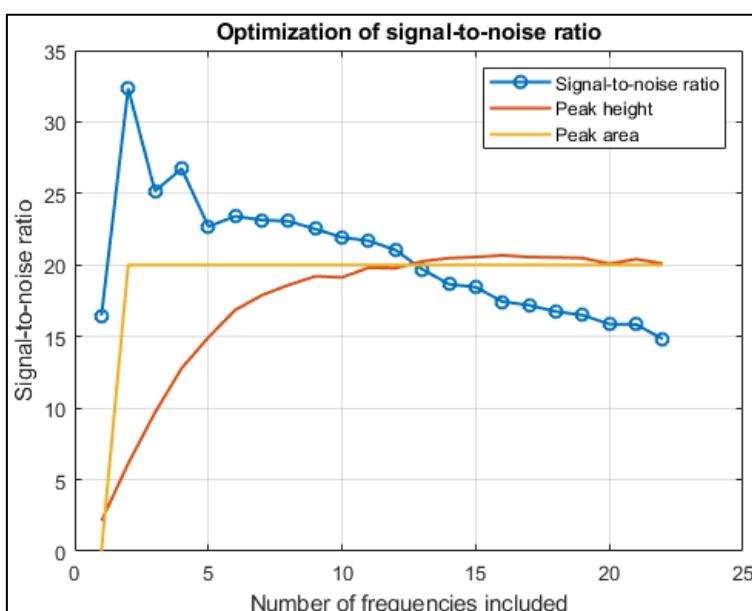
L'ottimizzazione del filtro di Fourier per il rapporto segnale/rumore (SNR) dei segnali del picco affronta lo stesso compromesso delle convenzionali funzioni di smoothing; vale a dire, l'SNR ottimale si ottiene quando l'altezza del picco è inferiore al massimo senza rumore. Ad esempio, lo script [GaussianSNRFrequencyReconstruction.m](#) mostra che per un picco Gaussiano, l'SNR ottimale viene raggiunto quando l'altezza del picco è circa la metà del valore reale, ma l'area del picco è la stessa. (Il rumore bianco ha [uguale ampiezza a tutte le frequenze](#), (pag. 28) mentre la maggior parte del segnale del picco è concentrata nelle prime poche frequenze).

Un esempio più 'drammatico' è mostrato nella figura a lato ([script](#)). In questo caso, il segnale (in alto a sinistra) sembra essere solo rumore casuale ad alta frequenza e il suo spettro di Fourier (in alto a destra, mostrato con una scala logaritmica) mostra che le componenti ad alta frequenza

dominano lo spettro su gran parte della sua gamma di frequenze. Il pannello in basso a sinistra mostra lo spettro di Fourier espanso nelle direzioni X e Y per mostrare più chiaramente la regione delle basse frequenze. Lì, la serie di dossi relativamente regolari, con picchi alla 1a, 20a e 40a frequenza, sono molto probabilmente il segnale effettivo. Partendo dall'ipotesi che le componenti al di sopra della 40a armonica siano sempre più dominate dal rumore, è possibile utilizzare la funzione del filtro di Fourier ([FouFilter.m](#)) per ridurre gradualmente le



armoniche più alte e ricostruire il segnale dalla trasformata di Fourier modificata (linea rossa). Il risultato (in basso a destra) mostra che il segnale contiene due picchi Lorentziani parzialmente sovrapposti che erano completamente oscurati dal rumore ad alta frequenza nel segnale originale.



Software per il filtraggio di Fourier

MATLAB. Il codice più semplice possibile per un filtro di Fourier elimina semplicemente tutte le frequenze al di sopra di un certo limite. Per farlo correttamente, è necessario prestare attenzione a utilizzare sia la componente seno *che* quella coseno (o equivalentemente la frequenza *e* la fase

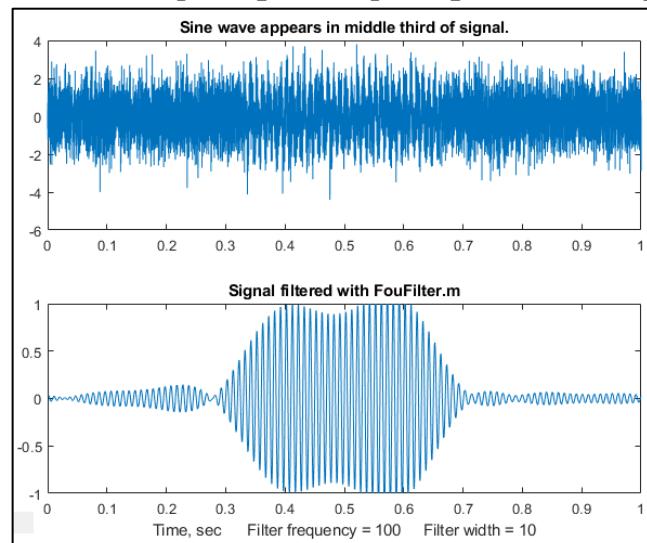
o la parte reale e quella immaginaria) della trasformata di Fourier. L'operazione deve tenere conto della struttura dell'immagine speculare della trasformata di Fourier di Matlab: le frequenze più basse sono agli estremi della fft e quelle più alte sono nella porzione *centrale*. Quindi, per passare le n frequenze più basse, si devono passare i primi n punti e gli ultimi n a zero per gli altri.

```
ffty=fft(y); % ffy is the fft of y
lfft=length(ffty); % Length of the FFT
ffty(n:lfft-n)=0; % Frequencies between n and lfft-n in the fft are set to zero.
fy=real(ifft(ffty)); % Real part of the inverse fft
```

La funzione di questo semplice filtro passa basso di Fourier è [flp.m](#). Questa è l'essenza minima di un filtro di Fourier, ma non è un filtro davvero pratico, perché il suo taglio brusco di solito si traduce in variazioni sulla linea di base, come mostrato sopra.

Funzione del filtro di Fourier generico. Per rendere il filtro di Fourier più utile in generale, si deve aggiungere del codice che includa non solo i filtri passa-basso, ma anche i filtri passa-alto, passa-banda ed escludi-banda, in più fornire una frequenza di taglio più graduale e variabile.

La funzione Matlab/Octave [FouFilter.m](#) è un filtro di Fourier più flessibile utilizzabile come filtro passa-basso, passa-alto o escludi-banda (notch) *con un rapporto di taglio variabile*. Questa funzione ha la forma **[ry, fy, ffilter, ffy]=FouFilter(y, samplingtime, centerfrequency, frequencywidth, shape, mode)**, dove y è il vettore della serie

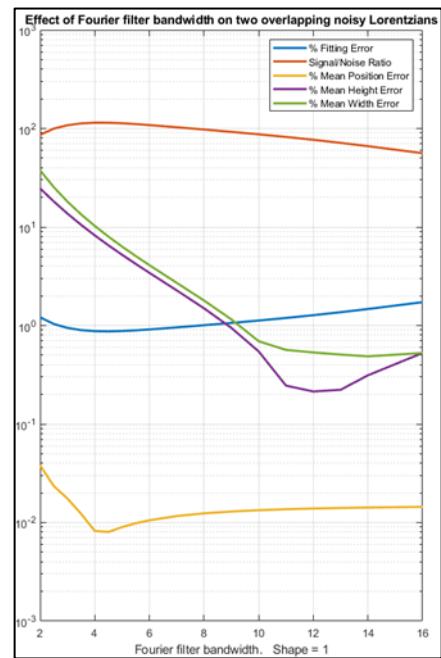


segnaletto filtrato in 'ry'. Può gestire segnali praticamente di qualsiasi lunghezza, limitati solo dalla memoria del computer. Ecco due script di esempio per chiamare FouFilter.m: [TestFouFilter.m](#) mostra un filtro passa-banda di Fourier applicato ad una sinusoida rumorosa da 100 Hz che appare nella terza metà della registrazione, mostrata nella prossima figura. Si vede che questo filtro è efficace nell'estrare il segnale dal rumore, ma che il tempo di risposta è lento. Lo script [TestFouFilter2.m](#) mostra un filtro passa-banda di Fourier applicato ad una sinusoida rumorosa di 100 Hz con la frequenza centrale del filtro variata da 50 a 150 Hz. Entrambi richiedono la funzione FouFilter.m nel path di Matlab/Octave.

La figura a lato ([script Matlab](#)) mostra l'effetto della larghezza di banda di un filtro passa-basso di Fourier applicato a un tipico segnale con rumore bianco. Il grafico semi-log mostra il rapporto segnale/rumore (rosso) e gli errori percentuali nei parametri del picco (altezza, larghezza e posizione) in funzione della larghezza del filtro. Viene anche mostrato il

temporale, 'samplingtime' è la durata totale del segnale campionato in secondi, millisecondi o microsecondi; 'centerfrequency' e 'frequencywidth' sono la frequenza centrale e la larghezza del filtro in Hz, KHz, o MHz, rispettivamente; 'Shape' determina la ripidità del taglio. Se $\text{shape} = 1$, il filtro è Gaussiano; all'aumentare di shape il filtro diventa sempre più rettangolare. Si imposta $\text{mode} = 0$ per il filtro passa-banda, $\text{mode} = 1$ per l'escludi-banda (notch).

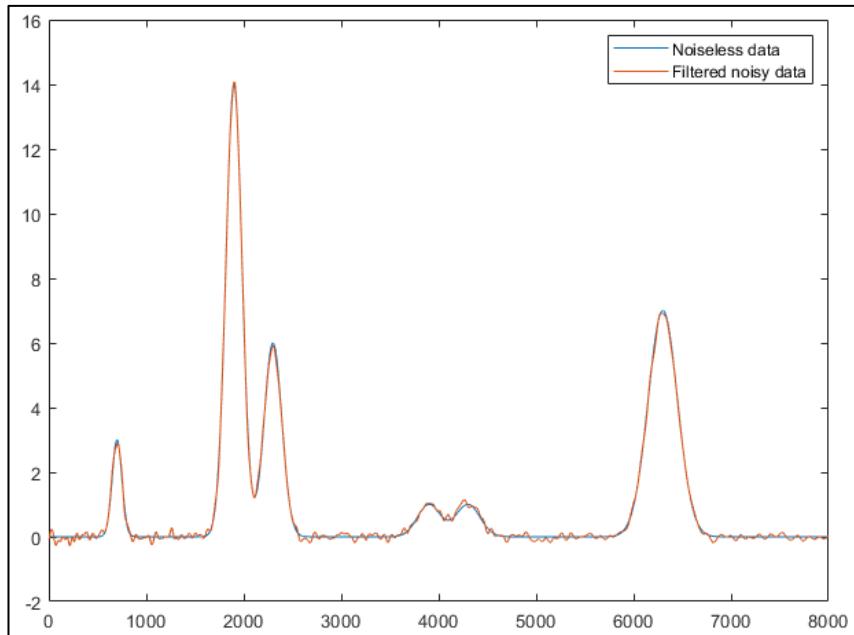
FouFilter restituisce il



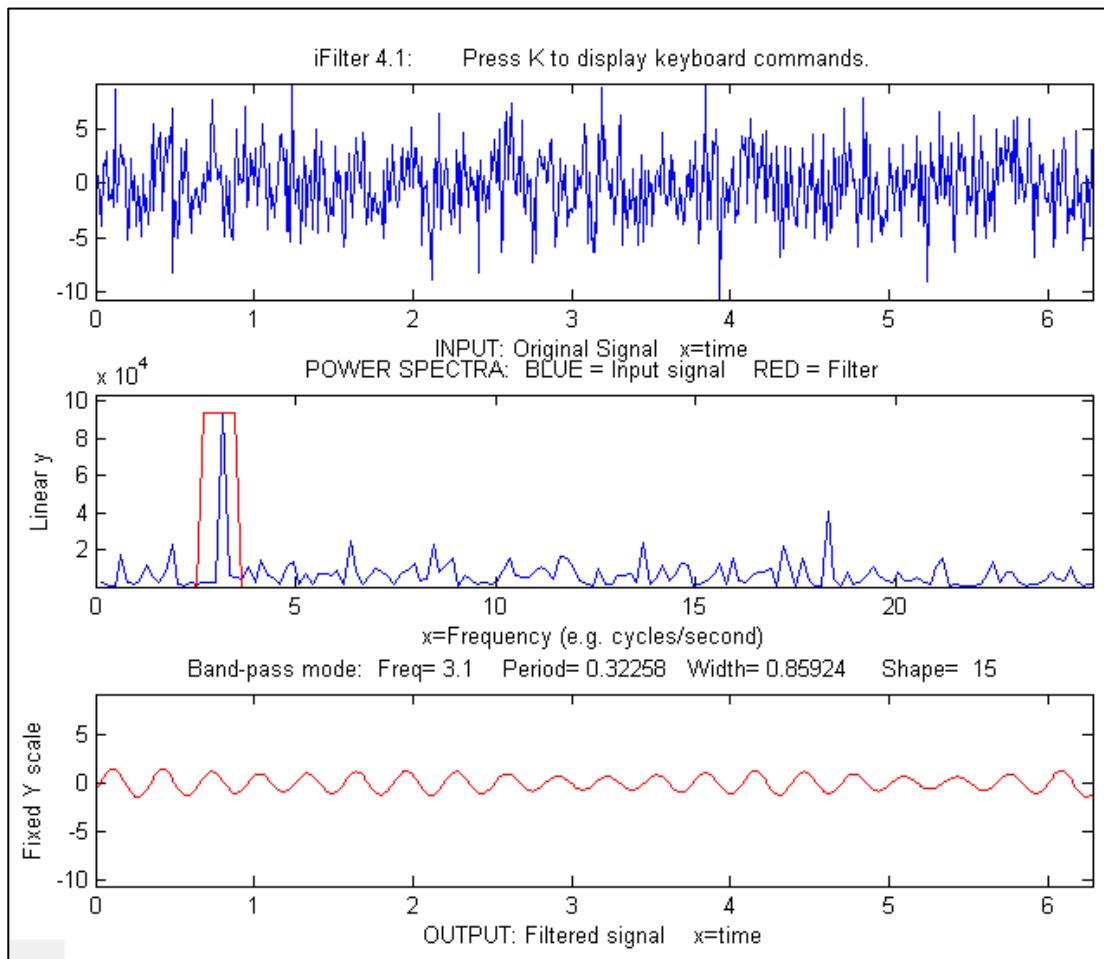
rapporto segnale/rumore. Come al solito, i risultati sono scarsi se la larghezza di banda è troppo bassa o troppo alta, ma in questo caso il rapporto segnale/rumore è migliore a una larghezza di banda relativamente bassa mentre la maggior parte delle misure dell'altezza e della larghezza del picco sono più accurate a valori molto più alti. [Un risultato leggermente diverso](#) si ottiene con `shape=2`, per cui la ripidità del taglio è maggiore. Come al solito, il compromesso è d'obbligo.

Lo strumento interattivo di smoothing dei dati Live Script include il filtro di Fourier tra molti altri metodi di smoothing. (A pagina 55; il link per il download: [DataSmoothing mlx](#)).

Filtro di Fourier segmentato. [SegmentedFouFilter.m](#) è una versione *segmentata* di FouFilter.m, che applica diverse frequenze centrali e larghezze a diversi segmenti del segnale. La sintassi è la stessa di FouFilter.m eccetto che i due argomenti di input `centerFrequency` e `filterWidth` devono essere vettori con i valori di `centerFrequency` e `filterWidth` per ciascun segmento. La funzione divide il segnale in un numero di segmenti di uguale lunghezza determinato da `centerFrequency` e `filterWidth`, che devono avere la stessa lunghezza. Per un aiuto e degli esempi digitare “`help SegmentedFouFilter`”. La figura seguente mostra "Example 2", che implementa un filtro passa-basso di Fourier con una larghezza di banda decrescente man mano che le larghezze dei picchi diventano più larghe da sinistra a destra. Se si guarda da vicino, si nota che il rumore casuale nel segnale filtrato, che era costante nel segnale originale, diminuisce all'aumentare della larghezza del



filtro.



Ulteriori applicazioni basate su Matlab che utilizzano un filtro *interattivo* di Fourier, [*iFilter.m*](#), mostrato sopra, consente di regolare i parametri del filtro premendo i tasti mentre se ne osserva dinamicamente l'effetto sul segnale. Questo è descritto a pagina 380. Esiste anche una versione per gli utenti Octave, [*ifilteroctave.m*](#) che utilizza tasti diversi per il centro del filtro e le regolazioni della larghezza.

Una dimostrazione di un filtro di Fourier in *tempo reale* viene trattata a pagina 339.

Wavelet e wavelet denoising

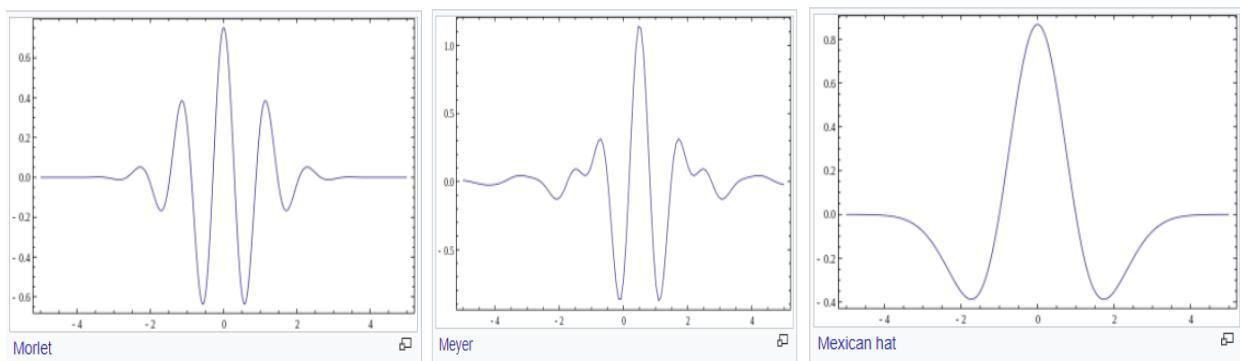
Le Wavelet sono letteralmente “ondine”, piccole forme d’onda oscillanti che iniziano da zero, si gonfiano al massimo e poi decadono rapidamente fino a zero. Si possono contrapporre alle onde seno e coseno, che continuano “perennemente”, ripetendosi all’infinito dal positivo al negativo. Nei paragrafi precedenti si è visto quanto sia utile utilizzare la Trasformata di Fourier di un segnale, che esprime un segnale come somma di onde seno e coseno, consentendo operazioni utili come convoluzione, deconvoluzione e filtraggio di Fourier. Ma c’è un aspetto negativo nella Trasformata di Fourier; copre l’*intera* durata del segnale, fornendo solo il contenuto *medio* della frequenza. Sebbene a pagina 98 abbiamo visto che è possibile utilizzare variazioni segmentate o risolte nel tempo della trasformata di Fourier per superare questa difficoltà, è disponibile un modo più sofisticato per risolvere questa limitazione dell’analisi di Fourier consiste nell’utilizzare le wavelet come set di base per rappresentare i segnali anziché le onde seno e coseno. Come le onde sinusoidali, le wavelet possono essere allungate o compresse lungo il loro asse “x” o asse del tempo per coprire le diverse frequenze. Ma a differenza delle onde sinusoidali, le wavelet possono essere *traslate lungo l’asse temporale* di un segnale per testare le variazioni temporali, perché le wavelet

sono di breve durata rispetto ai segnali con cui vengono utilizzate.

Le wavelet furono introdotte da matematici e fisici nei primi anni del 20° secolo e il successivo sviluppo è stato soprattutto matematico. Molti dei trattati sulle wavelet in letteratura sono finalizzati agli aspetti matematici formali, che sono stati “elaborati in modo estremamente dettagliato” (secondo il riferimento 82). Il sistema dei valori dei matematici – dimostrazione rigorosa, esplorazione esaustiva, assunzione di un background matematico e la necessità di una notazione compatta - le rendono difficili per i non specialisti. Per questo motivo, ci sono molte introduzioni “facili” all’argomento (riferimenti 79 - 82) che promettono di attenuare il colpo dell’astrazione matematica. Perciò, non si ripeteranno qui tutti quei dettagli matematici. Si cercherà, piuttosto, di mostrare cosa si possa ottenere utilizzando le wavelet *senza* comprendere tutta la matematica sottostante. Un particolare interesse viene posto alle situazioni in cui le wavelet funzionano meglio delle migliori tecniche convenzionali a disposizione, ma anche alle poche situazioni in cui le tecniche convenzionali rimangono superiori.

Una [trasformata wavelet](#) (WT) è una decomposizione di un segnale in un insieme di “funzioni base” costituite da contrazioni, espansioni e traslazioni di una funzione wavelet (rif. 85). Può essere calcolata mediante una ripetuta convoluzione del segnale (pag. 103) con la wavelet scelta, mentre viene traslata lungo la dimensione temporale (per misurare la variazione nel tempo) e quando la wavelet viene allargata o compressa (per misurare le diverse frequenze). Poiché vengono rilevate due dimensioni, il risultato è naturalmente una superficie 3D (tempo-frequenza-ampiezza) che può essere convenientemente visualizzato con un grafico [a isolinee](#) tempo-frequenza con diversi colori per rappresentare le ampiezze in quell’istante e a quella frequenza. Ovviamente, tali calcoli richiederanno algoritmi più complessi e tempi di esecuzione maggiori, spesso da 5 a 20 volte più lunghi rispetto ai metodi convenzionali. Questo potrebbe essere stato un problema agli albori dei computer, ma con i moderni processori veloci e la grande capacità di memoria è poco probabile che sia motivo di preoccupazione al momento.

Le wavelet vengono utilizzate per la visualizzazione, l’analisi, la compressione e il denoising di dati complessi. Ci sono decine di diverse forme di wavelet, che di per sé fanno una grande differenza dall’analisi di Fourier basata su onde sinusoidali. L’[articolo di Wikipedia sulle wavelet](#) ne menziona tre, mostrate di seguito, da sinistra a destra: la Meyer, la Morlet e il cappello messicano. Le wavelet sono costruite convenzionalmente in modo che l’area sottesa dalla curva sia zero e l’integrale dei loro quadrati sia 1.0.



In Matlab, il modo più semplice per accedere a questi strumenti è utilizzare il *Wavelet Toolbox*, se incluso nella licenza del sito Matlab del campus, aziendale o se lo si compra separatamente. Questo toolbox include un’interfaccia utente grafica (GUI) per un Analizzatore Wavelet, Multirisoluzione del Segnale e un "Wavelet Signal Denoiser", così come una [vasta raccolta di funzioni wavelet a riga di comando](#). La documentazione è disponibile su

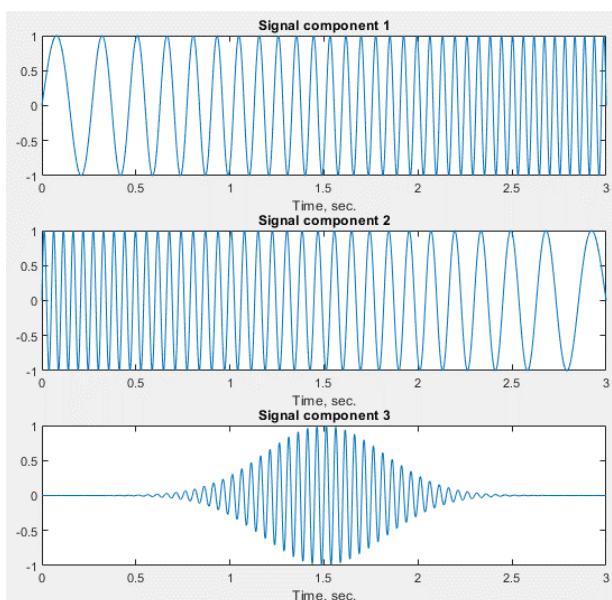
[https://www.mathworks.com/products/wavelet.html.](https://www.mathworks.com/products/wavelet.html)

Tuttavia, *non* è assolutamente necessario avere il Wavelet Toolbox, perché il codice è stato pubblicato su Internet in diversi linguaggi. Ad esempio, diversi documenti (riferimenti 84, 90) includono o fanno riferimento al codice Matlab che *implementa le wavelet utilizzando le sole funzioni native di Matlab* “fft”, “ifft” e “conv”. In questo capitolo verranno utilizzati tutti questi approcci software per descrivere le proprietà e le applicazioni delle wavelet alle misure scientifiche.

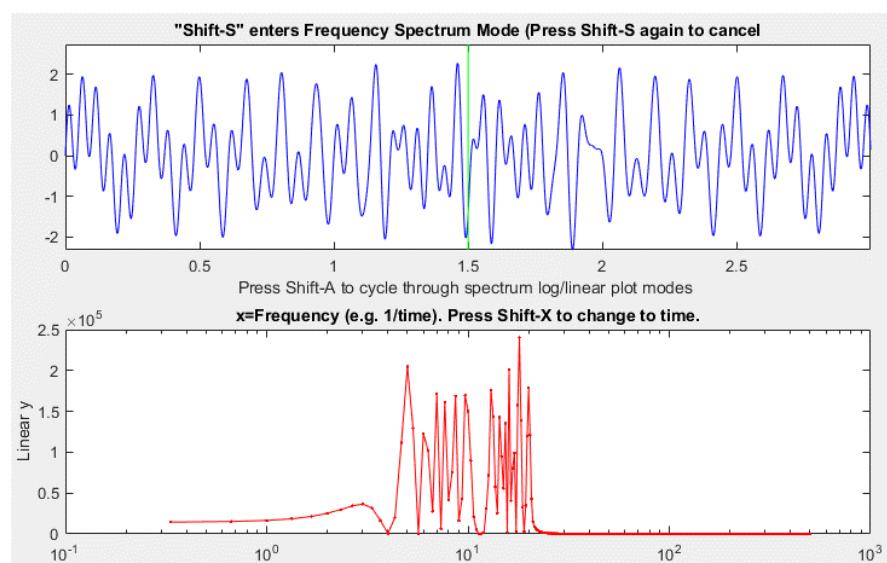
Visualizzazione ed analisi

Le wavelet sono piuttosto efficaci nel visualizzare segnali complicati e aiutare gli osservatori a dar loro un senso. Un buon esempio è fornito nel riferimento 84, che descrive un segnale campionato a 1000 Hz costituito da tre componenti sovrapposte inizialmente sconosciute allo sperimentatore.

Queste componenti sono mostrate nella figura seguente: (1) un'onda sinusoidale (detta ‘chirp’) ‘spazzolata’ da 5 Hz a 20 Hz, (2) un'altra onda sinusoidale (‘chirp’) spazzolata in ordine inverso da 20 Hz a 5 Hz, e infine (3) un'onda sinusoidale modulata-Gaussiana a 20 Hz col picco al centro del segnale.



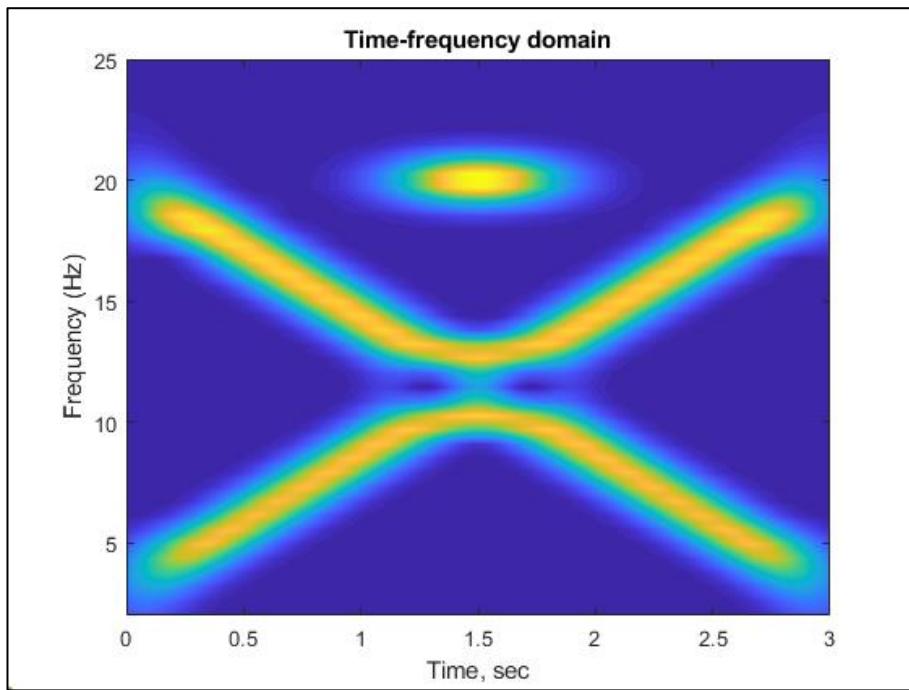
Quando questi tre vengono sovrapposti, la forma d'onda risultante, mostrata nel pannello superiore della figura sotto



(visualizzata in [iSignal.m](#)) è un guazzabuglio complicato che non offre alcun indizio sulla struttura sottostante. Lo spettro della trasformata di Fourier convenzionale, mostrato nel pannello inferiore, illustra solo che le componenti di frequenza del segnale che rientrano nell'intervallo compreso tra 3 Hz e 25 Hz circa. In effetti, lo spettro di Fourier è

fuorviante; suggerisce che potrebbero esserci *due* componenti, una ad una gamma di frequenza più alta dell'altra, con un piccolo intervallo intorno ai 12 Hz. Ma in realtà ci sono *tre* componenti, due delle quali coprono un'ampia gamma di frequenze e la terza è fissa a circa 12 Hz. Lo spettro di Fourier qui non aiuta.

Al contrario, il contorno tempo-frequenza-ampiezza, basato sulle wavelet, mostrato sotto, calcolato con la wavelet di Morlet dal codice [Matlab nel riferimento 86](#), aiuta a svelare le complessità, mostrando chiaramente tutte e tre le componenti. In questa schermata, il giallo corrisponde alle

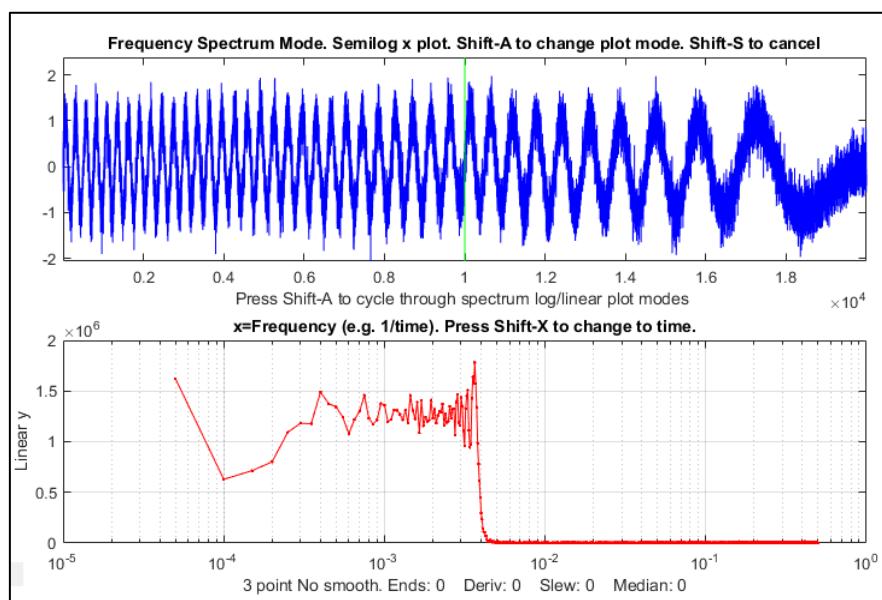


ampiezze maggiori e il blu a quelle più basse. L'onda sinusoidale Gaussiana modulata a 20 Hz è chiaramente visibile in alto al centro.

(Per inciso, c'è un'ambiguità riguardo alle due onde sinusoidali nel punto in cui si incrociano in frequenza nel mezzo del segnale e si annullano momentaneamente; continuano ad andare nella stessa direzione, formando una "X", oppure entrambe invertono la direzione, formando una "V" e il suo riflesso? I due comportamenti porterebbero allo stesso segnale finale. L'assunto più semplice sarebbe il primo).

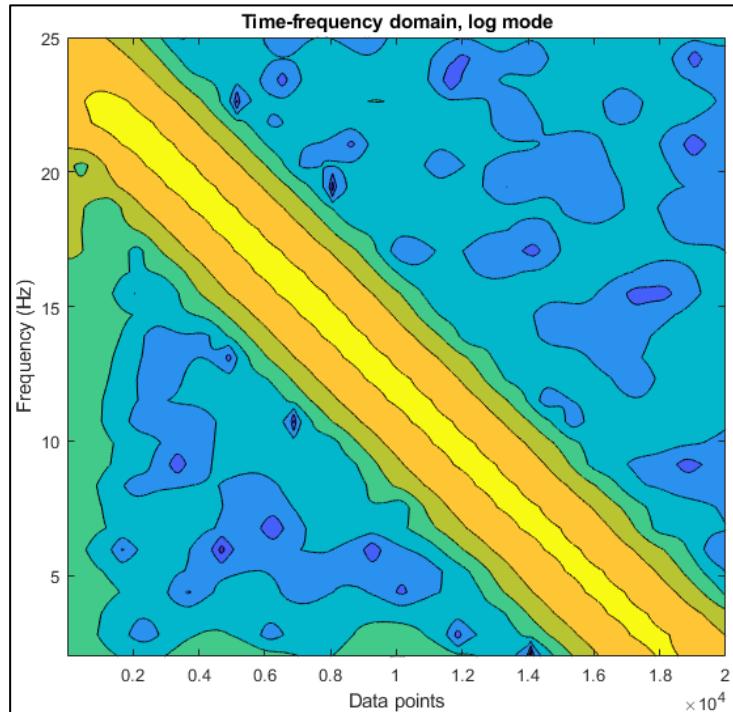
Un altro esempio è più vicino a una tipica applicazione scientifica: estrarre un segnale da un eccesso di rumore e interferenza. Si

basà sul segnale dei "picchi sepolti" utilizzato in precedenza, a pagina 99. Il segnale (pannello superiore nella schermata iSignal a lato) ha una coppia di picchi Gaussiani nascosti, totalmente sepolti in un'onda sinusoidale a frequenza variabile molto più forte e da un rumore bianco casuale. Lo spettro di Fourier, osservato qui nel pannello inferiore, ancora una volta



non offre alcun indizio evidente dei picchi sottostanti.

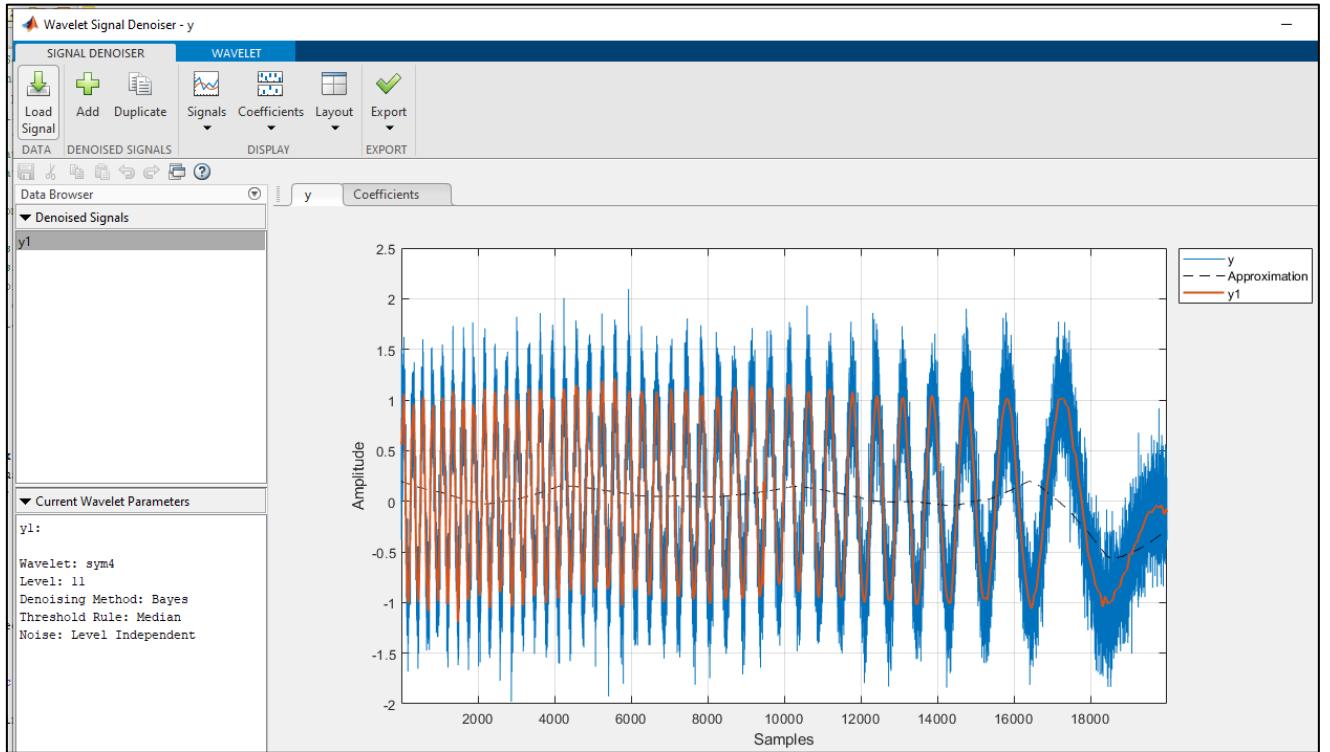
Qui è stata applicata la wavelet Morlet a questo segnale per creare la matrice tempo-frequenza-ampiezza mostrata a sinistra ([script](#) e [funzione wavelet Morlet](#)). Non farsi distrarre dalla grande striscia diagonale gialla; che corrisponde alla spazzolata dell'onda sinusoidale. Si osservino invece le due gobbe verdi più deboli in basso a sinistra, all'estremità delle basse frequenze, vicino ai punti 5000 e 10000. Quelle sono le due vette Gaussiane. Sulla base di questa osservazione, sarebbe giustificato eseguire uno [smoothing](#) o un [curve-fitting](#) in quella specifica regione, come fatto in precedenza a pagina 99. (Si può confrontare questo grafico con il display segmentato dello spettro di Fourier per questo segnale mostrato in quella pagina, che è più rozzo ma mostra informazioni simili. La wavelet è chiaramente uno strumento a risoluzione maggiore).



Riduzione del rumore con le wavelet

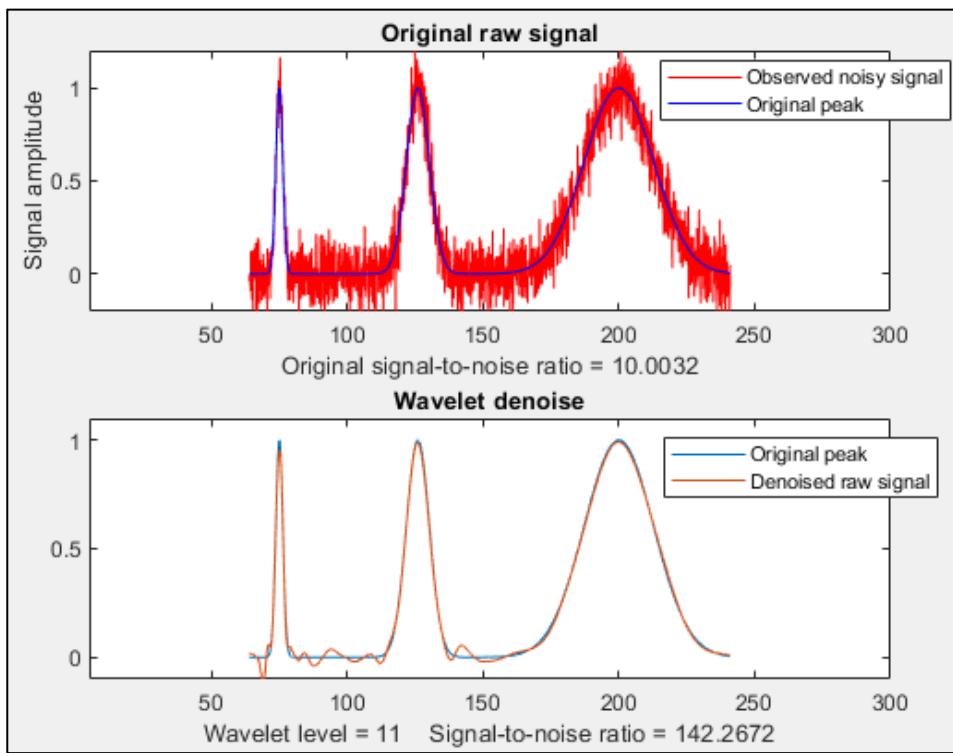
Nel contesto delle wavelet, il termine “denoising” significa ridurre il più possibile il rumore senza distorcere il segnale. Il denoising utilizza la matrice tempo-frequenza-ampiezza creata dalla trasformata wavelet. Si basa sul presupposto che il rumore indesiderato viene separato dal segnale desiderato e suddiviso nelle sue componenti di frequenze. Più in generale, nelle misure scientifiche, le componenti del segnale desiderate si trovano a frequenze relativamente *basse* mentre il rumore sta soprattutto tra le frequenze più *alte*. Il processo viene controllato sia selezionando il tipo di wavelet che scegliendo un numero intero positivo chiamato “livello (o parametro) di scala” della wavelet; più alto è il livello, più bassa è la divisione in frequenza tra segnale e rumore. (In tal senso, il livello della wavelet è qualitativamente simile alla larghezza nell'operazione di smoothing).

Anche qui, il Wavelet Toolbox di Matlab fornisce degli strumenti utili. Innanzitutto, c'è l'app con interfaccia grafica chiamata “Wavelet Signal Denoiser”. Il tipo di wavelet e il suo livello di scala sono selezionabili manualmente. È stata utilizzata per analizzare il segnale dei “picchi sepolti” descritto in precedenza, utilizzando la wavelet “sym4” ad un livello di scala relativamente alto di 11, perché i livelli più bassi fanno passare molto del segnale sinusoidale 'spazzolato' mentre livelli maggiori smorzano troppo i picchi Gaussiani. (Il numero nel nome della wavelet si riferisce al numero dei cosiddetti “[momenti nulli](#)”. Con più momenti nulli significa si possono rappresentare funzioni più complesse). Il risultato “Approximation” (la linea tratteggiata nel grafico sotto) è l'informazione sulla *bassa frequenza* nei dati, e si può chiaramente vedere che è una versione “denoised” del segnale originale (in blu). I due dossi ai numeri 5000 e 10000 del campione sono i due picchi Gaussiani.

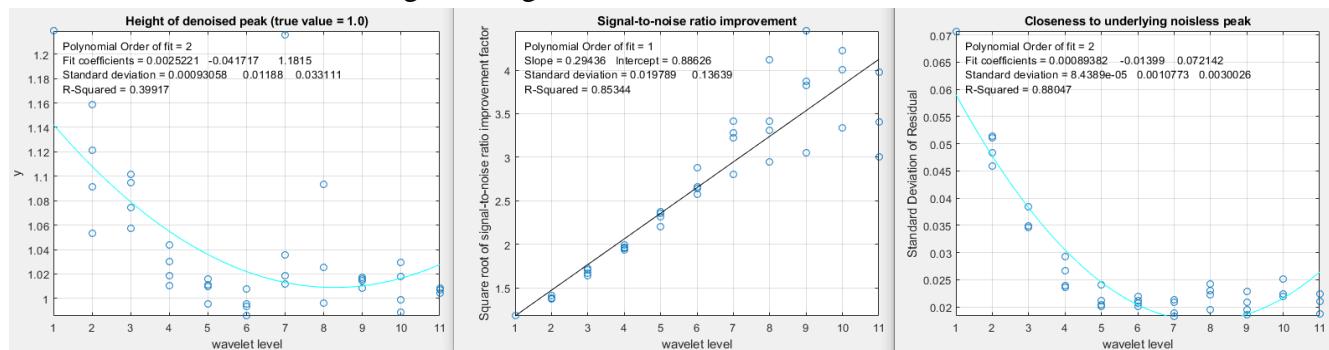


Quindi, sia la wavelet sym4 nel Wavelet Signal Denoiser, nella foto sopra, che la matrice tempo-frequenza-ampiezza della wavelet Morlet, mostrata precedentemente, evidenziano i picchi Gaussiani nascosti, ma li visualizzano in modo diverso.

Oltre alla app GUI, c'è anche una funzione di denoising a riga di comando chiamata “`wdenoise`” (sintassi: `wdenoise(noisydata, level, ...)`). La selezione del tipo e del livello di scala della wavelet viene impostata includendo argomenti di input opzionali della funzione. Il vantaggio di una *funzione*, rispetto ad una app GUI, è che è possibile scrivere script che confrontano rapidamente ed automaticamente molti settaggi diversi, o che confrontano i risultati di diversi metodi di riduzione del rumore convenzionali, oppure per automatizzare l'elaborazione batch di grandi set di dati memorizzati (vedere pagina 337). Ad esempio, la questione sulla selezione ottimale del livello di scala della wavelet può essere risolta dallo script [OptimizationOfWaveletLevel3peaks.m](#), che crea un segnale costituito da tre picchi rumorosi Gaussiani (o Lorentziani) con altezza unitaria, con larghezze diverse, cui aggiunge del rumore bianco, come nella figura seguente.



Lo script utilizza la funzione `wdenoise.m` per effettuare la 'pulizia' del segnale con la wavelet "coiflet" dal livello 1 all'11, misurando tre grandezze per ogni livello: (a) l'altezza dei picchi, (b) il miglioramento del rapporto segnale-rumore, e (c) la somiglianza al segnale sottostante senza rumore, come mostrato nei tre grafici seguenti.



Da questi grafici si può vedere che un livello di scala di 7 circa, è quello ottimale, in questo caso. Al di sopra di 7, il rapporto segnale-rumore (grafico centrale) continua a migliorare, ma i risultati non sono affidabili e tendono a sparpagliarsi troppo. (Passando ai picchi Lorentziani - riga 28 dello script - si ottengono risultati simili).

Lo script [WaveletsComparison.m](#) confronta cinque diversi tipi di wavelet con lo stesso segnale: BlockJS, bior5.5, coif2, sym8, e db4, tutte a livello di scala di 12 ([grafico](#)). I risultati sono simili ma il sym8 ha un leggero vantaggio. Per la maggior parte delle forme di picchi con del rumore bianco aggiunto, per lo smoothing i diversi tipi di wavelet si comportano in modo simile. Per i segnali con rumore pesato ad *alta* frequenza, la wavelet [bior5.5](#) funziona meglio delle altre ([script](#); [grafico](#)). Per gli impulsi quadrati, la wavelet [Haar è chiaramente superiore](#).

Un altro [script](#), `SmoothVsWavelets2Gaussians.m`, confronta cinque diverse tecniche di smoothing non-wavelet e due diverse wavelet, tutte utilizzano lo stesso segnale simulato costituito da due picchi Gaussiani con una *differenza di 50 volte* della larghezza del picco, con del rumore bianco aggiunto. Per ciascun metodo, vengono misurati gli errori percentuali di altezza, larghezza e area del picco, nonché la differenza tra il segnale nascosto sottostante e il segnale ripulito dal rumore (o con smoothing). Ciò illustra un vantaggio significativo che il denoising wavelet ha rispetto allo smoothing; *si adatta molto meglio alle differenze delle larghezze dei picchi*. In questa tabella è mostrato un riepilogo dei tipici risultati. (Il picco 1 è il picco stretto e il picco 2 è 50 volte più largo).

Risultati tipici Errori percentuali del piccol picco2				
Metodo	Residui	Altezza	Larghezza	Area
Originale	9.88%	6.29% 25.8%	6.31% -23.24%	-2.49% 0.86%
Gaussiano	2.53%	-3.34% -3.79%	5.72% 4.35%	-2.6% 0.82%
Segmentata	2.04%	-24.48% 3.09%	37.8% 0.21%	-7.07% 0.85%
Savitsky-Golay	2.93%	-2.08% 6.45%	8.66% -3.04%	-1.9% 0.86%
Filtro RC	3.29%	-6.53% 9.58%	16.06% -5.45%	-11.76% 0.86%
Filtro Hamming	2.91%	-5.12% 8.7%	8.19% -5.31%	-2.17% 0.86%
wavelet coif2	1.02%	1.78% 1.18%	-5.59% 0.22%	-6.54% 0.75%
wavelet db2	1.17%	11.47% 3.82%	3.36% 2.38%	-5.34% 0.81%

I “Residui” sono le differenze percentuali tra il segnale senza rumore in esame e il segnale con rumore casuale dopo il denoising; tiene conto sia del rumore residuo nel segnale che della distorsione della forma del segnale. Come si può vedere, *la wavelet coif2 esce in vantaggio per la maggior parte delle misure*. Ciò illustra i vantaggi pratici più significativi del denoising wavelet: (1) fornisce risultati che sono almeno altrettanto buoni, e spesso migliori, dei metodi convenzionali 55 di smoothing; (2) è più facile da usare perché si adatta automaticamente a diverse larghezze dei picchi; e (3) è più facile da ottimizzare perché nella maggior parte dei casi solo l'impostazione del livello di scala fa la differenza nei risultati pratici.

Tuttavia, ci sono alcune situazioni in cui i metodi convenzionali restano i migliori. Ad esempio, nel calcolare le derivate seconde di picchi rumorosi con larghezze variabili, uno smoothing Gaussiano-pesato segmentato (pag. 326) fornisce un rapporto segnale/rumore migliore di un denoising wavelet ([script](#); [grafico](#)), specialmente se il rapporto segnale-rumore è scarso ([grafico](#)), presumibilmente perché lo spettro di frequenza del rumore è così fortemente pesato sulle alte frequenze. Inoltre, il denoising wavelet non funziona affatto se l'ampiezza del rumore è *proporzionale* all'ampiezza del segnale, anziché *costante* ([script](#); [grafico](#)). Talvolta, se il rapporto segnale-rumore originale è pessimo, il denoising wavelet produce degli stretti spike artificiali nel segnale ripulito, anche quando si usa il [soft thresholding](#). Questi, però, sono casi speciali; ci sono molte altre situazioni in cui il denoising wavelet è davvero il metodo da scegliere, sempre che la maggiore lentezza delle wavelet non sia un problema.

Il miglioramento delle prestazioni del rapporto segnale-rumore del denoising wavelet viene comparato ai metodi tradizionali, in funzione del rapporto di smoothing, nel capitolo precedente sullo smoothing, pag. 56. Il metodo wavelet funziona meglio ma è 10 volte più lento persino dello smoothing Savitzky-Golay.

Il denoising wavelet è uno dei diversi tipi di smoothing della riduzione del rumore disponibili nel tool di smoothing dei dati Live Script descritto a pagina 55. Questo è un modo molto conveniente per confrontare il denoising wavelet con altre forme di riduzione del rumore, come la media mobile, il filtraggio Savitzky-Golay e Fourier.

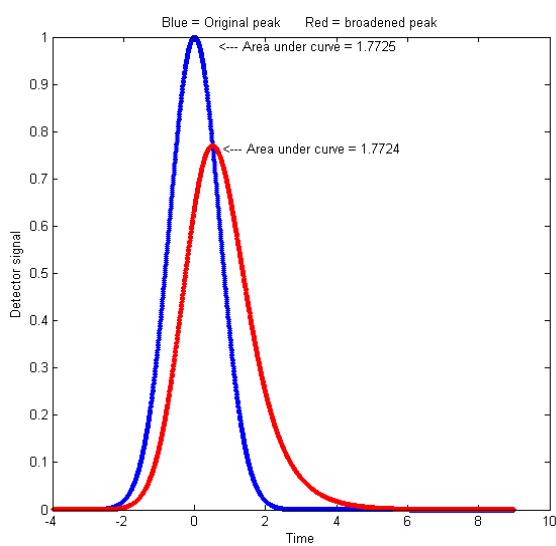
Per i programmati Python, esiste un pacchetto wavelet chiamato [PyWavelets](#) che ha oltre 100 filtri wavelet integrati, il supporto per wavelet personalizzate ed è compatibile con il toolbox wavelet di Matlab. Vedere anche “[A gentle introduction to wavelet for data analysis](#)” per una trattazione pressoché grafica basata su Python.

Integrazione e misura dell'area di un picco

L'integrazione simbolica delle funzioni e il calcolo degli integrali definiti sono argomenti che vengono presentati nei corsi di Calcolo elementari. L'integrazione numerica dei segnali digitalizzati viene applicata, nell'elaborazione analitica dei segnali, come metodo per misurare le aree sotto i

picchi nelle curve dei segnali.

Per esempio, le misure delle aree dei picchi sono importantissime in [cromatografia](#), una classe di tecniche di misure chimiche in cui una miscela di componenti viene fatta scorrere attraverso un tubo, o uno strato, preparati chimicamente, che consente ad alcune componenti della miscela di scorrere più velocemente di altre, seguito da un dispositivo chiamato *rilevatore* che misura e registra le



componenti dopo la separazione. Idealmente, le componenti sono sufficientemente separate in modo da formare un *picco* distinto nel segnale del rilevatore. Le altezze dei picchi sono [calibrate](#) alla concentrazione di quel componente misurando i picchi ottenuti da "soluzioni standard" in concentrazione nota. In cromatografia è normale misurare l'*area* dei picchi del rilevatore anziché l'*altezza*, perché l'area è meno influenzata dal rumore casuale e dai meccanismi di dilatazione del picco (dispersione) provocati dalle molecole di una determinata sostanza diluite e disperse anziché concentrate in un "grumo" di materiale attraversando il tubo o lo strato. Gli effetti di tali dispersioni, che derivano da molteplici fonti, causano

l'abbassamento, l'allargamento, e talvolta una certa asimmetria dei picchi cromatografici, ma hanno *pochissimo effetto sull'area totale sottesa dal picco*, se il numero totale di molecole resta lo stesso. Se la risposta del rilevatore è lineare rispetto alla concentrazione del materiale, l'*area* del picco resta proporzionale alla quantità totale della sostanza che attraversa il rilevatore, anche se l'*altezza* del picco è più piccola. Un esempio grafico è mostrato a lato ([codice Matlab/Octave](#)), che disegna il segnale del rivelatore in funzione del tempo, dove la **curva blu** rappresenta il segnale originale e la **curva rossa** mostra l'effetto dell'ampliamento provocato dalla dispersione. L'altezza del picco è inferiore e la larghezza è maggiore, ma l'*area* sotto la curva è quasi la stessa. Se l'entità dell'ampliamento cambia tra il tempo in cui vengono eseguiti gli *standard* e il periodo in cui vengono analizzati i *campioni* ignoti, allora *le misure delle aree dei picchi saranno più accurate e affidabili* di quelle delle altezze. (L'altezza del picco sarà proporzionale alla quantità di materiale solo se la larghezza e la forma del picco sono costanti). Ecco un altro esempio con un ampliamento maggiore: ([script](#) e [grafico](#)).

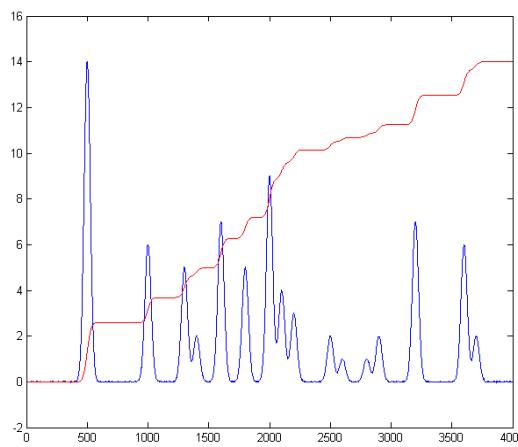
Le misure dell'area dei picchi vengono occasionalmente utilizzate anche nella *spettroscopia*, ad esempio nei metodi di [iniezione di flusso](#) e nell'assorbimento atomico in forni di grafite ([rif.87](#)). In tale applicazione, le curve di calibrazione basate sulle misure delle aree sono più lineari di quelle dell'altezza del picco perché la maggior parte dell'area di un picco viene misurata quando l'assorbanza transitoria è inferiore al massimo e dove viene seguita di più la [Legge di Beer](#).

Al contrario, le misure delle altezze dei picchi sono *più semplici da fare* e sono *meno soggette alle interferenze* dei picchi vicini e sovrapposti. Un ulteriore svantaggio della misura dell'area è che devono essere determinati i punti di inizio e di fine del picco, il che può risultare difficile soprattutto se i più picchi si sovrappongono. In linea di principio, il [curve fitting](#) può misurare le aree dei picchi anche se si sovrappongono, ma ciò richiede che le forme dei picchi siano note almeno approssimativamente (vedere, comunque, [PeakShapeAnalyticalCurve.m](#) descritto a pagina329).

I picchi cromatografici sono spesso descritti come una [funzione Gaussiana](#) o una [convoluzione](#) di una Gaussiana con una funzione esponenziale. Un confronto quantitativo dettagliato tra l'altezza del picco e la misura dell'area del picco è fornito a pagina 306: [Perché misurare l'area anziché l'altezza del picco?](#) (In spettroscopia, ci sono [altri meccanismi di espansione](#), come [l'espansione Doppler](#) causata dal movimento termico, che si traduce in una [funzione Gaussiana espansa](#)).

Prima dell'avvento dei computer, i ricercatori usavano una varietà di metodi intelligenti ma arcaici per calcolare le aree dei picchi:

- (a) si disegna il segnale su un foglio millimetrato, si ritaglia il picco con le forbici, quindi si pesa il ritaglio con una micro-bilancia rispetto a una sezione quadrata di area nota;
- (b) si contano i quadrati della griglia sotto una curva disegnata su carta millimetrata;ù
- (c) si usa un [integratore disco-sfera-cilindro](#) meccanico;
- (d) si usa una riga e la geometria per calcolare l'area di un [triangolo costruito con i lati tangenti a quelli del picco](#), e l'area geometrica di quel triangolo;
- (e) si calcola la somma cumulativa dell'ampiezza del segnale e si misurano le altezze dei gradini risultanti, come nella figura seguente. (Questo è un metodo precedentemente usato nella spettroscopia NMR del protone, in cui l'area di ogni picco o gruppo di picchi è proporzionale al numero di atomi di idrogeno equivalenti responsabili di quel picco).



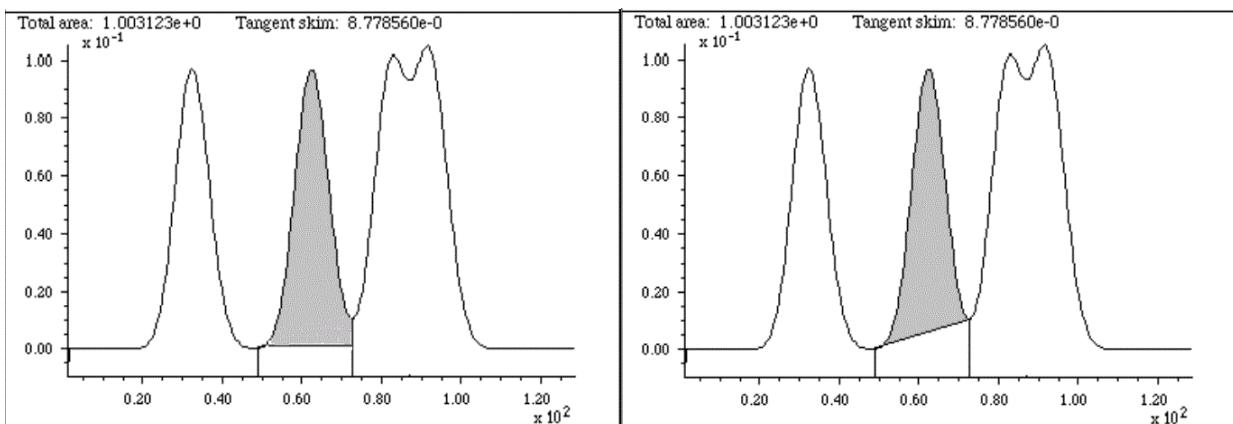
Ora che quasi tutti gli strumenti di misura hanno una certa potenza di calcolo, è possibile utilizzare metodi digitali più precisi e convenienti. Indipendentemente da come viene misurata, le *unità* dell'area del picco sono il *prodotto* di unità x e y. Quindi, in un cromato-gramma in cui x è il tempo in minuti e y è in volt, l'area è in volt-minuto. Negli spettri di assorbimento dove x è nm (nanometri) e y è l'assorbanza, l'area è assorbanza-nm. Per questo motivo, l'ampiezza numerica dell'area del picco sarà sempre diversa da quella dell'altezza del picco. Se si esegue un'analisi quantitativa di campioni noti

mediante una [curva di calibrazione](#), è necessario utilizzare lo stesso metodo di misurazione sia per gli standard che per i campioni, *anche se le misure sono imprecise*, purché l'*errore sia lo stesso* per tutti gli standard e i campioni (motivo per cui un metodo approssimativo come la costruzione di un triangolo funziona meglio del previsto).

Il metodo migliore per calcolare l'area di un picco dipende dal fatto che il picco sia isolato o sovrapposto ad altri picchi o se sia sovrapposto ad una linea di base diversa da zero. Per un picco isolato, Yuri Kalambet (riferimento 72) ha dimostrato che l'area della [Regola del Trapezio](#), come calcolata dalla funzione "trapz.m" di Matlab, è una stima efficiente dell'intera area del picco con un errore straordinariamente basso, anche se sono presenti solo pochi punti lungo l'ampiezza del picco, mentre la [regola di Simpson](#) è meno efficiente nell'integrazione dell'intera area. Per un picco Gaussiano, la regola del trapezio richiede solo 0.62 punti per ciascuna deviazione standard (2.5 punti entro la larghezza di base 4σ) per ottenere un errore di integrazione dello 0.1% soltanto. Una [simulazione digitale](#) supporta questo risultato. Per i picchi asimmetrici, però, sono necessari più punti.

La gestione dei picchi sovrapposti

Il modo classico per gestire il problema dei picchi sovrapposti consiste nel tracciare due linee verticali dai limiti sinistro e destro del picco fino all'asse x e poi misurare l'area totale delimitata dalla curva del segnale, dall'asse x (riga dove $y=0$), e dalle due linee verticali, mostrato come area ombreggiata nella figura.

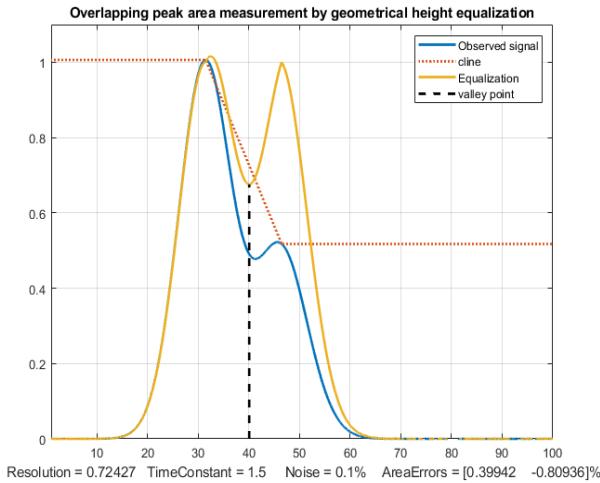


La misura dell'area per picchi Gaussiani sovrapposti, utilizzando il metodo del **taglio verticale** (l'area ombreggiata a sinistra) e quello del **taglio tangente** (l'area ombreggiata a destra).

Questo è spesso chiamato metodo del **taglio perpendicolare**; è un po' noioso da fare a mano, ma è un compito facile per un computer. I limiti sinistro e destro di ciascun picco sono presi sia (a) gli avvallamenti (minimi) tra due picchi che (b) i punti intermedi tra i centri di picchi adiacenti. Il metodo del punto a metà ha il vantaggio che l'SNR ad un segnale *massimo* è generalmente migliore che a un *minimo*, quindi è probabile che il massimo possa essere localizzato con maggiore precisione rispetto ai minimi. Il presupposto di base del metodo del taglio perpendicolare è che l'area persa tagliando i piedi di un picco è compensata includendo i piedi del picco adiacente.

Questo funziona abbastanza bene se i picchi sono Gaussiani, simmetrici, non troppo sovrapposti e non troppo diversi in altezza. Inoltre, la linea di base deve essere nulla; qualsiasi segnale di fondo estraneo deve essere sottratto prima della misura. Utilizzando questo metodo, è possibile stimare l'area del secondo picco nell'esempio sopra con una precisione di circa lo 0.3%. Gli ultimi due picchi, invece, danno errori maggiori del 4%, e questo solo perché i due picchi nell'esempio hanno la stessa altezza e larghezza; più in generale, l'errore è molto maggiore se i due picchi sono troppo sovrapposti. Come regola generale, affinché questo metodo sia accettabile, l'avvallamento tra i picchi deve essere piuttosto basso, forse un quarto o un quinto dell'altezza del picco adiacente inferiore. Esistono altri metodi geometrici che possono ridurre, in molti casi, tali errori. Lo sharpening dei picchi con i metodi che conservano l'area come l'auto-deconvoluzione (pag. 108) o quelli basati sulla derivata (pag. 75) riduce il grado di sovrapposizione e può ridurre notevolmente gli errori di misura dell'area commessi dal metodo del taglio verticale, per esempio come in Calc di Excel/OpenOffice [PeakSharpeningAreaMeasurementDemo.xlsx \(schermata\)](#). Inoltre, i *picchi* asimmetrici risultanti dall'[espansione esponenziale](#) si possono [simmetrizzare con l'addizione ponderata della sua derivata prima](#), rendendo le aree del taglio verticale [più accurate](#) (pag. 138). In entrambi i casi, potrebbe essere necessario intensificare maggiormente lo sharpening, se questo è l'unico modo per formare una valle tra i picchi di cui si desidera misurare le aree. Nel caso in cui un singolo picco sia sovrapposto a una linea di base diritta o vagamente curva, è possibile utilizzare il metodo del **taglio tangente**, che misura l'area tra la curva e una linea di base rettilinea disegnata lungo la base del picco (p.es. l'*area ombreggiata* nella figura precedente a destra). In generale, la parte più difficile del problema è l'incertezza nel determinare la forma della linea di base sotto i picchi e quando ogni picco inizia e quando finisce. Una volta determinati, si sottrae la linea di base da ogni punto tra quello iniziale e quello finale, si sommano e si moltiplica per l'intervallo dell'asse x. Per inciso, lo smoothing di un segnale rumoroso non modifica le aree dei picchi, ma può rendere più facile determinare i punti di inizio e fine del picco. Lo svantaggio dello smoothing è che aumenta la larghezza del picco e la sovrapposizione di picchi adiacenti. I metodi numerici di sharpening, per esempio, il [derivativo](#) e la [deconvoluzione di Fourier](#), possono aiutare col problema della sovrapposizione dei picchi, ed entrambe queste tecniche hanno l'utile proprietà di non modificare l'area totale dei picchi. Vedere [3peaks.pdf](#) per una serie di esempi (script Matlab [GLSDPerpDropDemo16.m](#)). Consultare anche l'appendice a pagina 364 per un altro esempio.

Altri metodi. Sebbene il metodo del taglio verticale rimanga lo standard, ci sono altri due metodi geometrici che possono funzionare meglio in qualche caso.



Il metodo della "equalizzazione", illustrato nella figura a lato, localizza il punto del taglio verticale. Viene costruito un insieme di tre segmenti rettilinei che toccano i massimi stimati dei due picchi, indicati dalla linea rossa tratteggiata etichettata con "cline" nella figura a lato. Il risultato della divisione del segnale originale, in blu, per questa linea, è un segnale temporaneamente normalizzato (la linea gialla) che ha più o meno le stesse altezze dei picchi. L'effetto di questo trattamento è quello di *rendere gli avvallamenti più profondi* tra i picchi, in modo che

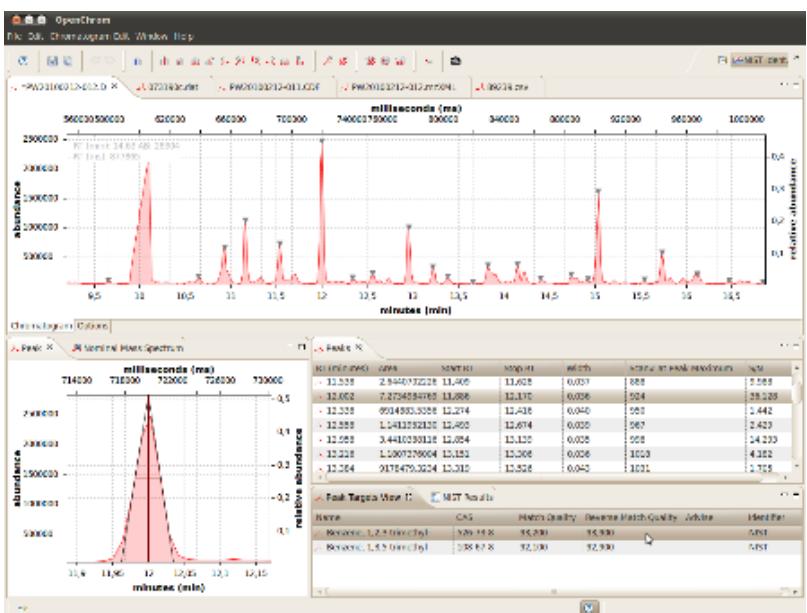
rimanga distinta per valori inferiori dell'altezza del secondo picco. Questo si usa solo allo scopo di determinare il punto di separazione tra i picchi, mostrato con una linea verticale nera, per poi essere abbandonato. Vengono poi calcolate le aree col taglio verticale sul segnale *originale* osservato (linea blu). Da notare che questo nuovo punto di separazione non è esattamente lo stesso nell'avvallamento del segnale originale, né è esattamente il punto a metà strada tra le due posizioni dei picchi. Questa operazione sarebbe difficile da eseguire manualmente, ma il software può farla facilmente, fornendo solo una stima iniziale delle due posizioni di picco basata sul segnale osservato.

Il metodo "riflessione/sottrazione", mostrato a lato, è più semplice, ma richiede che il picco più grande sia perfettamente simmetrico. Una stima del primo picco isolato viene costruita riflettendo la sua metà sinistra e usandola per sostituire la metà destra, ottenendo la linea tratteggiata rossa nella figura. Quindi quel picco stimato viene semplicemente sottratto dall'intero segnale per rivelare il secondo picco isolato (linea gialla tratteggiata). Le due aree vengono poi calcolate separatamente dalla funzione "trapz". Anche questo processo è facilmente automatizzabile, fornendo solo la posizione del primo picco. Funziona perfettamente solo se il picco più grande è simmetrico e se la separazione dei picchi è tale che la coda sinistra del picco più piccolo non incrementi significativamente l'altezza del primo picco.

Se il *profilo* dei picchi è noto, un buon modo per misurare le aree dei picchi sovrapposti consiste nell'usare l'*approssimazione dei minimi quadrati della curva*, come discusso a partire da pagina 165. Se le posizioni, le larghezze e le altezze sono sconosciute, e sono note solo i profili fondamentali dei picchi, allora si può impiegare il [metodo iterativo ai quadrati minimi](#). In alcuni casi, si può includere anche il background nel 'curve fitting'.

Per la gascromatografia e la spettrometria di massa in particolare, [OpenChrom di Philip Wenig](#) è un sistema di dati [open-source](#) che può direttamente importare file binari e testuali di dati cromatografici.

Comprende metodi per rilevare le linee di base e misurare le aree dei picchi in un cromatogramma. La documentazione



è ampia. È disponibile per Windows, Linux, Solaris e Mac OS X. A lato viene mostrata una schermata (cliccare per ingrandire). L'autore aggiorna regolarmente il programma e la documentazione. Un altro programma open-source disponibile gratuitamente per la spettroscopia di massa è "[Skyline](#)" del [MacCoss Lab Software](#), che è specificamente mirato al monitoraggio delle reazioni. Sono disponibili tutorial e video. Esistono anche software commerciali, come [Chrom&Spec software di Ampersand](#) e [LabSolutions di Shimadzu](#), che eseguono sofisticate analisi fattoriali, deconvoluzione dei picchi e così via.

Misura dell'area di un picco con gli spreadsheet.

[EffectOfDx.xlsx](#) ([schermata](#)) mostra che la semplice equazione $\text{sum}(y) \cdot dx$ misura accuratamente l'area di un picco Gaussiano isolato se ci sono almeno 4 o 5 punti visibili al di sopra della linea di base e includendo i punti esterni in più o meno ad almeno 2 o 3 deviazioni standard della Gaussiana. Mostra anche che una Gaussiana ampliata esponenzialmente deve includere più punti sul lato finale (lato destro, in questo caso) per ottenere la precisione migliore. [EffectOfNoiseAndBaseline.xlsx](#) ([schermata](#)) illustra l'effetto del rumore casuale e della linea di base, mostrando che l'area è più sensibile a una linea di base non nulla rispetto alla stessa quantità di rumore casuale.

[CumulativeSum.xls](#) ([schermata](#)) è un esempio di integrazione di un segnale di tipo picco mediante la somma cumulativa normalizzata; si possono incollare i propri dati nelle colonne A e B.

[CumulativeSumExample.xls](#) è un esempio con i dati.

I fogli di calcolo **Excel** e **Calc** [PeakDetectionAndMeasurement](#) e [CurveFitter](#) possono misurare le aree sotto i picchi Gaussiani di serie temporali, utilizzando [l'algoritmo findpeaks](#) e le tecniche del [curve fitting iterative non-lineari](#), rispettivamente. Ma nessuno dei due è versatile quanto uno specifico programma cromatografico come [OpenChrom](#).

Sharpening per la misura dell'area di picchi sovrapposti.

È stata creata una serie di spreadsheet template per misurare l'area, col taglio verticale, di picchi sovrapposti, utilizzando lo sharpening [anche lo sharpening derivativo](#). C'è un template vuoto a disposizione in cui poter applicare il Copia/Incolla dei propri dati ([PeakSharpeningAreaMeasurementTemplate.xlsxm](#)), una versione di esempio con dei campioni già inseriti e predisposti ([PeakSharpeningArea-MeasurementExample.xlsxm](#)) e un "demo" che crea e misura *dati simulati con aree note* ([PeakSharpeningAreaMeasurementDemo.xlsxm](#)) in modo da poter vedere come lo sharpening influenza sulla precisione della misura. Ci sono istruzioni molto brevi nella riga 2 di ciascuna di questi. Inoltre, ci sono *tooltip* su molte delle celle (indicate da un segno rosso nell'angolo in alto a destra della cella). Tutti e tre hanno pulsanti ActiveX cliccabili per una comoda regolazione interattiva dei fattori K2 e K4 dell'1% o del 10% per ogni click. Naturalmente, il problema è sapere quali valori usare per i fattori di ponderazione delle derivate 2^a e 4^a (K1 e K2). Tali valori dipendono dalla separazione, dalla larghezza e dall'altezza relativa dei due picchi, e si devono determinare sperimentalmente in base ad un proprio compromesso tra l'entità dello sharpening e il grado di distorsione della linea di base. Un buon punto di partenza per i picchi Gaussiani è $(\sigma^2)/30$ per il fattore della derivata 2^a e $(\sigma^4)/200$ per quello della derivata 4^a, dove σ è la deviazione standard della Gaussiana, poi si regolano per ottenere i picchi più stretti senza avere valori negativi. Non dare per scontato che aumentando le K fino al raggiungimento della risoluzione della linea di base si otterrà sempre la precisione migliore dell'area. I valori ottimali dipendono dal rapporto delle altezze dei picchi: a 1:1, a parità di larghezze e forme, il metodo del taglio verticale (pagina 135) funziona perfettamente senza sharpening, ma se c'è disuguaglianza tra le forme, le altezze o larghezze, i valori K danno errori inferiori ma esagerare con lo sharpening può sacrificare la precisione. Le due schermate [screen1](#) e [screen2](#), che usano gli stessi valori di K, mostrano che è possibile trovare valori K che diano un'eccellente precisione per il picco 2 su un certo intervallo di altezze relative, anche quando il picco più piccolo è molto basso.

Senza lo sharpening, le misure col taglio verticale sono impossibili perché non ci sono avvallamenti tra i picchi.

Il template [PeakSymmetrizationTemplate.xlsxm \(grafico\)](#) esegue la simmetrizzazione dei picchi espansi esponenzialmente mediante l'aggiunta ponderata della derivata prima. Vedere pagina 77. [PeakSymmetrizationExample.xlsxm](#) è un'applicazione con dati di esempio già inseriti. La procedura qui si applica prima della regolazione di k1 per ottenere picchi più simmetrici (valutando le pendenze sui lati opposti), poi si immette il tempo di inizio, quello dell'avvallamento e quello finale del grafico per la coppia di picchi che si desidera misurare nelle celle B4, B5 e B6, infine (opzionalmente) si regola il fattore di sharpening k2 della derivata seconda. Le aree di questi due picchi con i tagli verticali vengono riportate nelle colonne F e G. Questi spreadsheet hanno pulsanti Active-X per regolare il fattore di ponderazione della derivata prima (k1) nella cella J4 e quello per la derivata seconda k2 (cella J5). Esiste anche una versione demo che consente di determinare l'accuratezza delle aree dei picchi col taglio verticale in condizioni diverse generando internamente picchi sovrapposti di aree note, specificando l'asimmetria (B6), l'altezza relativa del picco (B3), la larghezza (B4) e il rumore (B5): [PeakSymmetrizationDemo.xlsxm \(grafico\)](#).

Misura dell'area di un picco con Matlab e Octave

Matlab e **Octave** hanno comandi nativi per la somma di elementi (“sum” e “cumsum” per la somma cumulativa) e per l'integrazione numerica trapezoidale (“trapz”). Per esempio, si considerino questi tre comandi Matlab.

```
>> x=-5:.1:5;
>> y=exp(-(x).^2);
>> trapz(x,y)
```

Queste righe calcolano accuratamente il valore numerico dell'area sottostante la curva di x,y, in questo caso una Gaussiana isolata, la cui area può essere mostrata come la [radice quadrata di pi](#), che è pari a 1.7725:

$$\begin{aligned} \int_{-\infty}^{\infty} e^{-x^2} dx &= \left[\int_{-\infty}^{\infty} e^{-x^2} dx \int_{-\infty}^{\infty} e^{-y^2} dy \right]^{1/2} \\ &= \left[\int_0^{2\pi} \int_0^{\infty} e^{-r^2} r dr d\theta \right]^{1/2} \\ &= \left[\pi \int_0^{\infty} e^{-u} du \right]^{1/2} \\ &= \sqrt{\pi} \end{aligned}$$

Se l'intervallo tra i valori di x, dx, è *costante*, allora l'area è semplicemente `y_i = sum(y) .* dx`. In alternativa, il segnale può essere integrato usando `y_i = cumsum(y) .* dx`, quindi l'area del picco sarà uguale [all'altezza del gradino risultante](#), `max(y_i) - min(y_i) = 1.7725`.

L'area di un picco è proporzionale al prodotto tra la sua altezza e la sua larghezza, ma la costante di proporzionalità dipende dalla forma del picco. Un picco Gaussiano puro con un'altezza *h* e [larghezza a metà altezza](#) *w* ha un'area totale di $1.064467 * h * w$. Un picco Lorentziano ha un'area totale di $(\pi/2) * h * w$. Un mix di Gaussiana-Lorentziana con la percentuale del carattere Gaussiano, *p*, ha un'area di $((100-p)/100) * ((\pi/2) * w * h) + (p/100) * (1.064467 * w * h)$. Il grafico [LorentzianVsGaussian.png](#) confronta picchi Gaussiani e Lorentziani con le stesse altezze e larghezze. Il Lorentziano ha più area nelle ali esterne, si deve misurare su un intervallo più ampio su entrambi i lati del picco. Per ottenere un'area entro l'1%, la si deve espandere fino a 64 volte la FWHM! (Vedere [LorentzianAreaProblem.m](#), [grafico](#)). Alcuni segnali reali in pratica hanno difficili

profili del picco (mix di Lorentziani o Gaussiani/Lorentziani) che sono troppo ravvicinati per consentire la misura delle aree teoriche direttamente mediante integrazione. La possibile soluzione include lo sharpening (pag. 138), il curve fitting iterativo (pag. 146), o anche la misura dell'altezza e della larghezza e poi il calcolo analitico dell'area utilizzando le precedenti espressioni analitiche.

I picchi nei segnali reali hanno alcune altre complicazioni: (a) Le forme dei picchi potrebbero non essere note; (b) possono essere sovrapposti su una linea di base variabile; (c) possono essere sovrapposti ad altri picchi; (d) c'è sempre del rumore casuale. Si può utilizzare la simulazione del segnale per testare l'influenza di tali complicazioni. Ad esempio, lo script Matlab/Octave [AreasOfIsolatedPeaks.m](#) crea un segnale simulato multi-picco e poi testa l'accuratezza della misura dell'area del picco di quel segnale con una larghezza della finestra di integrazione e un intervallo di correzione della linea di base specificati. (In tal caso, è il rumore il colpevole).

Codice per computer per il metodo del taglio verticale. Il seguente codice Matlab/Octave misura le aree di due picchi simmetrici sovrapposti nei vettori x, col metodo del taglio verticale. Le variabili m1 e m2 sono le stime delle posizioni sull'asse x dei due picchi, che vengono tipicamente determinate con qualche algoritmo di ricerca basato sulla derivata prima. La funzione "[val2ind](#)" restituisce l'indice della posizione del valore specificato in un vettore. La terza riga trova il punto a metà strada tra i due picchi. Le ultime due righe utilizzano la funzione "trapz" di Matlab per misurare le aree prima e dopo il punto di avvallamento.

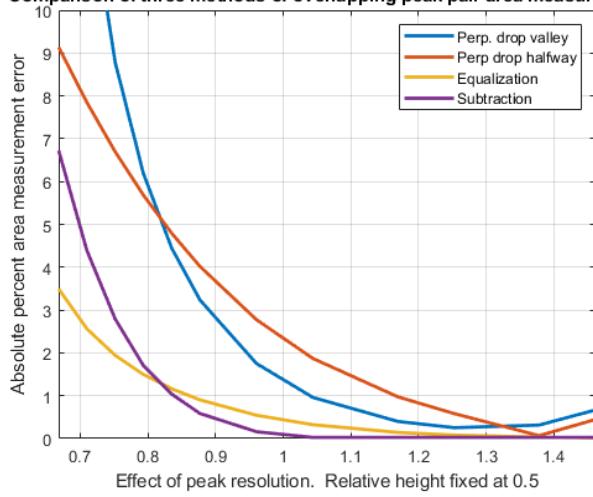
```
index1=val2ind(x,m1);
index2=val2ind(x,m2);
valleyindex=val2ind(x,(m1+m2)/2),
PDMesArea1=trapz(x(1:valleyindex),y(1:valleyindex));
PDMesArea2=trapz(x(valleyindex:length(x)),y(valleyindex:length(x));
```

In alternativa, si può sostituire "valleyindex" con `valleyy=min(y(index1:index2))`; `valleyindex=val2ind(y,valleyy)`; che usa il *minimo* tra i picchi anziché il punto medio. Ma il *metodo del punto medio ha il vantaggio* che l'SNR al massimo del picco è generalmente migliore che al minimo, e inoltre funziona anche quando non c'è un minimo distinguibile tra i picchi. La funzione [PerpDropAreas.m](#) usa il metodo del punto a metà strada per misurare le aree di qualsiasi numero di picchi sovrapposti, dato un elenco delle posizioni dei massimi dei picchi. Questi metodi funzionano bene se le *larghezze* dei picchi sono uguali o quasi. Tra i metodi alternativi c'è [EqualPerpDrop.m](#), che esegue la misura dell'area col metodo della "equalizzazione" e [EqualPerpDropTest.m](#), che mostra l'utilizzo della funzione applicata alla misura di due picchi EMG simulati e sovrapposti (Exponentially Modified Gaussian). Il codice Matlab/Octave per tutti questi metodi è contenuto nello script "[OverlapAreaComparison.m](#)". Nel caso del picco Gaussiano con una risoluzione di 0.7 e un rapporto di altezza da 1 a 0.5, l'errore percentuale relativo delle aree del picco è:

```
Peak 1 Peak 2
Perpendicular drop, valley point: -6.44% 12.89%
Perpendicular drop, half-way point: 3.91% -7.83%
Equalization method: 1.27% -2.54%
Subtraction method: -2.12% 4.25%
```

I parametri si possono cambiare dalla riga 5 alla 10 per eseguire il test con altre separazioni e altezze relative dei picchi. Il metodo dell'equalizzazione è spesso, ma non sempre, quello più accurato. (Nota: lo script richiede che le funzioni [val2ind.m](#), [halfwidth.m](#), [ExpBroaden.m](#) e [plotit.m](#) stiano nel path).

Comparison of three methods of overlapping peak pair area measurement



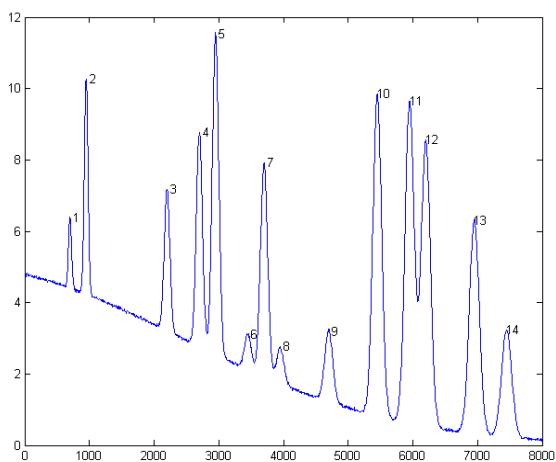
Un'analisi più approfondita di questi metodi mostra l'effetto della modifica della risoluzione del picco, mostrata a lato ([script](#), [grafico](#)) e della modifica dell'altezza del picco più piccolo, mostrato in basso a destra ([script](#), [grafico](#)). Questi script comprendono l'effetto del *rumore casuale* nel segnale, poiché il rumore può influenzare la posizione dei massimi e il punto di separazione tra i picchi, indipendentemente dal fatto che siano determinati manualmente o da un algoritmo computerizzato (come in questo caso); il rumore casuale è impostato dalla variabile "noise", che è il rumore bianco casuale frazionario aggiunto al segnale.

Inoltre, questi script includono l'effetto dell'*asimmetria* nel profilo dei picchi, che può causare errori nella misura dell'area con tutti questi metodi. Dopo tutto, la vera ragione per misurare l'area piuttosto che le altezze dei picchi è quella di [ridurre l'effetto di variazioni incontrollate nell'ampliamento dei picchi](#). L'asimmetria è fissata dalla variabile "TimeConstant", che è la costante di tempo della convoluzione esponenziale applicata al segnale che riduce l'altezza e allunga la metà destra. Entrambi sono a zero nelle figure precedenti per semplicità e per mostrare la migliore accuratezza possibile. Ad esempio, con una risoluzione di 1.0, un tau di 2 e il rumore impostato su 0.01 (1%), il metodo del taglio verticale e quello dell'equalizzazione superano gli altri metodi ([grafico](#)). Le cose sono molto più facili e più tolleranti nell'analisi *quantitativa* utilizzando una [curva di calibrazione](#), perché in quel caso la precisione assoluta dell'area non è realmente necessaria. È importante, piuttosto, la *riproducibilità* delle aree. Gli errori *sistematici* nella misura dell'area modificano semplicemente la *pendenza* e la curva di calibrazione, e se le condizioni sono le stesse, tra calibrazione e analisi (sempre un requisito in ogni caso), l'errore verrà completamente annullato. Per esempio, se si eseguono gli script precedenti con picchi molto asimmetrici (TimeConstant=3), una pessima risoluzione (resolution =0.68) e una quantità visibile di rumore (noise=5%), gli errori *sistematici* nella misura dell'area [sono abbastanza grandi](#) (5%-15%), ma ciononostante vengono prodotte delle buone curve di calibrazione lineari sia col [taglio verticale nel punto medio](#) che col [metodo dell'equalizzazione](#), in un range di altezze relative da 0.1 a 0.99, con coefficienti di correlazione di 0.999. La curva di calibrazione compensa l'errore sistematico e la misura dell'area integra più punti sul picco.

Tutti questi metodi possono produrre errori significativi se i picchi sono molto sovrapposti o molto asimmetrici. Tuttavia, l'asimmetria, che è il risultato di una *dilatazione esponenziale*, si può ridurre *prima* di calcolare le aree, col [metodo dell'addizione della derivata prima](#), che restringe i picchi e ne elimina l'asimmetria *senza modificarne le aree*. Altri metodi di sharpening, in particolare l'auto-deconvoluzione (pagina 108) si possono usare anche quando il picco da misurare è troppo debole o troppo poco risolto per consentirne una facile misura. In definitiva, nei casi più difficili, potrebbe essere necessario considerare l'uso del [curve fitting iterativo](#), sebbene sia certamente più complesso dal punto di vista matematico e soggetto ai suoi limiti.

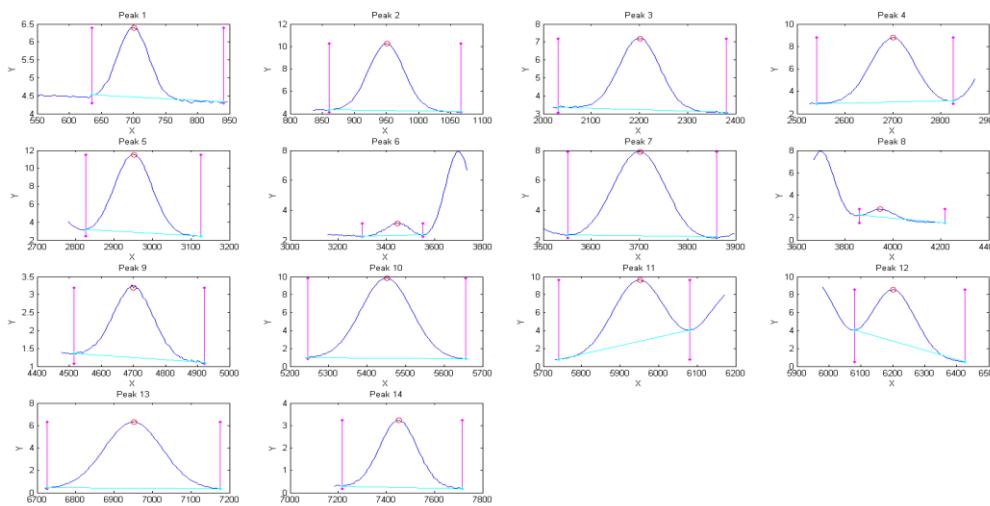
Rilevamento automatico di più picchi

Measurepeaks.m (La sintassi è **M=measurepeaks (x,y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth, plots)**) è una funzione che *velocemente ed automaticamente* rileva



i picchi in un segnale, utilizzando il metodo di passaggio per lo zero della derivata [descritto in precedenza](#), e ne misura le aree utilizzando il metodo del taglio verticale e quello tangente. Condivide i primi 6 argomenti di input con [findpeaksG](#). Restituisce una [tabella](#) contenente: il numero del picco, la posizione, l'altezza assoluta, la differenza tra picco e valle, l'area col taglio verticale e quello col taglio tangente per ciascun picco rilevato. Se l'ultimo argomento di input ("plots") è impostato a 1, [disegna il segnale](#) con picchi numerati (mostrati a sinistra) e inoltre [disegna anche i singoli picchi](#) (in blu) con il massimo (cerchi rossi), i punti di avvallamento (magenta) e le linee tangenti (ciano) contrassegnati come mostrato a destra. Le altezze e le posizioni x sono indicate con cerchi rossi, l'area col taglio verticale è l'area totale misurata tra le due linee verticali magenta fino allo zero e l'area col taglio tangente è quella tra la linea di base in ciano e il picco blu (che compensa una linea di base locale rettilinea). Digitare "[help measurepeaks](#)" e provare i sette esempi che ci sono, o eseguire

[HeightAndArea.m](#) per testare l'[accuratezza della misura dell'altezza e dell'area del picco](#) con segnali che abbiano più picchi con rumore, background e qualche sovrapposizione. In generale, i valori per l'altezza assoluta del picco e l'area col taglio verticale sono migliori per i picchi che non hanno un background, anche se sono leggermente sovrapposti, mentre i valori per la differenza picco-valle e per l'area col taglio tangente sono migliori per i picchi isolati su un background leggermente curvo. Nota: questa funzione usa lo [smoothing](#) (specificato dall'argomento di input SmoothWidth) solo per il *rilevamento* dei picchi; le misure vengono eseguite sui dati y *originali*. Se i dati originali sono rumorosi, la posizione degli avvallamenti potrebbe essere incerta, nel qual caso potrebbe essere utile eseguire a parte lo smoothing dei dati y prima di chiamare measurepeaks.m, utilizzando una qualsiasi funzione. (Lo smoothing non modifica l'area di un picco isolato).



[M,A]=autopeaks.m è fondamentalmente una combinazione di [autofindpeaks.m](#) e di [measurepeaks.m](#). Ha una sintassi simile a quella di [measurepeaks.m](#), tranne per il fatto che i parametri di rilevamento del picco (SlopeThreshold, AmpThreshold,

smoothwidth, peakgroup e smoothtype) possono essere omessi e la funzione calcolerà i valori di prova come fa [autofindpeaks.m](#). L'uso della semplice sintassi [M,A]=autopeaks(x, y) funziona bene in alcuni casi, altrimenti si provi [M,A]=autopeaks(x, y, n), usando diversi valori per n (all'incirca il numero di picchi che approssimerebbe il segnale registrato) fin quando non vengono rilevati i picchi da misurare. Proprio come [measurepeaks.m](#), restituisce una [tabella](#) M contenente il numero del picco, la posizione, l'altezza assoluta, la differenza picco-valle, l'area col taglio verticale e quella col taglio tangente per ogni picco rilevato, ma può anche, facoltativamente, restituire un vettore A

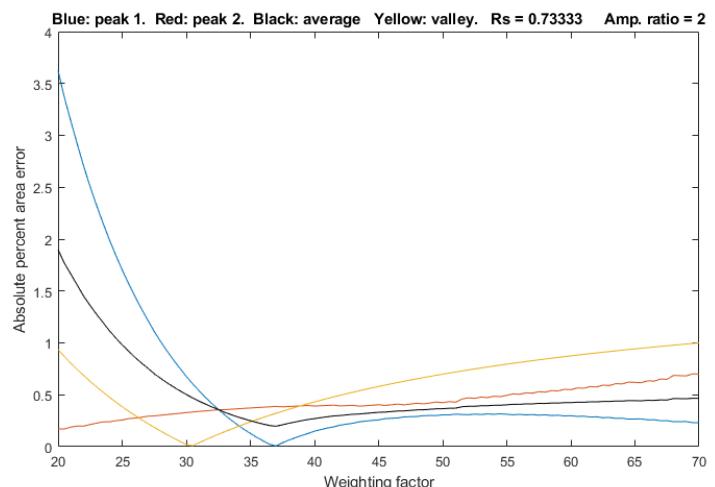
contenente i parametri di rilevamento calcolati (per usarli in altre funzioni di rilevamento ed approssimazione). Per un controllo più preciso sul rilevamento del picco, è possibile specificare tutti i parametri di rilevamento digitando: M=autopeaks (x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup). La funzione [autopeaksplot.m](#) è simile, ma disegna anche il segnale e i singoli picchi come fa measurepeaks.m (mostrato sopra). Lo script [testautopeaks.m](#) esegue tutti gli esempi nel file di help di autopeaks, con una pausa di 1 secondo tra ciascuno, stampando i risultati nella finestra di comando, disegnando e numerando i picchi (Window 1) e ciascun singolo picco (Window 2); richiede [gaussian.m](#) e [fastsmooth.m](#) nel path di Matlab. Autopeaks.m e autopeaksplot.m restituiscono una matrice **M**, che elenca ogni picco rilevato nelle righe, nello colonne ha le seguenti misure:

Peak	Position	PeakMax	Peak-valley	Perp drop	Tan skim
1	6.0000	1.3112	1.2987	1.7541	1.7121
2	. . . ecc.				

Per determinare l'effetto dello smoothing, dello sharpening, della deconvoluzione o di altri metodi di miglioramento del segnale sulle aree dei picchi sovrapposti misurate col metodo del taglio verticale, la funzione Matlab/Octave [ComparePDAreas.m](#) usa [autopeaks.m](#) per misurare le aree dei picchi dei segnali originali e processati, "orig" e "processed", e mostra un grafico a dispersione [scatter] delle aree dei dati originali rispetto a quelle dei dati processati per ciascun picco e restituisce le tabelle dei picchi, P1 e P2 rispettivamente, la pendenza, l'intercetta e i valori di R^2 , che idealmente dovrebbero essere 1, 0, e 1, se l'elaborazione non ha avuto alcun effetto sull'area del picco.

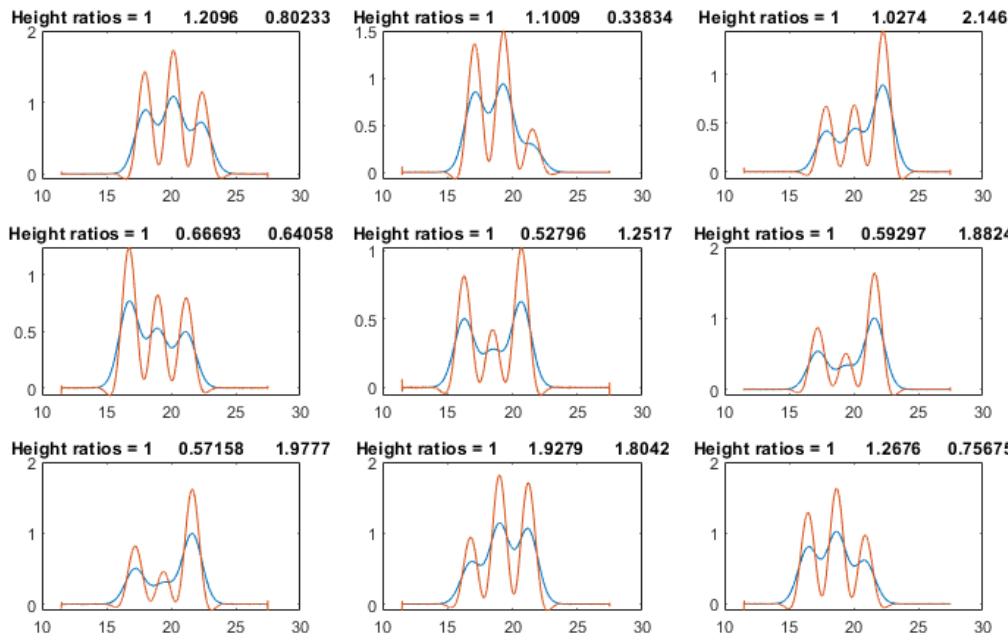
Le funzioni correlate [wmeasurepeaks.m](#) e [testwmeasurepeaks.m](#) utilizzano il *denoising wavelet* (pag. 130) anziché lo smoothing, ma questo fa poca differenza, perché le misure dei parametri dei picchi si basano sull'approssimazione ai quadrati minimi dei *dati originali*, non di quelli con *smoothing*, quindi il solito vantaggio del denoising wavelet di evitare la distorsione dello smoothing non si applica qui.

La funzione Matlab/Octave per la ricerca automatica dei picchi [findpeaksG.m](#) calcola l'area del picco assumendo che il suo profilo sia Gaussiano (o Lorentziano, per la variante [findpeaksL.m](#)). La funzione correlata [findpeaksT.m](#) usa il *metodo di costruzione del triangolo* per calcolare i parametri del picco. Anche per i picchi Gaussiani ben separati, le misure delle aree col metodo di costruzione del triangolo non è molto accurata; i risultati sono circa il 3% al di sotto dei valori corretti. (Tuttavia, questo metodo funziona meglio di [findpeaksG.m](#) quando i picchi sono notevolmente asimmetrici; vedere [triangulationdemo](#) per degli [esempi](#)). Al contrario, [measurepeaks.m](#) non fa supposizioni sulla forma del picco.



Lo sharpening del picco (pagina 74) può spesso aiutare nella misura delle aree dei picchi sovrapposti, creando (o approfondendo) le valli tra i picchi che sono necessarie per il metodo del taglio verticale. [SharpenedOverlapDemo.m](#) è uno script che determina automaticamente il grado ottimale di sharpening con le derivate pari che minimizzano gli errori di misurazione delle aree dei picchi di [due Gaussiane sovrapposte](#) col metodo del taglio verticale utilizzando la funzione

[autopeaks.m](#). Lo fa applicando diversi gradi di sharpening e disegnando gli errori delle aree



(differenza percentuale tra gli errori veri e quelli misurati) rispetto al fattore di ponderazione dello sharpening, come mostrato in figura. Mostra anche l'altezza della valle tra i picchi (linea gialla). Questo dimostra che:

- (1) il fattore ottimale di sharpening dipende

dalla larghezza e dalla separazione dei due picchi e dal rapporto sulle loro altezze,

- (2) il grado di sharpening non è eccessivamente critico, spesso esibendo un'ampia regione ottimale,
- (3) l'ottimo per i due picchi non è necessariamente lo stesso, e
- (4) l'ottimo per la misura dell'area solitamente non si ha nel punto in cui l'avvallamento è zero. (Per eseguire questo script si devono avere [gaussian.m](#), [derivxy.m](#), [autopeaks.m](#), [val2ind.m](#) e [halfwidth.m](#) nel path di ricerca di Matlab. Si scaricano da <https://terpconnect.umd.edu/~toh/spectrum/>).

Lo script [SharpenedOverlapCalibrationCurve.m](#) simula la costruzione e l'uso di curve di calibrazione di tre picchi gaussiani sovrapposti (le linee blu nei grafici dei segnali nella pagina successiva). Lo sharpening delle derivate pari (la linea rossa nei grafici del segnale) viene utilizzata per migliorare la risoluzione dei picchi per consentire la misura dell'area col taglio verticale. Una linea retta viene approssimata alla curva di calibrazione e viene calcolato l' R^2 , per dimostrare (1) la linearità della risposta e (2) l'indipendenza dai picchi adiacenti sovrapposti. Si possono modificare i seguenti parametri:

1. La risoluzione, Rs, modificando la larghezza del picco w nella riga 15. Inizialmente $w=2$, che rende $Rs=0.55$.
2. I rapporti delle altezze dei picchi, modificando il picco minimo e quello massimo nelle righe 21 e 22. (Il default è 0.2 e 1.0, un intervallo di rapporto 1:5). Naturalmente, se il picco 2 è *trop*po piccolo, non ci sarà un avvallamento tra i picchi.
3. Il numero di standard nelle curve di calibrazione, nella riga 24. Numeri più grandi danno risultati migliori.
4. Il numero di campioni simulati, nella riga 25. Numeri più grandi danno errori medi più affidabili.

[SymmetrizedOverlapCalibrationCurve.m](#) fa la stessa cosa per la simmetrizzazione di picchi Gaussiani sovrapposti espansi esponenzialmente tramite l'aggiunta della derivata prima. La variabile critica è "factor" nella riga 27, che, per ottenere i risultati migliori, dovrebbe corrispondere a, o leggermente superare, "tau", la costante di tempo esponenziale nella riga 19. Si devono avere

[gaussian.m](#), [derivxy.m](#), [autopeaks.m](#), [val2ind.m](#), [halfwidth.m](#), [fastsmooth.m](#) e [plotit.m](#) nel path di ricerca di Matlab.

iSignal (pag. 366) è una funzione Matlab interattiva scaricabile che esegue varie funzioni di signal processing descritte in questo tutorial, tra cui la misura dell'area del picco utilizzando la Regola di Simpson e col taglio verticale. Cliccare per visualizzare o **click destro > Save link as... qui**, o scaricare il [file ZIP](#) con i dati di esempio per un test. È mostrato di seguito applicando il metodo del taglio verticale ad una serie di quattro picchi di uguale area. In fondo al pannello si vede come gli intervalli delle misure, contrassegnati dalle linee verticali tratteggiate magenta, sono posizionate nei *minimi* degli avallamenti su entrambi i lati di ciascuno dei quattro picchi. Si può vedere quest'animazione visualizzandola in [Microsoft Word 365](#), altrimenti cliccare su [questo link](#).

Le seguenti righe di codice Matlab/Octave creano quattro picchi gaussiani sintetizzati al computer che *hanno tutti la stessa altezza* (1.000), *larghezza* (1.665) e *area* (1.772) ma con *diversi gradi di sovrapposizione dei picchi*, come nella figura a lato.

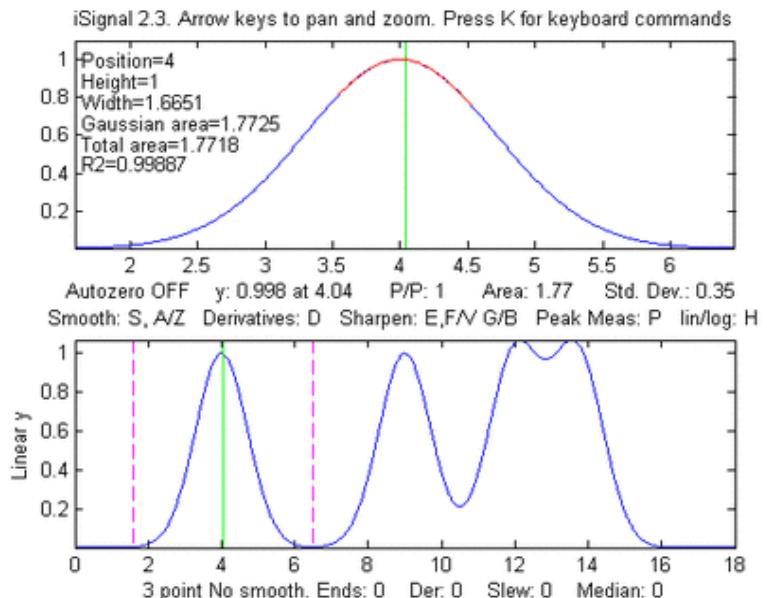
```
x=[0:.01:18];
y=exp(-(x-4).^2) + exp(-(x-9).^2) + exp(-(x-12).^2) + exp(-(x-13.7).^2);
isignal(x,y);
```

Usare **iSignal** per misurare le aree di ciascuno di questi picchi col metodo del taglio verticale, si usano i tasti pan e zoom per posizionare le due linee esterne del cursore (linee punteggiate in magenta) nell'avallamento da entrambi i lati del picco. Il totale di ciascuna area verrà mostrato sotto la finestra superiore.

	Peak #	Position	Height	Width	Area
1	4.00	1.00	1.661	1.7725	
2	9.001	1.0003	1.6673	1.77	
3	12.16	1.068	2.3	1.78	
4	13.55	1.0685	2.21	1.79	

I risultati dell'area sono ragionevolmente accurati in questo esempio solo perché il metodo del taglio verticale compensa approssimativamente la sovrapposizione parziale tra i picchi, ma solo se i picchi sono simmetrici, all'incirca uguali in altezza e senza background.

iSignal include un comando aggiuntivo (tasto **J**) che chiama la funzione [autopeaksplot](#), per rilevare automaticamente i picchi nel segnale e misurarne la loro posizione, l'altezza assoluta, la differenza picco-valle, l'area col taglio verticale e quello tangente. Richiede l'immissione della densità dei picchi (all'incirca il numero di picchi che approssimerebbero il segnale); maggiore è questo numero, più è sensibile ai picchi più stretti. Visualizza i picchi misurati proprio come fa la funzione di [measurepeaks](#) descritta. (Per tornare a iSignal, premere un qualsiasi tasto freccia).



Misura dell'area col curve fitting iterativo

In generale, La misura più flessibile dell'area di picchi sovrapposti, assumendo che il *profilo* base dei picchi sia noto o deducibile, viene fatta [coll'approssimazione iterativa ai quadrati minimi](#), per esempio utilizzando [peakfit.m](#), mostrato di seguito (per Matlab e Octave). Questa funzione può approssimare qualsiasi numero di picchi sovrapposti con i profili selezionati da un elenco. Usa la funzione "trapz" per calcolare l'area di ciascuno dei modelli componenti i picchi. Per esempio, utilizzando la funzione **peakfit** sugli stessi dati usati in precedenza, i risultati sono molto più accurati:

```
>> peakfit([x;y],9,18,4,1,0,10,0,0,0)
```

Peak #	Position	Height	Width	Area
1	4	1	1.6651	1.7725
2	9	1	1.6651	1.7725
3	12	1	1.6651	1.7725
4	13.7	1	1.6651	1.7725

La funzione interattiva [iPeak](#) (pagina 245), può essere usata anche per stimare le aree dei picchi. Ha il vantaggio di poter rilevare e misurare *tutti i picchi in un segnale in un'unica operazione*. Il metodo predefinito per misurare l'area in iPeak è la stima Gaussiana, assumendo che i picchi sono Gaussiani e approssimano la vetta del picco. Per esempio (utilizzando gli stessi vettori x e y definiti nella pagina precedente):

```
>> ipeak([x,y],10)
Peak # Position Height Width Area
1 4 1 1.6651 1.7727
2 9.0005 1.0001 1.6674 1.7754
3 12.16 1.0684 2.2546 2.5644
4 13.54 1.0684 2.2521 2.5615
```

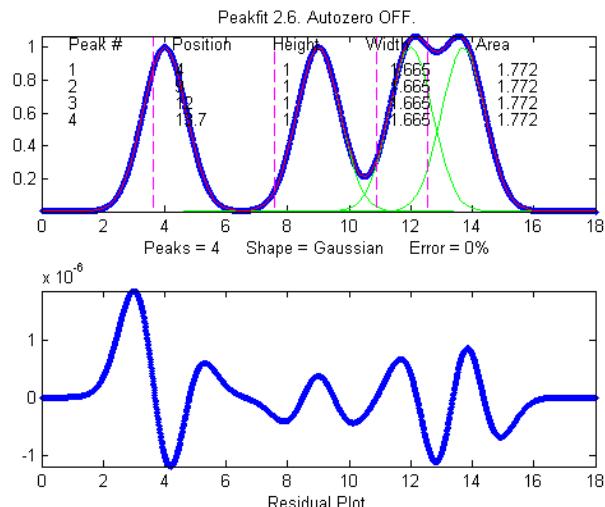
I picchi 1 e 2 vengono misurati accuratamente da *iPeak*, la le larghezze e le aree dei picchi 3 e 4 non sono precise perché i picchi si sovrappongono. Fortunatamente, *iPeak* ha una funzione interna, "peakfit" (attivata dal tasto N) che usa le stime delle posizioni e delle larghezze come prime ipotesi, col risultato di avere una buona accuratezza per tutti e quattro i picchi.

```
Fitting Error 0.0002165%
Peak# Position Height Width Area
1 4 1 1.6651 1.7724
2 9 1 1.6651 1.7725
3 12 1 1.6651 1.7725
4 13.7 0.99999 1.6651 1.7724
```

Quindi in questa situazione ideale artificiale, i risultati rispecchiano perfettamente le aspettative.

Correzione del background/linea di base

La presenza di una linea di base o di un segnale di fondo, cui i picchi si sovrappongono, influenzera notevolmente la misure dell'area del picco se non corretta o compensata. *iSignal*, *iPeak*, [measurepeaks](#) e **peakfit** hanno tutte diverse modalità di correzione della linea di base, per linee di se piatte, lineari inclinate e curve, in più, *iSignal* e *iPeak* hanno anche una funzione di sottrazione della



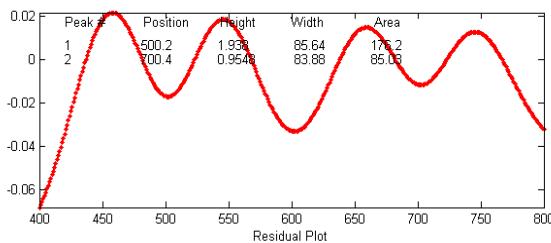
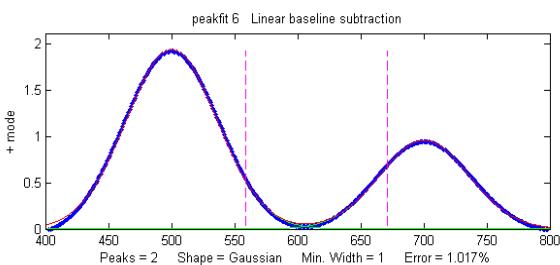
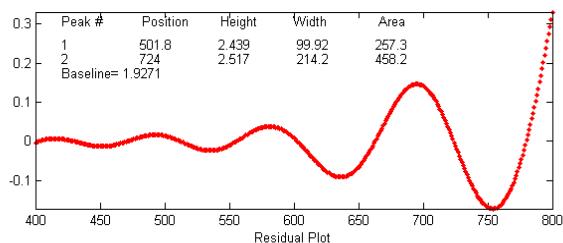
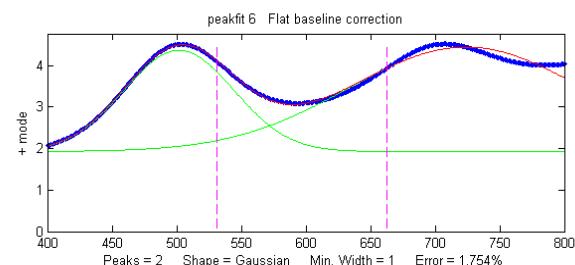
linea di base lineare multi-punto per parti che consente di stimare manualmente il background da sottrarre all'intero segnale. Se la linea di base è causata dai bordi di un forte picco adiacente sovrapposto, allora è possibile includere quel picco nell'operazione di curve-fitting, come si vede nell'Esempio 22 di pagina 397.

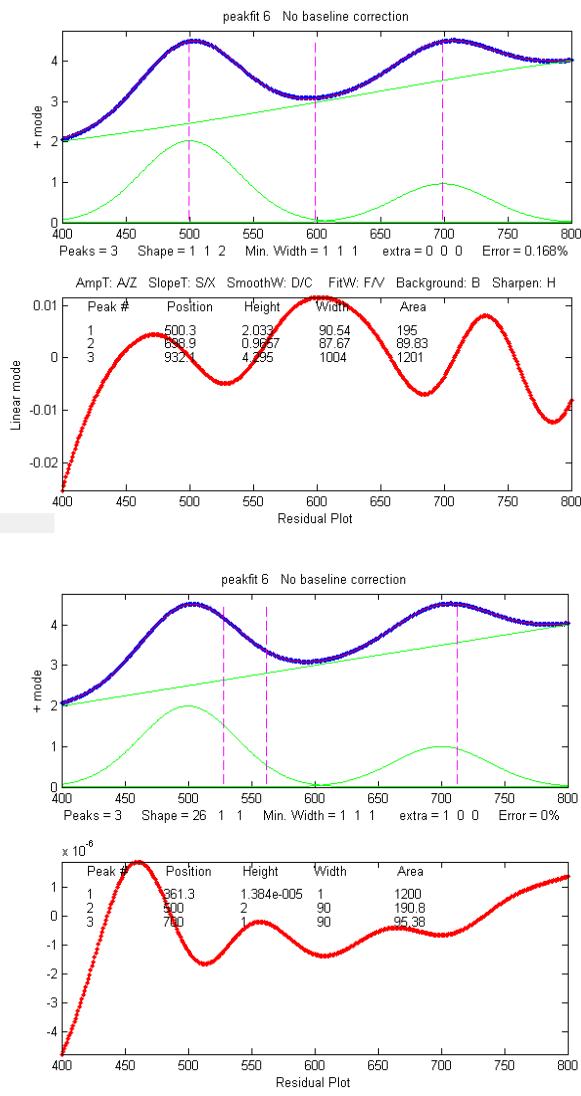
Lo script [AreasOfIsolatedPeaks2.m](#) mostra l'uso di peakfit.m per un segnale sperimentale simulato costituito da diversi picchi isolati su una linea di base inclinata diritta. In questo esempio, si presume che le posizioni dei picchi siano sufficientemente riproducibili da poter utilizzare un insieme predeterminato di segmenti di misurazione per approssimare ciascun picco separatamente e determinarne la posizione, l'altezza, la larghezza e l'area esatte.

La seguente riga di Matlab/Octave crea un segnale simulato costituito da due picchi Gaussiani privi di rumore e leggermente sovrapposti con altezze teoriche rispettivamente di 2,00 e 1,00 e aree di 191,63 e 95,81 unità. La linea di base è inclinata e lineare e leggermente maggiore in altezza della stessa altezza dei picchi, ma il problema più serio è che il segnale *non ritorna mai alla linea di base* abbastanza a lungo da facilitare la distinzione del segnale dalla linea di base.

```
>> x=400:1:800;y=2.*gaussian(x,500,90)+1.*gaussian(x,700,90)+2.*(.x./400);
```

Una semplice applicazione di iSignal, utilizzando la modalità di base 1 e il metodo det taglio verticale, sottostima seriamente entrambe le aree (168.6 e 81.78), perché la modalità 1, della linea di base, funziona solo quando il segnale torna completamente sulla linea di base locale alle estremità dell'intervallo approssimato, cosa che non capita qui.





Una misura automatica con il 'taglio tangente' di [measurepeaks](#) non è accurata in questo caso perché i picchi non scendono fino alla linea di base alle estremità del segnale e a causa della leggera sovrapposizione:

```
>> measurepeaks(x,y,.0001,.8,2,5,1)
Position PeakMax Peak-valley Perp drop Tan skim
1  503.67  4.5091  1.895   672.29 171.44
2  707.44  4.5184  0.8857  761.65 76.685
```

Un tentativo di utilizzare il curve fitting **peakfit.m** nella modalità di correzione 3 della linea di base (**peakfit([x;y],0,0,2,1,0,1,0,3)**, nella parte superiore più a sinistra della figura) fallisce perché la linea di base effettiva è inclinata, non piatta. La modalità lineare della linea di base **peakfit([x;y],0,0,2,1,0,1,0,1)**, in alto a destra della figura) non è molto migliore in questo caso (pagina 211). Un approccio più accurato consiste nell'impostare la modalità della linea di base a zero e nell'includere un *terzo* picco nel modello per approssimare la linea di base, per esempio o con un modello Lorentziano – **peakfit([x;y],0,0,3,[1 1 2])**, in basso a sinistra della figura - o con un modello "pendenza [slope]" - profilo 26 in peakfit versione 6, in basso a destra della figura. Quest'ultimo metodo fornisce sia l'errore di approssimazione più basso (meno dello 0.01%) sia le aree dei picchi più accurate (meno dello ½% di errore nell'area del picco):

```
>> [FitResults,FitError]=peakfit([x;y],0,0,3,[1 1 26])
Peak#      Position      Height      Width      Area
1  500  2.0001  90.005  190.77
```

```

2 700 0.99999 89.998 95.373
3 5740.2 8.7115e-007 1 1200.1

```

FitError =0.0085798

Si noti che in quest'ultimo caso il numero di picchi è 3 e l'argomento per il profilo è un vettore [1 1 26] che specifica due componenti gaussiane più il profilo 26 della "pendenza lineare". Se la linea di base sembra non essere lineare, si potrebbe preferire l'utilizzo di una quadrica (profilo 46; si veda l'esempio 38 a pagina 400). Se la linea di base sembra essere diversa su entrambi i lati del picco, si può tentare di modellarla con un profilo ad S (sigmoide), o un sigmoide in salita, profilo 10 ([click per il grafico](#)), **peakfit([x;y],0,0,2,[1 10],[0 0])**, o una sigmoide in discesa, profilo 23 ([click per il grafico](#)), **peakfit([x;y],0,0,2,[1 23],[0 0])**, in questi esempi lasciando il modello come una Gaussiana.

Picchi asimmetrici e ampliamento del picco: taglio verticale rispetto al curve fitting

[AsymmetricalAreaTest.m](#) è uno script Matlab/Octave che confronta l'accuratezza di diversi metodi di misura dell'area di un singolo picco rumoroso asimmetrico: (A) stima Gaussiana, (B) triangolazione, (C) taglio verticale, curve fitting (D) per Gaussiana espansa esponenzialmente e (E) due Gaussiane sovrapposte. [AsymmetricalAreaTest2.m](#) è simile tranne per il fatto che confronta la precisione (deviazione standard) delle aree. Per un picco singolo con una linea di base nulla, i metodi del taglio verticale e del curve fitting funzionano entrambi bene, entrambi notevolmente migliori della stima Gaussiana o della triangolazione. Il vantaggio dei metodi di curve fitting è che possono trattare in modo più accurato i picchi che si sovrappongono o che sono sovrapposti ad una linea di base.

Ecco un esperimento Matlab/Octave che crea un segnale contenente cinque picchi Gaussiani con le stesse altezze (1.0) e larghezze (3.0) iniziali ma con un successivo aumento *del grado di ampliamento esponenziale*, simile all'ampliamento dei picchi comunemente riscontrati in cromatografia:

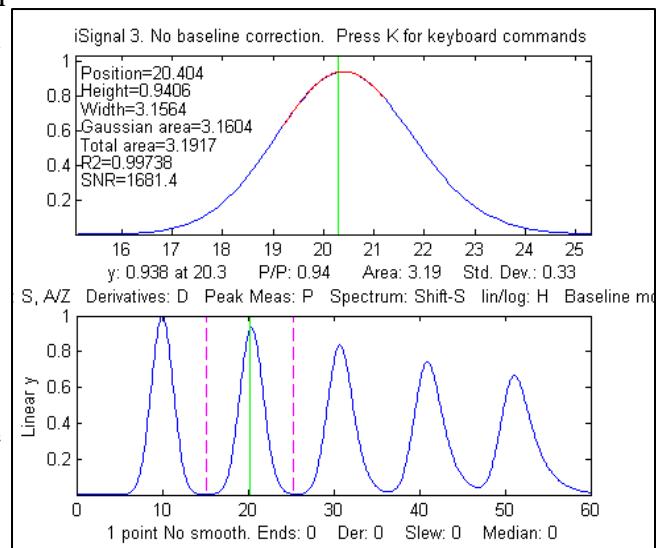
```

>> x=5:.1:65;
>> y=modelpeaks2(x, [1 5 5 5 5], [1 1 1 1 1], [10 20 30 40 50], [3 3 3 3
3], [0 -5 -10 -15 -20]);
>> isignal(x,y);

```

L'area teorica sotto queste Gaussiane è *sempre la stessa*: $1.0645 \times \text{Height} \times \text{Width} = 1 \times 3 \times 1.0645 = 3.1938$. Un perfetto algoritmo di misura dell'area restituirebbe questo numero per tutti e cinque i picchi.

All'aumentare dell'ampliamento da sinistra a destra, l'altezza del picco *diminuisce* (del 35% circa) e la larghezza *aumenta* (del 32% circa). Poiché l'area sotto il picco è proporzionale al *prodotto* dell'altezza e della larghezza del picco, queste due modifiche *approssimativamente si annullano vicendevolmente* e il risultato è che l'area è quasi indipendente dalla dilatazione (vedere il riepilogo dei risultati in [5ExponentialBroadenedGaussianFit.xlsx](#)). Il metodo del taglio verticale (pagina 140), **PerpDropAreas (x, y, min(x), max(x), positions)**, dove "positions" sono le posizioni sull'asse x dei



picchi originali, fornisce le aree [3.1933 3.1926 3.1738 3.1006 3.3045], un errore medio dell'1.4%, che non è del tutto perfetto.

La funzione Matlab/Octave per la ricerca dei picchi [findpeaksG.m](#), trova tutti e cinque i picchi e misura le loro aree assumendo una forma Gaussiana; questo funziona bene per il picco 1 non allargato ([script](#)), ma sottostima le aree all'aumentare dell'ampliamento nei picchi 2-5:

Peak	Position	Height	Width	Area
1	10.0000	1.0000	3.0000	3.1938
2	20.4112	0.9393	3.1819	3.1819
3	30.7471	0.8359	3.4910	3.1066
4	40.9924	0.7426	3.7786	2.9872
5	51.1759	0.6657	4.0791	2.8910

Il [metodo di costruzione del triangolo](#) (utilizzando [findpeaksT.m](#)) sottostima anche l'area del picco non diffuso 1 ed è meno preciso per i picchi allargati ([script](#); [grafico](#)):

Peak	Position	Height	Width	Area
1	10.0000	1.1615	2.6607	3.0905
2	20.3889	1.0958	2.8108	3.0802
3	30.6655	0.9676	3.1223	3.0210
4	40.8463	0.8530	3.4438	2.9376
5	50.9784	0.7563	3.8072	2.8795

La funzione automatizzata [measurepeaks.m](#) fornisce risultati migliori utilizzando il metodo del taglio verticale (5^a colonna della tabella).

```
>> M=measurepeaks(x,y,0.0011074,0.10041,3,3,1)
```

Peak	Position	PeakMax	Peak-val.	Perp drop	Tan skim
1	10	1	.99047	3.1871	3.1123
2	20	4	.94018	.92897	3.1839
3	30	709	.83756	.81805	3.1597
4	40	93	.74379	.70762	3.1188
5	50	095	.66748	.61043	3.0835

Utilizzando [iSignal](#) (pagina 366) e il metodo del taglio verticale manuale picco-picco si ottengono le aree 3.193, 3.194, 3.187, 3.178 e 3.231, una media di 3.1966 (vicine al valore teorico di 3.1938) ed una deviazione standard di solo 0.02 (0.63%). In alternativa, integrando il segnale, [cumsum\(y\).*dx](#), dove dx è la differenza tra valori adiacenti sull'asse x (0.1 in questo caso), e poi [misurando le altezze dei restanti passi](#), si ottengono risultati simili: 3.19, 3.19, 3.18, 3.17, 3.23. Con entrambi i metodi, le aree dei picchi non sono esattamente uguali come dovrebbero essere.

Ma si possono ottenere misure automatiche più accurate di tutti e cinque i picchi, utilizzando **peakfit.m** con profili multipli, una Gaussiana e quattro Gaussiane modificate esponenzialmente (profilo 5) con diversi fattori esponenziali (vettore extra):

```
>> [FitResults,FittingError]=peakfit([x;y],30,54,5,[1 5 5 5 5],  
[0 -5 -10 -15 -20],10, 0, 0)
```

Peak#	Position	Height	Width	Area
1	9.9933	0.98051	3.1181	3.2541
2	20.002	1.0316	2.8348	3.1128

```

3 29.985 0.95265 3.233   3.2784
4 40.022 0.9495   3.2186  3.2531
5 49.979 0.83202 3.8244  3.2974
FittingError = 2.184%

```

L'errore di approssimazione non è molto migliore della semplice approssimazione Gaussiana. Risultati migliori sono ottenibili utilizzando i risultati preliminari di posizione e larghezza ottenuti dalla [funzione findpeaks](#) o col curve fitting con una semplice approssimazione Gaussiana e utilizzando tali risultati come vettore di "start" (otto argomenti di input):

```

>> [FitResults,FittingError]=peakfit([x;y],30,54,5, [1 5 5 5 5], [0 -5 -
10 -15 -20], 10, [10 3.5 20 3.5 31 3.5 41 3.5 51 3.5], 0)

Peak# Position Height Width Area
1 9.9999 0.99995 3.0005 3.1936
2 20      0.99998 3.001   3.1944
3 30.001  1.0002  3.0006 3.1948
4 40      0.99982 2.9996 3.1924
5 49.999  1.0001  3.003   3.1243
FittingError = 0.02%

```

Risultati ancora più accurati per l'area si ottengono usando peakfit con una Gaussiana e quattro Gaussiane esponenzialmente modificate di *uguale larghezza* (profilo 8):

```

>> [FitResults,FittingError]=peakfit([x;y],30,54,5, [1 8 8 8 8], [0 -5 -
10 -15 -20], 10, [10 3.5 20 3.5 31 3.5 41 3.5 51 3.5], 0)

Peak# Position Height Width Area
1 10     1.0001   2.9995 3.1929
2 20     0.99998 3.0005 3.1939
3 30     0.99987 3.0008 3.1939
4 40     0.99987 2.9997 3.1926
5 50     1.0006   2.9978 3.1207
FittingError = 0.008%

```

Quest'ultimo approccio funziona perché, sebbene i picchi *allargati* abbiano chiaramente larghezze diverse (come mostrato nella semplice approssimazione Gaussiana), i picchi prima dell'allargamento hanno tutti la *stessa* larghezza. In generale, se ci si aspetta che i picchi debbano avere larghezze uguali o fisse, allora è meglio usare un [modello vincolato](#) che approssima tale conoscenza; si otterranno stime migliori delle proprietà ignote misurate, anche se l'errore di approssimazione sarà maggiore rispetto a un modello non vincolato.

Gli *svantaggi* del modello allargato esponenzialmente sono che:

- (a) potrebbe non corrispondere perfettamente all'effettivo processo di ampliamento fisico;
- (b) è più lento di una semplice approssimazione Gaussiana, e
- (c) a volte necessita di un aiuto, sotto forma di un vettore iniziale o dei vincoli di *uguale-larghezza*, come visto in precedenza, per ottenere i migliori risultati.

In alternativa, se l'obiettivo è misurare solo le *aree* dei picchi e non le posizioni e le larghezze, allora non è nemmeno necessario modellare l'ampliamento fisico per ciascun picco. Si può semplicemente mirare a una buona approssimazione utilizzando due (o più) semplici Gaussiane ravvicinate per ciascun picco e semplicemente *addizionando le aree* dell'approssimazione migliore. Per esempio, il 5° picco nell'esempio precedente (il più asimmetrico) si può ben approssimare con [due Gaussiane sovrapposte](#), ne risulta un'area totale di $1.9983 + 1.1948 = 3.1931$, molto vicina all'area teorica di 3.1938. Si possono anche utilizzare più Gaussiane se il profilo del picco è più complesso. Questa è detta "[regola della somma](#)" nel calcolo integrale: l'integrale della somma di due funzioni è uguale alla somma dei loro integrali. A titolo di dimostrazione, lo script

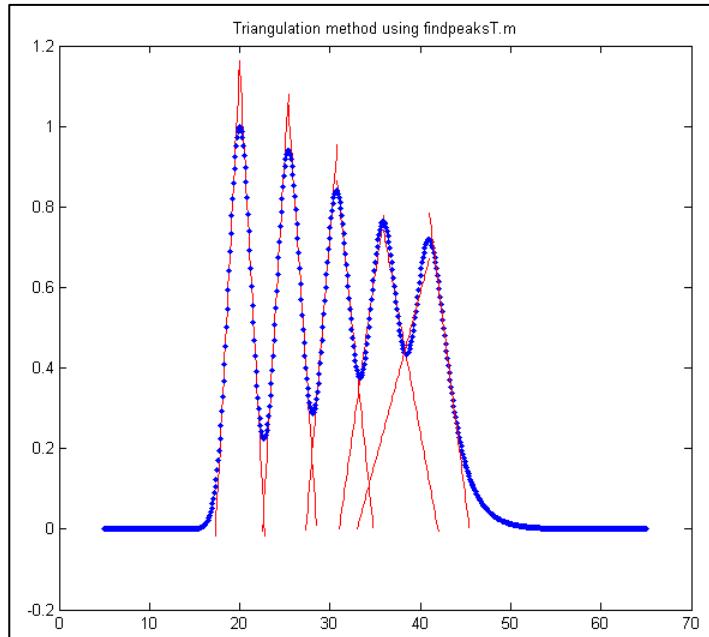
[SumOfAreas.m](#) mostra che anche picchi decisamente non Gaussiani si possono approssimare con più componenti Gaussiane e che l'area totale delle componenti si avvicina a quella del picco non-Gaussiano all'aumentare del numero delle componenti ([grafico](#)). Quando si utilizza questa tecnica, è meglio impostare il numero di tentativi (*NumTrials*, il 7° argomento di input della funzione *peakfit.m*) a 10 o più; inoltre, se il picco di interesse è su una linea di base, è necessario sommare solo le aree di quei picchi che partecipano all'approssimazione del picco stesso e *non* quelli che approssimano la linea di base.

Un'alternativa ai curve fitting con un modello espanso esponenzialmente consiste nell'usare [symmetrize.m](#) o [iSignal.m](#) su ciascun picco per convertirli in picchi simmetrici e poi approssimarli con un appropriato modello simmetrico (in questo caso una Gaussiana). Vedere pagina 77.

Con l'**avvicinamento** dei picchi, si può presentare una sfida più dura e più realistica.

```
>> y=modelpeaks2(x,[1 5 5 5 5],[1 1 1 1 1],[20 25 30 35 40],[3 3 3 3  
3],[0 -5 -10 -15 -20]);
```

In questo caso, il [metodo di costruzione del triangolo](#) fornisce aree di 3.1294, 3.202 3.3958, 4.1563 e 4.4039, sovrastimando seriamente le aree degli ultimi due picchi e *measurepeaks.m* utilizzando il metodo del taglio verticale (pagina 140) fornisce aree di [3.233 3.2108 3.0884 3.0647 3.3602, rispetto al valore teorico di 3.1938, migliori ma non perfette. Il metodo dell'altezza integrata a passi [*integration-step height method*] è quasi inutile perché i passi non sono più distinti chiaramente.



La funzione *peakfit* funziona meglio, utilizzando ancora il risultato

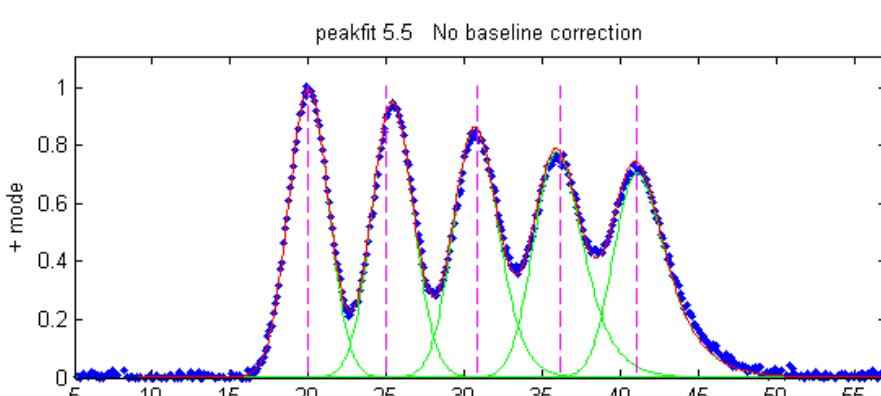
approssimativo di **findpeaksG.m** come valore di 'start' (8° argomento di input) per *peakfit*.

```
>> [FitResults,FittingError]=peakfit([x;y],30,54,5,[1 8 8 8 8],[0 -5 -10  
-15 -20],10, [20 3.5 25 3.5 31 3.5 36 3.5 41 3.5],0)
```

Peak#	Position	Height	Width	Area
1	20	0.99999	3.0002	3.1935
2	25	0.99988	3.0014	3.1945...
3	30	1.0004	2.9971	3.1918
4	35	0.9992	3.0043	3.1955
5	40.001	1.0001	2.9981	3.1915

FittingError = 0.01%

Successivamente, si crea una sfida [ancora più difficile](#) con altezze diverse dei picchi (1, 2, 3, 4 e 5, rispettivamente) e un p' di *rumore casuale addizionale*. Le aree teoriche (Height*Width*1.0645) sono 3.1938, 6.3876, 9.5814, 12.775 e 15.969.



```

>> y=modelpeaks2(x,[1 5 5 5 5],[1 2 3 4 5], [20 25 30 35 40], [3 3 3 3 3], [0 -5 -10 -15 -20])+.01*randn(size(x));

>> [FitResults,FittingError]=peakfit([x;y],30,54,5, [1 8 8 8 8], [0 -5 -10 -15 -20], 20, [20 3.5 25 3.5 31 3.5 36 3.5 41 3.5],0)

Peak#      Position      Height      Width Area
1  19.999  1.0015  2.9978  3.1958
2  25.001  1.9942  3.0165  6.4034
3  30          3.0056  2.9851  9.5507
4  34.997  3.9918  3.0076  12.78
5  40.001  4.9965  3.0021  15.966
FittingError = 0.2755

```

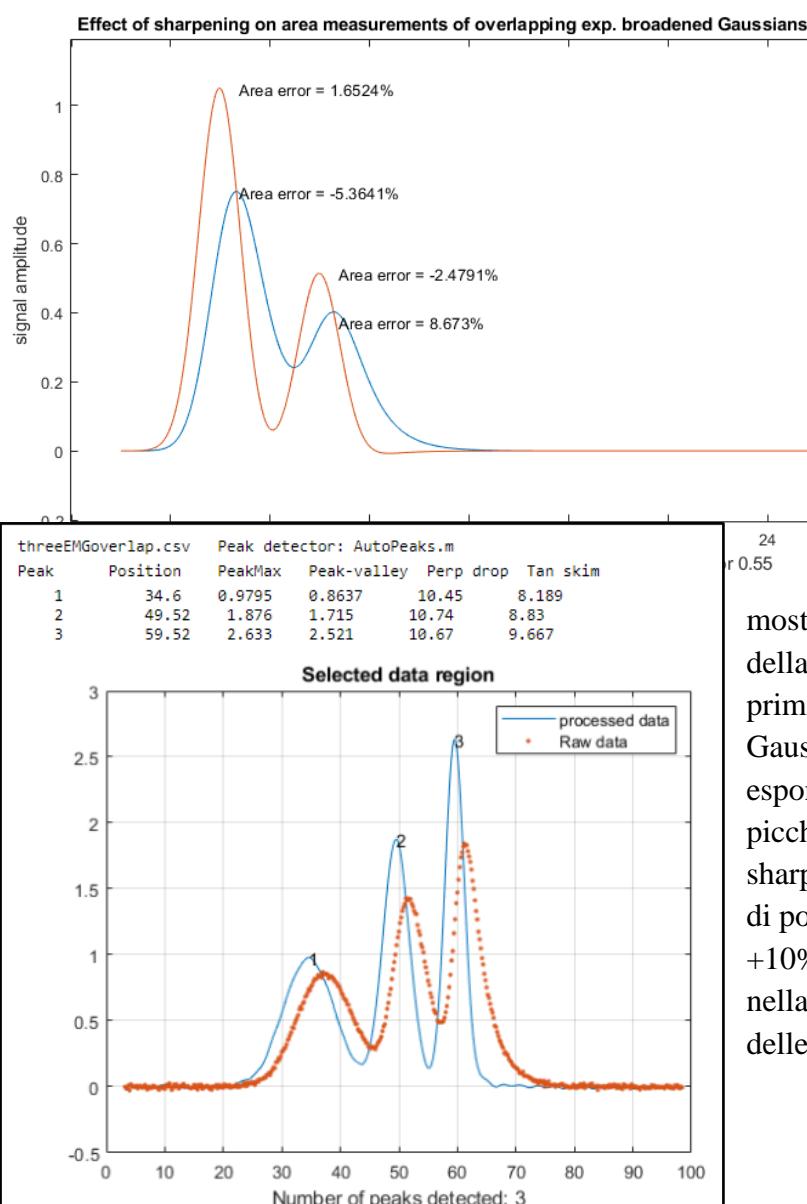
Le aree misurate in questo caso (l'ultima colonna) sono prossime ai valori teorici, mentre tutti gli altri metodi danno una precisione notevolmente inferiore. Maggiore è la sovrapposizione tra i picchi e più disuguali sono le altezze, peggiore è l'accuratezza dei metodi del taglio verticale e della costruzione del triangolo. Se i picchi sono così sovrapposti che i massimi separati non sono visibili, entrambi i metodi falliscono completamente, mentre il curve fitting può spesso recuperare un risultato ragionevole, ma *solo se è possibile fornire un valore approssimativo come ipotesi iniziale.*

Sebbene il curve fitting è generalmente il metodo più potente per trattare gli effetti combinati di picchi asimmetrici sovrapposti poggiati su un background non irrilevante, la tecnica più semplice e computazionalmente più veloce dello [sharpening con la derivata prima](#) (pagina 77) può risultare

utile come metodo per ridurre o eliminare gli effetti dell'ampliamento esponenziale, ottenendo una forma più semplice, più facile e più veloce da approssimare. Come nel caso del curve fitting, è più efficace se c'è un picco isolato con lo stesso ampliamento esponenziale perché quel picco può essere utilizzato per determinare più facilmente il valore migliore del fattore di ponderazione della derivata prima.

[SymmetrizedOverlapDemo.m](#),

mostrato a lato, mostra l'ottimizzazione della simmetrizzazione della derivata prima per la misura delle aree di due Gaussiane sovrapposte allargate esponenzialmente. Disegna e confronta i picchi originali (in blu) e quelli con sharpening (in rosso), poi prova i fattori di ponderazione della derivata prima da +10% a -10% del valore corretto di tau nella riga 14, disegna gli errori assoluti delle aree dei picchi rispetto ai valori dei



fattori. È possibile modificare la risoluzione modificando le posizioni del picco nelle righe 17 e 18 o la larghezza nella riga 13. Modificare l'altezza nella riga 16. Si devono avere derivxy.m, autopeaks.m e halfwidth.m nel path di ricerca di Matlab. Questo metodo tratta facilmente anche l'ampliamento col [doppio esponenziale](#), pagina 86, che non è facilmente gestibile col solo curve fitting.

La misura dell'area del picco con metodi multipli fa parte dello strumento di rilevamento del picco Live Script [PeakDetection mlx](#) descritto a pagina 245 e illustrato a lato. Questo strumento interattivo consente la [simmetrizzazione della derivata prima](#) opzionale dei picchi distorti, nonché lo sharpening simmetrico mediante [autodeconvoluzione di Fourier](#), per migliorare la risoluzione dei picchi sovrapposti e migliorare la precisione del picco è la misura.

Approssimazione delle curve A: Quadrati minimi lineare

L'obiettivo del curve fitting è quello di trovare i parametri di un modello matematico che descrive un insieme di dati (spesso rumorosi) in modo da minimizzare le differenze tra il modello e i dati. L'approccio più comune è il metodo dei "minimi quadrati lineari", chiamato anche "minimi quadrati polinomiali", una procedura matematica ben nota per trovare i coefficienti delle equazioni [polinomiali](#) che "meglio approssimano" l'insieme dei dati X,Y. Un'equazione polinomiale esprime la variabile dipendente Y come somma ponderata di una serie di funzioni a valore singolo della variabile indipendente X, più comunemente come una linea retta ($Y = a + bX$, dove **a** è l'*intercetta* e **b** è la *pendenza*), o una quadrica ($Y = a + bX + cX^2$), o una cubica ($Y = a + bX + cX^2 + dX^3$), e così via ai polinomi di ordine superiore. Questi coefficienti (**a**, **b**, **c**, ecc.) si possono usare per prevedere i valori di Y per ciascuna X. In tutti questi casi, Y è una *funzione lineare* di tutti i parametri **a**, **b**, **c** e/o **d**. *Questo è il motivo per cui la si chiama approssimazione "lineare" ai quadrati minimi, non perché il grafico di X rispetto a Y sia lineare.* solo per il polinomio di primo-ordine $Y = a + bX$ il grafico di X rispetto a Y è lineare. E se il modello *non* è descrivibile con una somma ponderata di funzioni a valore singolo, allora è possibile utilizzare un metodo dei minimi quadrati diverso, più laborioso dal punto di vista computazionale, il metodo "non-lineare", discusso a pagina 190.

"Approssimazione migliore [Best fit]" significa semplicemente che le differenze tra i valori Y effettivi misurati e i valori Y previsti dall'equazione del modello sono *minimizzati*. Non significa una "perfetta" approssimazione; nella maggior parte dei casi, l'approssimazione migliore ai minimi quadrati *non passa per tutti i punti* del set di dati. Soprattutto, un'approssimazione ai quadrati minimi *deve essere conforme al modello selezionato* - per esempio, una linea retta o una parabola quadratica - e ci saranno quasi sempre dei punti dati che non passano esattamente sul linea migliore dell'approssimazione, a causa di un errore casuale nei dati o perché il modello non è in grado di descrivere esattamente i dati.

Un'altra cosa: non è corretto dire "approssima i dati a ..." una linea retta o un altro modello; in realtà è il contrario: si sta approssimando un *modello ai dati*. I *dati* non vengono in alcun modo modificati; è il *modello* che viene regolato per approssimare i dati. (In effetti, in qualche caso speciale può risultare utile trasformare i dati prima del 'curve fitting'; vedere pagina 164).

Le approssimazioni migliori ai quadrati minimi possono essere calcolati con alcune calcolatrici portatili, co gli spreadsheet e programmi appositi (si veda [Dettagli Matematici](#) in seguito). Sebbene sia possibile stimare la linea retta più adatta tramite una stima visiva e una riga, il metodo ai quadrati minimi è più obiettivo e più facile da automatizzare. (Se si dovesse fornire un grafico X, Y a cinque diverse persone chiedendo loro di stimare visivamente la retta migliore, si otterrebbero

cinque risposte leggermente diverse, ma fornendo i dati a cinque diversi computer, la risposta sarebbe identica ogni volta).

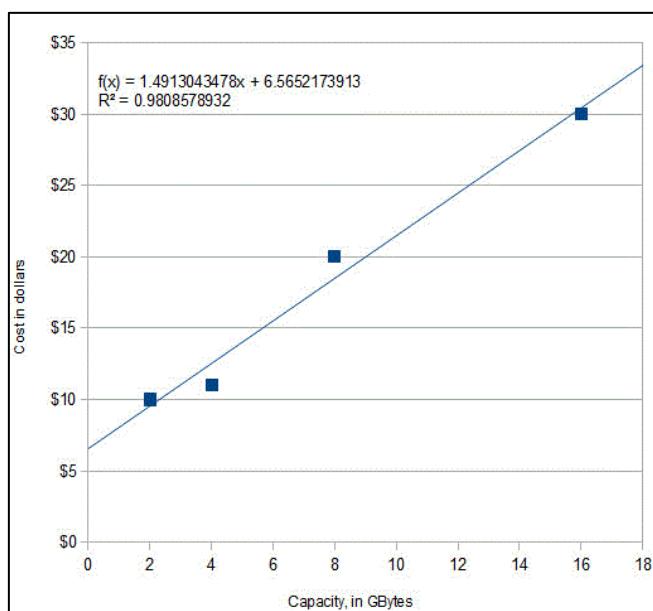
Esempi di approssimazioni polinomiali

Ecco un semplicissimo esempio: lo storico dei prezzi delle schede di memoria SD di diverse dimensioni pubblicizzati nel numero del 19 febbraio del 2012 del New York Times. (Sì, chiaramente i prezzi sono *molto* più bassi ora, ma questi erano davvero i prezzi in un grande magazzino nel 2012).

Capacità della Memoria (GBytes) Prezzo in dollari US

2	\$9.99
4	\$10.99
8	\$19.99
16	\$29.99

Qual è il rapporto tra la capacità di memoria ed il costo? Ovviamente, ci si aspetta che le schede di capacità maggiore costino di più di quelle più piccole e se si disegna il grafico del costo rispetto alla capacità (grafico a pagina seguente), si vedrà una relazione approssimativamente rettilinea. Un algoritmo ai quadrati minimi può calcolare i valori di "a" (intercetta) e "b" (pendenza) di una linea retta che sia una "approssimazione migliore" dei punti dati. Usando il calcolo lineare dei quadrati minimi, dove **X = capacità** e **Y = costo**, l'equazione della retta che più semplicemente descrive questi dati (arrotondando al centesimo più vicino) è:



$$\text{Costo} = \$6.56 + \text{Capacità} * \$1.49$$

Quindi, \$1.49 è la *pendenza* e \$6.56 è l'*intercetta*. (L'equazione è disegnata come linea continua che passa tra i punti dei dati nella figura). Fondamentalmente, questo ci dice che il costo di una scheda di memoria è costituito da un costo fisso di \$6.56 più \$1.49 per ogni GBytes di capacità. Come interpretarlo? I \$6.56 rappresentano i costi che sono gli stessi a prescindere dalla capacità della memoria: una stima ragionevole è che include cose come la confezione (le diverse schede hanno la stessa dimensione fisica e sono confezionate allo stesso modo), il

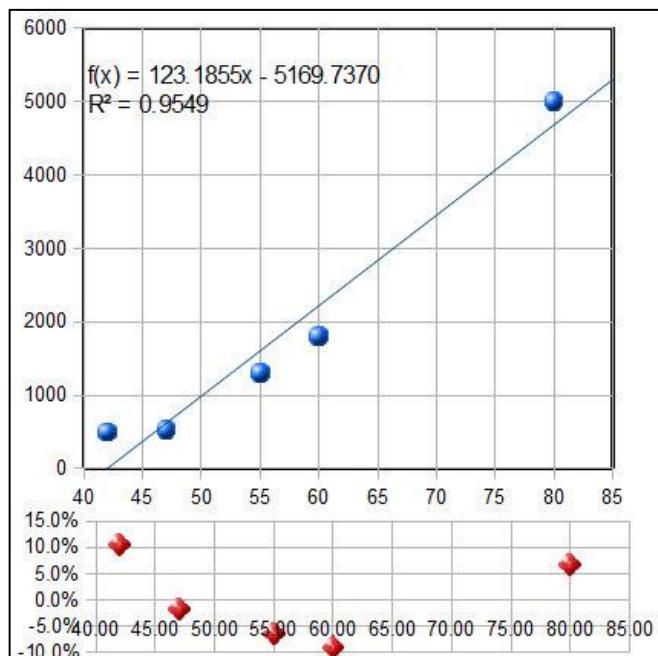
trasporto, il marketing, la pubblicità e lo spazio nel negozio al dettaglio. Il \$1.49 (1.49 dollari/Gbyte) rappresentano l'aumento del prezzo di vendita al dettaglio del chip del circuito integrato più grande nelle schede di capacità maggiore, soprattutto perché *hanno più valore per il consumatore* ma potrebbe anche essere più costoso costruirlo perché c'è più silicio, sono più complessi, o forse hanno maggiori scarti di produzione. Quindi, in questo caso la pendenza e l'intercetta hanno dei significati fisici ed economici.

Cosa possiamo fare con quest'informazione? Prima di tutto, possiamo vedere quanto si avvicinino a questa equazione i prezzi reali: approssimativamente *ma non perfettamente*. La retta dell'equazione passa *in mezzo* ai punti dati ma non passa esattamente *in* ciascuno di essi. Questo perché i prezzi al dettaglio sono influenzati anche da diversi fattori imprevedibili e casuali: la concorrenza locale, offerta, domanda, e persino arrotondamento dei prezzi al numero "intero" più vicino; tutti questi fattori costituiscono il "[rumore](#)" in tali dati. La procedura dei minimi quadrati calcola anche R^2 ,

detto il *coefficiente di determinazione* o il *coefficiente di correlazione*, che è un indicatore della "bontà dell'approssimazione". R^2 è esattamente 1.0000 quando l'approssimazione è perfetta, di meno quando l'approssimazione non lo è. Più ci si avvicina a 1.0000 meglio è. Un valore di R^2 di 0.99 solitamente indica una buona approssimazione; 0.999 è ottima. (Il valore R^2 viene calcolato come mostrato a pagina 169).

L'altro modo con cui si possono usare tali dati è quello per prevedere il prezzo di schede di altre capacità, se fossero disponibili, inserendo la capacità nell'equazione e valutarne il costo. Per esempio, una scheda da 12 Gbyte dovrebbe costare \$24.44 secondo questo modello. E una da 32 Gbyte dovrebbe costare \$54.29, ma *sarebbe una previsione oltre l'intervallo dei dati disponibili* - si chiama "estrapolazione" - *ed è molto rischiosa* perché non si conoscono davvero quali altri fattori possono influenzare i dati oltre l'ultimo punto. (Si potrebbe anche risolvere l'equazione della capacità in funzione del costo e utilizzarla per prevedere quanta capacità ci si potrebbe aspettare di acquistare per una data quantità di denaro se un tale prodotto fosse disponibile).

Come si è detto, erano i prezzi del 2012. Perché non fare un po' di "compiti a casa"? Cercare e provare ad approssimare i prezzi *correnti* e confrontarli. Si ottiene una pendenza inferiore, una intercetta più piccola o entrambe?

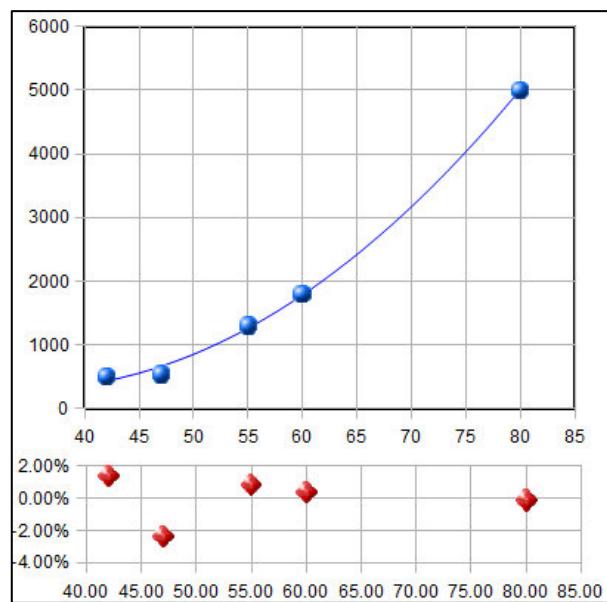


Ecco un altro esempio correlato: i prezzi storici dei televisori LCD a schermo piatto standard ad alta definizione (non UHD) in funzione delle dimensioni dello schermo, come pubblicizzato sul Web nella primavera del 2012 I prezzi dei cinque modelli selezionati, *simili ad eccezione della dimensione dello schermo*, sono disegnati rispetto alla dimensione dello schermo in pollici, nella figura a lato ed approssimati con un modello del primo ordine (una retta).

Come per il precedente esempio, l'approssimazione non è perfetta. L'equazione del modello approssimato è mostrata in alto sul grafico, assieme al valore di R^2 (0.9549) indicante che l'approssimazione non è

particolarmente buona. E si può vedere dalla linea di approssimazione che un 40-pollici avrà un *costo negativo!* È pazzesco. *Paghrebbero* per vendere questi televisori? Penso di no Chiaramente, qualcosa qui non va.

La bontà dell'approssimazione è mostrata ancora più chiaramente nel piccolo grafico in fondo alla figura, con i punti rossi. Questo mostra i "residui", le differenze tra ciascun punto e l'approssimazione ai quadrati minimi in quel punto. Si vede che le deviazioni dallo zero sono grandi ($\pm 10\%$), ma ancora più importante, *non sono completamente casuali*; formano una *curva a forma di U chiaramente visibile*. Questo suggerisce che il modello della retta usato potrebbe non essere quello ideale e si potrebbero avere risultati migliori cambiando modello.



(Oppure potrebbe essere solo un *caso*: il primo e l'ultimo punto potrebbero essere più alti del previsto, perché erano televisori insolitamente costosi per quelle dimensioni. Come accertarsene, se l'insieme dei dati non è molto accurato?)

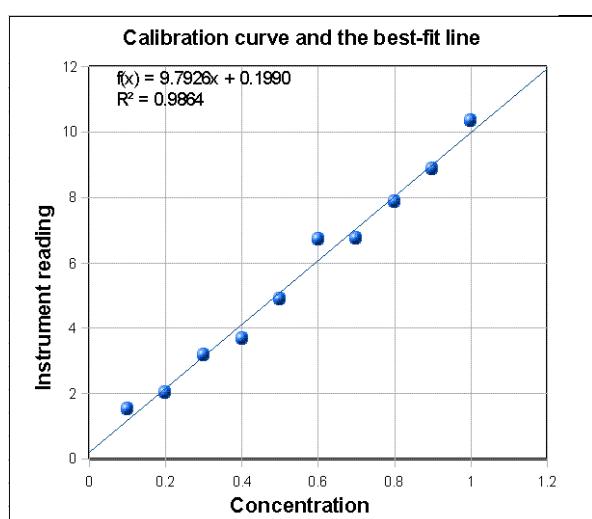
I calcoli ai quadrati minimi possono approssimare i dati non solo a linee rette, ma a *qualsiasi insieme di dati descrivibili da una polinomiale*, per esempio un'equazione del secondo ordine (quadratica) ($Y = a + bX + cX^2$). applicando un'approssimazione del secondo ordine a questi dati, si ottiene il grafico in figura. Ora il valore di R^2 è più alto, 0.9985, indicando che l'approssimazione è migliore (ma ancora una volta non proprio perfetta), e che i residui (punti rossi in basso) sono più piccoli e più casuali. Ciò non dovrebbe davvero essere una sorpresa, per la stessa natura di questi dati particolari. La dimensione di uno schermo TV è sempre indicata come la *lunghezza della diagonale*, da un angolo dello schermo al suo angolo opposto, ma la quantità di materiale, la difficoltà di fabbricazione, il peso e i requisiti di alimentazione dello schermo dovrebbero essere tutti in scala con l'*area dello schermo*. L'area è proporzionale al *quadrato* della misura lineare, quindi l'inclusione di un termine X^2 nel modello è più che ragionevole in questo caso. Applicando un'approssimazione quadratica, con un termine X^2 , si prevede che il set da 40 pollici costerà meno di \$500, il che è più sensato dell'approssimazione lineare. (L'effettiva interpretazione del significato dei coefficienti dell'approssimazione migliore **a**, **b** e **c** è, tuttavia, impossibile a meno che non si sappia di più sulla produzione e la commercializzazione dei televisori). La procedura dei minimi quadrati ci permette di modellare i dati con un'equazione polinomiale più o meno semplice. Il punto qui è che un modello quadratico è giustificato non solo perché si adatta meglio ai dati, ma in questo caso è giustificato perché è *previsto in linea di principio* basato sul rapporto tra la lunghezza e l'area. (Per inciso, come ci si potrebbe aspettare, i prezzi sono diminuiti considerevolmente dal 2012; nel 2024, un set Vizio 4K da 65" era disponibile presso Sam's Club per meno di \$380, molto più economico di quanto previsto da questa approssimazione).

In generale, l'approssimazione di *qualsiasi* insieme di dati con un polinomio di ordine superiore, come una quadratica, cubica o superiore, ridurrà l'errore di approssimazione rendendo i valori di R^2 più prossimi a 1.000. Questo perché un modello di ordine superiore ha più coefficienti variabili che il programma può regolare per approssimare i dati. Per esempio, si potrebbero approssimare i dati dei prezzi della scheda SD con una quadratica (grafico), ma non c'è alcun motivo per farlo e l'approssimazione migliorerebbe di poco. Il pericolo è che si potrebbe "approssimare il rumore", cioè regolarsi sul rumore casuale di quel *quel set di dati*, mentre un'altra misura con del rumore casuale diverso fornirebbe risultati diversi. Infatti, utilizzando un ordine polinomiale *uno meno il numero dei punti*, l'approssimazione sarà perfetta con $R^2=1.000$. Per esempio, i dati della scheda SD hanno solo 4 punti e se si approssimano questi dati a un polinomio del 3^o ordine (cubica), si otterrà un'approssimazione matematicamente *perfetta* (grafico), ma che non ha senso nel mondo reale (il prezzo scende al di sopra di x=14 Gbyte). È davvero privo di significato e fuorviante - *qualsiasi* dato di 4-punti si adatterebbe perfettamente a un modello cubico, *anche il rumore casuale puro!*

L'unica giustificazione per l'utilizzo di un polinomio di ordine superiore è che c'è motivo di credere,

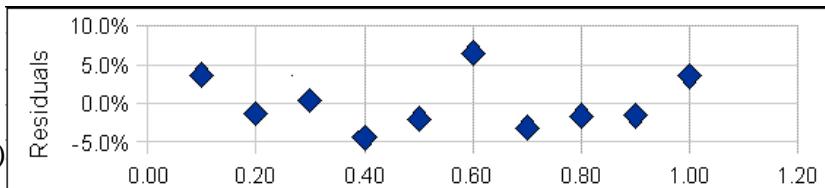
o è stato osservato, che ci sia una *consistente* non-linearietà nei dati, come nell'esempio precedente del prezzo del TV.

Il grafico qui mostra un terzo esempio, preso dal campo della chimica analitica: sono i dati di calibrazione in linea retta dove X = concentrazione e Y = la lettura di qualche strumento che dovrebbe essere linearmente proporzionale alla concentrazione X (in altre parole, $Y = a + bX$). Se si sta leggendo questo



online, si può [cliccare per scaricare questi dati](#). I punti blu sono i dati. Non cadono tutti su una linea retta perfetta a causa del rumore casuale e dell'errore di misurazione nelle letture dello strumento ed eventualmente anche degli errori volumetrici nelle concentrazioni degli standard (che di solito vengono preparati in laboratorio diluendo una soluzione madre). Per questo primo gruppo di dati, la **b** è 9.7926 e l'intercetta **c** è 0.199. In chimica analitica, la pendenza della curva di calibrazione è spesso chiamata "sensibilità". L'intercetta indica la lettura dello strumento che ci si aspetterebbe se la concentrazione fosse zero. Normalmente, gli strumenti vengono regolati ("azzerati") dall'operatore per fornire una lettura di zero per una concentrazione di zero, ma il rumore casuale e la deriva dello strumento possono far sì che l'intercetta sia diversa da zero per qualsiasi calibrazione. In questo caso, i dati sono infatti generati dal computer, il valore "vero" della pendenza era esattamente 10 e dell'intercetta era esattamente zero prima venisse aggiunto il rumore, e il rumore veniva aggiunto da un generatore di numeri casuali normalmente distribuiti e centrati sullo zero. La presenza del rumore ha provocato uno scostamento del 2% circa su questa misura della pendenza. (Se ci fosse stato un numero maggiore di punti, i valori calcolati della pendenza e dell'intercetta sarebbero stati quasi certamente migliori. In media, la precisione delle misure della pendenza e dell'intercetta migliora con la *radice quadrata del numero di punti* nei dati). Con questo numero di punti, è *matematicamente* possibile utilizzare un grado polinomiale ancora più alto, fino a uno in meno del numero di punti dati, ma non è *fisicamente* ragionevole nella maggior parte dei casi; per esempio, si potrebbe approssimare perfettamente un polinomio del 9°-grado, ma il **risultato è alquanto insolito** ([link al grafico](#)). Nessuno strumento analitico ha una curva di calibrazione che si comporta in questo modo!

Un grafico dei residui per i dati di calibrazione (a lato) solleva una domanda. Ad eccezione del sesto punto (a una concentrazione di 0,6) gli altri punti sembrano formare



una curva a forma di U, indicando che un'equazione quadratica potrebbe essere un modello migliore per quei punti rispetto a una linea retta. Si può rifiutare il 6° punto come valore "anomalo", dovuto forse a un errore nella preparazione di quella soluzione standard o alla lettura dello strumento? L'eliminazione di quel punto [migliorerebbe la qualità dell'approssimazione](#) ($R^2=0.992$ anziché 0.986) specie se si [utilizzasse un'approssimazione quadratica](#) ($R^2=0.998$). L'unico modo per saperlo con certezza è ripetere quella preparazione della soluzione standard e la calibrazione e vedere se quella forma a U persiste tra i residui. Molti strumenti danno una risposta di calibrazione molto lineare, mentre altri possono mostrare una risposta leggermente non lineare in alcune circostanze ([per esempio](#)). Ma in realtà, i dati di calibrazione usati per *questo* esempio sono stati generati dal computer per essere *perfettamente lineari*, con numeri casuali normalmente distribuiti aggiunti per simulare il rumore. Quindi quel sesto punto *non è un valore anomalo* e i dati sottostanti non sono realmente curvi, ma *non lo si saprebbe in un'applicazione reale*. Sarebbe stato un errore eliminare quel 6° punto ed usare una quadrica in questo caso. Morale: non eliminare punti solo perché sembrano un po' strani, a che non si abbia una buona ragione, e non usare approssimazioni di ordine più alto solo perché si adatta meglio se si sa che lo strumento fornisce una risposta lineare in quelle circostanze. Anche errori casuali perfettamente distribuiti normalmente possono occasionalmente dare deviazioni individuali che si discostano abbastanza dalla media e potrebbero indurre a pensare che siano valori anomali. Non ci si faccia ingannare. (*Piena confessione*: l'esempio precedente è stato ottenuto "[spulciando](#)" tra decine di set di dati generati casualmente con del rumore aggiunto, per cercare quello che, sebbene casuale, *sembrasse* avere un valore anomalo).

Risolvere l'equazione di calibrazione per la concentrazione. Una volta stabilita la curva di calibrazione, questa può essere utilizzata per determinare le concentrazioni di campioni ignoti misurati

sullo stesso strumento, ad esempio *risolvendo l'equazione per la concentrazione come una funzione della lettura dello strumento*. Il risultato per il caso lineare è che la concentrazione del campione C_x è dato da $C_x = (S_x - \text{intercetta})/\text{pendenza}$, dove S_x è il segnale dato dalla soluzione campione e la "pendenza" e la "pendenza" sono il risultato dell'approssimazione ai quadrati minimi. Se viene usata un'approssimazione quadratiche, allora si deve usare una "[equazione quadratica](#)" più complessa per risolvere la concentrazione (vedere [QuadraticEquation.m](#)), ma il problema di risolvere l'equazione della calibrazione per la concentrazione diventa [incredibilmente complesso per approssimazioni con polinomi di ordine superiore](#). (Nota: La concentrazione e le letture dello strumento si possono registrare con qualsiasi unità di misura conveniente se le stesse unità vengono usate sia per la calibrazione che per le misure ignote).

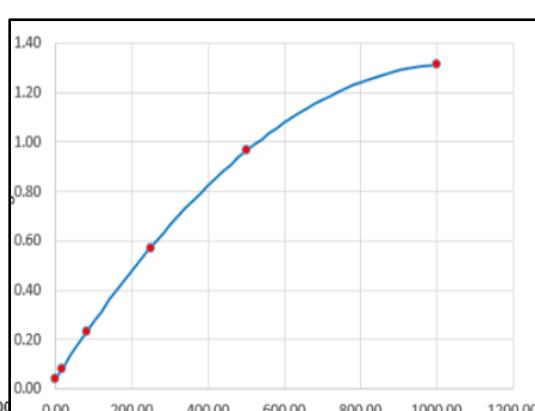
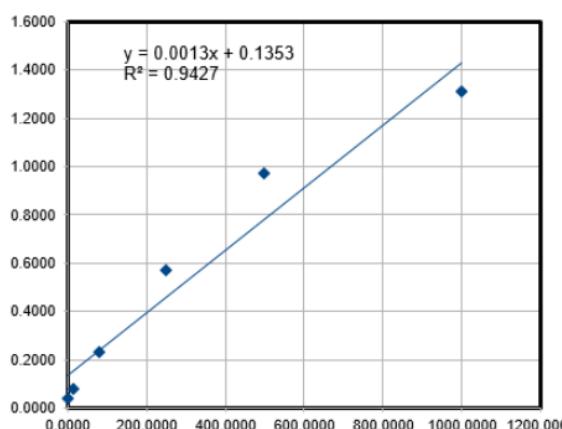
Affidabilità dei risultati col curve fitting

Quanto sono affidabili la pendenza, l'intercetta e gli altri coefficienti polinomiali ottenuti dai calcoli dei quadrati minimi sui dati sperimentali? L'unico fattore più importante è l'adeguatezza del modello scelto; è fondamentale che il modello (ad esempio lineare, quadratico, gaussiano, ecc.) sia una buona corrispondenza con l'effettiva forma dei dati. Lo si può fare scegliendo un modello basato sul comportamento noto e atteso di quel sistema (come usare un modello di calibrazione lineare per uno strumento noto per dare una risposta lineare in quelle condizioni) o scegliendo un modello che dà sempre residui sparsi casualmente che non mostrano una forma regolare. Ma anche con un modello perfetto, la procedura dei minimi quadrati applicata a insiemi ripetitivi di misurazioni non darà sempre gli stessi risultati a causa di un errore casuale (rumore) nei dati. Se si dovesse ripetere più volte l'intero set di misure e fare i calcoli dei minimi quadrati su ciascun set di dati, le deviazioni standard dei coefficienti varierebbero direttamente con la deviazione standard del rumore e inversamente con la radice quadrata del numero di punti in ogni approssimazione, a parità di tutte le altre. Il problema, ovviamente, è che non è sempre possibile ripetere tutte le misure più volte. Si potrebbe avere *un* solo blocco di misure e ogni esperimento potrebbe essere molto costoso da ripetere. Quindi, sarebbe fantastico se avessimo un metodo di scelta rapida che ci consenta di *predire* le deviazioni standard dei coefficienti da una *singola misura* del segnale, senza doverla ripetere.

Verranno descritti tre metodi generali per predire la deviazioni standard dei coefficienti polinomiali: la [propagazione algebrica degli errori](#), la [simulazione Monte Carlo](#) e il [metodo di campionamento bootstrap](#).

Propagazione algebrica degli errori

Il metodo classico si basa sulle [regole per la propagazione degli errori matematici](#). La propagazione degli errori dell'intero metodo di approssimazione della curva può essere descritta in [algebra in forma chiusa](#) suddividendo il metodo in una serie di semplici differenze, somme, prodotti e rapporti, e applicando le [regole della propagazione degli errori](#) a ciascun passo. I risultati di questa procedura per un'approssimazione ai minimi quadrati del primo ordine (linea retta) sono mostrati nelle ultime tre righe dell'insieme di equazioni in [Dettagli Matematici](#), a pagina 169. Essenzialmente, queste equazioni fanno uso delle deviazioni dalla linea dei



minimi quadrati
(i "residui") per
stimare le
deviazioni
standard della
pendenza e

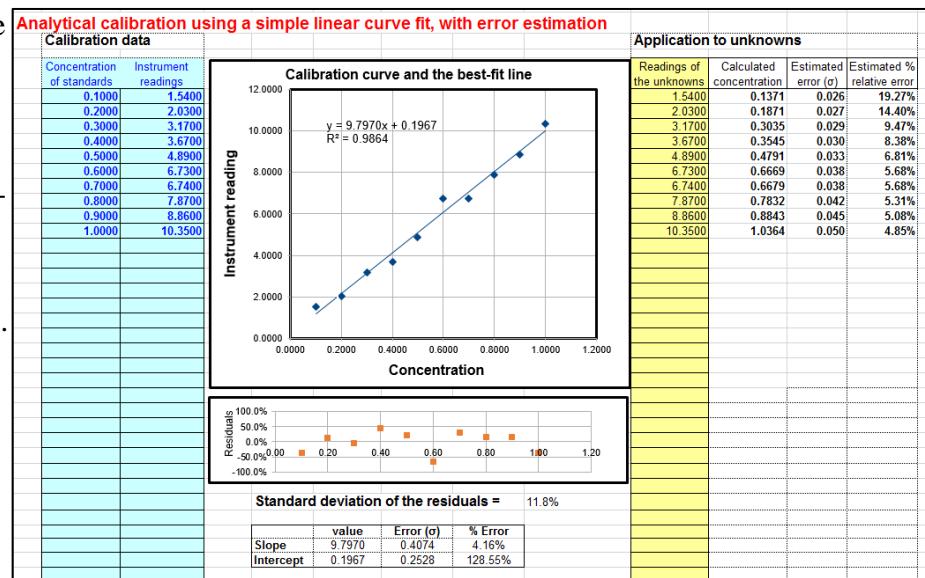
dell'intercetta, basandosi sul presupposto che il rumore in quel singolo insieme di dati sia *casuale* e sia rappresentativo del rumore che si otterrebbe ripetendo le misure. *Poiché queste previsioni si basano solo su un singolo set di dati, sono valide solo nella misura in cui tale set sia tipico di altri che potrebbero essere ottenuti con misurazioni ripetute.* Se i propri errori casuali sono *piccoli* durante l'acquisizione dei dati, si otterrà un'ingannevole approssimazione di *bell'aspetto*, ma anche le stime della deviazione standard della pendenza e dell'intercetta saranno troppo *basse*, mediamente. Se gli errori casuali sono *grandi* in quei dati, si otterrà un'approssimazione ingannevolmente *brutta*, ma le stime della deviazione standard saranno troppo *alte*, mediamente. Questo problema peggiora quando il numero di punti è piccolo. Questo non vuol dire che non valga la pena calcolare le deviazioni standard previste della pendenza e della intercetta, ma si tenga presente che queste previsioni sono accurate solo se il numero di punti dati è grande (e solo se il rumore è casuale e normalmente distribuito). Attenzione: se le deviazioni dalla linearità nei dati sono *sistematiche* e non *casuali* - per esempio, se si cerca di approssimare una linea retta a dei dati curvi filtrati con smoothing (a sinistra, pagina precedente), allora le stime delle deviazioni standard della pendenza e dell'intercetta da queste ultime due equazioni *saranno troppo alte*, perché presumono che le deviazioni siano causate da rumore casuale che varia da misura a misura, mentre in effetti dei dati curvi filtrati senza rumore casuale (a destra, pagina precedente) darà la *stessa* pendenza e intercetta da misura a misura.

Nell'applicazione della calibrazione analitica, la concentrazione del campione C_x è data da C_x = (S_x - intercetta)/pendenza, dove S_x è il segnale dato dalla soluzione campione. L'incertezza dei tre termini contribuisce all'incertezza di C_x. La deviazione standard di C_x può essere stimata dalle deviazioni standard di pendenza, intercetta e S_x utilizzando le [regole per la propagazione degli errori matematici](#). Ma il problema è che, in chimica analitica, la manodopera e il costo della preparazione e gestione di un gran numero di soluzioni standard spesso ne limita il numero a un insieme piuttosto piccolo, per standard statistici, quindi queste stime della deviazione standard sono spesso pessime.

Un foglio di calcolo che esegue questi calcoli sulla propagazione degli errori per i propri dati di calibrazione analitica del primo ordine (lineari) può essere scaricato da

<http://terpconnect.umd.edu/~toh/models/CalibrationLinear.xls>.

Per esempio, col caso della calibrazione lineare nella sezione precedente, dove il valore "vero" della pendenza era 10 e quello dell'intercetta era zero, questo spreadsheet (la cui



schermata è mostrata a lato) prevede che la pendenza sia 9.8 con una deviazione standard di 0.407 (4.2%) e che l'intercetta sia 0.197 con una deviazione standard di 0.25 (128%), entrambi ben entro due deviazioni standard dei valori veri. Questo spreadsheet esegue anche i calcoli della propagazione dell'errore per le concentrazioni calcolate di ciascuna incognita nelle ultime due colonne a destra. Nell'esempio in questa figura, le letture dello strumento degli standard sono acquisite come incognite, mostrando che il range degli errori percentuali previsti variano dal 5% al 19% del valore vero di questi standard. (Si noti che la deviazione standard della concentrazione è maggiore ad alte concentrazioni rispetto alla deviazione standard della pendenza, e considerevolmente maggiore a basse

concentrazioni a causa della maggiore influenza dell'incertezza nell'intercetta). Per un'ulteriore discussione e degli esempi, si veda "[The Calibration Curve Method with Linear Curve Fit](#)". La funzione Matlab/Octave [plotit.m](#) usa il metodo algebrico per calcolare le deviazioni standard dei coefficienti quadrati minimi per qualsiasi ordine polinomiale.

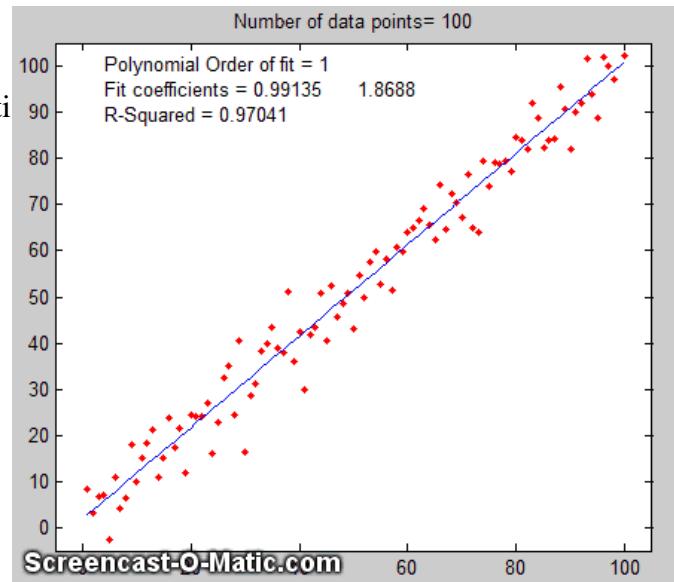
Simulazione Monte Carlo

Il secondo modo per stimare le deviazioni standard dei coefficienti dei minimi quadrati consiste nell'eseguire una simulazione dei numeri casuali (un tipo di [simulazione Monte Carlo](#)). Richiede la conoscenza (dalle misurazioni precedenti) della deviazione standard media del rumore casuale nei dati. Utilizzando un computer, si costruisce un modello dei dati sul normale intervallo di valori di X e Y (p.es. $Y = \text{intercept} + \text{slope} * X + \text{noise}$, dove **noise** è il vettore del rumore nei dati), si calcola la pendenza e l'intercetta per ciascun insieme rumoroso di dati simulato, poi si ripete il processo molte volte (solitamente poche migliaia) con diversi insiemi di rumore casuale, e alla fine si calcola la deviazione standard di tutte le pendenze e delle intercette risultanti. Questo viene normalmente fatto con rumore casuale normalmente distribuito (ad esempio, le funzioni RAND o RANDN di cui dispongono molti linguaggi di programmazione). Questi generatori di numeri casuali producono rumore "bianco", ma [si possono derivare altri colori del rumore](#). Se il modello è buono e il rumore nei dati è ben caratterizzato in termini di distribuzione in frequenza e dipendenza dall'ampiezza del segnale, i risultati saranno una stima molto buona delle deviazioni standard attese dei coefficienti dei minimi quadrati. (Se il rumore non è costante, ma varia con i valori di X e Y, o se il rumore non è bianco o non è distribuito normalmente, allora quel comportamento deve essere incluso nella simulazione).

Un [esempio animato](#) è mostrato a lato (visionabile leggendolo in [Microsoft Word 365](#), altrimenti cliccare su [questo link](#)), per un caso di un set di dati rettilineo di 100-punti con una pendenza=1, intercetta=0 e una deviazione standard del rumore aggiunto pari allo 5% del valore massimo di y. Per ogni serie ripetuta di dati simulati, i coefficienti di approssimazione (pendenza e intercetta misurate ai minimi quadrati) sono leggermente diversi a causa del rumore.

Ovviamente, questo metodo prevede la programmazione di un computer per calcolare il modello e non è conveniente come valutare una semplice espressione algebrica. Ma ci sono due importanti vantaggi di questo metodo: (1) ha una grande generalità; può essere applicato a metodi di approssimazione della curva che sono troppo complicati per i calcoli classici di propagazione dell'errore algebrici in forma chiusa, anche [ai metodi iterativi non lineari](#); e (2) le sue previsioni si basano sul rumore medio nei dati, non sul rumore in un singolo set di dati. Per questo motivo, fornisce stime più affidabili, in particolare quando il numero di punti in ciascun set di dati è piccolo. Tuttavia, non è sempre possibile applicare questo metodo perché non sempre si conosce la deviazione standard media del rumore casuale nei dati. È possibile eseguire facilmente questo tipo di calcolo in Matlab/Octave e negli spreadsheet (pagina 169).

Si può scaricare uno script Matlab/Octave che confronta la simulazione Monte Carlo col metodo algebrico precedente da <http://terpconnect.umd.edu/~toh/spectrum/LinearFiMC.m>. Eseguendo questo script con set di dati di diverse dimensioni ("NumPoints" nella riga 10), si può vedere che la devia-



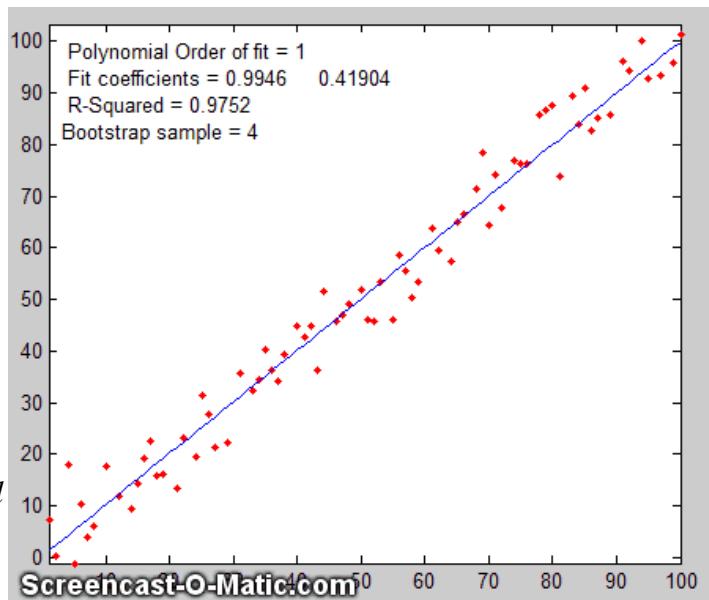
zione standard prevista dal metodo algebrico oscilla molto da una esecuzione all'altra quando NumPoints è piccolo (p.es. 10), ma le predizioni Monte Carlo sono molto stabili. Quando NumPoints è grande (p.es. 1000), entrambi i metodi concordano abbastanza bene.

Il metodo Bootstrap

Il terzo metodo è il "[bootstrap](#)", una procedura che prevede la scelta di sotto-campioni casuali con la sostituzione da un singolo set di dati e l'analisi di ciascun campione allo stesso modo (p.es. con l'approssimazione ai quadrati minimi). Ogni campione viene restituito al set di dati dopo il campionamento, in modo che (a) un particolare punto dai dati originali possa apparire più volte in ciascun campione e (b) il numero di elementi in ogni sotto-campione bootstrap sia uguale al numero di elementi nel set di dati originale. Questo si spiega meglio con un semplice esempio. Si consideri un insieme di dati con 10 coppie x,y assegnate alle lettere dalla *a* alla *j*. Il set di dati originale è rappresentato come $[a\ b\ c\ d\ e\ f\ g\ h\ i\ j]$, e alcuni sotto-campioni di bootstrap tipici potrebbero essere $[a\ b\ b\ d\ e\ f\ f\ h\ i\ i]$ o $[a\ a\ c\ c\ e\ f\ g\ g\ i\ j]$. Ogni campione bootstrap contiene lo stesso numero di punti, ma con circa un terzo delle coppie di dati saltate, un terzo duplicato e un terzo invariato. (Ciò equivale a pesare un terzo delle coppie con un fattore 2, un terzo con 0 e lasciare un terzo non ponderato). Si dovrebbe utilizzare un computer per generare centinaia di campioni bootstrap come questi e per applicare la procedura di calcolo in esame (in questo caso quella lineare dei minimi quadrati) a ciascun insieme.

Se *non* ci fosse rumore nei dati e se il modello venisse scelto correttamente, allora tutti i punti dei dati originali e ciascun sotto-campione bootstrap ricadrebbero *esattamente nel profilo del modello* e il risultato dei minimi quadrati sarebbe virtualmente lo *stesso per ogni sotto-campione*.

Tuttavia, se *c'è* rumore nei dati, ogni sotto-campione bootstrap darebbe un *risultato leggermente diverso* (ad esempio, i coefficienti polinomiali dei minimi quadrati), poiché ogni il sotto-campione ha un sottoinsieme diverso del rumore casuale. Ciò è illustrato dall'animazione a lato (visualizzabile scaricando la versione [Microsoft Word 365](#), altrimenti [cliccare su questo link](#)), per lo stesso set di dati in linea retta di 100 punti utilizzato sopra. Si può vedere che *la variazione dei coefficienti per il best-fit tra i sotto-campioni è la stessa della simulazione Monte Carlo* sopra. Maggiore è la quantità di rumore casuale nei dati,



maggior sarà l'intervallo dei risultati da campione a campione nel set di bootstrap. Questo consente di stimare l'incertezza della quantità in esame, proprio come nel metodo Monte-Carlo precedente. La differenza è che il metodo Monte-Carlo si basa sul presupposto che il rumore sia noto, casuale e possa essere accuratamente simulato da un generatore di numeri casuali su un computer, mentre il metodo bootstrap utilizza il *rumore effettivo nell'insieme dei dati* a portata di mano, come il metodo algebrico, tranne per il fatto che non necessita di una soluzione algebrica di propagazione dell'errore. Il metodo bootstrap condivide quindi la sua generalità con l'approccio Monte Carlo, ma è limitato dal presupposto che il rumore in quel (possibilmente piccolo) singolo set di dati sia rappresentativo del rumore che otterebbe con misurazioni ripetute. Il metodo bootstrap, tuttavia, non può stimare correttamente gli errori dei parametri risultanti da [una cattiva scelta del modello](#). Il metodo è esaminato in dettaglio nella sua [ampia letteratura](#). Questo tipo di calcolo bootstrap è facilmente

eseguibile in [Matlab/Octave](#) e può anche essere eseguito (con maggiore difficoltà negli [spreadsheet](#)).

Confronto dei metodi per la previsione dell'errore.

Lo script Matlab/Octave [TestLinearFit.m](#) confronta *tutti e tre* questi metodi (simulazione Monte Carlo, il metodo algebrico e il bootstrap) per un'approssimazione ai quadrati minimi di un set lineare del primo ordine di 100-punti. Ogni metodo viene ripetuto su diversi set di dati con la stessa pendenza media, intercetta e rumore casuale, quindi la deviazione standard (SD) delle pendenze (SDslope) e delle intercette (SDint) vengono compilate e incolonnate nella tabella seguente.

NumPoints = 100 SD of the Noise = 9.236 x-range = 30		
Simulation	Algebraic equation	Bootstrap method
SDslope SDint	SDslope SDint	SDslope SDint
Mean SD: 0.1140 4.1158	0.1133 4.4821	0.1096 4.0203

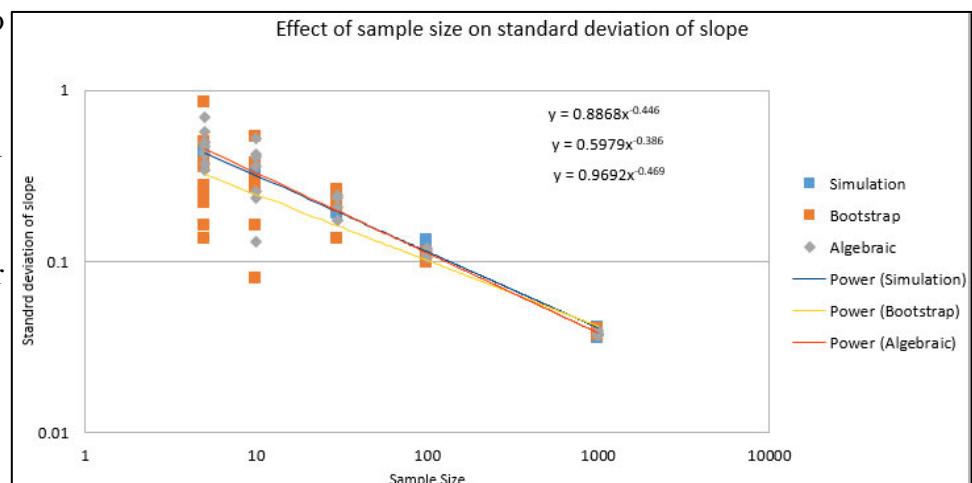
(Lo script si può scaricare da <http://terpconnect.umd.edu/~toh/spectrum/TestLinearFit.m>).

Generalmente, le deviazioni standard medie ("Mean SD") dei tre metodi concordano molto bene, ma i metodi algebrico e bootstrap fluttuano più della simulazione Monte Carlo ogni volta che viene eseguito questo script, perché sono basati sul rumore in uno *singolo* set di 100 punti, mentre la simulazione Monte Carlo riporta la media di molti set di dati. Naturalmente, il metodo algebrico è più semplice e veloce da calcolare rispetto agli altri metodi. Tuttavia, una propagazione algebrica della soluzione dell'errore non è sempre possibile da ottenere, mentre i metodi Monte Carlo e bootstrap non dipendono da una soluzione algebrica e si possono rapidamente applicare ad situazioni di approssimazioni più complicate come quello [iterativo non-lineare ai quadrati minimi](#), come si vedrà in seguito.

Effetto del numero dei punti sulla precisione dell'approssimazione con i minimi quadrati

Gli spreadsheet [EffectOfSampleSize.ods](#) o [EffectOfSampleSize.xlsx](#), che raccolgono i risultati di molte esecuzioni di [TestLinearFit.m](#) con un numero diverso di punti ("NumPoints"), dimostra che la deviazione standard della pendenza e dell'intercetta *diminuisce* se il numero di punti *aumenta*; in media, le *deviazioni standard sono inversamente proporzionali alla radice quadrata del numero di punti*, che è coerente con l'osservazione che la pendenza di un grafico log-log è all'incirca 1/2.

Questi grafici davvero drammatizzano il problema delle piccole dimensioni del campione, ma questo deve essere bilanciato con il costo per ottenere più punti. Ad esempio, nella calibrazione chimica analitica, è possibile ottenere un numero maggiore di punti di calibrazione preparando e misurando più soluzioni standard o leggendo ripetutamente ciascuna con un numero minore di standard. Il primo approccio tiene conto sia degli errori volumetrici nella preparazione delle soluzioni sia del rumore casuale nelle letture dello strumento, ma la manodopera e il costo per preparare e l'eseguire un gran numero di soluzioni standard e poi smaltirle in modo sicuro, sono limitanti. Quest'ultimo approccio è meno costoso ma è meno affidabile perché tiene conto solo del rumore casuale nelle letture dello

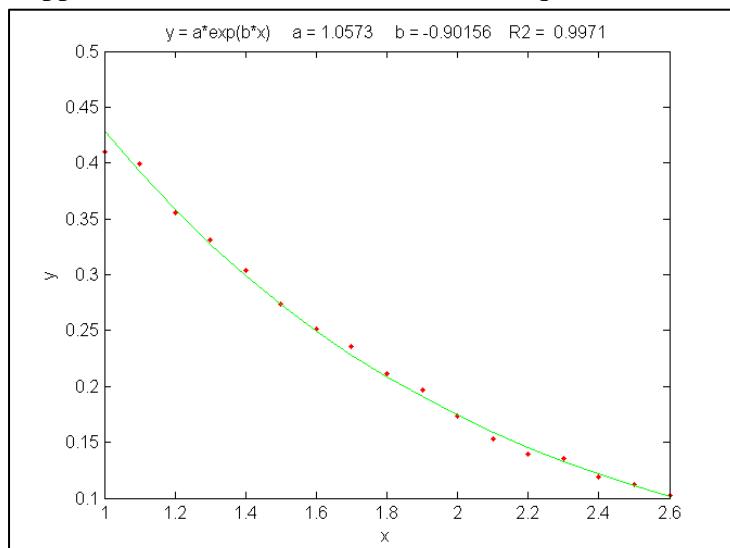


strumento. Nel complesso, è meglio affinare le tecniche di laboratorio e le impostazioni dello strumento per ridurre al minimo l'errore piuttosto che tentare di compensare effettuando molte letture.

Eseguire lo smoothing o no? Di solito è meglio che un segnale rumoroso non subisca lo [smoothing](#) prima dei calcoli dei minimi quadrati, perché così facendo non migliorerà l'affidabilità dei risultati dei minimi quadrati, ma farà sì che sia la propagazione algebrica degli errori che i calcoli bootstrap *sottostimino pesantemente* la deviazione standard dei risultati dei minimi quadrati. È possibile dimostrarlo con lo script [TestLinearFit.m](#) impostando SmoothWidth nella riga 10 a qualcosa di maggiore di 1, che eseguirà lo smoothing dei dati prima dei calcoli del "least-squares". Ciò non ha alcun effetto significativo sulla *effettiva* deviazione standard calcolata con il metodo Monte Carlo, ma riduce in modo significativo quella *prevista* calcolata dalla propagazione algebrica di della propagazione degli errori e (specialmente) col metodo bootstrap. Per ragioni simili, se il rumore è [rosa anziché bianco](#), anche le stime dell'errore col bootstrap saranno troppo basse. Al contrario, se il rumore è [blu](#), come avviene nei segnali elaborati che sono stati sottoposti a una sorta di processo di [differenziazione](#) o che sono stati [deconvoluti](#) da qualche processo di blurring, allora gli errori previsti dalla propagazione algebrica degli errori e dai metodi di bootstrap saranno *alti*. (Lo si può dimostrare in proprio eseguendo [TestLinearFit.m](#) con le modalità di rumore rosa e blu selezionate nelle righe 23 e 24). Conclusione: la previsione dell'errore funziona meglio per il rumore *bianco*.

Trasformare relazioni non-lineari

In qualche caso, una relazione fondamentalmente non lineare si può trasformare in una forma suscettibile di approssimazione della curva polinomiale mediante una trasformazione delle coordinate (ad esempio prendendo il logaritmo o il reciproco dei dati), per poi applicare il metodo dei minimi quadrati all'equazione lineare risultante. Per esempio, il segnale nella prossima figura proviene dalla simulazione di un decadimento esponenziale che ha la forma matematica $Y = a \exp(bX)$, dove X =tempo, Y =intensità del segnale, a è il valore di Y quando $X=0$ e b è il decadimento costante. Questo è un problema fondamentalmente non lineare perché Y è una funzione non-lineare del parametro b . Tuttavia, prendendo il logaritmo naturale da entrambi i lati dell'equazione, si ottiene $\ln(Y) = \ln(a) + bX$. In questa equazione, Y è una funzione *lineare* di entrambi i parametri $\ln(a)$ e b , quindi può essere approssimata dal metodo dei minimi quadrati per stimare $\ln(a)$ e b , da cui si ottiene a calcolando $\exp(\ln(a))$. In questo esempio, i valori "veri" dei coefficienti sono $a = 1$ e $b = -0.9$, ma a ciascun punto è stato aggiunto del rumore casuale, con una deviazione standard pari al 10% del valore di quel punto, al fine di simulare una misura sperimentale tipica in laboratorio. Una stima dei valori di $\ln(a)$ e b , dati solo i punti rumorosi, può essere determinata dall'approssimazione della curva ai minimi quadrati di $\ln(Y)$ rispetto a X .



L'applicazione di un'approssimazione esponenziale ai quadrati minimi (linea continua) a dei dati

*rumorosi (punti)
per stimare la costante di decadimento.*

L'equazione più adatta, mostrata dalla linea continua verde nella figura, è $Y = 0.959 \exp(-0.905 X)$, ovvero, $a = 0.959$ e $b = -0.905$, che sono ragionevolmente prossimi ai valori attesi di 1 e -0.9, rispettivamente. Pertanto, anche in presenza di un sostanziale rumore casuale (10% di deviazione standard relativa), è possibile ottenere stime ragionevoli dei parametri dell'equazione (entro il 4% circa). Il requisito più importante è che il modello deve essere buono, cioè che l'equazione selezionata per il modello descriva accuratamente il comportamento del sistema (eccetto per il rumore). Spesso questo è l'aspetto più difficile perché i modelli non sono sempre noti con certezza. In Matlab e in Octave, l'approssimazione si può eseguire con una sola riga di codice:
`polyfit(x, log(y), 1)`, che restituisce $[b \log(a)]$. (In Matlab e Octave, "log" è il logaritmo naturale, "log10" è il logaritmo in base 10).

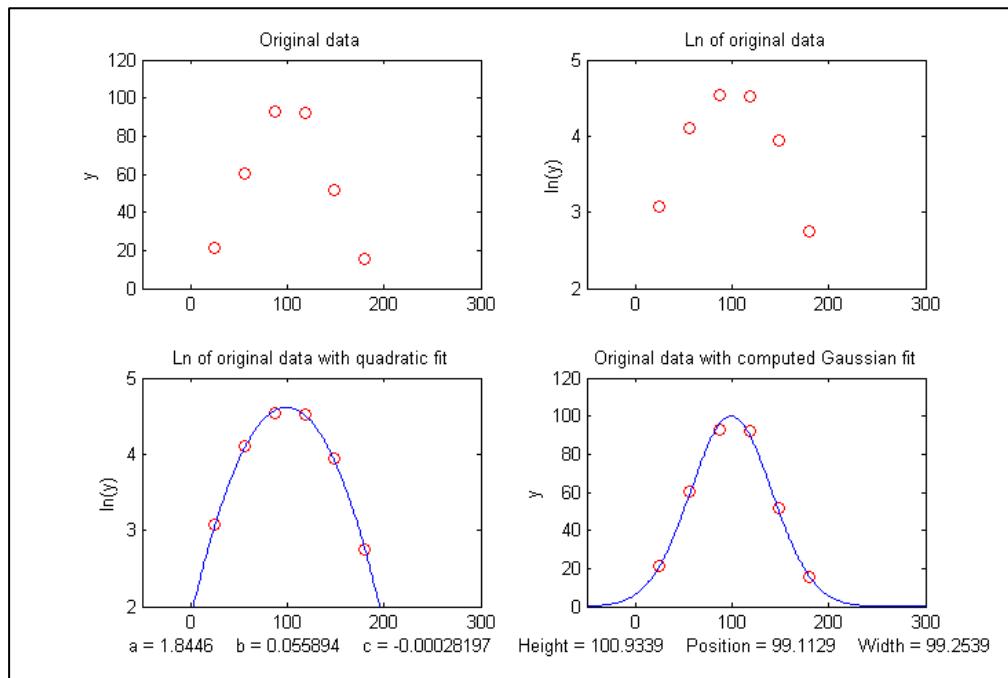
Un altro esempio di linearizzazione di una relazione esponenziale è esplorato a pagina 319: [Segnale e Rumore nel Mercato Azionario](#).

Altri esempi di relazioni non lineari ma linearizzabili mediante trasformazione delle coordinate includono le relazioni logaritmiche ($Y = a \ln(bX)$) e di elevamento a potenza ($Y = aX^b$). Metodi di questo tipo erano molto comuni ai tempi prima dei computer, quando era difficile approssimare qualcosa che non fosse una linea retta. È ancora utilizzato oggi per estendere la gamma di relazioni funzionali che possono essere gestite dalle comuni routine dei minimi quadrati lineari disponibili nei fogli di calcolo e nelle calcolatrici portatili. (La funzione Matlab/Octave [trydatatrans.m](#) prova otto diverse trasformazioni semplici di dati su un dato insieme di dati x,y e approssima i dati trasformati a una linea retta o a un polinomio). Tuttavia, solo poche relazioni non lineari possono essere gestite con una semplice trasformazione dei dati. Per approssimare *qualsiasi* funzione arbitraria, potrebbe essere necessario ricorrere al metodo di approssimazione *iterativo*, che verrà trattato in [Curve Fitting C](#).

Semplice approssimazione di picchi Gaussiani e Lorentziani con trasformazione dei dati

Un esempio interessante dell'uso della trasformazione per convertire una relazione non lineare in una forma suscettibile di approssimazione con una curva polinomiale è l'uso della trasformazione logaritmica naturale (\ln) per convertire un picco [Gaussiano](#) positivo, che ha la forma funzionale fondamentale $\exp(-x^2)$, in una parabola della forma $-x^2$, che è approssimabile con una funzione polinomiale del secondo ordine (quadratica) ($y = a + bx + cx^2$). L'equazione per un picco Gaussiano è $y = h * \exp(-(x-p) / (1/(2*sqrt(ln(2)))*w)^2)$, dove h è l'altezza del picco, p è la posizione sull'asse x del massimo, w è la larghezza intera a metà del massimo. Il logaritmo naturale di y [si può dimostrare che è](#) $\log(h) - (4 p^2 \log(2)) / w^2 + (8 p x \log(2)) / w^2 - (4 x^2 \log(2)) / w^2$, che è una forma quadratica nella variabile indipendente x perché la somma di x^2 , x , e i termini costanti. Esprimendo ciascuno dei parametri del picco h , p e w in termini dei tre coefficienti quadratici, [un po' di algebra](#) (per cortesia della [Wolfram Alpha](#)) mostrerà che tutti e tre i parametri del picco (altezza, posizione massima e larghezza) possono essere calcolati dai tre coefficienti quadratici a , b e c . L'altezza del picco è data da $\exp(a - c * (b / (2 * c))^2)$, la posizione è data da $-b / (2 * c)$ e la larghezza a metà altezza da $2.35482 / (\sqrt{2} * \sqrt{-c})$. Questo è chiamato "Algoritmo di Caruana"; [vedere Streamlining Digital Signal Processing: A "Tricks of the Trade" Guidebook, Richard G. Lyons, ed., pag. 298, dal nome di Rich Caruana, uno scienziato informatico e ricercatore americano](#). L'area sotto il picco Gaussiano può [anche essere calcolata ed è circainformativamente dimostrato che](#) $\approx 1.064467 * \text{altezzah} * \text{larghezzaw}$.

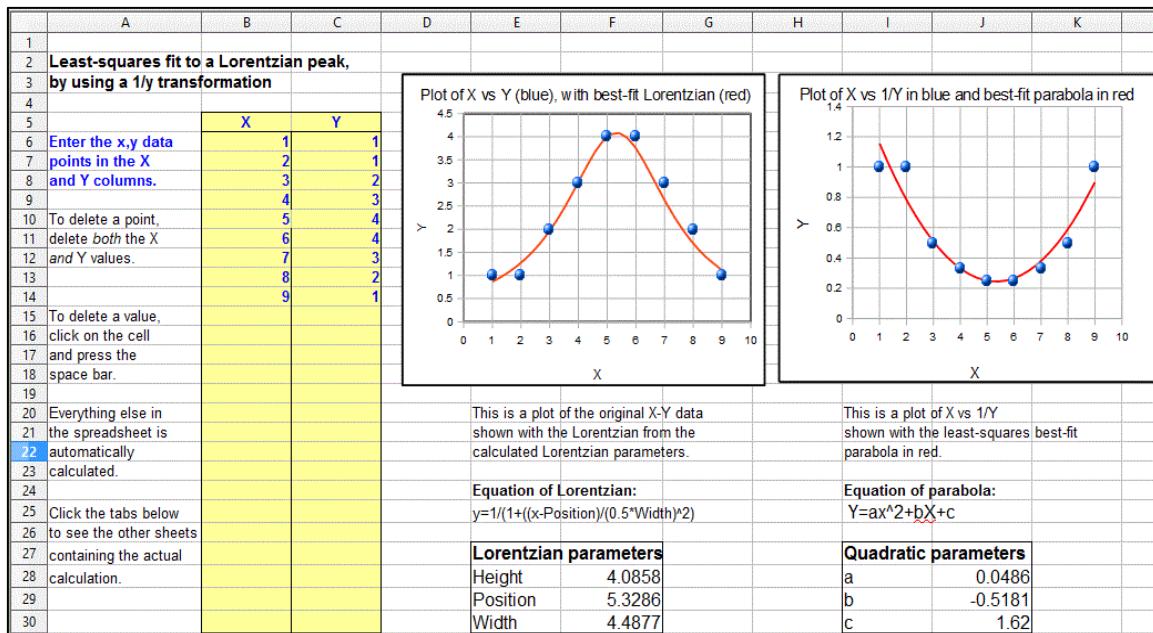
Un vantaggio di questo tipo di approssimazione della curva gaussiana, rispetto alla semplice stima visiva, è illustrato nella figura seguente. Il segnale è un picco Gaussiano sintetizzato con una altezza effettiva del picco di esattamente 100 unità, una posizione effettiva di 100 unità e una semi-



larghezza di 100 unità, ma è *scarsamente campionata* soltanto ogni 31 unità sull'asse x. Il [set di dati risultante](#), mostrato dai punti rossi in alto a sinistra, *ha solo 6 punti sul picco stesso*. Se dovessimo prendere il massimo di questi 6 punti (il 3° punto da sinistra, con $x=87$, $y=95$) come il massimo del picco, otterremmo solo un adattamento approssimativo ai valori reali della posizione (100) e dell'altezza (100) del picco. Se dovessimo prendere la distanza tra il 2° e il 5° punto come larghezza del picco, si otterrebbe solo $3 \times 31 = 93$, rispetto al valore reale di 100. Se si tentasse di calcolare l'*area* del picco da queste misure, si otterrebbe $1.064467 \times 95 \times 93 = 9404.6$, molto inferiore al valore teorico di $1.064467 \times \text{height} \times \text{width} = 10644.67$. Queste sono tutte stime molto scarse. *Tuttavia*, prendendo il *log naturale* dei dati (in alto a destra) si ottiene una *parabola* che può essere adattata con un'approssimazione dei minimi quadrati quadratica (mostrata dalla linea blu in basso a sinistra). Dai tre coefficienti dell'approssimazione quadratica, si possono calcolare valori molto più accurati dei parametri del picco Gaussiano, mostrato in basso nella figura: altezza=100.93; posizione=99.11; larghezza=99.25; area= 10663. Il grafico in basso a destra mostra l'approssimazione Gaussiana risultante (in blu) visualizzata con i dati originali (punti rossi). La precisione dei parametri di questo picco (circa l'1% in questo esempio) è limitata solo dal rumore nei dati; *molto più accurato, con un piccolo costo computazionale*.

La figura sopra è stata creata in Matlab (o Octave), usando [questo script](#). (La funzione Matlab/Octave [gaussfit.m](#) esegue il calcolo per un insieme di dati x,y. Si può anche scaricare uno spreadsheet che esegue gli stessi calcoli; è disponibile nei formati OpenOffice Calc ([link per il download](#), [Schermata](#)) ed [Excel](#)). Il metodo è semplice e molto veloce, ma affinché questo metodo funzioni correttamente, il set di dati *non deve contenere zeri o punti negativi*; se il rapporto segnale/rumore è molto scarso, può essere utile saltare quei punti o anteporre un leggero smoothing dei dati per ridurre questo problema. Inoltre, il segnale originale del picco Gaussiano deve essere un picco singolo isolato con una linea di base nulla, cioè deve tendere a zero lontano dal centro del picco. In pratica, ciò significa che qualsiasi linea di base diversa da zero deve essere sottratta dai dati prima di applicare questo metodo. (Un approccio più generale, ma più lento, all'approssimazione dei picchi Gaussiani, che funziona per i set di dati con zeri e numeri negativi e anche per i dati con più picchi sovrapposti, è il metodo di [curve fitting iterativo non-lineare](#), che verrà trattato in seguito, a pag. 190).

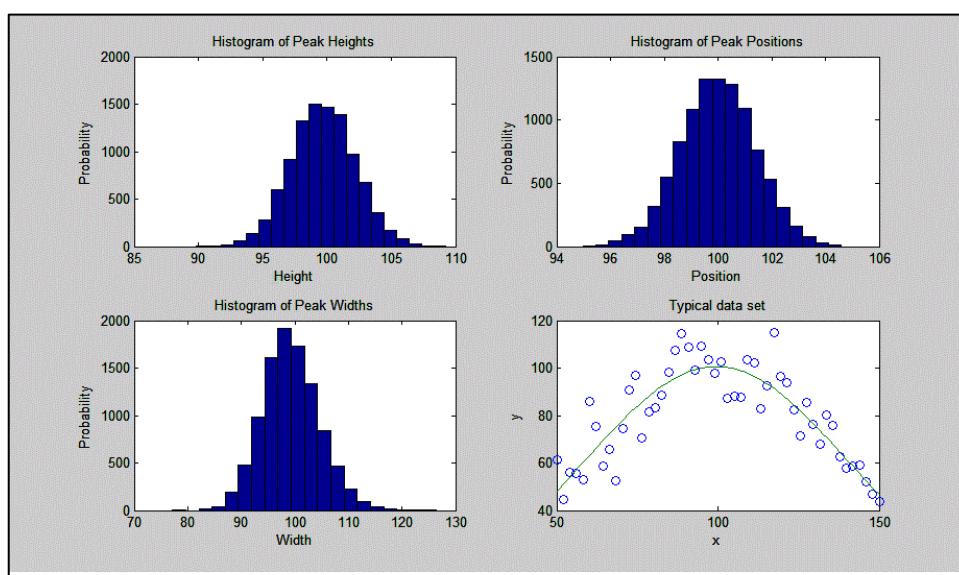
Un metodo simile può essere derivato per un picco Lorentziano, che ha la forma fondamentale $y=h/(1+((x-p)/(0.5*w))^2)$, adattando una quadratica al reciproco di y. Come per il picco Gaussiano, tutti e tre i parametri del picco (altezza **h**, posizione del massimo **p** e larghezza **w**) si possono calcolare dai tre coefficienti **a**, **b** e **c** dell'approssimazione quadratica: $h=4*a/(4*a*c-b^2)$, $p=-b/(2*a)$, and $w=\sqrt{((4*a*c)-b^2)/a}/\sqrt{a}$. Proprio come per il caso della Gaussiana, il set di dati non deve contenere alcun valore y zero o negativo. La funzione Matlab/Octave lorentzfit.m esegue il calcolo per un insieme di dati x,y, e gli spreadsheet Calc ed Excel LorentzianLeastSquares.ods e LorentzianLeastSquares.xls eseguono gli stessi calcoli (illustrati di seguito).



(A proposito, un modo rapido per testare uno dei metodi precedenti è usare questo *set minimale di dati di un picco*: $x=5, 20, 35$ e $y=5, 10, 5$, che ha altezza, posizione e larghezza pari a 10, 20 e 30, rispettivamente, per un singolo picco isolato asimmetrico di qualsiasi forma, assumendo solo una linea di base nulla). Da provare.

Per applicare i metodi di cui sopra a segnali contenenti *due o più* picchi Gaussiani o Lorentziani, è necessario prima individuare tutti i massimi dei picchi, in modo che possano adeguatamente elaborare i gruppi di punti centrati su ciascun picco con gli algoritmi appena discussi. Questo viene discusso a pagina 227.

Tuttavia, c'è uno svantaggio nell'usare metodi di trasformazione delle coordinate per convertire relazioni non lineari in una semplice forma polinomiale, e cioè che anche il rumore è influenzato dalla trasformazione, con il risultato che la propagazione dell'errore dai dati originali ai risultati



finali è spesso difficile da prevedere. Ad esempio, nel metodo appena descritto per misurare l'altezza, la posizione e la larghezza dei picchi Gaussiani o Lorentziani, i risultati dipendono non solo dall'ampiezza del rumore nel segnale, ma anche da quanti punti attraverso il picco vengono presi per l'approssimazione. Nel prendere più punti lontano dal centro del picco, dove i valori y si avvicinano a zero, il logaritmo naturale di questi punti tende all'infinito negativo all'avvicinarsi di y a zero. Il risultato è che il rumore di quei punti di bassa ampiezza viene eccessivamente amplificato ed ha un effetto sproporzionato sull'approssimazione della curva. Ciò è in contrasto con la solita aspettativa che la qualità dei parametri derivati dall'approssimazione, migliora con la radice quadrata del numero di punti ([pagina 214](#)). Un ragionevole compromesso in questo caso è prendere solo i punti nella metà superiore del picco, con i valori Y fino alla metà del massimo del picco. In tal caso, la propagazione dell'errore (prevista da una [simulazione Monte Carlo](#) con rumore casuale costante normalmente distribuito) mostra che le deviazioni standard relative dei parametri del picco misurati sono direttamente proporzionali al rumore nei dati e inversamente proporzionali alla radice quadrata del numero di punti dati (come previsto), ma che le costanti di proporzionalità differiscono:

- (a) la deviazione standard relativa dell'altezza del picco = $1.73 * noise / \sqrt{N}$,
- (b) la deviazione standard relativa della posizione del picco = $noise / \sqrt{N}$,
- (c) la deviazione standard relativa dell'ampiezza del picco = $3.62 * noise / \sqrt{N}$,

dove *noise* è la deviazione standard del rumore nei dati e *N* è il numero dei punti presi per l'approssimazione ai quadrati minimi. Da questi risultati si vede che la misura della *posizione* del picco è più precisa, seguita dall'*altezza* mentre la *larghezza* è la meno precisa. Se si includessero punti lontani dal massimo del picco, in cui il rapporto segnale/rumore è molto basso, i risultati sarebbero peggiori del previsto. Queste previsioni dipendono dalla conoscenza del rumore nel segnale; se è disponibile per la misura un solo campione di quel rumore, non vi è alcuna garanzia che il campione sia rappresentativo, specialmente se il numero totale di punti nel segnale misurato è piccolo; la deviazione standard di campioni piccoli è notoriamente variabile. Inoltre, queste previsioni si basano su una simulazione con rumore *bianco costante normalmente distribuito*; se si variasse il rumore effettivo col livello del segnale o col valore dell'asse x, o se la distribuzione della probabilità fosse stata qualcosa di diverso dal normale, quelle previsioni non sarebbero state accurate. In questi casi, il [metodo bootstrap](#) (pagina 162) ha il vantaggio di campionare il rumore effettivo nel segnale.

Il codice Matlab/Octave per questa simulazione Monte Carlo, si può scaricare da <http://terpconnect.umd.edu/~toh/spectrum/GaussFitMC.m>; visualizza [schermata](#). Una simulazione simile (<http://terpconnect.umd.edu/~toh/spectrum/GaussFitMC2.m>, visualizza la [schermata](#)) confronta questo metodo con l'approssimazione dell'intero picco Gaussiano col metodo iterativo in [Curve Fitting 3](#), trovando che la precisione dei risultati è solo leggermente migliore col metodo iterativo (più lento).

Nota 1: Se si sta leggendo online, con un click-destro su qualsiasi link dei file .m precedenti e poi si seleziona “**Save Link As...**” per scaricarli localmente ed usarli in Matlab/Octave.

Nota 2: Nelle tecniche di approssimazione della curva descritte qui e nei prossimi due capitoli, non è richiesto che l'intervallo dell'asse x tra i punti sia uniforme, come si presume in molte altre tecniche di elaborazione del segnale descritte in precedenza. Gli algoritmi di curve fitting accettano tipicamente un insieme di valori arbitrariamente spaziati sull'asse x e un corrispondente insieme per l'asse y.

Dettagli matematici e software per i quadrati minimi lineare

L'approssimazione ai quadrati minimi per un set di dati x,y si può calcolare utilizzando solo l'aritmetica di base. Di seguito sono riportate le equazioni rilevanti per il calcolo della pendenza e dell'intercetta dell'equazione di approssimazione del primo ordine, $y = \text{intercetta} + \text{pendenza} \cdot x$, nonché la deviazione standard prevista della pendenza e dell'intercetta e il coefficiente di determinazione, R^2 , che è un indicatore della "bontà dell'approssimazione". (R^2 è 1.0000 se l'approssimazione è perfetta altrimenti è inferiore).

```
n = numero dei punti x,y  
sumx = Σx  
sumy = Σy  
sumxy = Σx*y  
sumx2 = Σx*x  
meanx = sumx / n  
meany = sumy / n  
slope = (n*sumxy - sumx*sumy) / (n*sumx2 - sumx*sumx)  
intercept = meany-(slope*meanx)  
ssy = Σ(y-meany)^2  
ssr = Σ(y-intercept-slope*x)^2  
R2 = 1-(ssr/ssy)  
Deviazione standard della pendenza = SQRT(ssr/(n-2))*SQRT(n/(n*sumx2 - sumx*sumx))  
Deviazione standard dell'intercetta = SQRT(ssr/(n-2))*SQRT(sumx2/(n*sumx2 - sumx*sumx))
```

(In queste equazioni, Σ rappresenta la sommatoria; per esempio, Σx significa la somma di tutti i valori di x , e $\Sigma x*y$ indica la somma di tutti i prodotti $x*y$, ecc.)

Le ultime due righe prevedono la deviazione standard della pendenza e dell'intercetta basandosi solo su quel campione di dati, assumendo che le deviazioni dalla linea siano casuali e normalmente distribuite. Queste sono stime della variabilità delle pendenze e delle intercette che è probabile che si ottengano se si ripetono le misurazioni dei dati più e più volte nelle stesse condizioni, assumendo che le deviazioni dalla linea retta siano dovute a una *variabilità casuale* e non un errore sistematico causato dalla non linearità. Se le deviazioni sono casuali, di volta in volta saranno leggermente diverse, facendo variare la pendenza e l'intercetta da una misura all'altra, con una deviazione standard prevista da queste ultime due equazioni. Tuttavia, se le deviazioni sono causate da una non linearità sistematica, saranno le stesse da misura a misura, nel qual caso la previsione di queste ultime due equazioni non sarà rilevante, e si potrebbe fare meglio usando un'approssimazione polinomiale come una quadratica o una cubica.

L'affidabilità di queste stime della deviazione standard dipende dall'assunzione di deviazioni casuali e dal numero di punti nell'approssimazione della curva; migliorano con la radice quadrata del numero di punti. Si può scrivere un [insieme leggermente più complesso di equazioni](#) per approssimare un'equazione del secondo ordine (quadratico o parabolico) ad un insieme di dati; invece di una pendenza e di una intercetta, vengono calcolati tre coefficienti, **a**, **b** e **c**, che rappresentano i coefficienti dell'equazione quadratico **ax²+bx+c**.

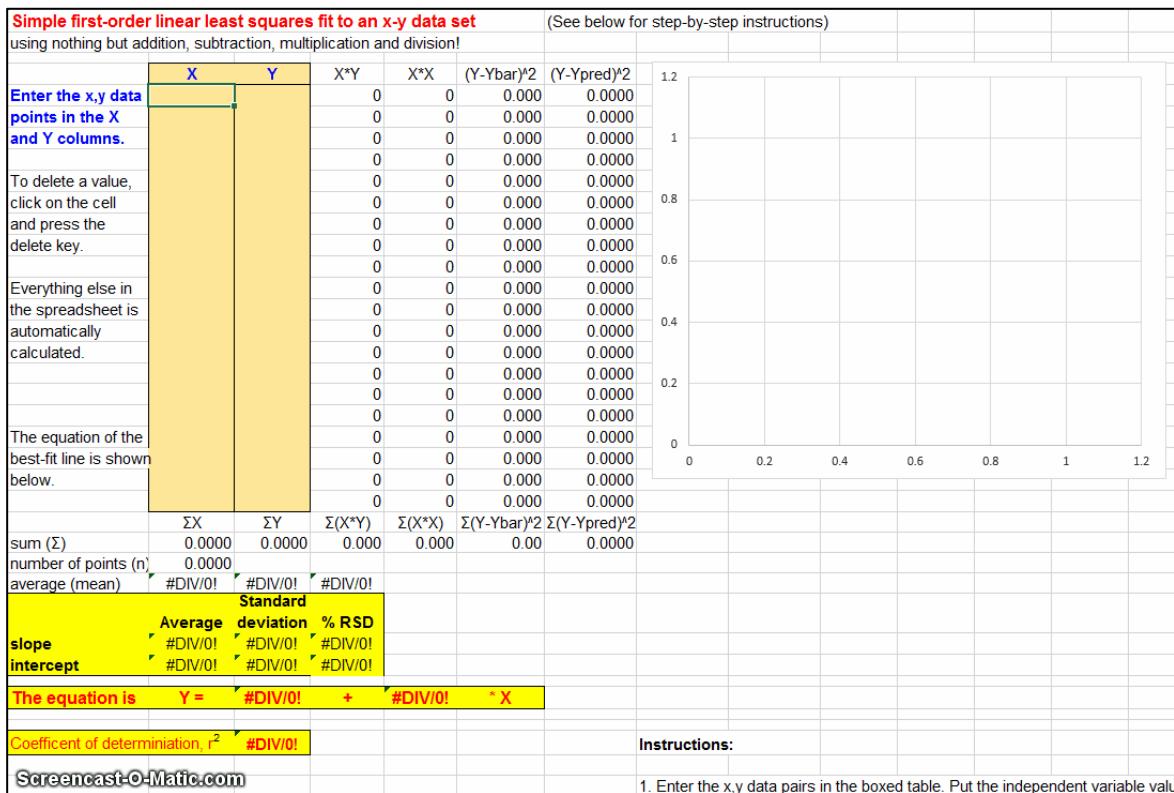
Questi calcoli si possono eseguire manualmente passo passo, con l'aiuto di una calcolatrice o di uno spreadsheet, con un [programma](#) scritto in qualsiasi linguaggio, come uno [script Matlab o Octave](#).

Siti Web: [Wolfram Alpha](#) include alcune funzionalità per [l'analisi di regressione](#) ai quadrati minimi, incluse approssimazioni lineari, polinomiali, esponenziali e logaritmiche. [Statpages.org](#) può

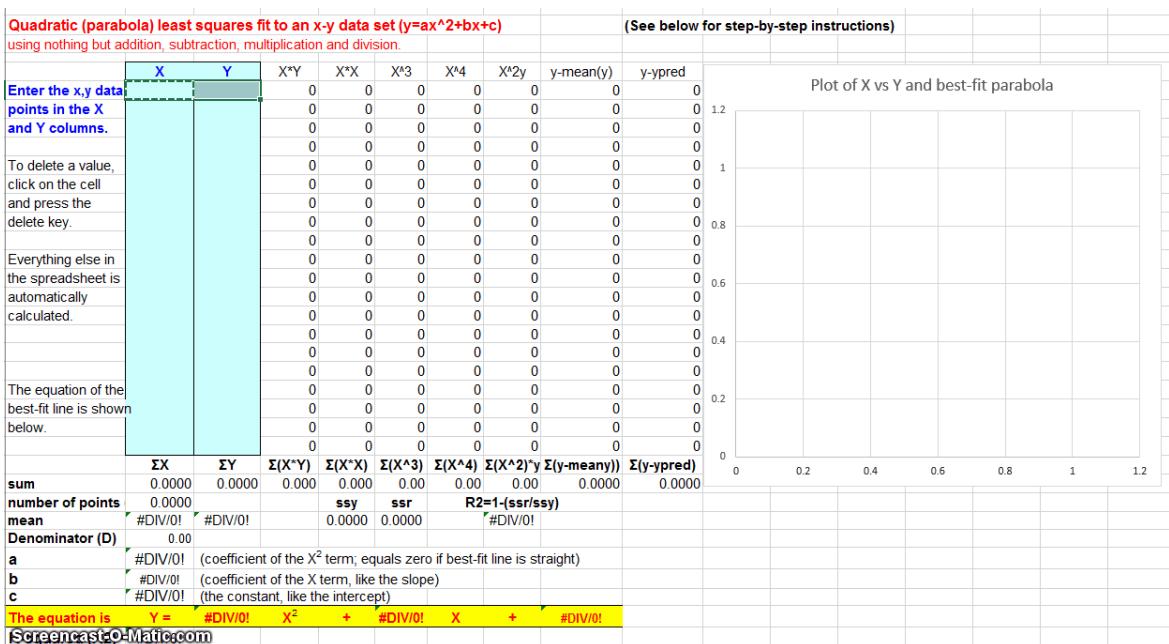
eseguire una vasta gamma di calcoli statistici e test, e ci sono diversi siti Web specializzati nel disegno e nella visualizzazione dei dati con la capacità di eseguire il curve-fitting, tra cui in particolare [Plotly](#) e [MyCurveFit](#). Siti Web come questi possono essere molto utili quando si lavora da uno smartphone, un tablet o un computer che non dispone di un software di calcolo adeguato. Se si sta leggendo questo online, si può fare **Ctrl-Click** su questi link per aprirli nel browser.

Spreadsheet per i quadrati minimi lineari

Gli spreadsheet possono eseguire facilmente i calcoli descritti in precedenza. I seguenti fogli di calcolo ([LeastSquares.xls](#) e [LeastSquares.odt](#) per l'approssimazione lineare e [QuadraticLeastSquares.xls](#) e [QuadraticLeastSquares.ods](#) per quelle quadratiche), utilizzano le espressioni di cui sopra per calcolare e disegnare rispettivamente l'approssimazione ai minimi quadrati lineare e quella quadratica (parabola). (Visualizzate in Word 365 o sul Web, queste animazioni mostrano il risultato dell'inserimento di un piccolo insieme di dati punto per punto). Il vantaggio dei fogli di calcolo è che sono molto personalizzabili per la propria applicazione e possono essere distribuiti su dispositivi mobili come tablet e smartphone. Per approssimazioni di una linea retta, si possono usare le funzioni native *slope* e *intercept*.



Animazione dell'immissione dei dati in un template per l'approssimazione ai quadrati minimi del primo ordine (lineare) (<https://terpconnect.umd.edu/~toh/spectrum/LeastSquares.GIF>)



Animazione dell'immissione dei dati in un template per l'approssimazione ai quadrati minimi del secondo ordine (quadratica).

(<https://terpconnect.umd.edu/~toh/spectrum/QuadraticLeastSquares.GIF>)

La funzione LINEST. I fogli di calcolo moderni hanno anche funzionalità *native* per calcolare approssimazioni ai quadrati minimi polinomiali di *qualsiasi* ordine. Per esempio, si può usare la funzione LINEST sia in [Excel](#) che in [OpenOffice Calc](#) per calcolare polinomiali e altre approssimazioni ai quadrati minimi a curvilinee. Oltre ai coefficienti polinomiali per l'approssimazione migliore, la funzione LINEST calcola contemporaneamente anche i valori dell'errore standard, il coefficiente di determinazione (R^2), il valore dell'errore standard per la y stimata, la statistica F, il numero di gradi di libertà, somma dei quadrati di regressione e la somma dei quadrati dei residui. Un inconveniente significativo di LINEST, rispetto all'elaborazione della matematica utilizzando la serie di espressioni matematiche descritte sopra, è che è più difficile adattarsi a un numero variabile di punti dati e rimuovere punti dati sospetti o cambiare l'ordine del polinomio. LINEST è una *funzione di matrice*, il che significa che quando si inserisce la formula in una cella, verranno utilizzate più celle per l'output della funzione. *Non si può modificare una funzione LINEST come qualsiasi altra funzione dello spreadsheet*. Per specificare che LINEST è una funzione di matrice, eseguire le operazioni seguenti. Evidenziare l'intera formula, incluso il segno " $=$ ". Su un Macintosh, si tiene premuto il tasto "mela" e si preme "**Enter**". Su un PC si tengono premuti i tasti "Ctrl" e "Shift" e si preme "**Enter**". Excel aggiunge delle parentesi " $\{ \}$ " attorno alla formula, per mostrare che si tratta di una matrice. Notare che non è possibile immettere manualmente i caratteri " $\{ \}$ "; se lo si fa, Excel tratterà il contenuto della cella come caratteri e non come una formula. *Evidenziare la formula completa e premere il tasto "mela" o "Ctrl", "Maiusc" e "Invio" è l'unico modo per inserire una formula matriciale*. Queste istruzioni del Colby College possono aiutare: <http://www.colby.edu/chemistry/PChem/notes/linest.pdf>.

Nota pratica: se si sta lavorando con un template che utilizza la funzione LINEST e si desidera modificare il numero di punti, il modo più semplice per farlo è selezionare le righe o le colonne contenenti i dati, click desto sull'*intestazione* della riga o della colonna (1,2,3 o A, B, C, ecc.) ed usare **Insert** o **Delete** nel menù del click-destro. Se lo si fa in questo modo, verrà aggiustata la funzione LINEST per far riferimento *automaticamente* alle righe o alle colonne. È più facile che provare a modificare direttamente la funzione LINEST. (Se si stanno inserendo le righe o le colonne, si deve trascinare-copiare le formule dalle vecchie righe o colonne in quelle vuote appena inserite).

Si veda [CalibrationCubic5Points.xls](#) per un esempio.

Applicazioni per la calibrazione analitica e la misura

Ci sono specifiche versioni di questi spreadsheet che applicano il 'curve fitting' alle curve di calibrazione (creando grafici delle misure del segnale rispetto agli standard noti della concentrazione) e che calcolano anche le concentrazioni dei campioni ignoti (l'intero set è scaricabile come [CalibrationSpreadsheets.zip](#)). Naturalmente, questi fogli di calcolo possono essere utilizzati per quasi tutte le applicazioni di calibrazione delle misure; è sufficiente modificare le etichette delle colonne e degli assi per adattarli alla propria applicazione. Una applicazione tipica di questi modelli di fogli di calcolo all'analisi XRF (fluorescenza a raggi X) è mostrata in questo video su YouTube: <https://www.youtube.com/watch?v=U3kzgVz4HgQ>

Un altro [set di spreadsheet](#) viene usato per eseguire le [simulazioni Monte Carlo](#) del processo di calibrazione e di misurazione utilizzando diversi metodi di calibrazione analitici ampiamente utilizzati, inclusi i minimi quadrati del primo ordine (linea retta) e del secondo ordine (linea curva). Sono inclusi i tipici errori sistematici e casuali sia nelle misure del segnale che in quelle volumetriche, allo scopo di dimostrare come la non-linearità, le interferenze ed gli errori casuali si combinano per influenzare il risultato (la cosiddetta "propagazione degli errori").

Per l'approssimazione dei *picchi*, [GaussianLeastSquares.odt](#), è uno spreadsheet OpenOffice che approssima una funzione quadratica al logaritmo naturale di $y(x)$ e calcola l'altezza, la posizione e la larghezza della Gaussiana che approssima meglio $y(x)$. C'è anche una versione Excel ([GaussianLeastSquares.xls](#)). [LorentzianLeastSquares.ods](#) e [LorentzianLeastSquares.xls](#) approssimano una funzione quadratica al reciproco di $y(x)$ e calcola l'altezza, la posizione e la larghezza della Lorentziana che approssima meglio $y(x)$. Si noti che per nessuna di queste approssimazioni, i dati possono contenere punti a zero o negativi e la linea di base (il valore di y molto lontano dal centro del picco) dev'essere zero. Si veda il [Fitting Peaks](#), precedente.

Matlab e Octave

[Matlab](#) e [Octave](#) hanno semplici funzioni native per l'approssimazione ai quadrati minimi: [polyfit](#) e [polyval](#). Per esempio, se si ha un insieme di punti x,y nei vettori "x" e "y", allora i coefficienti per l'approssimazione ai quadrati minimi saranno dati da `coef=polyfit(x,y,n)`, dove "n" è l'ordine del polinomio di approssimazione: $n = 1$ per una linea retta, 2 per una quadratica (parabola), ecc. I coefficienti polinomiali 'coef' vengono dati in ordine di potenza decrescente di x. Per un'approssimazione con una retta ($n=1$), `coef(1)` è la pendenza ("b") e `coef(2)` è l'intercetta ("a"). Per un'approssimazione quadratica ($n=2$), `coef(1)` è il termine di x^2 ("c"), `coef(2)` è il termine di x ("b") e `coef(3)` è il termine costante ("a").

L'equazione di approssimazione si può valutare con la funzione [polyval](#), per esempio, `fity=polyval(coef,x)`. Questo funziona per qualsiasi ordine ("n") del polinomio di approssimazione. Si possono disegnare i dati assieme all'equazione approssimata con la funzione `plot: plot(x,y,'ob',x,polyval(coef,x),'-r')`, che disegna i dati come cerchi blu e l'equazione come una linea rossa. Si possono disegnare i residui scrivendo `plot(x,y-polyval(coef,x))`.

Quando il numero dei punti è piccolo, si potrebbe notare che la curva approssimata viene visualizzata come una serie di segmenti rettilinei, il che può apparire brutto. Si può ottenere un grafico più uniforme dell'equazione dell'approssimazione, valutata con divisioni più fini della x, definendo `xx=linspace(min(x),max(x))`; e poi utilizzando xx anziché x per valutare e disegnare l'approssimazione: `plot(x,y,'ob',xx,polyval(coef,xx),'-r')`.

`[coef,S] = polyfit(x,y,n)` restituisce i coefficienti polinomiali coef e una struttura [struttura](#)

'S' utilizzata per ottenere le [stime degli errori](#).

```
>> [coef,S]=polyfit(x,y,1)
coef =
1.4913 6.5552
S =
R: [2x2 double]
df: 2
normr: 2.2341
>> S.R
ans =
-18.4391 -1.6270
0 -1.1632
```

Il vettore delle deviazioni standard dei coefficienti [si può calcolare da S con l'espressione \$\text{sqrt}\(\text{diag}\(\text{inv}\(S.R\) * \text{inv}\(S.R'\)\) . * S.normr.^2 ./ S.df\)\$](#) ', nello stesso ordine dei coefficienti.

Il Metodo Matriciale

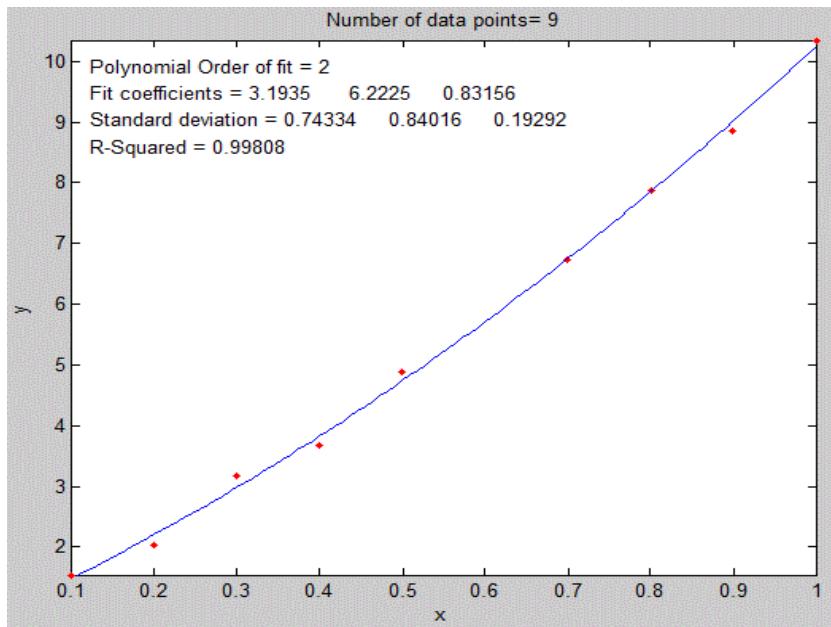
In alternativa, è possibile eseguire i calcoli della polinomiali ai quadrati minimi per i vettori riga x,y *senza* usare la funzione nativa polyfit di Matlab/Octave utilizzando il [metodo matriciale](#) col simbolo "/" di Matlab, che significa "divisione matriciale destra". I coefficienti di una approssimazione del primo ordine sono dati da $y / [x; ones(size(y))]$ e di una del secondo ordine (quadratica) da $y / [x.^2; x; ones(size(y))]$. Per polinomi di ordine superiore, basta aggiungere un'altra riga alla matrice del denominatore, per esempio, un'approssimazione del terzo ordine sarà $y / [x.^3; x.^2; x; ones(size(y))]$ e così via. I coefficienti vengono restituiti nello stesso ordine di polyfit, in potenze decrescenti di x (ad esempio, per un'approssimazione del primo ordine, prima la *pendenza* (il termine di x^1) poi l'*intercetta* (il termine di x^0). Utilizzando l'esempio dell'approssimazione del primo ordine ai prezzi delle schede di memoria SD:

```
>>x=[2 4 8 16];
>> y=[9.99 10.99 19.99 29.99];
>> polyfit(x,y,1)
ans =
1.4913 6.5552
>> y/[x;ones(size(y))]
ans =
1.4913 6.5552
```

Questo mostra che i risultati della *pendenza* e *dell'intercetta per la funzione polyfit e per il metodo matriciale sono gli stessi*. (I risultati della pendenza e dell'intercetta possono essere gli stessi, ma la funzione polyfit ha il vantaggio di poter anche calcolare le *stime degli errori* con un piccolo sforzo aggiuntivo, come descritto in precedenza, pagina 159).

La funzione plotit.m

il grafico seguente è stato generato dalla funzione Matlab/Octave [plotit.m](#), nella forma plotit(data) o



`plotit(data,polyorder)`. Questa funzione usa *tutte le tecniche citate nel precedente paragrafo*. Accetta 'dati' sotto forma di un singolo vettore, o una coppia di vettori "x" e "y", o una matrice 2xn o nx2 con x nella prima riga o colonna e y nella seconda, e disegna i punti in rosso. Se viene passato l'argomento di input opzionale "polyorder", `plotit` approssima un polinomio di ordine "polyorder" ai dati e disegna l'approssimazione come una linea blu e visualizza i

coefficienti e la misura della bontà dell'approssimazione R^2 nell'angolo superiore sinistro del grafico.

Nella prossima pagina c'è un esempio Matlab/Octave sull'uso di `plotit.m` per eseguire la trasformazione delle coordinate descritta, a pagina 164, per approssimare una relazione esponenziale, che mostra sia i dati esponenziali originali che quelli trasformati con un'approssimazione lineare nelle finestre [figure\(2\)](#) e [figure\(1\)](#), rispettivamente. Se si sta leggendo online, [click per il download](#).

```
x=1:.1:2.6;
a=1;
b=-.9;
y=a.*exp(b.*x);
y=y+y.*1.*rand(size(x));
figure(1)
[coeff,R2]=plotit(x,log(y),1);
ylabel('ln(y)');
title('Plot of x vs the natural log (ln) of y')
aa=exp(coeff(2));
bb=coeff(1);
yy= aa.*exp(bb.*x);
figure(2)
plot(x,y,'r.',x,yy,'g')
xlabel('x');
ylabel('y');
title(['y = a*exp(b*x) a = ' num2str(aa) ' b = ' num2str(bb) ' R2 = ' num2str(R2) ] );
```

Nella versione 5 o 6 la sintassi di `plotit` può essere `[coef, RSquared, StdDevs] =plotit(x,y,n)`. Restituisce i coefficienti dell'approssimazione '`coeff`', in potenze decrescenti di x, le deviazioni standard di quei coefficienti '`StdDevs`' nello stesso ordine e il valore di R-squared. Per calcolare le deviazioni standard *relative*, basta digitare `StdDevs./coef`. Per esempio,

lo script seguente calcola una linea retta con cinque punti, una pendenza di 10, una intercetta di zero e rumore pari a 1.0. Usa poi plotit.m per disegnare e approssimare i dati ad un modello lineare del primo ordine (linea retta) e calcola la deviazione standard stimata della pendenza e dell'intercetta, eseguendolo ripetutamente, si osserverà che la pendenza e l'intercetta misurate sono di solito entro due deviazioni standard rispettivamente di 10 e zero. Da provare con diversi valori di "Noise".

```
NumPoints=5;
slope=10;
Noise=1;
x=round(10.*rand(size(1:NumPoints)));
y=slope*x+Noise.*randn(size(x));
[coef,RSquared,StdDevs]=plotit(x,y,1)
```

Confrontare due insiemi di dati. Plotit è utilizzabile anche per confrontare due diversi vettori di variabili dipendenti (ad esempio y_1 e y_2) se *condividono le stesse variabili indipendenti* x , ad esempio per determinare la somiglianza di due diversi spettri misurati sulle stesse lunghezze d'onda come è stato fatto a pagina 12: **[coeff,R2]=plotit(y1,y2,1);**

R^2 è una misura della somiglianza. Più R^2 è prossimo a 1.000, più sono simili. Se y_1 e y_2 sono due misurazioni dello *stesso* segnale con un diverso rumore casuale, il grafico mostrerà una dispersione casuale di punti lungo una linea retta con una pendenza, $\text{coeff}(1)$, di 1.00. Se y_1 e y_2 sono lo stesso segnale con ampiezze diverse, la pendenza della linea sarà uguale al loro rapporto medio. Se i punti sono curvi e si ripetono, la differenza tra i due vettori y è maggiore del rumore casuale.

La sintassi può essere facoltativamente **plotit(x,y,n,datastyle,fitstyle)**, dove **datastyle** e **fitstyle** sono stringhe opzionali che specificano lo stile e il colore della linea e del simbolo, nella convenzione standard di Matlab. Le stringhe, tra virgolette singole, sono composte da un elemento di una o tutte le seguenti 3 colonne:

```
b blu . punto - solido
g verde o cerchio : punteggiato
r rosso x segno-x -. trattino-punto
c ciano + più -- tratteggiato
m magenta * asterisco (none) nessuna linea
y giallo s quadrato
k nero d rombo
w bianco v triangolo (giù)
^ triangolo (sù)
< triangolo (sinistra)
> triangolo (destra)
p pentagramma
h esagramma
```

Per esempio, **plotit(x,y,3,'or','-g')** disegna i dati come cerchi rossi e l'approssimazione come una linea continua verde (il valore predefinito è punti rossi e un linea blu, rispettivamente).

Si può usare plotit.m in Matlab per linearizzare e disegnare altre [relazioni non-lineari](#), come:

```

y = a exp(bx) : [coeff,R2]=plotit(x,log(y),1); f=exp(coeff(2)); b=coeff(1);
y = a ln(bx): [coeff,R2]=plotit(log(x),y,1); a=coeff(1); b=log(coeff(2));
y=axb: [coeff,R2]=plotit(log(x),log(y),1); a=exp(coeff(2)); b=coeff(1);
y=start(1+rate)x: [coeff,R2]=plotit(x,log(y),1); start=exp(coeff(2));
rate=exp(coeff(1))-1;

```

Quest'ultima è l'espressione per l'*interesse composto*, trattato a pagina 319: [Segnale e Rumore nel Mercato Azionario](#).

Non si dimentichi che in Matlab/Octave, "log" significa *logaritmo naturale*; il logaritmo in *base-10* è indicato con "log10".

Stima degli errori dei coefficienti. La funzione [plotit](#) ha anche una [routine bootstrap](#) che calcola l'errore del coefficiente col metodo bootstrap (deviazione standard STD e deviazione standard relativa RSD) e restituisce i risultati nella matrice "BootResults" (di dimensioni 5 x polyorder+1). È possibile modificare il numero di campioni di bootstrap nella riga 101. Il calcolo viene attivato includendo un 4^o argomento di *output*, p.es.

```
[coef,RSquared,StdDevs, BootResults] = plotit(x,y,polyorder).
```

Questo vale per qualsiasi ordine della polinomiale. Per esempio:

```

>> x=0:100;
>> y=100+(x*100)+100.*randn(size(x));
>> [FitResults, GOF, baseline, coeff, residual, xi, yi, BootResults] =
plotit(x,y,1);

```

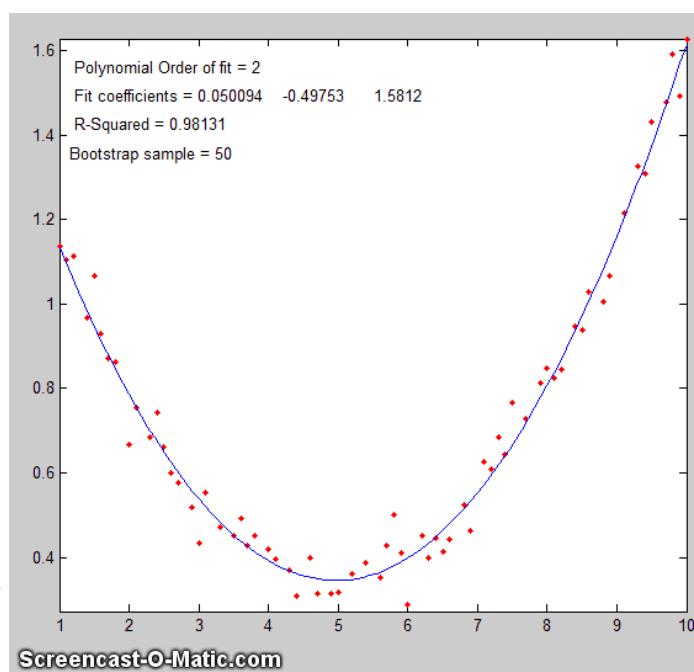
Le istruzioni precedenti calcolano una linea retta con una intercetta e una pendenza di 100, più del rumore casuale con una deviazione standard di 100, quindi approssima una retta a quei dati e stampa una tabella di stime dell'errore di bootstrap, con la pendenza nella prima colonna e l'intercetta nella seconda:

```

Bootstrap Results
Mean: 100.359  88.01638
STD:  0.204564 15.4803
STD (IQR):  0.291484 20.5882
% RSD:      0.203832 17.5879
% RSD (IQR): 0.290441 23.3914

```

La variazione [plotfita](#) anima il processo di bootstrap per scopi didattici, [come mostrato nell'animazione a lato](#) un'approssimazione quadratica. È necessario includere gli argomenti di output, ad esempio:



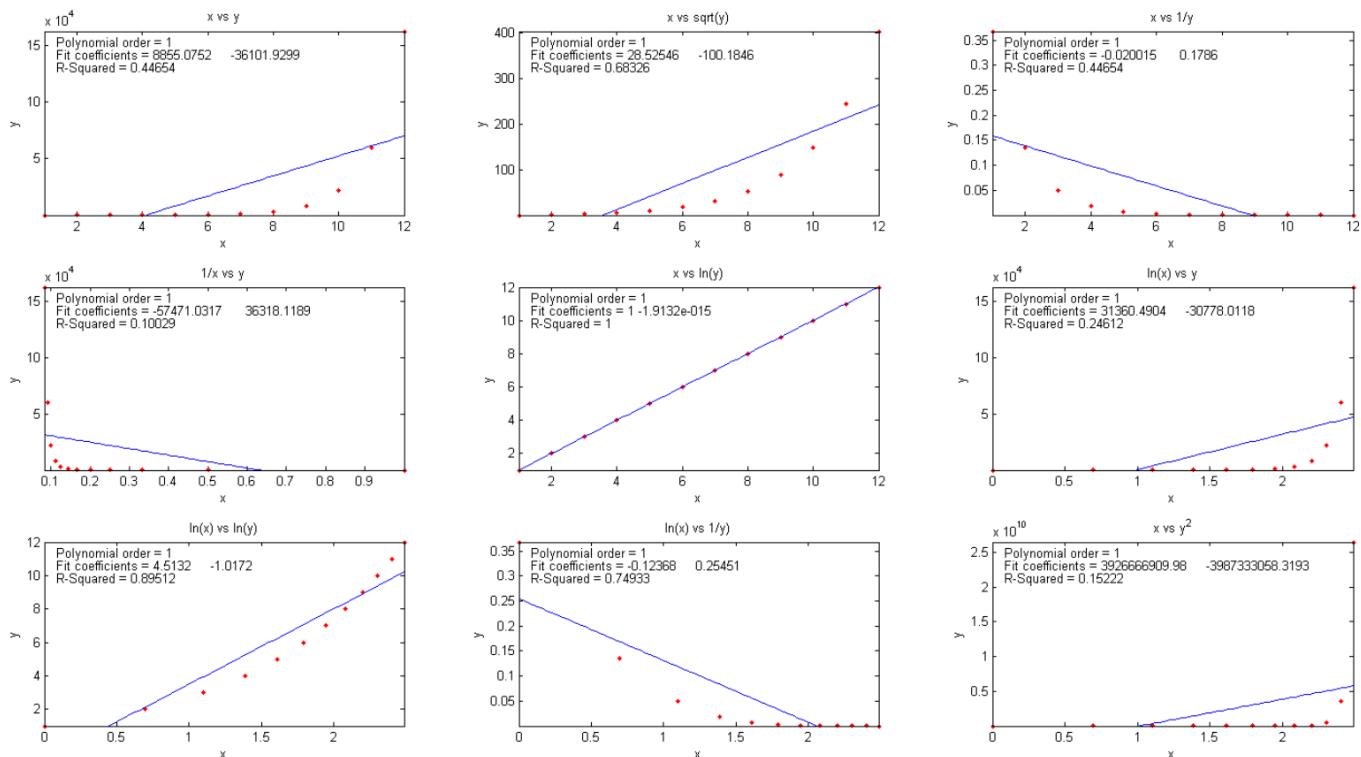
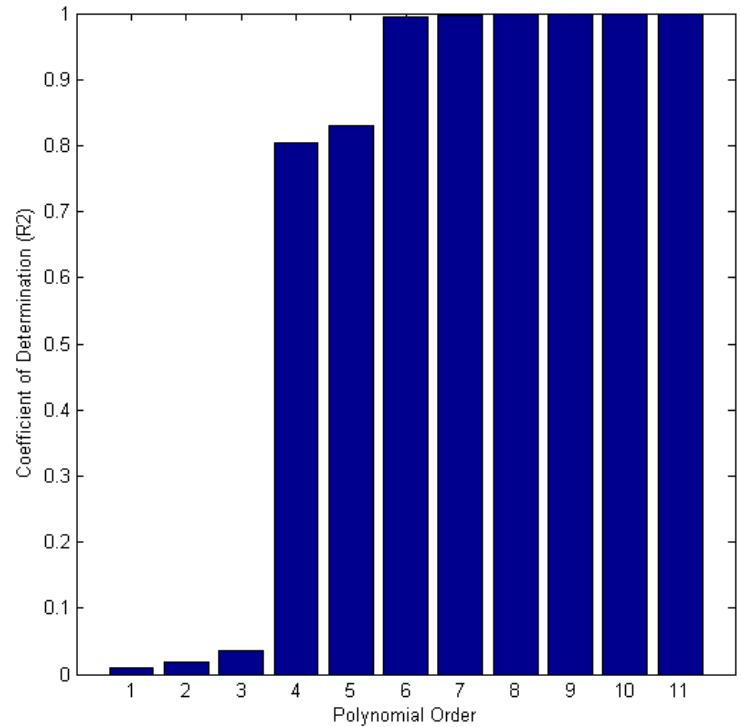
```
[coef, RSquared, BootResults] = plotfita([1 2 3 4 5 6],[1 3 4 3 2 1],2);
```

La variazione [logplotfit](#) disegna e adatta $\log(x)$ rispetto a $\log(y)$, per i dati che seguono una [relazione di legge di potenza](#) o che coprono un intervallo numerico molto ampio.

Confronto di ordini polinomiali. La funzione [trypoly\(x,y\)](#) approssima i dati in x,y con una serie di polinomi dal grado 1 a $\text{length}(x)-1$ e restituisce i coefficienti di determinazione (R^2) di ciascuna approssimazione come un vettore, mostrando che, per *qualsiasi* dato, il coefficiente di determinazione R^2 si avvicina a 1 quando l'ordine polinomiale si avvicina a $\text{length}(x)-1$. La variante [trypolyplot\(x,y\)](#) crea un grafico a barre come quello mostrato a lato.

Confronto delle trasformazioni dei dati.

La funzione [trydatatrans\(x, y, polyorder\)](#) prova 8 diverse semplici trasformazioni dei dati x,y, approssima i dati trasformati a un polinomio di ordine 'polyorder', mostra i risultati [graficamente in un matrice 3 x 3 di piccoli grafici](#) e restituisce i valori di R^2 in un vettore. Nell'esempio seguente, per $\text{polyorder}=1$, è il 5° ad essere il migliore, ovvero x rispetto a $\ln(y)$. Un esempio è mostrato di seguito.



Approssimazione dei picchi con Gaussiana o Lorentziana singola

Una semplice funzione utente Matlab/Octave che approssima una funzione Gaussiana singola a un segnale x,y è [gaussfit.m](#), che implementa il metodo di approssimazione quadratico di x rispetto a $\ln(y)$ [descritto sopra](#). Ha la forma `[Height,Position,Width]=gaussfit(x,y)`.

Per esempio,

```
>> x=50:150;
>> y=100.*gaussian(x,100,100)+10.*randn(size(x));
>> [Height,Position,Width]=gaussfit(x,y)
```

restituisce [Height,Position,Width] raggruppati intorno a 100,100,100. Una funzione simile per i picchi Lorentziani è [lorentzfit.m](#), che assume la forma

```
[Height,Position,Width]=lorentzfit(x,y).
```

Una variante espansa della funzione gaussfit.m è [bootgaussfit.m](#), che fa la stessa cosa ma opzionalmente disegna anche i dati e l'approssimazione e calcola le stime dell'errore casuale nell'altezza, la larghezza e la posizione della funzione Gaussiana approssimata col metodo di campionamento bootstrap. Per esempio:

```
>> x=50:150;

>> y=100.*gaussian(x,100,100)+10.*randn(size(x));

>> [Height, Position, Width, BootResults]=bootgaussfit(x,y,1);
```

Funziona come nell'esempio precedente, ma mostra anche le stime dell'errore in una tabella (pagina seguente) e restituisce la matrice 3x5 BootResults. Digitare "help bootgaussfit" per un aiuto.

Height	Position	Width
Bootstrap Mean:	100.84	101.325
Bootstrap STD:	1.3458	0.63091
Bootstrap IQR:	1.7692	0.86874
Percent RSD:	1.3346	0.62266
Percent IQR:	1.7543	0.85737
98.341	2.0686	2.9735
2.1035		
3.0237		

È importante che il segnale rumoroso non venga filtrato con lo [smoothing](#) se le previsioni di errore di bootstrap devono essere accurate. Lo smoothing fa sì che il metodo bootstrap sottostimi seriamente la precisione dei risultati.

Le funzioni gaussfit.m e lorentzfit.m sono semplici e facili, ma non funzionano bene con picchi molto rumorosi o per picchi multipli sovrapposti. A titolo di dimostrativo, [OverlappingPeaks.m](#) è uno script che mostra come utilizzare gaussfit.m per misurare [due picchi Gaussiani parzialmente sovrapposti](#). Richiede un'attenta selezione delle regioni ottimali di dati intorno alla cima di ciascun picco. Si provi a cambiare la posizione e l'altezza relative del secondo picco o ad aggiungere rumore (riga 3) e si osservi come influiscono sulla precisione. Questa funzione richiede le funzioni gaussian.m, gaussfit.m e peakfit.m nel pathdi ricerca di Matlab. Lo script esegue anche misure con il [metodo iterativo](#) (pag. 190) utilizzando peakfit.m, che è [più accurato ma richiede più tempo per i calcoli](#).

Le funzioni solo-per-Matlab [iSignal.m](#) (pag. 366) e [ipf.m](#) (pag. 405), i cui compiti principali sono l'approssimazione di *picchi*, possono anche eseguire l'approssimazione di *polinomi* di qualsiasi ordine (**Shift-o**).

Le versioni recenti di Matlab hanno un comodo strumento per l'approssimazione interattiva controllato manualmente (anziché programmato) della curva polinomiale nella finestra Figure. Se si sta leggendo online, cliccare per un video di esempio: [\(link esterno a YouTube\)](#).

Il *Matlab Statistics Toolbox* include due tipi di funzioni di bootstrap, "[bootstrp](#)" e "[jackknife](#)". Per aprire la pagina di riferimento nel browser della guida di Matlab, digitare "doc bootstrp" o "doc jackknife".

Approssimazione delle curve B: Spettroscopia Multicomponente

L'analisi spettroscopica delle miscele, quando lo spettro della miscela è la semplice somma degli spettri dei componenti noti che possono sovrapporsi ma non sono identici, può essere eseguita utilizzando speciali metodi di calibrazione basati su un tipo di minimi quadrati lineari chiamato *regressione lineare multipla*. Questo metodo è ampiamente utilizzato dai moderni spettrometri come quelli a serie di diodi, nella trasformata di Fourier e spettrometri a scansione controllati digitalmente (poiché la perfetta riproducibilità della lunghezza d'onda è un requisito fondamentale). Per comprendere la matematica richiesta, è utile introdurre un po' di base sull'[algebra matriciale](#) (nota anche come algebra lineare), che è solo una notazione abbreviata per trattare i segnali espressi come equazioni con un termine per ciascun punto. Poiché quest'area della matematica potrebbe non far parte del background matematico di tutti, in questa sezione si eseguiranno alcune derivazioni matematiche matriciali elementari.

Definizioni:

n = numero dei diversi componenti chimici nella miscela

s = numero dei campioni

s₁, s₂ = campione 1, campione 2, ecc.

c = concentrazione molare

c₁, c₂ = componente 1, componente 2, ecc.

w = numero di lunghezze d'onda a cui viene misurato il segnale

w₁, w₂ = lunghezza-d'onda 1, lunghezza-d'onda 2, ecc.

Σ = sensibilità analitica (pendenza di un grafico di A rispetto a c)

A = segnale analitico

\mathbf{M}^T = matrice trasposta della matrice \mathbf{M} (righe e colonne invertite).

\mathbf{M}^{-1} = [matrice inversa](#) della matrice \mathbf{M} .

Presupposti:

Il segnale analitico, A (come l'assorbanza nella spettroscopia di assorbimento, l'intensità della fluorescenza nella spettroscopia di fluorescenza e la riflettanza nella spettroscopia di riflettanza) è direttamente proporzionale alla concentrazione, c. La costante di proporzionalità, che è la pendenza

di un grafico di A rispetto a c, è $\mathbf{\Sigma}$.

$$\mathbf{A} = \mathbf{\Sigma} \mathbf{c}$$

Il segnale totale è la somma dei segnali di ogni componente in una miscela:

$$A_{\text{totale}} = A_{c1} + A_{c2} + \dots \text{ per tutti gli } n \text{ componenti.}$$

Calibrazione multivariata col metodo dei minimi quadrati classico (CLS)

Questo metodo viene solitamente applicato all'analisi spettroscopica quantitativa di una miscela di componenti, a patto di poter misurare gli spettri dei singoli componenti e assumendo che il segnale totale della miscela sia semplicemente la somma dei segnali per ciascun componente della miscela. La misura degli spettri delle concentrazioni note dei diversi componenti consente di determinare la loro sensibilità analitica $\mathbf{\Sigma}$ a ciascuna lunghezza d'onda. Quindi ne consegue che:

$$A_{w1} = \mathbf{\Sigma}_{c1,w1} c_{c1} + \mathbf{\Sigma}_{c2,w1} c_{c2} + \mathbf{\Sigma}_{c3,w1} c_{c3} + \dots \text{ per tutti gli } n \text{ componenti.}$$

$$A_{w2} = \mathbf{\Sigma}_{c1,w2} c_{c1} + \mathbf{\Sigma}_{c2,w2} c_{c2} + \mathbf{\Sigma}_{c3,w2} c_{c3} + \dots$$

e così via per tutte le lunghezze d'onda - w3, w4, ecc. Non è pratico scrivere tutti questi termini singolarmente, soprattutto perché ci possono essere *centinaia* di lunghezze d'onda nei moderni spettrometri con array-di sensori. Inoltre, nonostante la massa di dati, queste non sono altro che equazioni lineari; i calcoli richiesti qui sono piuttosto semplici e sicuramente molto facili da far fare ad un computer. Quindi, *c'è davvero bisogno di una notazione altrettanto semplice* che sia più compatta. Per fare ciò, è convenzionale usare **lettere in grassetto** per rappresentare un *vettore* (come una colonna o una riga di numeri in uno spreadsheet) o un *amatrice* (come un *blocco* di numeri in uno spreadsheet). Per esempio, **A** potrebbe rappresentare l'elenco delle assorbanze in ciascuna lunghezza d'onda in uno spettro di assorbimento. Quindi questo grande insieme di equazioni lineari può essere scritto:

$$\mathbf{A} = \mathbf{\Sigma} \mathbf{C}$$

dove **A** è il vettore lungo w dei segnali misurati (p.es. lo spettro del segnale) della miscela, $\mathbf{\Sigma}$ è la matrice rettangolare $n \times w$ dei valori noti di $\mathbf{\Sigma}$ per ciascuno degli n componenti a ciascuna delle w lunghezze d'onda, e **C** è il vettore lungo n delle concentrazioni di tutti i componenti. $\mathbf{\Sigma} \mathbf{C}$ significa che $\mathbf{\Sigma}$ "pre-moltiplica" **C**; ovvero, *ogni colonna della matrice $\mathbf{\Sigma}$ viene moltiplicata punto-per-punto* per il vettore **C**.

Se si dispone di una soluzione campione contenente quantità sconosciute di questi n componenti, se ne misura lo spettro **A** e si cerca di calcolare il vettore concentrazione delle concentrazioni **C**. Per risolvere l'equazione di matriciale di cui sopra per **C**, il numero delle lunghezze d'onda w dev'essere uguale o maggiore del numero dei componenti n . Se $w = n$, allora abbiamo un sistema di n equazioni in n incognite che può essere risolto pre-moltiplicando entrambi i lati dell'equazione per $\mathbf{\Sigma}^{-1}$, la matrice inversa di $\mathbf{\Sigma}$, e usando la proprietà che ogni matrice moltiplicata per il suo inverso è l'unità:

$$\mathbf{C} = \mathbf{\Sigma}^{-1} \mathbf{A}$$

Poiché gli spettri sperimentali reali sono soggetti a rumore casuale (p.es. rumore fotonico e quello del rivelatore), la soluzione sarà più precisa se vengono utilizzati i segnali a un numero maggiore di lunghezze d'onda, ovvero se $w > n$. Questo è facilmente realizzabile senza aumentare la manodopera utilizzando un moderno spettrofotometro a matrice di sensori. Ma allora l'equazione non può essere risolta con una semplice inversione di matrice, perché la matrice $\mathbf{\Sigma}$ è $w \times n$ e *una matrice inversa esiste solo per le matrici quadrate*. Tuttavia, una soluzione può essere ottenuta in questo caso pre-moltiplicando entrambi i lati dell'equazione per l'espressione $(\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^T$, supponendo

che $\mathbf{\Sigma}$ sia diverso da zero:

$$(\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^T \mathbf{A} = (\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^T \mathbf{\Sigma} \mathbf{C} = (\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} (\mathbf{\Sigma}^T \mathbf{\Sigma}) \mathbf{C}$$

dove $\mathbf{\Sigma}^T$ è la *trasposta* di $\mathbf{\Sigma}$ (righe e colonne invertite). Ma la quantità $(\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} (\mathbf{\Sigma}^T \mathbf{\Sigma})$ è una matrice moltiplicata per la sua inversa e quindi è unitaria. Quindi, possiamo semplificare il risultato con:

$$\mathbf{C} = (\mathbf{\Sigma}^T \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^T \mathbf{A}$$

Questa è spesso detta “[equazione normale](#)”. In questa espressione, $\mathbf{\Sigma}^T \mathbf{\Sigma}$ è una matrice quadrata di ordine n , il numero delle componenti. Nella maggior parte delle applicazioni pratiche, n , il numero di componenti chimici, è relativamente piccolo, forse solo da 2 a 5. Il vettore \mathbf{A} di lunghezza w , è il numero delle lunghezze d'onda. Può essere abbastanza grande, forse diverse centinaia in uno spettrometro ad array di diodi. Più lunghezze d'onda vengono utilizzate, più efficacemente verrà mediata il rumore casuale (sebbene non aiuti a utilizzare lunghezze d'onda nelle regioni spettrali in cui nessuno delle componenti produce segnali analitici). La determinazione della regione di lunghezza d'onda ottimale deve essere generalmente determinata empiricamente. Tutti i componenti che contribuiscono allo spettro devono essere considerati e inclusi nella matrice $\mathbf{\Sigma}$. Questo metodo di calcolo è chiamato “Classical Least Squares” (Minimi Quadrati Classici) o semplicemente “CLS”, così chiamato perché è un metodo piuttosto vecchio introdotto molto prima che i computer lo rendessero pratico.

Vengono comunemente realizzate tre estensioni del metodo CLS:

- (a) Se si hanno s campioni multipli ignoti da misurare, si possono calcolare tutti in una volta utilizzando la stessa notazione di cui sopre, combinando i loro spettri in una matrice $w \times s$ detta \mathbf{A} , che risulterà in una matrice $n \times s$ detta \mathbf{C} . (Questa funzionalità è utilizzata nell'Appendice: "Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo" a pagina354).
- (b) Per tener conto dello spostamento della linea di base causato dalla deriva, dallo sfondo e dalla dispersione della luce, viene aggiunta una colonna di 1 alla matrice $\mathbf{\Sigma}$. Ciò ha l'effetto di introdurre nella soluzione un componente aggiuntivo con uno spettro piatto; indicato come "correzione del background".
- (c) Per tenere conto del fatto che la precisione della misura può variare con la lunghezza d'onda, si è soliti eseguire una soluzione *pesata* dei minimi quadrati che de-enfatizza le regioni di lunghezza d'onda in cui la precisione è scarsa:
- (d) $\mathbf{C} = (\mathbf{\Sigma}^T \mathbf{\Sigma}^{-1} \mathbf{\Sigma})^{-1} \mathbf{\Sigma}^T \mathbf{V}^{-1} \mathbf{A}$

dove \mathbf{V} è una matrice diagonale $w \times w$ delle varianze a ciascuna lunghezza d'onda. Nella spettroscopia di assorbimento, dove la precisione della misura è scarsa nelle regioni spettrali dove l'assorbanza è molto alta (e sia il livello di luce che il rapporto segnale-rumore sono quindi bassi), è comune usare la trasmittanza T o il suo quadrato T^2 come fattori delle lunghezze d'onda.

Il metodo CLS è in linea di principio applicabile a qualsiasi numero di componenti sovrapposti. La sua precisione è limitata dalla precisione con cui sono noti gli spettri dei singoli componenti, dalla quantità di rumore nel segnale, dall'entità della sovrapposizione degli spettri e dalla linearità delle curve analitiche di ciascun componente (la misura con cui le ampiezze del segnale sono proporzionali alla concentrazione). In pratica, il metodo non funziona bene con strumenti vecchio stile con una regolazione meccanica della lunghezza d'onda, a causa dell'insufficiente riproducibilità della lunghezza d'onda. Questo perché molti dei punti dati finiscono per trovarsi sui *lati* delle bande spettrali, dove *anche piccoli errori nella riproducibilità delle impostazioni della lunghezza d'onda tra le misure comporterebbe grandi cambiamenti nell'intensità*. Tuttavia, è particolarmente adatto

agli strumenti a serie di diodi e a trasformata di Fourier, che hanno una riproducibilità della lunghezza d'onda estremamente buona. Il metodo dipende anche dalla linearità del segnale analitico rispetto alla concentrazione. Specificatamente nella spettrofotometria di assorbimento, ci sono ben note [deviazioni strumentali dalla linearità della curva analitica](#) che pongono un limite alle prestazioni di questo metodo, ma questo problema può essere evitato applicando l'approssimazione iterativa della curva (pag. 190) e la [Convoluzione di Fourier](#) (pag. 103) allo spettro di *trasmissione*, un'idea che verrà sviluppata in seguito, a pag. 268.

Calibrazione inversa dei minimi quadrati (ILS)

ILS è un metodo che può essere utilizzato per misurare la concentrazione di un analita in campioni in cui lo spettro dell'analita *non* è noto in anticipo. Mentre il metodo dei minimi quadrati classico (classical least-squares: CLS) modella il segnale in ciascuna lunghezza d'onda come la somma delle concentrazioni dell'analita moltiplicata per la sensibilità analitica, i metodi dei minimi quadrati inversi (inverse least-squares: ILS) utilizzano l'approccio inverso e modellano la concentrazione dell'analita c in ciascun campione come somma dei segnali A a ciascuna lunghezza d'onda moltiplicati per i coefficienti di calibrazione m che esprimono come la concentrazione di quel componente sia correlata al segnale in ciascuna lunghezza d'onda:

$$c_{s1} = m_{w1}A_{s1,w1} + m_{w2}A_{s1,w2} + m_{w3}A_{s1,w3} + \dots \text{ per tutte le } w \text{ lunghezze d'onda.}$$

$$c_{s2} = m_{w1}A_{s2,w1} + m_{w2}A_{s2,w2} + m_{w3}A_{s2,w3} + \dots$$

e così via per tutti gli s campioni. In forma matriciale

$$\mathbf{C} = \mathbf{AM}$$

dove \mathbf{C} è il vettore lungo s delle concentrazioni dell'analita negli s campioni, \mathbf{A} è la matrice $w \times s$ dei segnali misurati alle w lunghezze d'onda negli s campioni, e \mathbf{M} è il vettore lungo w dei coefficienti di calibrazione.

Si supponga ora di avere una serie di s campioni standard che sono tipici del tipo di campione che si desidera misurare e che contengono un intervallo di concentrazioni di analiti che coprono l'intervallo delle concentrazioni che ci si aspetta di trovare in altri campioni di quel genere. Questo servirà come *set di calibrazione*. Si misura lo spettro di ciascuno dei campioni in questo set di calibrazione e si inseriscono questi dati in una matrice $w \times s$ dei segnali misurati \mathbf{A} . Quindi si misurano le concentrazioni di analita in ciascuno dei campioni *con un metodo analitico affidabile e indipendente* e si inseriscono i dati in un vettore lungo s delle concentrazioni \mathbf{C} . Insieme, questi dati consentono di calcolare il vettore di calibrazione \mathbf{M} risolvendo l'equazione precedente. Se il numero di campioni nel set di calibrazione è maggiore del numero di lunghezze d'onda, la soluzione dei minimi quadrati è:

$$\mathbf{M} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{C}$$

(Si noti che $\mathbf{A}^T \mathbf{A}$ è una matrice quadrata di dimensione w , il numero delle lunghezze d'onda, che dev'essere inferiore a s). Questo vettore di calibrazione può essere utilizzato per calcolare le concentrazioni di analiti di altri campioni, che sono simili a quelli presenti nel set di calibrazione, dagli spettri misurati dei campioni:

$$\mathbf{C} = \mathbf{AM}$$

Chiaramente, tutto questo funzionerà bene solo se i campioni analitici sono simili in composizione al set di calibrazione. Il vantaggio di questo metodo è che lo spettro di un campione ignoto può essere misurato in modo molto più rapido ed economico rispetto ai metodi di riferimento standard più laboriosi utilizzati per misurare il set di calibrazione, ma se le incognite sono abbastanza simili

al set di calibrazione, le concentrazioni calcolate dall'equazione di cui sopra saranno sufficientemente accurate per molti scopi.

Software per la spettroscopia a lunghezze d'onda multiple

Spreadsheet (Fogli di calcolo)

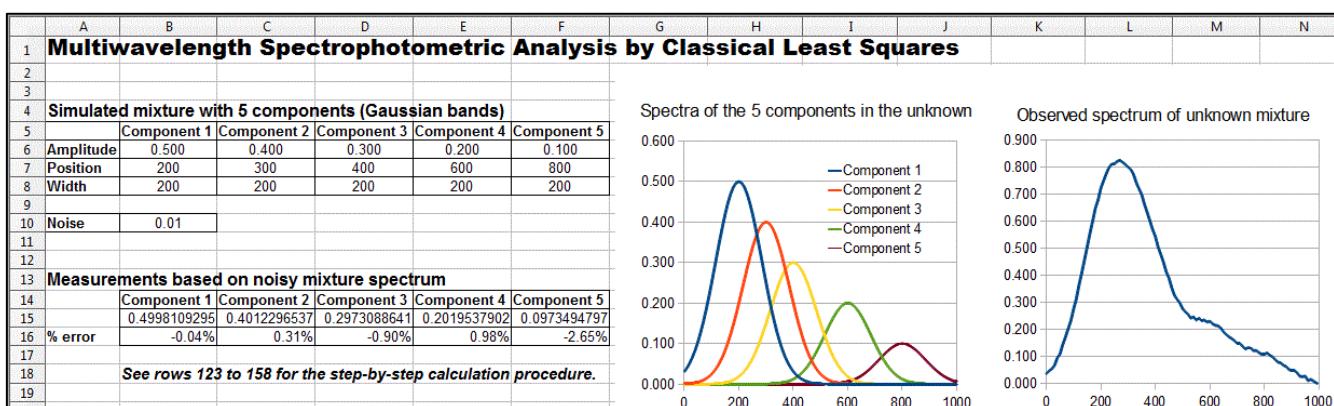
I moderni fogli di calcolo moderni hanno delle funzioni basilari per la gestione di matrici e possono essere utilizzati per la calibrazione multicomponente, ad esempio [Excel](#) e [OpenOffice Calc](#). Gli spreadsheet [RegressionDemo.xls](#) e [RegressionDemo.ods](#) (per Excel e per Calc, rispettivamente) mostrano la classica procedura dei minimi quadrati per un spettro simulato di una miscela di 5 componenti misurata a 100 lunghezze d'onda. Di seguito è mostrata una schermata. I calcoli matriciali descritti che risolvono per la concentrazione dei componenti sulla miscela ignota:

$$\mathbf{C} = (\mathbf{E}^T \mathbf{E})^{-1} \mathbf{E}^T \mathbf{A}$$

vengono eseguite in questi fogli di calcolo dalle funzioni matriciali TRANSPOSE (trasposta di matrice), MMULT (moltiplicazione di matrice) e MINVERSE (inversa di matrice), disposte passo-passo nelle [righe da 123 a 158 di questo spreadsheet](#). In alternativa, tutte queste operazioni matriciali si possono combinare in un'unica grande equazione in una sola cella, risultando, però, meno leggibile.

C = MMULT (MMULT (MINVERSE (MMULT (TRANSPOSE (E);E)); TRANSPOSE (E));A)

dove **C** è il vettore delle 5 concentrazioni di tutti i componenti della miscela, **E** è la matrice rettangolare 5×100 delle sensibilità note (p.es. assorbimenti) per ciascuna delle 5 componenti a ciascuna delle 100 lunghezze d'onda, e **A** è il vettore dei segnali misurati a ciascuna delle 100 lunghezze d'onda (p.es. lo spettro del segnale) della miscela ignota. (Nota: le funzioni matriciali dello spreadsheet come queste devono essere inserite con **Ctrl-Shift-Enter**, non col solo **Enter** come al solito. Si veda "[Guidelines and examples of array formulas](#)".



Dimostrazione in OpenOffice Calc della procedura CLS per la misura di una miscela ignota di 5 componenti a 100 lunghezze d'onda

In alternativa, si possono saltare tutti i dettagli sopra e utilizzare la funzione nativa **LINEST**, sia in [Excel](#) che in [OpenOffice Calc](#), che esegue questo tipo di calcolo in una sola istruzione. Questo è illustrato in [RegressionTemplate.xls](#), nella cella Q23. (Una piccola modifica della sintassi della funzione, mostrata nella cella Q32, esegue un calcolo di *linea di base corretta*). Un vantaggio significativo della funzione **LINEST** è che può calcolare automaticamente gli errori standard dei coefficienti e il valore R^2 nella stessa operazione; usando Matlab o Octave, che richiederebbe un po'

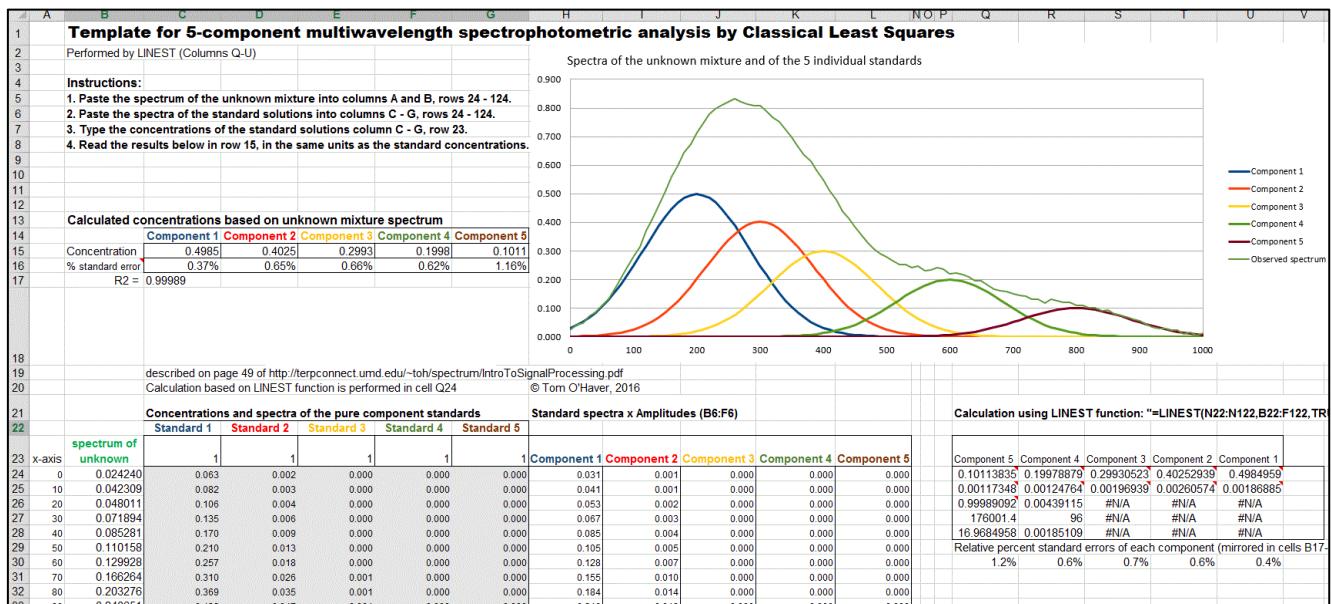
di lavoro extra. (LINEST è anche una funzione matriciale e dev'essere anch'essa inserita digitando **Ctrl-Shift-Enter**, e non semplicemente **Enter**). Da notare che questa è la *stessa* funzione LINEST utilizzata precedentemente per [il polinomio dei minimi quadrati](#) (pagina 154) la differenza è che lì, le colonne multiple di valori sono *calcolate*, prendendo per esempio le potenze (quadrati, cubi, ecc.) delle x, mentre nel metodo CLS multicomponente, le colonne multiple dei valori x sono spettri *sperimentali* delle diverse soluzioni standard. La *matematica* è la stessa ma l'*origine dei dati x* è molto diversa.

Un template per eseguire un'analisi di 5 componenti sui propri dati, con le istruzioni passo-passo, è disponibile come [RegressionTemplate.xls](#) e [RegressionTemplate.ods](#) ([Grafico a pagina dopo](#)) dalla demo precedente). Si esegue il copia-e-incolla dei propri dati nelle colonne B - G. È necessario modificare le formule se il numero di punti o di componenti è diverso da questo esempio. Il modo più semplice per aggiungere più lunghezze d'onda è selezionare un'intera riga ovunque tra la riga 40 e la fine, click destro sul numero della riga a sinistra e si seleziona **Insert**. Questo inserirà una nuova riga vuota e aggiusterà automaticamente tutte le formule delle celle (inclusa la funzione LINEST) e il grafico per includere la nuova riga. Ripetere per le volte che è necessario. Infine, si seleziona l'intera riga appena prima dell'inserimento (ovvero l'ultima riga non vuota) e si trascina la copia verso il basso per riempire tutte le nuove righe vuote. La modifica del numero di componenti è più difficile: comporta l'inserimento o l'eliminazione di colonne tra la C e la G e tra la H e la L, nonché l'aggiustamento delle formule nelle righe 15 e 16 e anche in Q29-U29.

I fogli di calcolo di questo tipo, sebbene facili da usare una volta costruiti, devono essere modificati con cura per applicazioni aventi un numero diverso di componenti o un numero diverso di lunghezze d'onda, il che è scomodo e può essere soggetto a errori. Tuttavia, è possibile costruire questi fogli di calcolo in modo tale che si adeguino *automaticamente* a qualsiasi numero di componenti o lunghezze d'onda. Questo viene fatto utilizzando due nuove funzioni:

- (a) la funzione [COUNT](#) nelle celle B18 e F18, che conta il numero di lunghezze d'onda nella colonna A e il numero di componenti nella riga Q22-U22, rispettivamente, e
- (b) la funzione [INDIRECT](#) (vedere pagina 345) nella cella Q23 e nelle righe 12 e 13, che consente di *calcolare nello spreadsheet* l'indirizzo di una cella o di un intervallo di celle (in base al numero di lunghezze d'onda e di componenti appena contati) anziché usare un intervallo fissato.

Questa tecnica è utilizzata in [RegressionTemplate2.xls](#) e in due esempi che mostrano lo *stesso template* con i dati immessi per un diverso numero di lunghezze d'onda e per miscele di 5 componenti a 100 lunghezze d'onda ([RegressionTemplate2Example.xls](#)) e per 2 componenti a 59 lunghezze d'onda ([RegressionTemplate3Example.xls](#)). Ispezionando le funzioni LINEST nella cella Q23, si vedrà che sono le stesse in entrambi questi due modelli di esempio, anche se il numero di lunghezze d'onda e il numero dei componenti è diverso. Si dovrà comunque aggiustare il grafico per coprire l'intervallo dell'asse x desiderato. Cfr. pagina 345.



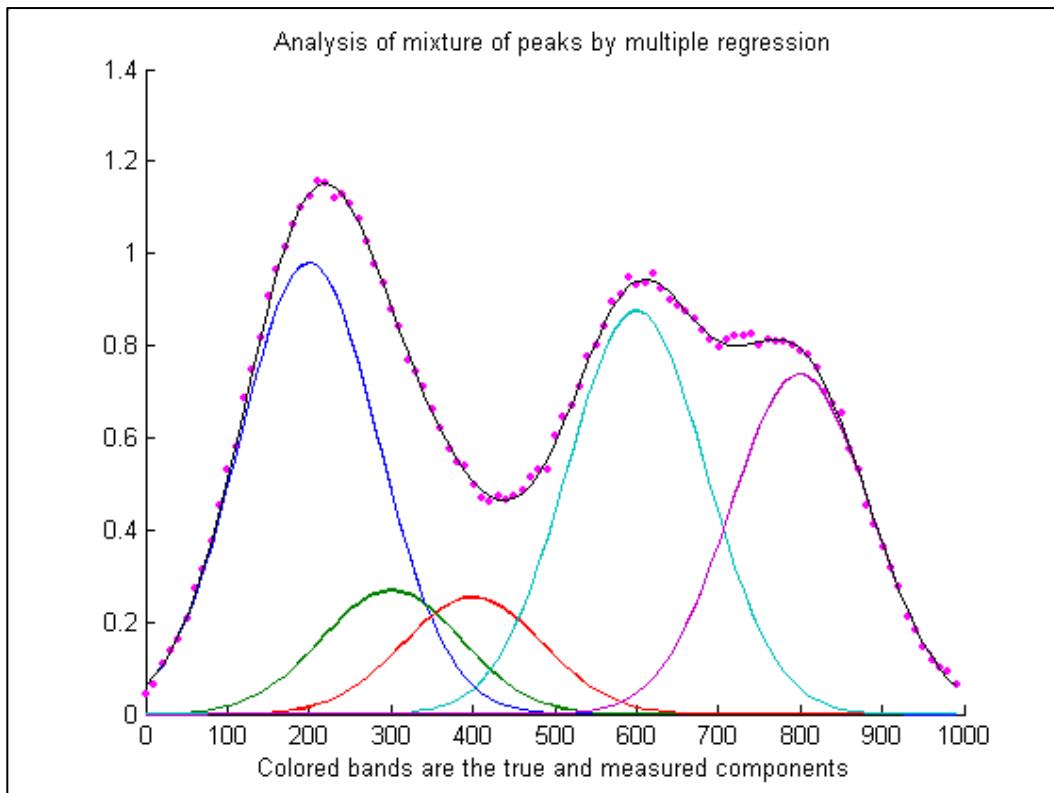
Template Excel per la misura di una miscela ignota di 5 componenti a 100 lunghezze d'onda.

Matlab/Octave e Python

Matlab/Octave e Python sono davvero i linguaggi naturali per l'analisi multicomponente perché gestiscono tutti i tipi di matematica matriciale in modo molto semplice, compatto e veloce e si adattano prontamente a qualsiasi numero di lunghezze d'onda o numero di componenti senza trucchi speciali. In Matlab/Octave, la notazione è molto compatta: la trasposizione della matrice A è A' , l'inversa di A è $\text{inv}(A)$ e la moltiplicazione tra matrici è indicata con un asterisco (*). Quindi la soluzione al metodo dei Minimi Quadrati Classico sopra è scritta in notazione Matlab/Octave come

$$\mathbf{C} = \text{inv}(\mathbf{E}' * \mathbf{E}) * \mathbf{E}' * \mathbf{A}$$

dove E è la matrice rettangolare delle sensibilità a ciascuna lunghezza d'onda per ciascun componente e A è lo spettro osservato della miscela. Si noti che la notazione Matlab/Octave non è solo più corta di quella nel foglio di calcolo, ma è anche più vicina alla notazione matematica tradizionale. In modo ancora più compatto, si può scrivere $C = A/E$, utilizzando l'operatore di Matlab [barra or "divisione destra"](#), che restituisce gli stessi risultati ma è in linea di principio più accurato rispetto alla precisione numerica del computer (solitamente trascurabile rispetto al rumore nel segnale; vedi pagina 332).



Lo script [RegressionDemo.m](#) (per Matlab o Octave) mostra la classica procedura dei minimi quadrati per uno spettro di assorbimento simulato di una miscela di 5 componenti a 100 lunghezze d'onda, illustrata sopra. La maggior parte di questo script è solo per la generazione dei segnali e per il disegno; l'effettiva regressione dei minimi quadrati viene eseguita su una riga:

```
MeasuredAmp = ObservedSpectrum*A'*inv(A*A')
```

dove vengono utilizzati simboli diversi per le variabili: "A" è una matrice contenente lo spettro di ciascuno dei componenti in ciascuna delle sue righe e "ObservedSpectrum" è lo spettro osservato della miscela ignota. In questo esempio, i punti rappresentano lo spettro osservato della miscela (con rumore) e le cinque bande colorate rappresentano i cinque componenti della miscela, i cui spettri sono noti ma le cui concentrazioni nella miscela sono sconosciute. La linea nera rappresenta "l'approssimazione migliore" allo spettro osservato calcolato dal programma. In questo esempio, le concentrazioni dei cinque componenti vengono misurate con una precisione di circa l'1% relativo (limitata dal rumore nello spettro osservato). Confrontando [RegressionDemo.m](#) al suo spreadsheet equivalente, [RegressionDemo.ods](#), entrambi in esecuzione sullo stesso computer, si può vedere che il codice Matlab/Octave calcola e disegna i risultati più rapidamente: un calcolo a 5 componenti con uno spettro da 1000 punti impiega meno di 0.01 secondi su un moderno PC desktop o laptop. La tecnica CLS è abbastanza veloce (soprattutto in Matlab) da poter essere applicata in tempo reale alla cromatografia 2D con rivelatori a matrice, dove viene acquisito uno spettro completo più volte al secondo sull'intero cromatogramma. Vedere "Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo" a pagina 354.

Estensioni:

(a) L'estensione a **campioni ignoti multipli**, ciascuno col proprio "ObservedSpectrum" è semplice in Matlab/Octave. Se si hanno "s" campioni, basta semplicemente assemblare i loro spettri osservati in una *matrice* con "s" righe e "w" colonne ("w" è il numero delle lunghezze d'onda), poi usare la formula come prima:

```
MeasuredAmp = ObservedSpectrum*A'*inv(A*A')
```

Il "MeasuredAmp" risultante sarà una *matrice* "s" × "n" anziché un vettore di lunghezza n ("n" è il numero di componenti misurati). Questo è un ottimo esempio della comodità della natura vettoriale/matriciale di questo linguaggio. ([RegressionDemoMultipleSamples.m](#) ne è una dimostrazione).

(b) L'estensione alla "**correzione del background**" si ottiene facilmente in Matlab/Octave aggiungendo una colonna di 1 alla matrice **A** contenente lo spettro dell'assorbimento di ciascun componente:

```
background=ones(size(ObservedSpectrum)) ;  
A=[background A1 A2 A3] ;
```

dove A1, A2, A3... sono i vettori degli spettri di assorbimento dei singoli componenti.

(c) Anche la **Regressione della trasmissione ponderata** si esegue rapidamente:

```
MeasuredAmp=(T*T).*A)\(ObservedSpectrum.*T) ;
```

dove T è il vettore dello spettro di trasmissione. Qui, viene usata la divisione matriciale con la barra rovesciata "\\" come scorciatoia per la classica soluzione matriciale dei quadrati minimi (cfr. <http://www.mathworks.com/help/techdoc/ref/mldivide.html>).

La funzione [cls.m](#): Normalmente, la matrice di calibrazione **M** viene assemblata dai segnali misurati sperimentalmente (p.es. gli spettri) dei singoli componenti della miscela, ma è anche possibile approssimare un modello generato al computer di profili base (p.es. Gaussiane, Lorentziane, ecc.) ad un segnale per determinare se tale segnale si possa rappresentare come somma ponderata di picchi, con profili semplici, sovrapposti. La funzione [cls.m](#) calcola un modello di questo tipo costituito dalla somma di qualsiasi numero di picchi di *di forma, larghezza e posizione nota*, ma di *altezza ignota*, e lo approssima al set x,y di dati rumorosi. La sintassi è

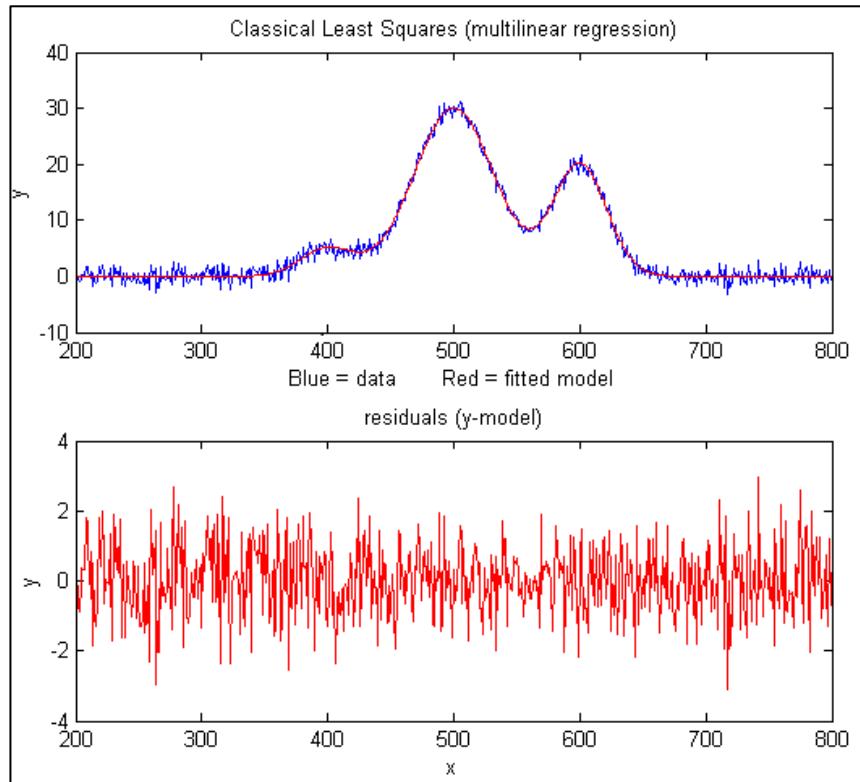
```
heights=cls(x, y, NumPeaks, PeakShape, Positions, Widths, extra)
```

dove x e y sono i vettori dei dati misurati (p.es. x potrebbe essere la lunghezza d'onda e y potrebbe essere l'assorbanza a ciascuna di tale lunghezza d'onda), 'NumPeaks' è il numero dei picchi, 'PeakShape' è il numero del profilo (1=Gaussiana, 2=Lorentziana, 3=logistica, 4=Pearson, 5=Gaussiana esponenzialmente espansa; 6=Gaussiane di pari larghezze; 7=Lorentziane di pari larghezze; 8=Gaussiane di pari larghezze esponenzialmente espansa, 9=impulso esponenziale, 10=sigmoide, 11=Gaussiana a larghezza fissa, 12=Lorentziana a larghezza fissa; 13=mix di Gaussiane/Lorentziane; 14=BiGaussiana, 15=BiLorentziana), 'Positions' è il vettore delle posizioni

dei picchi sull'asse x (una voce per ogni picco), 'Widths' è il vettore delle larghezze dei picchi in unità x (una voce per ogni picco) e 'extra' è il parametro aggiuntivo richiesto dai profili esponenzialmente espansi, Pearson, mix di Gaussiane/Lorentziane, BiGaussiana e BiLorentziana. Cls.m restituisce un vettore delle altezze misurate di ciascun picco.

La funzione [cls2.m](#) è simile alla [cls.m](#), eccetto che misura anche la linea di base (assumendo che sia piatta), utilizzando l'estensione alla "correzione del background" descritta in precedenza, e restituisce un vettore contenente il background B e le altezze misurate dei picchi H per ciascun picco 1,2,3, p.es., [B H1 H2 H3...].

Lo script dimostrativo [clsdemo.m](#) ([a lato](#)) crea i dati di un modello rumoroso, lo approssima con cls.m, calcola l'accuratezza delle altezze misurate, poi ripete il calcolo *per un'approssimazione iterativa dei quadrati minimi* ("INLS", trattato a pagina 190) utilizzando la funzione Matlab/Octave [peakfit.m](#), utilizzando le posizioni e le larghezze dei picchi conosciute solo come *ipotesi iniziale* ("start"). Si può vedere che CLS è più veloce e (solitamente) più accurato, specialmente se i picchi sono molto sovrapposti. (Questo script richiede le funzioni cls.m, modelpeaks.m e peakfit.m nel path di ricerca di Matlab/Octave). Un risultato tipico è:



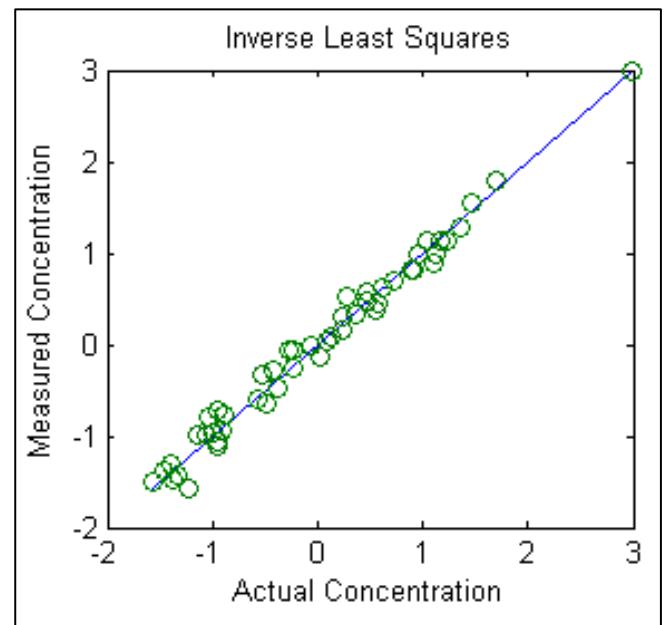
```
Figure window(1) : Classical Least-squares (multilinear regression)
Elapsed time is 0.012 seconds.
Average peak height accuracy = 0.9145%
```

```
Figure window(2) : Iterative non-linear peak fitting with peakfit.m
Elapsed time is 0.39 seconds.
Average peak height accuracy = 1.6215%
```

D'altra parte, INLS può essere migliore di CLS se ci sono *slittamenti insospettabili* delle posizioni dei picchi e/o nelle ampiezze tra la calibrazione e la misura (ad esempio causati dalla deriva della calibrazione dello spettrometro o dal cambiamento di temperatura, pressione o di soluzioni variabili), perché INLS può tracciare e compensare i cambiamenti di posizione e larghezza del picco. Lo si può verificare cambiando la variabile "PeakShift" (riga 16) a un valore diverso da zero in [clsdemo.m](#).

Regressione lineare ponderata: La funzione Matlab/Octave "[tfit.m](#)" simula la misura dello spettro di assorbimento di una miscela di tre componenti con la regressione lineare *ponderata e non*, mostra l'effetto della quantità di rumore nel segnale, l'entità della sovrapposizione degli spettri e la linearità delle curve analitiche di ciascuna componente. Questa funzione confronta anche i risultati con [un metodo più avanzato](#) descritto in seguito (riga 66) che applica l'approssimazione allo spettro dell'atrasmissione anziché a quello dell'assorbanza (pagina 268), che migliora la linearità e l'accuratezza del metodo.

La tecnica dei **Quadrati Minimi Inversa (ILS)** viene dimostrata in Matlab da [questo script](#) e dal grafico a lato. La matematica, [descritta sopra](#) a pagina 182, è simile al metodo Classico dei Quadrati Minimi e si può eseguire con qualsiasi metodo Matlab/Octave, Python o con spreadsheet descritto in questa sezione. Questo esempio si basa su un set di dati reali derivato dalla [spettroscopia di riflettanza nel infrarosso vicino \(NIR\)](#) di campioni di pasta di grano agricolo analizzati per il contenuto proteico. Sono presenti 50 campioni di calibrazione misurati a 6 lunghezze d'onda. Questi campioni di calibrazione erano già stati analizzati per il [contenuto proteico](#) con un affidabile (ma lento) [metodo di riferimento](#). Lo scopo di questa calibrazione è stabilire se i risultati del metodo della riflettanza dell'infrarosso vicino, più rapido e semplice, sono correlati al contenuto proteico determinato dal metodo di riferimento. Questi risultati indicano che lo sono, almeno per questo set di 50 campioni di grano, e quindi è probabile che la spettroscopia nell'infrarosso vicino dovrebbe fare un buon lavoro di stima del contenuto proteico di campioni ignoti simili.



Il punto è che i campioni ignoti devono essere simili ai campioni della calibrazione (eccetto ovviamente per il contenuto proteico), ma questa è una situazione analitica molto comune nei *controlli di qualità* industriali e agricoli, dove molti campioni di prodotti o colture di un tipo prevedibile simile devono spesso essere testati in modo rapido ed economico, spesso sul campo utilizzando semplici strumenti portatili. Potrebbe essere necessario un bel po' di tempo e fatica per calibrare lo strumento all'inizio, ma una volta fatto, può essere applicato in modo rapido ed economico al tipo di campione per il quale è stato calibrato. La spettroscopia di riflettanza nell'infrarosso vicino (NIR), in combinazione con i minimi quadrati inversi o dei metodi matematici correlati più sofisticati, è ampiamente utilizzata nell'industria, nell'agricoltura, nelle scienze alimentari e nelle applicazioni ambientali.

Vale la pena notare che il metodo di cui sopra, in particolare i metodi nell'infrarosso vicino, per campioni agricoli, è stato introdotto negli anni 1950-60, dal Dr. Karl Norris presso il Beltsville Agricultural Research Center nel Maryland, appena in fondo alla strada rispetto all'istituto dell'autore nel College Park. La vecchia storia è raccontata dallo stesso Dr. Norris in <https://journals.sagepub.com/doi/10.1255/jnirs.941>.

Nota: Se si sta leggendo questo articolo online, si può fare click destro del mouse su uno qualsiasi dei collegamenti dei file m sopra e selezionare **Save Link As...** per scaricarli nel proprio computer per usarli in Matlab.

I Quadrati Minimi Classici in Python

L'espressione equivale in Python per la "Equazione Normale" è

```
C = inv(E.T.dot(E)).dot(E.T).dot(A)
```

dove "inv()" indica la matrice inversa, l'espressione "E.T" indica la *trasposta* della matrice E e ".dot" indica il *prodotto scalare*. Questo è probabilmente più esplicito rispetto alla versione Matlab più compatta:

```
C = inv(E'*E)*E'*A;
```

Da ricordarsi che prima di farlo in Python si devono eseguire i seguenti import:

```
numpy as np
from numpy.linalg import inv
```

La coppia di script corrispondenti, [NormalEquationDemo.py](#) e [NormalEquationDemo.m](#) confronta questi aspetti in Python e in Matlab. Entrambi gli script utilizzano *la stessa sequenza di operazioni e gli stessi nomi di variabile* per creare e plottare un vettore di segnale simulato rumoroso A che è la

somma di tre picchi fortemente sovrapposti di forma nota (G_1 , G_2 e G_3) ma di ampiezza sconosciuta, poi misura le ampiezze C nel segnale rumoroso mediante i Quadrati Minimi Classici. L'errore percentuale tra le altezze dei picchi reali e quelle misurate viene stampato accanto a ciascun picco. Le somiglianze del codice sono sorprendenti, i risultati sono gli stessi e i tempi di esecuzione sono simili per le versioni Matlab e

Python. L'effettivo calcolo delle concentrazioni C da parte del CLS richiede solo una riga in entrambi i linguaggi: $C = \text{inv}(E.T.\cdot\text{dot}(E))\cdot\text{dot}(E.T)\cdot\text{dot}(A)$ in Python e $C = \text{inv}(E'\cdot E)\cdot E'\cdot A;$ in Matlab.

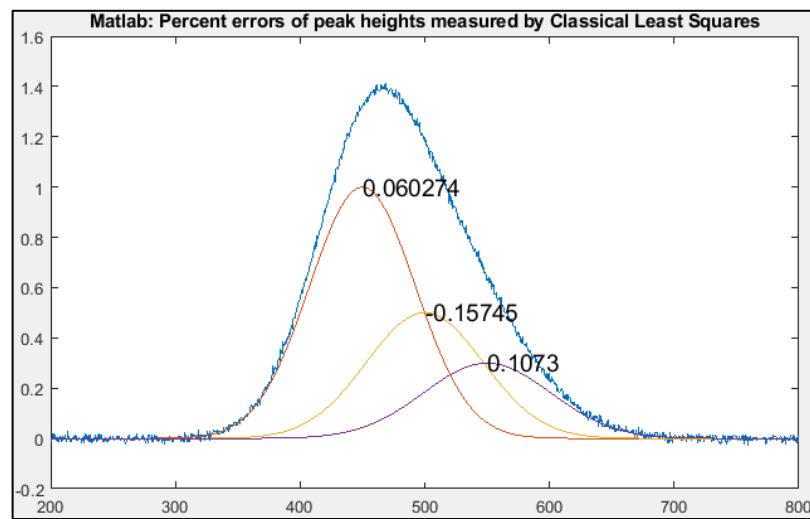
Approssimazione delle curve C: Approssimazione iterativa non-lineare

L'approssimazione ai quadrati minimi, descritta in "[Approssimazione delle curve A](#)" a pagina 154, è semplice e veloce, ma è limitata a situazioni in cui la variabile dipendente è modellabile da un polinomio con coefficienti lineari.

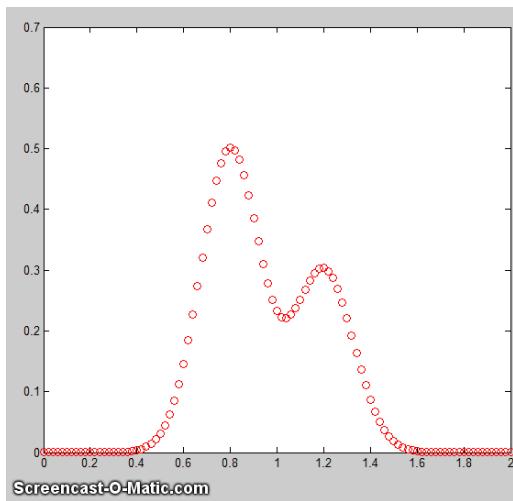
Il metodo più generale per approssimare i dati a qualsiasi modello è il [metodo iterativo](#), che è un tipo di procedura "tenta e riprova" in cui i parametri del modello vengono aggiustati in modo sistematico finché l'equazione non si approssima ai dati quanto richiesto. Suona quasi come un approccio a "forza bruta", e lo è. Infatti, prima dei computer, questo metodo non era molto usato. Ma ora, per la sua grande generalità, unita ai progressi nella velocità del computer e nell'efficienza degli algoritmi, i metodi iterativi sono ampiamente utilizzati più che mai. Esistono ampie applicazioni di questa tecnica nei settori della ricerca biomedica e farmacologia, delle scienze ambientali, della fisica e delle scienze dei materiali, dell'ingegneria e dello sviluppo di farmaci, delle neuroscienze e del neuro-imaging, dell'agronomia e della biomedicina. Ma qui ci si concentrerà sull'applicazione al peakfitting: cioè, determinare se è possibile modellare una forma di segnale complessa come la somma di diversi componenti semplici la cui altezza, posizione, larghezza e forma sono inizialmente ignote.

Un metodo iterativo procede nel seguente modo generale:

- (1) Selezionare un modello per i dati, ad esempio, nell'esempio seguente, la somma di due picchi Gaussiani;



- (2) Effettuare le ipotesi iniziali su tutti i parametri non lineari regolabili; (vale a dire, per l'applicazione dell'approssimazione dei picchi, questi sarebbero la posizione e l'ampiezza dei picchi che si presuppone siano presenti nei dati);
- (3) Un programma calcola il modello e lo confronta con l'insieme dei dati, calcolando un errore di approssimazione;
- (4) Il programma modifica sistematicamente i parametri, torna al passo precedente e lo ripete finché non viene raggiunta la precisione richiesta dell'approssimazione.



Una tecnica popolare per eseguire questa operazione è il [Metodo del Simplex di Nelder-Mead](#). Questa è un modo per organizzare ed ottimizzare i cambiamenti dei parametri (passo 4, sopra) per ridurre il tempo richiesto per approssimare la funzione al grado richiesto di accuratezza. Potrebbe sembrare complicato, ma con i personal computer contemporanei, l'intero processo richiede *in genere solo una frazione di secondo* o pochi secondi, a seconda della complessità del modello e del numero di parametri regolabili indipendentemente del modello. L'animazione qui ([script Matlab](#)) illustra il funzionamento del processo iterativo per un'approssimazione di 2 picchi

Gaussiani non vincolati a un piccolo insieme di dati x , y . Questo modello ha *quattro variabili non lineari* (le posizioni e le ampiezze delle due Gaussiane, determinate dall'iterazione) e *due variabili lineari* (le altezze delle due Gaussiane, che sono direttamente determinate dalla [regressione](#) per ciascuna iterazione di prova). Per consentire l'osservazione del processo in azione, questa animazione viene *artificialmente rallentata* da (a) un disegno passo-passo, (b) fornendo un'ipotesi iniziale sbagliata, e (c) aggiungendo il comando "`pause()`" nel ciclo di interazione. Senza tutto questo rallentamento, *l'intero processo richiede normalmente solo 0,05 secondi circa su un PC o laptop standard*, a seconda principalmente del numero di variabili non lineari da iterare. Funziona anche se i picchi sono così sovrapposti da unirsi in un picco unico, come mostrato dallo script [Demofitgauss2AnimatedBlended.m](#), e la corrispondente [animazione](#), ma potrebbero essere necessarie più iterazioni ed è più sensibile a qualsiasi rumore casuale nei dati (aggiunto nella riga 13).

Nota: lo script che ha creato [questa animazione](#), [Demofitgauss2animated.m](#), richiede che le funzioni [gaussian.m](#), [fitgauss2animated.m](#) e [fminsearch.m](#) stiano nel path di Matlab/Octave. Come ausilio didattico, una versione modificata di questo script, [Demofitgauss2animatedError.m](#), disegna l'errore di approssimazione rispetto al numero di iterazione, mostrando che una scarsa stima iniziale (valore di partenza) può essere dannosa, [richiede molte più iterazioni](#) (e più tempo) per trovare una buona approssimazione, il che non è molto efficiente dal punto di vista di un osservatore umano che può stimare visivamente i parametri del picco in modo molto più intelligente. Una buona ipotesi richiede [meno iterazioni](#). Un modo per ottenere ottimi valori iniziali per i segnali di tipo picco è quello di far precedere l'approssimazione della curva da un veloce [algoritmo di rilevamento dei picchi](#) per determinare il numero dei picchi, le posizioni e larghezze approssimative (come verrà spiegato in una prossima sezione), ma questo funzionerà solo quando i picchi da approssimare hanno massimi distinti e non sono [fusi in un unico picco](#). Come ci si potrebbe aspettare, un'[approssimazione a 3-picchi](#), con *sei* parametri non-lineari da ottimizzare, richiederà [più iterazioni](#).

La principale difficoltà dei metodi iterativi è che a volte non riescono a convergere verso una

soluzione ottimale complessiva nei casi difficili, specialmente se gli si dà una pessima stima iniziale. Per capire come ciò possa accadere, si pensi a un'analogia fisica: si mette un robot cieco in un ambiente collinare e lo si programma per camminare fino alla vetta più alta, dandogli solo un lungo bastone per sondare nelle sue immediate vicinanze e per sentire se il terreno è più alto o più basso attorno a sé. Se è più alto, cammina in quella direzione e sonda quella zona. Si ferma quando tutte le aree intorno sono più basse di dove si trova. Funziona bene se c'è una sola collina omogenea. Ma cosa succede se ci sono più collinette o se il terreno è roccioso e irregolare? È molto probabile che il robot si fermi su una delle alteure più piccole, non vedendo che potrebbe esserci una collina ancora più alta nelle vicinanze. Percorrere l'*intero* ambiente secondo una griglia per *tutte* le alteure richiederebbe moltissimo tempo.

L'approccio standard consiste nel riavviare il processo con variazioni casuali delle prime ipotesi, ripetere più volte e prendendo quello con l'errore di approssimazione più basso. Tale processo viene automatizzato nelle funzioni di approssimazione [descritte a pagina 384](#). Se ciò non bastasse, possiamo fare le nostre ipotesi iniziali. Con tutto questo, non sorprende che l'approssimazione iterativa della curva richieda più tempo della regressione lineare: con i tipici personal computer, un'approssimazione iterativa potrebbe richiedere frazioni di secondo mentre una regressione richiederebbe frazioni di millisecondo. Tuttavia, è abbastanza veloce per parecchi scopi.

La [precisione](#) dei parametri del modello misurata dall'approssimazione iterativa (pagina 202), come la classica dei quadrati minimi, dipende fortemente da un buon modello, una compensazione accurata per il background/linea di base, il rapporto segnale/rumore e dal numero di punti attraverso la caratteristica che si vuole misurare. Non è pratico prevedere le deviazioni standard dei parametri del modello misurato utilizzando l'approccio algebrico, ma sono applicabili sia i metodi della [simulazione Monte Carlo che il bootstrap](#) (pag. 162).

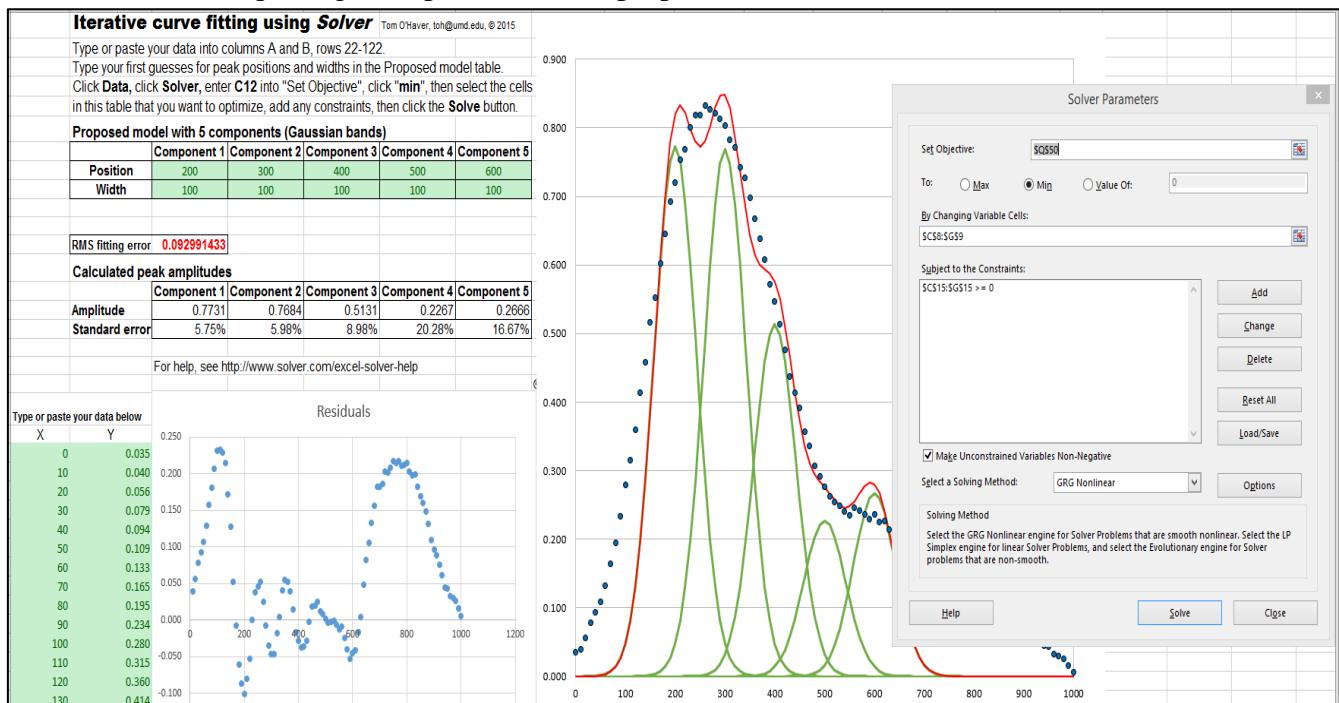
Nota: il termine "deconvoluzione spettrale" o "deconvoluzione di banda" è spesso usato per riferirsi a questa tecnica, ma in questo libro "deconvoluzione" si riferisce specificamente a quella di *Fourier*, un concetto indipendente trattato a altrove. Nella deconvoluzione di Fourier, la forma del picco in esame è *ignota*, ma si presume che la funzione di ampliamento sia *nota*; mentre, nell'approssimazione iterativa della curva dei minimi quadrati, è esattamente il contrario: la forma del picco deve essere nota ma la larghezza del processo di ampliamento, che determina l'ampiezza e la forma dei picchi nei dati registrati, è sconosciuta. Pertanto, il termine "deconvoluzione spettrale" è ambiguo: potrebbe significare la deconvoluzione di Fourier di una funzione di risposta da uno spettro, oppure potrebbe significare la decomposizione di uno spettro nei suoi diversi picchi componenti. Questi sono processi diversi; da non confondere. Vedere pagina 299.

Spreadsheet e programmi autonomi

Sia *Excel* che *OpenOffice Calc* hanno una funzionalità "[Solver](#)" che cambierà le celle indicate nel tentativo di produrre uno specifico obiettivo; questa si può usare nell'approssimazione del picco per minimizzare l'errore tra i dati e il modello calcolato proposto, come un insieme di bande Gaussiane sovrapposte. L'ultima versione comprende [tre diversi metodi di soluzione](#). [Questo esempio di spreadsheet Excel \(schermata\)](#) mostra come viene usato per approssimare quattro componenti Gaussiane ad un insieme di dati x,y che sono già stati inseriti nelle colonne A e B, dalla riga 22 alla 101 (lì si possono copiare e incollare i propri dati).

Dopo aver inserito i dati, si fa una stima visiva di quanti picchi Gaussiani potrebbero essere necessari per rappresentare i dati, le loro posizioni e le loro larghezze, e si digitano i valori stimati nella tabella "Proposed model". Lo spreadsheet calcola i valori per l'approssimazione migliore per le *altezze* con una [regressione multilineare \(pagina 180\)](#) nella tabella 'Calculated amplitudes', disegna i dati e l'approssimazione. Disegna anche i "residui", che sono le *differenze* punto-per-punto

tra i dati e il modello; idealmente i residui dovrebbero essere nulli o almeno piccoli. (Regolare la scala dell'asse x di questi grafici per adattarli ai propri dati).



Il passaggio successivo consiste nell'utilizzare la funzione **Solver** per "mettere a punto" la posizione e la larghezza di ciascun componente per minimizzare la % di errore di approssimazione (in rosso) e per rendere il grafico dei residui il più casuale possibile: si clicca su **Data** nella barra superiore, si clicca **Solver** (in alto a destra) per aprire la casella Solver, in cui si digita "C12" in "Set Objective", si clicca "min", si selezionano le celle in "Proposed Model" che si vuol ottimizzare, si aggiungono tutti i vincoli desiderati nella casella "Subject to the Constraints" e si clicca i pulsante **Solve**. Solver automaticamente ottimizza la posizione, la larghezza e l'ampiezza di tutte le componenti e viene visualizzata l'approssimazione migliore. (Si può vedere che il Solver ha modificato le voci selezionate nella tabella "proposed model", ridotto l'errore dell'approssimazione (cella C12, in rosso) e reso i residui più piccoli e più casuali). Se l'approssimazione fallisce, cambiare i valori iniziali, cliccare su **Solver** e cliccare sul pulsante **Solve**. È possibile automatizzare il processo di cui sopra e ridurlo alla pressione di un singolo tasto funzione utilizzando le *macro*, come descritto a pagina 308.

Quindi, quanti componenti Gaussiani sono necessari per approssimare i dati? Un modo per capirlo è guardare il grafico dei residui (che mostra la differenza punto per punto tra i dati e il modello approssimato) e aggiungere componenti fino a quando i residui non diventano *casuali, e non ondeggianti*, ma questo funziona solo se i dati non hanno subito *smoothing* prima dell'approssimazione. Ecco un esempio: un insieme di dati reali che sono approssimati con una sequenza crescente di [due Gaussiane](#), [tre Gaussiane](#), [quattro Gaussiane](#) e [cinque Gaussiane](#). Nel guardare questa sequenza di schermate, si vedrà diminuire l'errore percentuale dell'approssimazione, il valore R^2 diventa prossimo a 1.000 e i residui diventano più piccoli e più casuali. (Si noti che nell'approssimazione a 5 componenti, la prima e l'ultima componente non sono *picchi* nell'intervallo delle x 250-600 dei dati, ma si riferiscono al *background*). Non è necessario provare un'[approssimazione a 6 componenti](#) perché i residui sono già casuali con 5 e con più componenti si "approssima il rumore" diventando probabilmente instabile con un risultato molto diverso se si approssima un altro campione con un rumore diverso.

Se si utilizza un foglio di calcolo per questo tipo di approssimazione, bisogna creare un foglio di calcolo personalizzato per ogni problema, con il giusto numero di righe per i dati e con il numero di componenti desiderato. Per esempio, il template [CurveFitter.xlsx](#) è solo per un segnale da 100 punti

con un modello di 5 componenti Gaussiane. È facile estendere a un numero maggiore di punti inserendo righe tra la 22 e la 100, dalle colonne A alla N e trascinare giù, copiando le formule nelle nuove celle (p.es. [CurveFitter2.xlsx](#) è stato esteso a 256 punti). Per gestire altri numeri di componenti o forme, si dovrebbe inserire o eliminare le colonne tra C e G e tra Q e U, e modificare le formule, come è stato fatto in questo set di template per [2 Gaussiane](#), [3 Gaussiane](#), [4 Gaussiane](#), [5 Gaussiane](#) e [6 Gaussiane](#).

Se i picchi sono sovrapposti su una linea di base, si può *includere un modello per la linea di base* come una delle componenti. Per esempio, se desiderano approssimare 2 picchi Gaussiani su una linea di base lineare e inclinata, si seleziona un template a *3 componenti* e si cambia una delle Gaussiane nell'equazione di una linea retta($y=mx+b$, dove m è la pendenza e b è l'intercetta). Un template per questo caso particolare è [CurveFitter2GaussianBaseline.xlsx \(grafico\)](#); in questo caso non cliccare su "Make Unconstrained Variables Non-Negative" (Rendi le Variabili non vincolate Non-Negative), perché il modello della linea di base potrebbe richiedere variabili negative, come in questo esempio particolare. Se si desidera utilizzare un'altra forma per i picchi o un'altra forma per la linea di base, è necessario modificare l'equazione nella riga 22 delle colonne da C a G e trascinare e copiare la cella modificata fino all'ultima riga, come è stato fatto per cambiare la forma del picco Gaussiano in una Lorentziana in [CurveFitter6Lorentzian.xlsx](#). Oppure si può fare in modo che le colonne da C a G contengano equazioni per un *diverso* profilo del picco o della linea di base. Per i profili della Gaussiane espansi esponenzialmente, si deve usare [CurveFitter2ExpGaussianTemplate.xlsx](#) per due picchi sovrapposti ([schermata](#)). In questo caso, ogni picco ha *quattro* parametri: altezza, posizione, larghezza e lambda (che determina l'asimmetria - l'entità dell'ampliamento esponenziale).

L'utilizzo di un foglio di calcolo ha un grande vantaggio: è facile aggiungere *vincoli* alle variabili determinate dall'iterazione, ad esempio per vincolarle ad essere maggiori di zero o a rientrare tra due limiti, o ad essere uguale, ecc. Ciò costringerà le soluzioni ad aderire ad aspettative valide ed eviterà soluzioni non fisiche. Ciò è particolarmente importante per forme complesse come la Gaussian ampliata esponenzialmente appena discussa nel paragrafo precedente. È possibile farlo aggiungendo questi vincoli utilizzando la casella "Subject to the Constraints:" (Soggetto ai vincoli) al centro della casella "Solver Parameters" (vedere il grafico nella pagina precedente). Per i dettagli, vedere <https://www.solver.com/excel-solver-add-change-or-delete-constraint?>

Il punto, da tutto questo, è che si possono fare -in effetti, si *devono* fare- molte personalizzazioni per ottenere un template che si adatti ai propri dati. Al contrario, la funzione Matlab/Octave [peakfit.m \(pagina 384\)](#) automaticamente si adatta a qualsiasi numero di punti ed è facilmente impostabile su oltre 40 diverse forme di picchi (grafico a pagina 411) e a qualsiasi numero di picchi, semplicemente cambiando gli argomenti di input. Utilizzando la funzione di *Interactive Peak Fitter*, [ipf.m](#) in Matlab (pagina 405), si può *premere un solo tasto* per cambiare istantaneamente il profilo, il numero, la modalità della linea di base del picco (pagina 211), o per ri-calcolare l'approssimazione con diversi valori iniziali o con un sottoinsieme bootstrap dei dati (per stimare gli errori dei parametri del picco). È molto più semplice e veloce del foglio di calcolo. Ma al contrario, un *reale vantaggio degli spreadsheet* in questa applicazione è che è relativamente facile aggiungere i propri *profili personalizzati e i vincoli*, anche complicati, utilizzando formule standard dello spreadsheet. E se si sta assumendo personale, probabilmente è più facile trovare un programmatore di fogli di calcolo esperto che un programmatore Matlab. Quindi, se non si è sicuri di quale usare, il consiglio è quello di provare entrambi i metodi e decidere da soli.

Per i programmatori Python, ci sono [scipy.optimize.minimize](#) e [LMFit packages](#), un'estensione del metodo [Levenberg-Marquardt](#). Vedere pagina 432 per un confronto Matlab/Python sul fitting iterativo.

Sono disponibili numerosi componenti aggiuntivi e macro per l'approssimazione iterativa non lineare, di curve scaricabili per [Excel](#) e [OpenOffice](#). Per esempio, il [Dr. Roger Nix](#) della Queen Mary University di Londra ha sviluppato uno [spreadsheet Excel/VBA](#) molto carino per l'approssimazione dei dati della spettroscopia photoelettronica a raggi X (XPS), ma potrebbe essere utilizzato per approssimare altri tipi di dati spettroscopici. Viene fornito anche un foglio di 4 pagine di istruzioni.

Ci sono anche molti esempi di programmi stand-alone, sia [freeware](#) che commerciali, tra cui [PeakFit](#), [Data Master 2003](#), [MyCurveFit](#), [Curve Expert](#), [Origin](#), [ndcurvemaster](#) e il [linguaggio R](#).

Matlab e Octave

Matlab e Octave hanno una comoda ed efficiente funzione chiamata "[fminsearch](#)" che utilizza il metodo Nelder-Mead. È stata originariamente progettata per trovare i valori minimi di funzioni, ma può essere applicata all'approssimazione dei quadrati minimi creando una cosiddetta [funzione anonima](#) (ovvero "una funzione *lambda*") che calcola il modello, lo confronta con i dati e restituisce l'errore di approssimazione. Ad esempio, scrivendo **parameters** =
fminsearch(@(lambda)(fitfunction(lambda, x, y)), start) si esegue un'approssimazione iterativa dei dati nei vettori x,y a un modello descritto in una funzione creata precedentemente chiamata **fitfunction**, utilizzando le ipotesi iniziali nel vettore "start". I parametri dell'approssimazione migliore nel vettore "parameters", nello stesso ordine con cui appaiono in "start".

Nota: Per gli utenti Octave, la funzione fminsearch è contenuta nel pacchetto aggiuntivo "Optim" (usare l'ultima versione 1.2.2 o successive), scaricabile da [Octave Forge](#). È una buona idea installare tutti questi pacchetti aggiuntivi nel caso in cui siano necessari; seguire le istruzioni sulla pagina web di [Octave Forge](#). Per gli utenti Matlab, fminsearch è una funzione nativa, sebbene ci siano altre routine di ottimizzazione nell'Optimization Toolbox facoltativo, che non è necessario per gli esempi e i programmi in questo documento.

Un semplice esempio è l'approssimazione dell'[equazione del corpo nero](#) allo spettro di un corpo incandescente per stimarne la temperatura del colore. In questo caso, c'è solo *un* parametro non-lineare, la temperatura. Lo script [BlackbodyDataFit.m](#) mostra la tecnica, posizionando lo spettro misurato sperimentalmente nei vettori "wavelength" e "radiance" e quindi chiamando fminsearch con la funzione di approssimazione [fitblackbody.m](#). Le lampadine a incandescenza, del tipo che era comune nell'illuminazione domestica prima dei LED, sono esempi di radiatori a corpo nero. (Se una sorgente di corpo nero non è termicamente omogenea, potrebbe essere possibile modellarla come somma di due o più regioni di diversa temperatura, come nell'esempio 3 di [fitshape1.m](#)).

Un'altra applicazione è dimostrata dalla demo nativa di Matlab [fitdemo.m](#) e dalla sua corrispondente funzione di approssimazione [fitfun.m](#), che modellano la somma di due decadimenti esponenziali. Per vederlo, digitare "fitdemo" nella finestra dei comandi di Matlab. (Octave non ha questa funzione demo).

Approssimazione dei picchi

Molti metodi di misura strumentali producono segnali sotto forma di picchi di varie forme; una richiesta comune è quella di misurare posizioni, altezze, larghezze, e/o aree di tali picchi, anche quando sono rumorosi o sovrapposti tra loro. Questo non può essere fatto con i metodi dei minimi quadrati lineari, perché tali segnali non possono essere modellati come polinomi con coefficienti lineari (le posizioni e le larghezze dei picchi non sono funzioni lineari), quindi vengono invece utilizzate tecniche di approssimazione iterativa, spesso usando Gaussiane, Lorentziane, o alcune

altri profili semplici fondamentali come modello.

Lo script dimostrativo Matlab/Octave [Demofitgauss.m](#) illustra l'approssimazione di una funzione Gaussiana ad un insieme di dati, utilizzando la funzione di approssimazione [fitgauss.m](#). In questo caso, ci sono due parametri non lineari: la posizione del picco e la sua larghezza (l'altezza è un parametro lineare ed è determinata dalla regressione in un unico passaggio nella riga 9 della funzione di approssimazione [fitgauss.m](#) e viene restituita nella variabile globale "c"). Rispetto ai metodi polinomiali più semplici dei quadrati minimi per misurare i picchi (pagina 165), il metodo iterativo ha il vantaggio di utilizzare tutti i punti sull'intero picco, compresi lo zero e i punti negativi, e può essere applicato a più picchi sovrapposti come mostrato nello script [Demofitgauss2.m](#).

Per accogliere la possibilità che la linea di base possa spostarsi, si può aggiungere una colonna di tutti 1 alla matrice A, [proprio come è stato fatto nel metodo CLS](#) (pag. 180). Questo ha l'effetto di introdurre nel modello una componente aggiuntiva che è semplicemente una linea piatta; la sua ampiezza viene restituita insieme alle altezze dei picchi nel vettore globale "c"; [Demofitgaussb.m](#) e [fitgauss2b.m](#) illustrano questa aggiunta. ([Demofitlorentzianb.m](#) e [fitlorentzianb.m](#) per i picchi Lorentziani).

Questa tecnica di approssimazione dei picchi può essere facilmente estesa a qualsiasi numero di picchi sovrapposti dello stesso tipo utilizzando la *stessa* funzione fitgauss.m, che si adatta facilmente a qualsiasi numero di picchi, a seconda della lunghezza del vettore della prima ipotesi "start" *lambda* passato alla funzione come argomenti di input, insieme ai vettori dei dati *t* e *y*:

```
1 function err = fitgauss(lambda, t,y)
2 % Fitting functions for a Gaussian band spectrum.
3 % T. C. O'Haver. March 2006
4 global c
5 A = zeros(length(t),round(length(lambda)/2));
6 for j = 1:length(lambda)/2,
7     A(:,j) = gaussian(t, lambda(2*j-1),lambda(2*j))';
8 end
9 c = A\y'; % c = abs(A\y') for positive peak heights only
10 z = A*c;
11 err = norm(z-y');
```

Se nel modello ci sono *n* picchi, la lunghezza della *lambda* è $2n$, una voce per ciascuna variabile iterata ([position1 width1 position2 width2....ecc.]). Il ciclo "for" (righe 5-7) costruisce una matrice $n \times \text{length}(t)$ contenente il modello per ciascun picco separatamente, utilizzando una funzione utente per il profilo del picco (in questo caso [gaussian.m](#)), poi calcola il vettore lungo *n* delle altezze *c* tramite la [regressione dei quadrati minimi](#) nella riga 9, utilizzando la [notazione abbreviata di Matlab](#) "`\`". (Per vincolare l'approssimazione a valori *positivi* delle altezze dei picchi, sostituire $A\y'$ con $\text{abs}(A\y')$ nella riga 9). I risultati delle altezze dei picchi vengono utilizzati per calcolare *z*, la somma di tutti gli *n* profili, con la [moltiplicazione matriciale](#) nella riga 10, e poi calcola "err", la differenza quadratica media tra il modello *z* e i dati effettivi *y*, nella riga 11 con la funzione Matlab 'norm' e restituito alla funzione chiamante ('fminsearch'), che ripete il processo molte volte, provando diversi valori delle posizioni e delle larghezze dei picchi finché il valore di "err" non è sufficientemente basso.

Questa funzione di approssimazione sarebbe chiamata dalla funzione fminsearch di Matlab in questo modo:

```
params=fminsearch(@(lambda)(fitgauss(lambda, x,y)),[50 20])
```

dove le parentesi quadre contengono un vettore con le ipotesi iniziali per posizione e larghezza di ciascun picco ([position1 width1 position2 width2....ecc.]). L'argomento di output 'params' restituisce una matrice $2 \times n$ con le migliori posizioni e larghezze per ciascun picco, mentre le

altezze dei picchi sono contenute nella vettore globale `c`, lungo `n`. Si possono definire funzioni di approssimazione simili per altre forme di picco semplicemente chiamando la corrispondente funzione del profilo del picco, come [lorentzian.m](#) nella riga 7. (Nota: affinché questo e altri script come Demofitgauss.m o Demofitgauss2.m funzionino sulla propria versione di Matlab, tutte le funzioni che chiamano devono essere anticipatamente caricate in Matlab, in questo caso fitgauss.m e gaussian.m. Gli script che chiamano sotto-funzioni devono avere tali funzioni nel path di ricerca di Matlab. Le funzioni, al contrario, possono avere tutte le loro sotto-funzioni richieste definite all'interno della funzione principale stessa e quindi possono essere autonome, come lo sono i prossimi due esempi).

Funzione semplificata di approssimazione del picco per scopi generali

La funzione [fitshape2.m](#) (sintassi: `[Positions, Heights, Widths, FittingError] = fitshape2(x, y, start)`) estraе tutto questo insieme in una *funzione* semplificata generica di Matlab/Octave per approssimare più tipi di profilo sovrapposti ai dati contenuti nelle variabili vettoriali `x` e `y`. Il modello è la somma ponderata di un numero qualsiasi di profili elementari che sono definite matematicamente in funzione di `x`, con due variabili che il programma determinerà indipendentemente per ciascun picco - posizioni e larghezze - oltre alle altezze (cioè i pesi della somma ponderata). Si deve fornire il vettore iniziale di prima ipotesi 'start', nella forma [posizione1 larghezza1 posizione2 larghezza2 ...ecc.], che specifica la prima ipotesi di posizione e larghezza per ciascuna componente (una coppia di posizione e larghezza per ciascun picco nel modello). La funzione restituisce i parametri dell'approssimazione migliore nei vettori `Positions`, `Heights`, `Widths`, e calcola l'errore percentuale tra i dati e il modello in `FittingError`. Inoltre disegna i dati come punti e il modello approssimato come una linea.

La cosa interessante di questa funzione è che *l'unica parte che definisce la forma del modello è l'ultima riga*. In fitshape2.m, quella riga contiene inizialmente l'espressione per un *picco Gaussiano di altezza unitaria*, ma o si può cambiare in *qualsiasi espressione o algoritmo* che calcola una unità-altezza `g` come una funzione di `x` con due parametri ignoti 'pos' e 'wid' (posizione e larghezza, rispettivamente, per le forme dei picchi, ma potrebbero rappresentare qualsiasi cosa per altri tipi di funzione, come l'impulso esponenziale, sigmoidale, ecc.); tutto il resto nella funzione fitshape.m può rimanere lo stesso. Tutto sta nell'ultima riga. Questo rende fitshape.m una buona piattaforma per sperimentare diverse espressioni matematiche come modelli proposti per approssimare i dati. Ci sono anche altre due varianti di questa funzione per i modelli con *una* variabile iterata più l'altezza del picco ([fitshape1.m](#)) e *tre* variabili iterate più l'altezza del picco ([fitshape3.m](#)). Ognuna ha esempi illustrativi nel file della guida (digitare "help fitshape..."). **NEW** Una versione alternativa è [FitMultipleShapes2](#), che consente di specificare una qualsiasi delle 16 funzioni del profilo del picco in base al numero. Gli esempi sono nel file help interno. La sintassi è:

```
[Positions,Heights,Widths,FittingError]=FitMultipleShapes2(x,y,shape,start,m)
```

Tipi di forma variabile

Il [profilo Voigt](#), Pearson, [Breit-Wigner-Fano](#), la mix Gauss-Lorentz e le Gaussiane e Lorentziane espanso esponenzialmente sono definiti non solo dalla posizione del picco, altezza e larghezza, ma anche da *un parametro aggiuntivo che ottimizza la forma del picco*. Se quel parametro è *uguale e noto* per tutto un gruppo di picchi, lo si può passare come argomento di input aggiuntivo alla funzione del profilo, come mostrato nella funzione demo [VoigtFixedAlpha.m](#) per il profilo Voigt, che viene calcolato come una convoluzione di componenti Gaussiane e Lorentziane con larghezze differenti. Se il parametro della forma (*alpha*, il rapporto delle due larghezze) può essere *diverso* per ciascun picco del gruppo e deve essere determinato per iterazione (proprio come la posizione e la

larghezza), allora la routine deve essere modificata per accogliere *tre*, anziché *due*, variabili iterate, come mostrato nella funzione demo [VoigtVariableAlpha.m](#). Sebbene l'errore di approssimazione sia più basso con le variabili alfa, il tempo di esecuzione è più lungo e i valori alfa così determinati non sono molto stabili, rispetto al rumore nei dati e ai valori delle ipotesi iniziali, soprattutto per picchi multipli. (Queste sono funzioni autonome [self-contained]). La versione 9.5 della funzione Matlab/Octave generica [peakfit.m](#) include tipi di profili fissi e variabili per Pearson, ExpGaussian, Voigt e mix di Gaussiana/Lorentziana, nonché la logistica a 3-parametri o funzione di Gompertz (i cui tre parametri sono chiamati Bo, Kh e L, anziché posizione, larghezza e fattore di forma). Lo script [VoigtShapeFittingDemonstration.m](#) utilizza la versione 9.5 di peakfit.m per approssimare un singolo profilo Voigt e per calcolare la componente larghezza della Gaussiana, della Lorentziana e l'alfa. Calcola il profilo Voigt teorico con l'aggiunta di rumore casuale per il realismo. Lo script [VoigtShapeFittingDemonstration2.m](#) fa lo stesso per *due* profili Voigt sovrapposti, utilizzando sia il modello con l'alfa fisso che quello con l'alfa variabile (numeri della forma 20 e 30). (Richiede voigt.m, halfwidth.m e peakfit.m nel path di ricerca di Matlab). Lo script [GLBlendComparison](#) confronta il Voigt col più semplice mix Gauss/Lorentziana, mostrando che sono quasi identici con una differenza relativa dello 0.3%.

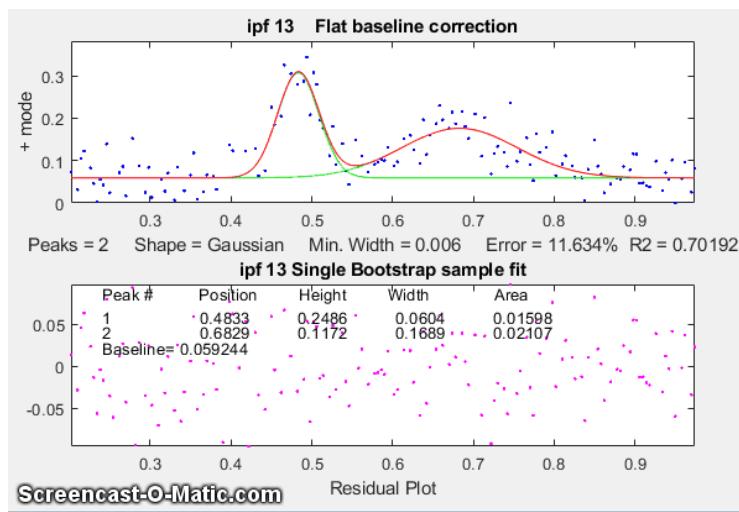
Se *non* si conosce la forma dei picchi, si può usare peakfit.m o ipf.m (pagina 405) per provare profili diversi per vedere se una delle le forme standard incluse in quei programmi approssimano i dati; cercare di trovare un picco nei dati che sia tipico, isolato e che abbia un buon rapporto segnale/rumore. Per esempio, le funzioni Matlab [ShapeTestS.m](#) e [ShapeTestA.m](#) testano i dati nei loro argomenti di input x, y, assumendo che sia un singolo picco isolato, lo approssimano con *diversi modelli candidati* utilizzando peakfit.m, disegnare ciascuna approssimazione in una finestra propria e stampare una tabella con gli errori di approssimazione nella finestra dei comandi.

[ShapeTestS.m](#) prova sette diversi modelli di picco *simmetrici* e [ShapeTestA.m](#) ne prova sei *asimmetrici*. Quello con l'errore di approssimazione più basso (e R^2 più prossimo a 1.000) è probabilmente il candidato migliore. [Provare gli esempi](#) nei file di help per ciascuna di queste funzioni. Ma attenzione, se c'è troppo rumore nei dati, i risultati possono essere fuorvianti. Ad esempio, anche se la forma effettiva del picco è qualcosa di diverso da una Gaussiana, è probabile che il modello delle Gaussiane multiple si approssima leggermente meglio perché ha più gradi di libertà e può "approssimare il rumore". La funzione Matlab peakfit.m ha molte più forme tra cui scegliere, ma è ancora un elenco *finito* e c'è sempre la possibilità che il profilo cercato non sia disponibile nel software che si sta utilizzando o che semplicemente non è descrivibile da una funzione matematica.

I segnali con *picchi di forme diverse in un segnale* possono essere approssimati dalla funzione [fitmultiple.m](#), che prende come argomenti di input un vettore dei tipi di picco e un vettore per i profili. La sequenza dei tipi di picco e dei profili deve essere determinata in anticipo. Per vedere come viene utilizzato, eseguire [Demofitmultiple.m](#).

Si possono creare le proprie funzioni di approssimazione per qualsiasi scopo; *non* sono limitate a singole espressioni algebriche ma possono essere algoritmi multi-passo arbitrariamente complessi. Per esempio, nel metodo TFit per la spettroscopia quantitativa di assorbimento (pag. 268), un modello di spettro di trasmissione ampliato strumentalmente è approssimato ai dati dello spettro di trasmissione osservato, utilizzando una [funzione di approssimazione](#) che esegue la [convoluzione di Fourier](#) (pag. 103) del modello di spettro di trasmissione con la funzione nota di fenditura dello spettrometro. Il risultato è un metodo alternativo di calcolo dell'assorbanza che consente l'ottimizzazione del rapporto segnale-rumore ed estende la gamma dinamica e la linearità della calibrazione della spettroscopia di assorbimento ben oltre i limiti normali.

Funzioni di Fitting per Matlab e Octave



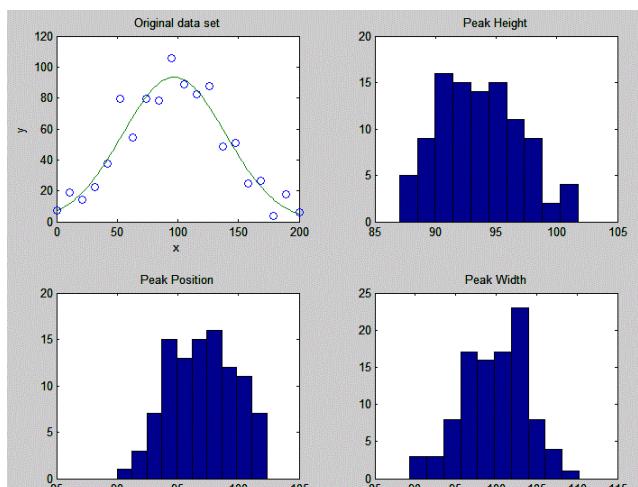
Qui vengono descritte funzioni per l'approssimazione dei picchi più complete con funzionalità aggiuntive: un set integrato di profili elementari selezionabile, un vettore "start" per le ipotesi iniziali, se non ne fornisce uno, una gestione delle linee di base, la possibilità di stimare le incertezze nei parametri dei picchi, ecc. Queste funzioni accettano segnali di qualsiasi lunghezza, compresi quelli con valori x non interi e non uniformi, e possono approssimare qualsiasi numero di picchi

con forme Gaussiane, Gaussiane di uguale larghezza, Gaussiane a larghezza fissa, Gaussiane esponenzialmente allargate, Gaussiane esponenzialmente allargate di uguale larghezza, Gaussiane biforcate, Lorentziane, Lorentziane a larghezza fissa, Lorentziane di uguale larghezza, Lorentziane esponenzialmente allargate, Lorentziane biforcate, distribuzione logistica, funzione logistica, triangolare, funzione alfa, Pearson 7, impulso esponenziale, sigmoide in su, sigmoide in giù, mix di Gaussiane/Lorentziane, Breit-Wigner-Fano e Voigt. ([Un grafico](#) che mostra i profili di base disponibili è a pagina 411).

Ci sono due diverse versioni, una [a riga di comando](#) denominata [peakfit.m](#), per Matlab o [Octave](#) e una [interattiva operante da tastiera](#) chiamata [ipf.m](#) o [ipfoctave.m](#) (pagina 405). Per aggiungere come elemento nei propri programmi e per automatizzare l'approssimazione di un gran numero di segnali, [peakfit.m](#) è migliore; ipf.m è l'ideale per esplorare alcuni segnali per determinarne il migliore intervallo di approssimazione, i profili, il numero di picchi, la modalità di correzione della linea di base, ecc. Entrambe le funzioni consentono operazioni semplificate fornendo valori di default per qualsiasi argomento di input non specificato; ad esempio, i valori iniziali, se non specificati negli argomenti di input, sono stimati dal programma in base alla lunghezza e all'intervallo dell'asse x dei dati. Rispetto alla funzione fitshape.m descritta sopra, [peakfit.m](#) ha un gran numero di profili integrati, non richiede (sebbene possa essere fornita) la prima ipotesi della posizione e della larghezza di ogni componente e dispone di funzioni per la correzione del background e altre funzioni utili per migliorare la qualità e l'affidabilità delle approssimazioni. Entrambe queste funzioni sono state ampiamente testate su dati sperimentali reali da centinaia di ricercatori in molti campi diversi).

Queste funzioni possono facoltativamente stimare la deviazione standard attesa e l'intervallo interquartile dei parametri di picco utilizzando il [metodo bootstrap di campionamento](#) (pagina 162). Vedere [DemoPeakfitBootstrap](#) per una dimostrazione autonoma [self-contained] di questa funzione.

L'effetto del rumore casuale sull'incertezza dei parametri di picco determinata dall'approssimazione iterativa dei quadrati minimi è prontamente stimata dal [metodo bootstrap di campionamento](#) (presentato a



pagina 162). Una semplice dimostrazione della stima bootstrap della variabilità di un'approssimazione dei quadrati minimi iterativa a un singolo picco Gaussiano rumoroso è data dalla funzione Matlab/Octave "[BootstrapIterativeFit.m](#)", che crea un singolo set di dati x, y costituito da un singolo picco Gaussiano rumoroso, estrae campioni di bootstrap da quel set di dati, esegue un'approssimazione iterativa al picco su ciascuno dei campioni di bootstrap e disegna le distribuzioni (istogrammi) delle altezze, posizione e larghezza dei campioni di bootstrap. La sintassi è `BootstrapIterativeFit(TrueHeight, TruePosition, TrueWidth, NumPoints, Noise, NumTrials)` dove `TrueHeight` è la reale altezza del picco Gaussiano, `TruePosition` è il valore reale sull'asse x del massimo, `TrueWidth` è la semi-larghezza reale (FWHM), `NumPoints` è il numero di punti presi per l'approssimazione ai quadrati minimi, `Noise` è la deviazione standard del rumore casuale (normalmente distribuito) e `NumTrials` è il numero di campioni bootstrap.

Un tipico esempio di `BootstrapIterativeFit(100,100,100,20,10,100)`; viene visualizzato nella figura a lato, sopra. I risultati, visualizzati nella finestra di comando, sono:

```
>> BootstrapIterativeFit(100,100,100,20,10,100);
Peak Height Peak Position Peak Width
mean: 99.27028 100.4002 94.5059
STD: 2.8292 1.3264 2.9939
IQR: 4.0897 1.6822 4.0164
IQR/STD Ratio: 1.3518
```

Una funzione dimostrativa simile per *due* picchi Gaussiani sovrapposti è disponibile in "[BootstrapIterativeFit2.m](#)". Digitare "help BootstrapIterativeFit2" per ulteriori informazioni. In entrambe queste simulazioni, vengono calcolate la deviazione standard (STD), nonché l'[intervallo interquartile](#) (IQR) di ciascuno dei parametri del picco. Questo viene fatto perché l'intervallo interquartile è molto meno influenzato dai *valori anomali*. La distribuzione dei parametri di picco misurati dall'approssimazione iterativa è spesso non-normale, mostrando una frazione maggiore di grandi deviazioni dalla media rispetto a quanto previsto per una distribuzione normale. Questo perché la procedura iterativa a volte converge su un risultato anormale, specialmente per approssimazioni di picchi multipli con molti parametri variabili. (Lo si potrebbe vedere negli istogrammi disegnati da queste simulazioni, specialmente per il picco più debole in [BootstrapIterativeFit2](#)). In questi casi, la deviazione standard sarà troppo alta a causa dei valori anomali, e il rapporto IQR/STD sarà molto inferiore al valore di 1,34896 previsto per una distribuzione normale. In tal caso, una stima migliore della deviazione standard della porzione centrale della distribuzione (senza i valori anomali) è IQR/1.34896.

È importante sottolineare che il [metodo bootstrap](#) predice solo l'effetto del rumore casuale sui parametri del picco per un modello fissato dell'approssimazione. Non considera la possibilità di imprecisioni dei parametri causate dall'utilizzo di un intervallo di dati non ottimale, o dalla scelta di un modello imperfetto, o da una compensazione imprecisa per il background/linea di base, che sono tutti almeno parzialmente soggettivi e quindi oltre la gamma di influenze che possono essere facilmente trattate da statistiche casuali. Se i dati hanno un rumore casuale relativamente piccolo o hanno subito uno smoothing per ridurre il rumore, è probabile che la selezione del modello e la correzione della linea di base saranno le principali fonti di imprecisione per i parametri del picco, che non sono ben previste dal metodo bootstrap.

Per la misura quantitativa dei picchi, è istruttivo confrontare il metodo iterativo del "least-squares" con i metodi più semplici, e meno impegnati computazionalmente. Per esempio, la misura dell'altezza di un singolo picco di larghezza e posizione incerta si potrebbe fare prendendo semplicemente il massimo del segnale in quella regione. Se il segnale è rumoroso, si otterrà

un'altezza del picco più precisa se il segnale viene prima filtrato con lo [smoothing](#) (pag. 38). Ma lo smoothing può distorcere il segnale e ridurre le altezze dei picchi. L'uso di un metodo iterativo di approssimazione, assumendo solo che la forma del picco sia nota, può fornire la migliore accuratezza possibile e precisione, senza richiedere lo smoothing anche in condizioni di rumore elevato, ad es. quando il rapporto segnale / rumore è 1, come nello script demo [SmoothVsFit.m](#):

```
True peak height = 1    NumTrials = 100  SmoothWidth = 50
```

Method	Maximum y	Max Smoothed y	Peakfit
Average peak height	3.65	0.96625	1.0165
Standard deviation	0.36395	0.10364	0.1157

Se viene misurata l'*area* del picco anziché l'*altezza*, non è necessario lo smoothing (a meno che non si localizzi l'inizio e la fine del picco), ma l'approssimazione fornisce comunque la migliore precisione. Vedere [SmoothVsFitArea.m](#).

È istruttivo anche confrontare il metodo iterativo dei minimi quadrati con l'*approssimazione dei minimi quadrati classica*, discussa a pagina 179, che può approssimare anche picchi in un segnale. La differenza è che nel metodo dei minimi quadrati classico, le posizioni, le larghezze e le forme di tutte le singole componenti sono tutte note in anticipo; le sole incognite sono le amplificazioni (p.es. le altezze) delle componenti della miscela. Nell'approssimazione iterativa non-lineare, d'altra parte, le posizioni, le larghezze e le altezze dei picchi sono *tutte ignote* a priori; la sola cosa nota è il *profilo* base dei picchi. L'approssimazione iterativa non lineare della curva è più difficile da eseguire (per il computer, in ogni caso) e più soggetta a errori, ma è necessario se si deve tener traccia degli spostamenti nella posizione o della larghezza del picco o per scomporre un segnale complesso con picchi sovrapposti in componenti fondamentali conoscendo solo la loro forma. Lo script Matlab/Octave “[CLSvsINLS.m](#)” confronta il metodo dei minimi quadrati classico (CLS) con diverse varianti del metodo iterativo (INLS) per misurare le altezze di tre picchi Gaussiani in un segnale di test rumoroso su un normale PC Windows, a dimostrazione che minore è il numero di parametri noti, più veloce e accurato è il calcolo dell'altezza del picco.

Method	Positions	Widths	Execution time	% Accuracy
CLS	known known	0.00133	0.30831	
INLS	unknown unknown	0.61289	0.6693	
INLS	known unknown	0.16385	0.67824	
INLS	unknown known	0.24631	0.33026	
INLS	unknown known (equal)	0.15883	0.31131	

Un altro confronto tra più tecniche di misurazione è presentato nel [Caso di Studio D](#) (pagina 291).

Nota: Se si sta leggendo online, si può fare click destro del mouse su uno qualsiasi dei link dei file m e selezionare **Save Link As...** per scaricarli nel proprio computer, e poi metterli nel path di ricerca di Matlab per usarli.

Accuratezza e precisione dei parametri del picco

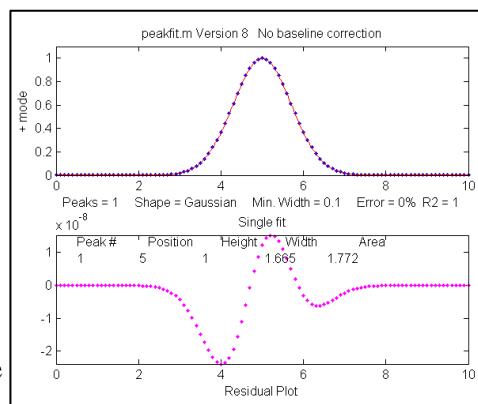
Il curve fitting iterativo viene spesso utilizzato per misurare posizione, altezza e larghezza dei picchi in un segnale, specialmente quando sono molto sovrapposti. Ci sono quattro principali fonti di errore nella misura iterativa di tali picchi. Questa sezione fa uso della funzione [peakfit.m](#). Le istruzioni sono [qui](#) oppure digitare "help peakfit". (Una volta che c'è peakfit.m nel path di ricerca, si può copiare, incollare, trascinare qualsiasi dei seguenti esempi di codice a singola o più righe nell'editor di Matlab o di Octave o nella finestra di comando e premere **Enter** per eseguirlo).

a. Modelli errati.

Profilo del picco. Se si ha un modello sbagliato per il picco, non ci si può aspettare che i risultati siano accurati; per esempio, quando i picchi effettivi sono di forma Lorentziana, ma si approssima con un modello Gaussiano o *viceversa*. Ad esempio, un singolo picco Gaussiano isolato a $x = 5$, con un'altezza di 1.000 virtualmente si approssima perfettamente a un modello Gaussiano, utilizzando la funzione Matlab [peakfit](#) (pagina 384), come mostrato a lato. (Il 5° argomento di input per la funzione peakfit indica il profilo del picco da usare nell'approssimazione; "1" significa Gaussiana. Vedere pagina 386 per l'elenco dei numeri dei profili).

```
>> x=[0:.1:10];y=exp(-(x-5).^2);
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,1,1)
Peak# Position Height Width Area
1 5 1 1.6651 1.7725
MeanFitError = 7.8579e-07 R2= 1
```

I risultati in "FitResults" sono, da sinistra a destra, il numero del picco, la posizione, l'altezza, l'ampiezza e l'area del picco. Il MeanFitError, o semplicemente "errore di approssimazione", è la radice quadrata della somma dei quadrati delle differenze tra i dati e il modello, come percentuale del segnale massimo nella regione approssimata. Le versioni recenti di peakfit restituiscono anche l'R2, "R-al quadrato" o coefficiente di determinazione, che è esattamente 1 per un'approssimazione perfetta. Notare l'accordo tra l'area (1.7725) con quella *teorica* della curva di $\exp(-x^2)$, che è la [radice quadrata di pi greco](#). Se si sta leggendo online, cliccare per la [soluzione di Wolfram Alpha](#). Ma questo stesso picco, se approssimato col modello errato (un modello *Logistico*, profilo numero 3), fornisce un errore di approssimazione dell'1,4% ed errori di altezza e larghezza del 3% e del 6%, rispettivamente. Tuttavia, l'errore dell'area è solo dell'1,7%, poiché gli errori di altezza e larghezza si annullano parzialmente. Quindi non è necessario disporre di un modello perfetto per ottenere una misurazione dell'area decente.



```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,1,3)
```

Peak#	Position	Height	Width	Area

```
1 5.0002 0.96652 1.762 1.7419
```

```
MeanFitError =1.4095
```

Quando si approssima con un modello *Lorentziano* ancora più sbagliato (profilo numero 2, mostrato a lato, di seguito), questo picco restituisce un errore di approssimazione del 7% e per altezza, larghezza e area, 8%, 20% e 17%, rispettivamente.

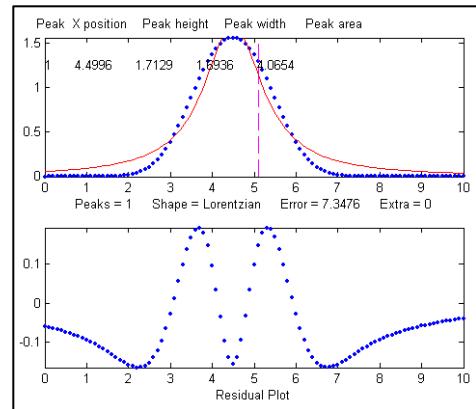
```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,1,2)
```

Peak#	Position	Height	Width	Area
1	5	1.0876	1.3139	2.0579

Peak#	Position	Height	Width	Area
1	5	1.0876	1.3139	2.0579

```
MeanFitError =5.7893
```

Ma è improbabile che la stima di un modello sia così lontana; è molto più probabile che i picchi effettivi siano una combinazione ignota di forme di picco, come la Gaussiana mescolata con un po' di Lorentziana o viceversa, oppure qualche modifica leggermente asimmetrica di una forma simmetrica standard. Quindi, se utilizzi un modello disponibile che è almeno *vicino* al profilo effettivo, gli errori dei parametri potrebbero non essere così gravi e potrebbero, in effetti, essere migliori di altri metodi di misura.



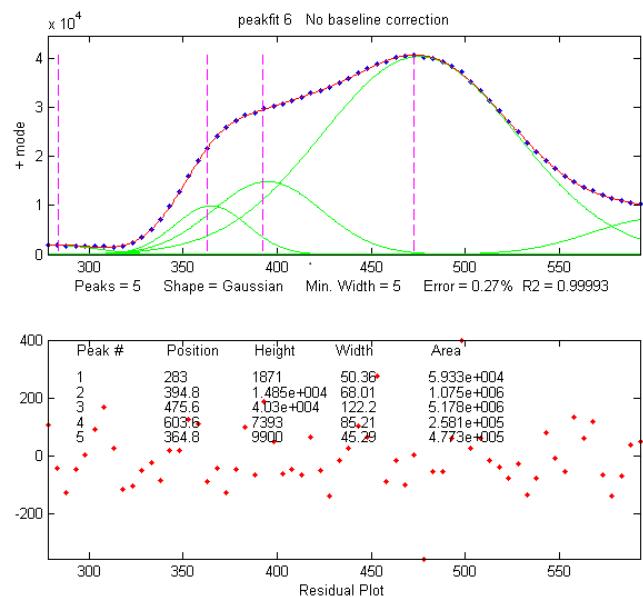
Quindi chiaramente più grandi sono gli errori di approssimazione, maggiori sono gli errori dei parametri, ma gli errori dei parametri ovviamente non sono *uguali* a quello dell'approssimazione (sarebbe *troppo* facile). Inoltre, l'*altezza* e la *larghezza* sono i parametri più soggetti ad errori. Le *posizioni* vengono misurate accuratamente anche se il modello è sbagliato, se il picco è simmetrico e non è troppo sovrapposto ad altri picchi.

Una buona approssimazione non è di per sé la prova che la funzione del profilo scelto sia quella corretta. In alcuni casi, la funzione sbagliata può dare un'approssimazione che sembra perfetta. Ad esempio, il grafico sopra a lato mostra [un'approssimazione a dei dati reali](#) con un modello di 5 picchi Gaussiana che mostra una percentuale minima degli errori residui che sembrano casuali, di solito un indicatore di una buona approssimazione. Ma in effetti, *in questo caso il modello è sbagliato*; i dati provengono da un dominio sperimentale dove il profilo in esame è *fondamentalmente non-Gaussiano* ma, in alcuni casi, è molto simile ad una Gaussiana. Come altro esempio, un insieme di dati costituito da picchi con un [profilo Voigt](#) si può approssimare con una [somma pesata di una Gaussiana e una Lorentziana](#) quasi come un vero [modello Voigt](#), anche se questi modelli *non sono matematicamente uguali*; la differenza nell'errore di approssimazione è così piccola che verrebbe probabilmente [oscurata dal rumore casuale](#) se fosse un segnale sperimentale reale. Lo script [GLBlendComparison](#) confronta il Voigt col più semplice mix Gauss/Lorentziana, mostrando che sono quasi identici con una differenza relativa dello 0.3%. La stessa cosa può accadere nelle forme del segnale sigmoidale: una coppia di semplici funzioni logistiche a 2 parametri sembra abbastanza bene approssimare [questi dati di esempio](#), con un errore di approssimazione inferiore all'1%; non ci sarebbe motivo per dubitare della bontà dell'approssimazione a meno che il rumore casuale non sia sufficientemente basso tale da poter vedere che i residui sono ondulati. Infatti, una logistica a 3 *parametri* (funzione [Gompertz](#)) [si adatta molto meglio](#) e i residui sono casuali, non ondulati, indicando

una migliore approssimazione. In questi casi non è possibile dipendere esclusivamente da ciò che *sembra* una buona approssimazione per determinare se l'approssimazione è col modello ottimale; a volte è necessario saperne di più sulla forma del picco prevista in quel tipo di esperimento, soprattutto se i dati sono rumorosi. Nella migliore delle ipotesi, se si ottiene una buona approssimazione con residui casuali non ondulati, si può solo affermare che i dati *sono coerenti col* modello proposto.

In alcune applicazioni l'accuratezza del modello *non* è così importante. Si prenda l'esempio delle *applicazioni di analisi quantitativa*, dove le altezze dei picchi o le aree misurate mediante approssimazione vengono utilizzate per determinare la *concentrazione della sostanza che ha creato il picco* costruendo una [curva di calibrazione](#) (pag. 439) in base a soluzioni standard preparate in laboratorio con concentrazioni note. In tal caso, la necessità di utilizzare il modello di picco esatto è minima, se la forma del picco sconosciuto è *costante e indipendente dalla concentrazione*. Se viene usato il modello sbagliato, la R^2 dell'approssimazione sarà scarsa (molto al di sotto di 1.000) e le altezze e le aree misurate saranno imprecise, ma l'*errore sarà lo stesso* per i campioni ignoti e per gli

standard di calibrazione, quindi l'errore verrà annullato. Di conseguenza, l' R^2 per la curva di calibrazione può essere molto alto (0.9999 o più) e le *concentrazioni misurate non saranno meno accurate di quanto sarebbero state con un modello perfetto*. Anche così, è utile usare la forma più accurata possibile, perché la R^2 dell'approssimazione funzionerà meglio da indicatore se qualcosa va storto durante l'analisi (come un aumento del rumore o la comparsa di un picco interferente da una sostanza estranea). Vedere [PeakShapeAnalyticalCurve.m](#) per una dimostrazione Matlab/Octave.



nel modello si verifica quando si ha il *numero dei picchi* errato, per esempio se il segnale ha effettivamente *due* picchi ma si tenta di approssimarne solo *uno*. Prendiamo prima un caso ovvio. Nell'esempio seguente, il codice Matlab genera un segnale con due picchi Gaussiani a $x = 4$ e $x = 6$ con altezze rispettivamente di 1.000 e 0.5000 e larghezze di 1.665, più del rumore casuale con una deviazione standard del 5% dell'altezza del picco più grande (un rapporto segnale-rumore di 20):

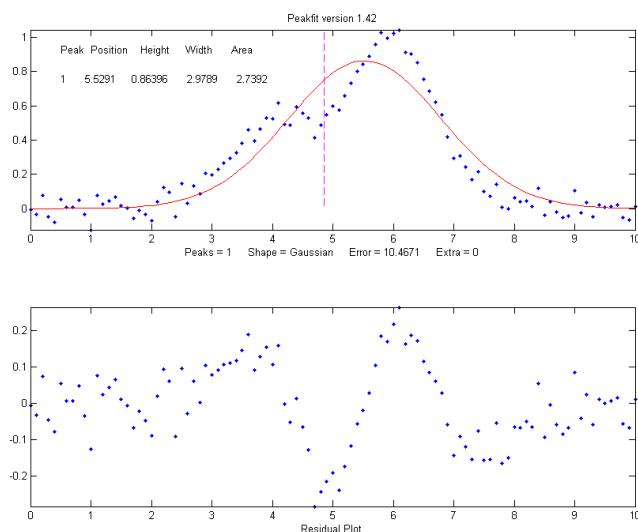
```
>> x=[0:.1:10];  
  
>> y=exp(-(x-6).^2)+.5*exp(-(x-4).^2)+.05*randn(size(x));
```

In un esperimento reale, normalmente non si conoscerebbero le posizioni, le altezze e le larghezze dei picchi; si userebbe l'approssimazione per *misurare* tali parametri. Si supponga che, sulla base dell'esperienza precedente o di alcune approssimazioni preliminari di prova, si sia stabilito che il *profilo* ottimale sia Gaussiano, ma non si sa con certezza quanti picchi ci sono in questo gruppo. -se si comincia approssimando questo segnale con un *singolo* picco Gaussiano, si ottiene:

```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,1,1)
```

Peak#	Position	Height	Width	Area
-------	----------	--------	-------	------

```
1 5.5291 0.86396 2.9789 2.7392
MeanFitError = 10.467
```



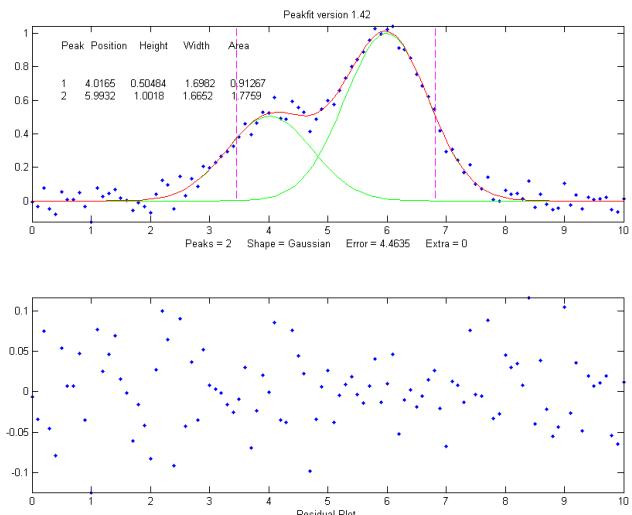
Ovviamente, questo non è giusto. Il grafico dei residui (pannello inferiore) mostra una struttura "ondulata" chiaramente visibile nella dispersione casuale dei punti a causa del rumore casuale nel segnale. Ciò significa che l'errore di approssimazione non è dovuto al solo rumore casuale; è un indizio che il modello non è del tutto completo.

Tuttavia, un'approssimazione con *due* picchi produce risultati molto migliori (il 4° argomento di input per la funzione peakfit indica il numero di picchi da utilizzare).

```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,2,1)
```

```
Peak# Position Height Width Area
1 4.0165 0.50484 1.6982 0.91267
2 5.9932 1.0018 1.6652 1.7759
```

MeanFitError = 4.4635



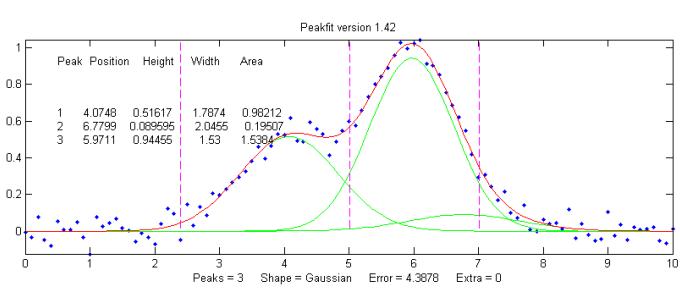
Ora i residui hanno una dispersione casuale dei punti, come ci si aspetterebbe se il segnale fosse stato accuratamente approssimato tranne che per il rumore casuale. Inoltre, l'errore di approssimazione è inferiore alla metà dell'errore con un solo picco. In effetti, l'errore di approssimazione è proprio quello che ci aspetteremmo in questo caso sulla base del rumore casuale del 5% nel segnale (stimando la deviazione standard relativa dei punti nella linea di base visibili ai bordi del segnale). Poiché si tratta di una simulazione in cui si conoscono in anticipo i

veri valori dei parametri (picchi in $x = 4$ e $x = 6$ con altezze rispettivamente di 1.0 e 0.50 e larghezze di 1.665), si possono calcolare gli errori dei parametri (la differenza tra le posizioni, le altezze e le larghezze reali dei picchi e i valori misurati). Si noti che sono abbastanza accurati (in questo caso entro circa l'1% relativo all'altezza del picco e il 2% sulle larghezze), *che è migliore del rumore casuale del 5% in questo segnale a causa dell'effetto medio dell'approssimazione a più punti nel segnale.*

Tuttavia, se passare da un picco a due picchi fornisce un'approssimazione migliore, perché non provare con *tre*? Se non ci fosse rumore nei dati e se la forma del picco fosse perfettamente coincidente col modello, l'errore di approssimazione sarebbe già stato essenzialmente nullo col modello a due picchi. L'aggiunta di un terzo picco al modello gli attribuirebbe un'altezza incredibilmente piccola. Ma negli esempi qui, come nei dati reali, c'è sempre del rumore casuale e il risultato è che l'altezza del terzo picco non sarà zero. Portando a tre il numero dei picchi si ottengono i seguenti risultati:

```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,3,1)
```

Peak# Position Height Width Area



1 4.0748 0.51617 1.7874
0.98212

2 6.7799 0.089595 2.0455
0.19507

3 5.9711 0.94455 1.53 1.5384

MeanFitError = 4.3878

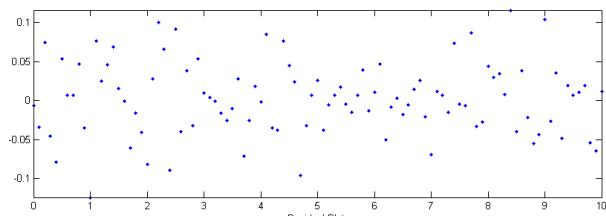
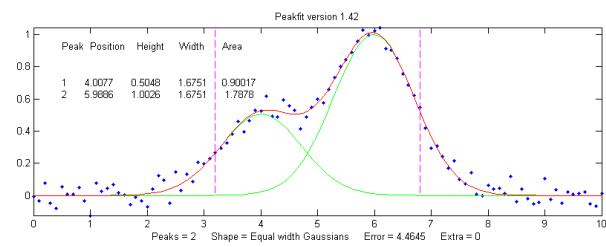
L'algoritmo ora ha cercato di approssimare un ulteriore picco di bassa ampiezza (picco numero 2 in questo caso) situato in $x = 6,78$. L'errore di approssimazione è inferiore a quello per i 2 picchi, ma solo leggermente, e i residui non, visivamente, meno casuali dell'approssimazione a 2 picchi.

Quindi, non sapendo nient'altro, un'approssimazione a 3 picchi potrebbe essere rifiutato solo su questa base. Infatti, c'è un grave svantaggio nell'approssimare più picchi di quelli presenti nel segnale: *aumentano gli errori dei parametri* nei picchi presenti. Ancora una volta, è possibile dimostrarlo perché so conoscono in anticipo i veri valori dei parametri dei picchi: chiaramente, le posizioni, le altezze e le larghezze dei due picchi reali che stanno nel segnale (picchi 1 e 3) sono significativamente meno accurate di quelli dell'approssimazione a 2 picchi.

Inoltre, se si ripete quell'approssimazione con lo *stesso segnale* ma con un *diverso campione* di rumore casuale (simulando una misura ripetuta di un segnale sperimentale stabile in presenza di rumore casuale), il terzo picco aggiuntivo nell'approssimazione a 3 picchi rimbalzerà dappertutto (perché questo si approssima al *rumore* casuale, non agli effettivi picchi nel segnale).

```
>> x=[0:.1:10];
```

```
>> y=exp(-(x-6).^2)+.5*exp(-((x-4).^2)+.05*randn(size(x));
```



```

>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,3,1)

Peak# Position Height Width Area
1 4.115  0.44767  1.8768  0.89442
2 5.3118  0.09340    2.6986  0.26832
3 6.0681  0.91085  1.5116  1.4657

MeanFitError = 4.4089

```

Con questi nuovi dati, due dei picchi (i numeri 1 e 3) hanno conservato all'incirca la stessa posizione, altezza e larghezza ma il picco numero 2 è notevolmente cambiato rispetto all'esecuzione precedente. Ora c'è una ragione ancora più convincente per rifiutare il modello a 3 picchi: la soluzione a 3 picchi *non è stabile*. E poiché questa è una simulazione in cui si conoscono le risposte giuste, si può verificare che l'accuratezza delle altezze dei picchi è sostanzialmente inferiore (circa il 10% di errore) del previsto con questo livello di rumore casuale nel segnale (5%). Se si dovesse eseguire un'approssimazione a 2 picchi sugli stessi nuovi dati, otterremmo misure molto migliori delle altezze.

```

>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,2,1)

Peak# Position Height Width Area
1 4.1601  0.49981  1.9108  1.0167
2 6.0585  0.97557  1.548   1.6076

MeanFitError = 4.4113

```

Se questo viene ripetuto più volte, risulta che i parametri dei picchi in $x = 4$ e $x = 6$ sono, in media, misurati in modo più accurato dall'approssimazione a 2 picchi. In pratica, il modo migliore per valutare un modello è quello di approssimare più misure ripetute dello stesso segnale (se ciò è fattibile sperimentalmente) e di calcolare la deviazione standard dei valori dei parametri del picco. Nel vero lavoro sperimentale, ovviamente, di solito non si *conoscono* le risposte giuste in anticipo, quindi è per questo che è importante usare metodi che funzionano bene quando le si *conoscono*. Ecco un esempio di un insieme di dati reali che corrispondeva a una successione di [2](#), [3](#), [4](#) e [5](#) Gaussiane, finché i residui non sono diventati casuali. Con ogni componente aggiunto, l'errore di approssimazione diventa più piccolo e i residui diventano più casuali. Ma oltre i 5 componenti, c'è poco da guadagnare aggiungendo più picchi al modello. Un altro modo per determinare il numero minimo di picchi necessari è disegnare l'errore di approssimazione rispetto al numero dei picchi nel modello; il punto in cui l'errore raggiunge un minimo per poi aumentare, sarebbe l'approssimazione con la "[combinazione ideale per avere la migliore approssimazione senza termini in eccesso o non necessari](#)" come si legge in Wikipedia. La funzione Matlab/Octave [testnumpeaks.m](#) (**R = te-**
stnumpeaks(x, y, peakshape, extra, NumTrials, MaxPeaks)) applica questa idea approssimando i dati x, y a una serie di modelli di forma peakshape contenente da 1 a MaxPeaks picchi nel modello. Il numero corretto di picchi è il modello con l'errore di approssimazione più basso oppure, se due o più modelli hanno circa lo stesso errore, il modello con il numero di picchi *minore*. Lo script demo Matlab/Octave [NumPeaksTest.m](#) utilizza questa funzione con segnali rumo-

rosi generati dal computer contenenti 3, 4, 5 o 6 picchi selezionati dall'utente. Con dati molto rumorosi, tuttavia, la tecnica non è sempre affidabile.

Vincoli sull'ampiezza dei picchi. Infine, c'è un'altra cosa che si può fare per migliorare la precisione della misurazione dei parametri e riguarda le larghezze dei picchi. In tutte le simulazioni precedenti, si è assunto che *tutti* i parametri dei picchi fossero ignoti e indipendenti gli uni dagli altri. In alcuni tipi di misure, però, ci si aspetta che le larghezze dei picchi di ciascun gruppo di picchi adiacenti siano tutte uguali, basandosi su principi o su esperimenti precedenti. Questa è una situazione comune nella chimica analitica, specialmente nella spettroscopia atomica e nella cromatografia, dove le larghezze dei picchi sono determinate in gran parte da fattori strumentali che sono gli stessi per tutti i picchi in una data regione.

Nella simulazione corrente, le larghezze reali dei picchi sono infatti pari a 1,665, ma tutti i risultati sopra mostrano che le larghezze *misurate* sono vicine ma non del tutto uguali, a causa al rumore casuale nel segnale. Le larghezze disuguali sono una conseguenza del rumore casuale, non delle vere differenze nell'ampiezza del picco. Ma si può introdurre un vincolo di *equi-larghezza* nell'approssimazione col profilo 6 (Gaussiane di pari larghezze) o col profilo 7 (Lorentziana di pari larghezze). Utilizzando il profilo 6 sullo stesso set di dati dell'esempio precedente:

```
>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,2,6)

Peak# Position Height Width Area

1 4.0293 0.52818 1.5666 0.8808

2 5.9965 1.0192 1.5666 1.6997

MeanFitError = 4.5588
```

Questa approssimazione con "larghezze uguali" forza tutti i picchi all'interno di un gruppo ad avere esattamente la stessa larghezza, ma tale larghezza è determinata dal programma in base ai dati. Il risultato è un errore di approssimazione *leggermente superiore* (in questo caso 4.5% anziché 4.4%), ma - forse sorprendentemente - le misurazioni dei parametri sono solitamente *più accurate* e *più riproducibili* (In particolare, le deviazioni standard relative sono in media inferiori per l'approssimazione a larghezza uguale rispetto a una a larghezza non vincolata per gli stessi dati, assumendo ovviamente che le vere larghezze dei picchi siano uguali). *Questa è un'eccezione* all'aspettativa generale che errori di approssimazione inferiori determinino errori inferiori dei parametri. È un'illustrazione della regola generale secondo cui *più si conosce la natura dei segnali e più il modello scelto aderisce a tale conoscenza, migliori sono i risultati*. In questo caso si sapeva che la forma del picco fosse Gaussiana (anche se si poteva verificare quella scelta provando altri profili candidati). Si è determinato che il numero di picchi era 2 esaminando i residui e gli errori dell'approssimazione per i modelli con 1, 2 e 3 picchi. E poi è stato introdotto il vincolo delle larghezze uguali dei picchi all'interno di ogni gruppo, sulla base della conoscenza preliminare dell'esperimento anziché sull'ispezione dei residui e degli errori di adattamento. Ecco un altro esempio, con dati sperimentali reali da una misura in cui ci si *aspetta che le larghezze dei picchi adiacenti siano uguali*, mostra il risultato di una approssimazione non vincolata ed una con larghezze uguali; gli errori dell'approssimazione sono leggermente più grandi per la equi-larghezza, ma in questo caso è preferibile. *Non ci si può*

aspettare che tutti gli esperimenti producano picchi di uguale larghezza, ma quando lo fanno, è meglio usare quel vincolo.

Profili a larghezza fissa. Andando oltre le *larghezze uguali* (nella versione 7.6 di peakfit e successive), si può specificare anche i profilo a *larghezza fissa* (forme numero 11, 12, 34-37), in cui le *larghezze dei picchi sono note in anticipo*, ma non sono necessariamente uguali, e vengono passate come *vettore* come 10° argomento di input, un elemento per ciascun picco, piuttosto che essere determinato dai dati come nella precedente misura con le equi-larghezze. Introducendo questo vincolo nell'esempio precedente e fornendo una larghezza precisa come decimo argomento di input: **peakfit([x' y'], 0, 0, 2, 11, 0, 0, 0, 0, [1.6661.666])**

```
Peak# Position Height Width Area
1 3.9943 0.49537 1.666 0.8785
2 5.9924 0.98612 1.666 1.7488
MeanFitError = 4.8128
```

Rispetto alla precedente approssimazione con equi-larghezze, l'errore del 4,8% qui è maggiore (perché ci sono meno gradi di libertà per minimizzare l'errore), ma gli errori dei parametri, in particolare le altezze dei picchi, sono *più accurate* perché le informazioni sulla larghezza fornite nell'argomento di input sono più accurate (1.666) rispetto a quelle calcolate dall'approssimazione con equi-larghezze (1.5666). Ancora una volta, non tutti gli esperimenti producono picchi di larghezza nota, ma quando lo fanno, è meglio utilizzare quel vincolo. Ad esempio, vedere l'[Esempio 35](#) (pagina 399) e lo script Matlab/Octave [WidthTest.m](#) (risultati tipici per un mix di Gaussiane/Lorentziane mostrata di seguito, dove più vincoli ci sono, maggiore è l'errore di approssimazione ma minori sono gli errori dei parametri, se i vincoli sono accurati).

Errore percentuale relativo	Errore di approssimazione	Errore della Posizione	Errore dell'altezza	Errore della larghezza
Fattore di forma e larghezze senza vincoli: profilo 33	0.78%	0.39%	0.80%	1.66%
Fattore di forma fisso e larghezze variabili: profilo 13	0.79%	0.25%	1.3%	0.98%
Fattore di forma fisso e larghezze fisse: profilo 35	0.8%	0.19%	0.695	0.0%

Regressione lineare multipla(peakfit versione 9 o successive). Infine, si noti che se anche le *posizioni* sono note restano ignote solo le *altezze*, non è nemmeno necessario utilizzare il metodo di approssimazione iterativo; si può, più facilmente e velocemente, usare la tecnica della *regressione multilineare* (chiamata anche [tecnica classica dei quadrati minimi](#), pagina 179) implementata dalla funzione [cls.m](#) e dalla versione 9 di [peakfit.m](#) come profilo numero 50. Sebbene la regressione multilineare determini un errore di approssimazione leggermente *maggior* (e un R^2 più basso), gli erro-

ri nelle altezze dei picchi misurati sono spesso *minori*, come in questo esempio tratto da [peak-fit9demo.m](#), dove le altezze reali dei [tre picchi Gaussiani sovrapposti](#) sono 10, 30 e 20.

Multilinear regression results (known position and width) :

Peak Position Height Width Area

1	400	9.9073	70	738.22
2	500	29.995	85	2714
3	560	19.932	90	1909.5

%fitting error=1.3048 R2= 0.99832 %MeanHeightError=0.427

Risultati dei quadrati minimi iterativi non lineari e non vincolati:

Peak Position Height Width Area

1	399.7	9.7737	70.234	730.7
2	503.12	32.262	88.217	3029.6
3	565.08	17.381	86.58	1601.9

%fitting error=1.3008 R2= 0.99833 %MeanHeightError=7.63

Ciò dimostra chiaramente come metodi di misurazione diversi possono *sembrare* gli stessi, e fornire errori di approssimazione quasi uguali, e tuttavia differire notevolmente nella precisione della misura dei parametri. Lo script simile [peakfit9demoL.m](#) è la stessa cosa con i picchi Lorentziani).

[SmallPeak.m](#) è uno script dimostrativo che confronta tutte queste tecniche applicate allo stimolante problema di misurare l'altezza di un piccolo picco fortemente sovrapposto e completamente oscurato da un picco molto più grande Confronta approssimazioni iterative senza vincoli, equi-larghezze e con posizioni fisse (utilizzando peakfit.m) con un quadrati minimo classico in cui sono ignote le *solo* altezze (utilizzando [cls.m](#)). Aiuta a distribuire le quattro finestre, in modo da poter osservare la drammatica differenza nella stabilità dei diversi metodi. Una tabella conclusiva degli errori relativi percentuali delle altezze mostra che *più ci sono vincoli, migliori sono i risultati* (ma solo se i vincoli sono *giustificati*). La chiave sta nel sapere su quali parametri si può fare affidamento per essere costanti e su quali si deve consentire di variare.

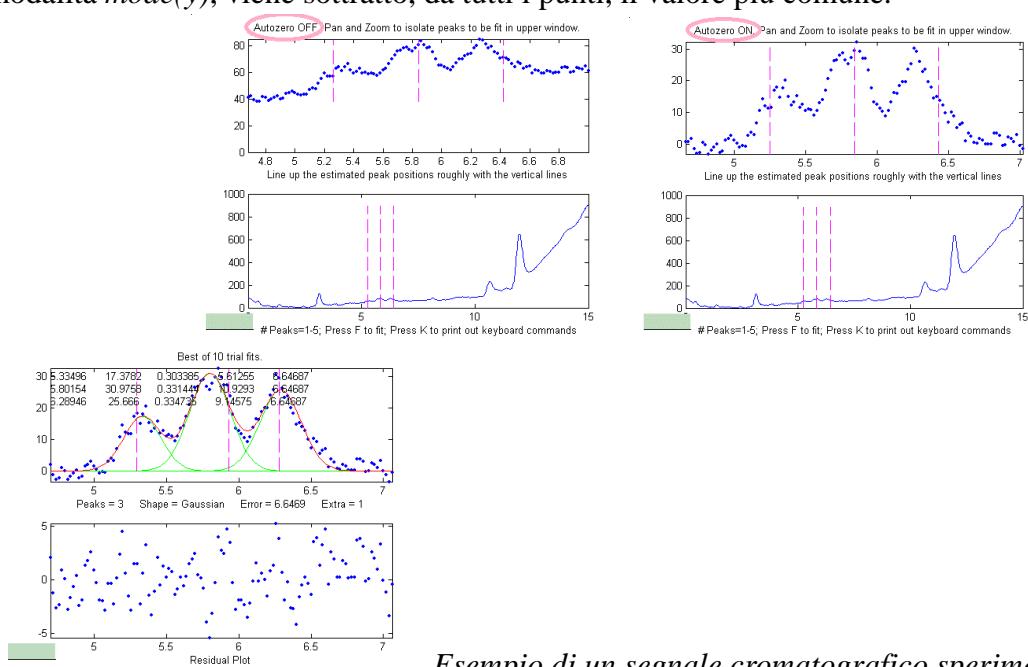
Ecco un video ([MorePeaksLowerFittingError.mp4](#)) di un esperimento con dati reali utilizzando l'approssimazione interattiva [ipf.m](#) (pagina 405) con un segnale sperimentale complesso in cui sono stati eseguiti diverse approssimazioni utilizzando profili dal 4 al 9 con larghezze variabili, uguali e fisse dei picchi Gaussiani. L'errore di approssimazione *decresce gradualmente dall'11% iniziale all'1.4%* man mano che vengono utilizzati più picchi, ma *questo è davvero giustificato?* Se l'obiettivo è semplicemente quello di ottenere una buona approssimazione, per esempio per stimare il rumore casuale nel segnale, allora si faccia tutto il necessario. Ma se l'obiettivo è quello di estrarre alcune informazioni utili dai parametri del modello, allora è necessaria una conoscenza più specifica dell'esperimento: quanti picchi sono realmente attesi; ci si aspetta davvero che le larghezze dei

picchi siano limitate? Da notare che in questo caso i residui (pannello in basso) non sono casuali hanno sempre un carattere distintamente "ondulato", suggerendo che i dati potrebbero aver subito uno smoothing prima dell'approssimazione (non è una buona idea: vedere <http://wmbriggs.com/blog/?p=195>). Quindi esiste una reale possibilità che alcuni di quei 9 picchi stiano semplicemente "approssimando il rumore", come verrà discusso più avanti a pagina 283.

b. Correzione del background

I picchi misurati in molti strumenti scientifici sono talvolta sovrapposti a un non specificato background o linea di base. Normalmente, il protocollo sperimentale è progettato per ridurre al minimo il background o per compensarlo, ad esempio sottraendo un segnale "vuoto" dal campione reale. Ma anche così, spesso c'è un background residuo che non può essere eliminato completamente sperimentalmente. L'origine e la forma di quel background dipendono dallo specifico metodo di misura, ma spesso è una forma ampia, inclinata o curva, e i picchi di interesse sono, a confronto, caratteristiche strette e sovrapposte a quello sfondo. In alcuni casi, la linea di base può essere un altro picco di interferenza che si sovrappone ai picchi di interesse. La presenza del ha un effetto relativamente scarso sulle *posizioni* dei picchi, ma è impossibile misurare accuratamente l'altezza, la larghezza e le aree del picco a meno che non si faccia qualcosa per tenerne.

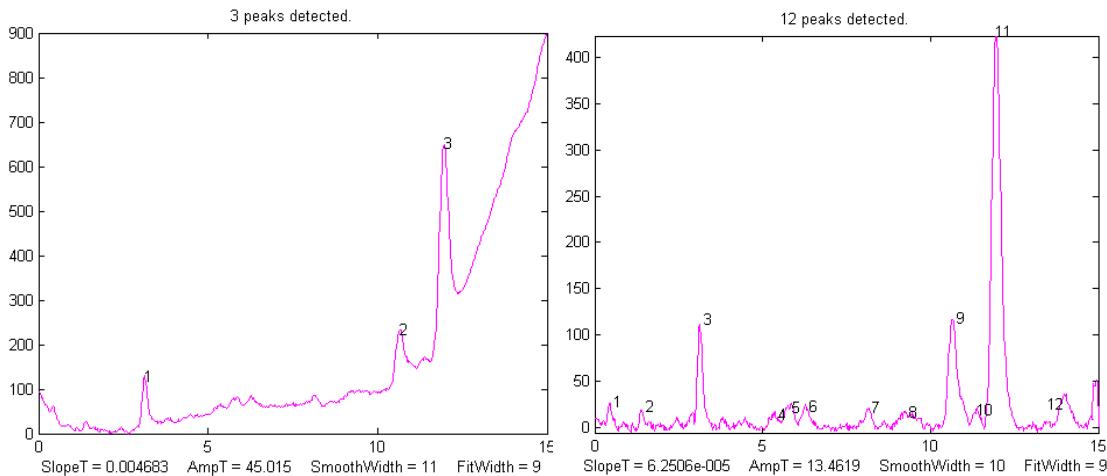
In letteratura sono descritti vari metodi per stimare e sottrarre il background in questi casi. L'ipotesi più semplice è che il background possa essere approssimato come una semplice funzione nel gruppo locale di picchi che si approssimano insieme, ad esempio come una linea costante (piatta), retta (lineare) o curva (quadratica). Questa è la base delle modalità "BaselineMode" nelle funzioni [ipf.m](#), [iSignal.m](#) e [iPeak.m](#), che vengono selezionate dal tasto **T** per scorrere attraverso le modalità *OFF*, *linear*, *quadratic*, *flat* e *mode(y)*. Nella modalità *flat*, nel calcolo dell'approssimazione viene inclusa una linea di base costante. In quella *linear*, verrà automaticamente sottratta una retta tra le due estremità del segmento di segnale nel pannello superiore *prima dell'approssimazione iterativa*. Nella modalità *quadratic*, viene sottratta una linea di base parabolica. Nelle ultime due modalità è necessario regolare il pan e lo zoom per isolare il gruppo di picchi sovrapposti da approssimare, in modo che il segnale torni al linea di base locale alle estremità sinistra e a destra della finestra. Nella modalità *mode(y)*, viene sottratto, da tutti i punti, il valore più comune.



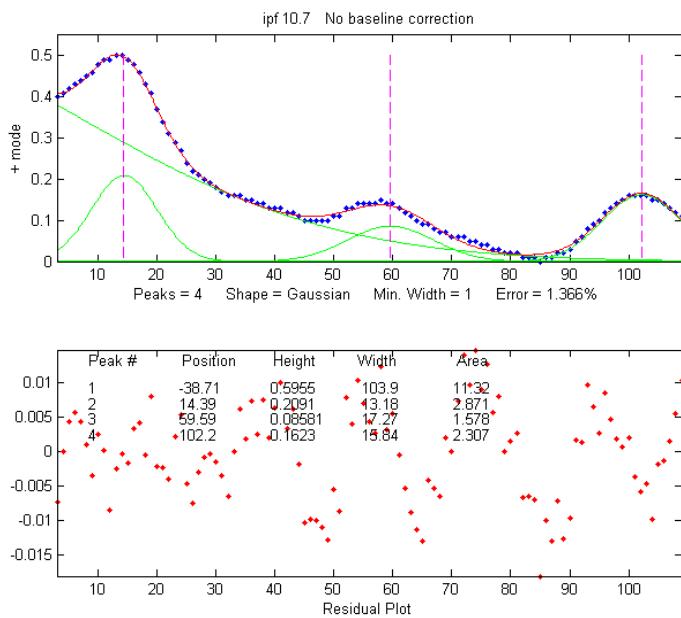
Esempio di un segnale cromatografico sperimentale in ipf.m. Da sinistra a destra, (1) Dati originali con picchi su una linea di base inclinata. I tre deboli picchi di

interesse vengono selezionati utilizzando i controlli pan e zoom, regolati in modo che il segnale ritorni sulla base locale ai lati del segmento visualizzato nella finestra superiore; (2) La linea di base lineare viene sottratta quando BaselineMode è impostato su 1 in ipf.m premendo il tasto **T**; (3) La regione selezionata viene approssimata con tre picchi Gaussiani, attivati premendo **3, G, F** (che significano 3 picchi, Gaussiani, Fit [=approssima]). Premere **R** per stampare la tabella dei picchi.

In alternativa, potrebbe essere meglio sottrarre prima lo sfondo dall'intero segnale, prima di eseguire ulteriori operazioni. Come prima, l'ipotesi più semplice è che lo sfondo sia lineare a tratti, cioè può essere approssimato come una serie di piccoli segmenti rettilinei. Questa è la base della modalità di sottrazione dello sfondo a più punti in [ipf.m](#), [iPeak.m](#) e in [iSignal](#). L'utente inserisce il numero di punti che si ritiene sia sufficiente per definire la linea di base, quindi si fa clic nel punto in cui si ritiene che la linea di base si trovi lungo l'intera lunghezza del segnale nella visualizzazione inferiore dell'intero segnale (ad esempio sugli avvallamenti tra i picchi). Dopo aver cliccato l'ultimo punto, il programma esegue l'interpolazione tra i punti cliccati e sottrae lo sfondo lineare a tratti dal segnale originale.



Da sinistra a destra, (1) I dati originali con i picchi sovrapposti ad una linea di base. (2) Il background sottratto dall'intero segnale utilizzando la funzione di rimozione multi-punto in [iPeak.m](#) ([ipf.m](#) e [iSignal.m](#) hanno la stessa funzione).



A volte, anche in assenza di una linea di base effettiva, i picchi possono sovrapporsi tanto che il segnale non ritorna mai alla linea di base, facendo sembrare che ce ne sia una da correggere. Ciò può verificarsi soprattutto con profili che hanno lati gradualmente inclinati, come il Lorentziano, [mostrato in questo esempio](#). L'approssimazione senza la correzione della linea di base in questo caso potrebbe funzionare.

In qualche caso, lo sfondo può essere modelato come un ampio picco il cui massimo cade *al di fuori* dell'intervallo dei dati acquisiti,

come nell'esempio reale a lato. Potrebbe essere possibile approssimare il picco fuori schermo includendo semplicemente *un picco extra nel modello per tenere conto della linea di base*. Nell'esempio a lato, sopra, sono visibili tre evidenti picchi, sovrapposti a una linea di base inclinata. In questo caso, il segnale si approssimava bene a quattro, anziché tre, Gaussiane di larghezze variabili, con un

errore di solo l'1,3%. L'ampia Gaussiana aggiuntiva, con un picco a $x = -38,7$, funge da linea di base. (Ovviamente, non si dovrebbero usare forme di uguale larghezza per questo, perché il picco di fondo è più ampio degli altri). [Un altro esempio di dati reali](#) mostra quattro picchi sullo schermo di altezze e larghezze molto diverse su un'ampia linea di base. Un tale segnale può risultare difficile da approssimare perché il punto di partenza per la maggior parte delle approssimazioni iterative è che tutti i picchi abbiano circa la stessa larghezza. Quindi, in alcuni casi, potrebbe essere necessario assegnare un vettore di "start" personalizzato. Utilizzando la [ipf.m](#) (pag. 405) i tasti 'C' e 'W' possono aiutare.

In un altro esempio di dati reali di uno [spettro sperimentale](#), viene usata la modalità di sottrazione della linea di base ("BaselineMode"), descritta precedentemente, insieme a un modello a di 5 Gaussiane, con una componente Gaussiana che approssima l'ampio picco che può essere parte dello sfondo e le altre quattro che approssimano i picchi. Questo si approssima molto bene ai dati (un errore dello 0,5%), ma un'approssimazione come questa può essere difficile da ottenere, perché ci sono tante altre soluzioni con errori leggermente superiori; potrebbero essere necessari diversi tentativi. Può essere utile specificare i *valori iniziali* per le variabili iterate, piuttosto che usare le scelte di default; tutti i programmi software qui descritti hanno questa capacità.

La funzione Matlab/Octave [peakfit.m](#) accetterà una forma di picco che è un *vettore di numeri di profili diversi*, che può essere utile per la correzione della linea di base. Ad esempio, si consideri un picco Gaussiano debole su una linea retta di base inclinata, utilizzando un'approssimazione a 2 componenti con una componente Gaussiana e una linea retta a pendenza variabile ('slope', profilo 26), specificata utilizzando il vettore ([1 26]) come argomento della forma:

```
x=8:.05:12;y=x+exp(-(x-10).^2);
[FitResults,GOF]= peakfit([x;y],0,0,2,[1 26],[1 1],1,0)

Peak#    Position      Height      Width      Area
1   10   1   1.6651   1.7642
2   4.485   0.22297   0.05   40.045
GOF = 0.0928   0.9999
```

Se la linea di base appare curva anziché diritta, la si può modellare con una *quadratic a*(profilo 46) anziché una pendenza lineare (peakfit *versione 8* e successive).

Se la linea di base sembra essere diversa su entrambi i lati del picco, si può provare a modellarla con una *forma a S* (sigmoide), o una sigmoide all'insù, profilo 10 ([click per il grafico](#)), `peakfit([x;y],0,0,2,[1 10],[0 0])`, o una sigmoide all'ingiù, profilo 23 ([click per il grafico](#)), `peakfit([x;y],0,0,2,[1 23],[0 0])`, in questi esempi lasciando il picco modellato come un Gaussiano.

Se il segnale è molto debole rispetto alla linea di base, l'approssimazione può essere aiutata aggiungendo delle ipotesi iniziali approssimative ("start") con la funzione "polyfit" per generarle in modo automatico per la linea di base inclinata. Per esempio, con *due* picchi sovrapposti e un'approssimazione con 3 picchi con `peakshape=[1 1 26]`.

```
x=4:.05:16;
y=x+exp(-(x-9).^2)+exp(-(x-11).^2)+.02.*randn(size(x));
start=[8 1 10 1 polyfit(x,y,1)];
peakfit([x;y],0,0,3,[1 1 26],[1 1 1],1,start)
```

Una tecnica simile si può impiegare in uno [spreadsheet](#), come mostrato in [CurveFitter2GaussianBaseline.xlsx \(grafico\)](#).

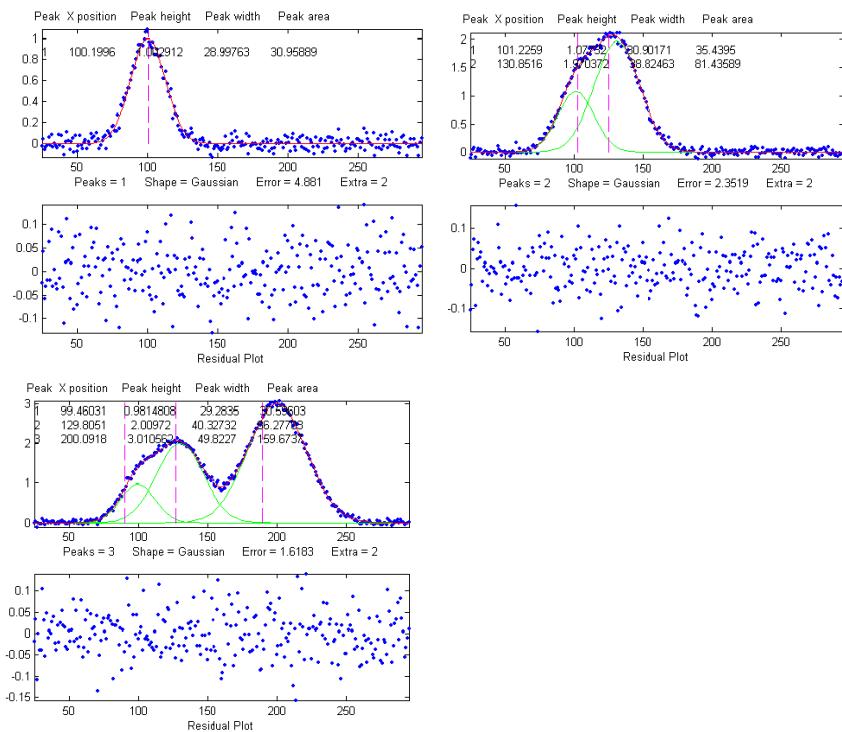
Lo svantaggio dell'inclusione della linea di base come componente variabile è che aumenta il numero di gradi di libertà, aumenta il tempo di esecuzione e aumenta la possibilità di approssimazione instabili. Indicare dei valori iniziali può aiutare.

c. Rumore casuale nel segnale.

Qualsiasi segnale sperimentale ha una certa quantità di rumore casuale, il che significa che i singoli punti si disperdono casualmente sopra o sotto i loro valori reali. Normalmente si assume che lo scostamento sia ugualmente al di sopra o al di sotto del segnale vero in modo che la media a lungo termine si avvicini al valore medio reale; il rumore a "media a zero" come si dice spesso. Il problema pratico è che ogni registrazione del segnale contiene solo un campione finito del rumore. Se viene eseguita un'altra registrazione del segnale, conterrà un altro campione indipendente del rumore.

Questi campioni di rumore non sono infinitamente lunghi e quindi non rappresentano la vera natura a lungo termine del rumore. Questo presenta due problemi: (1) un singolo campione del rumore non sarà a "media a zero" e quindi i parametri del modello più adatto non saranno necessariamente uguali ai valori reali, e (2) l'entità del rumore durante un campione potrebbe non essere tipico; il rumore potrebbe essere stato casualmente maggiore o minore della media durante quel periodo. Ciò significa che i metodi matematici di "propagazione dell'errore", che cercano di stimare il probabile errore nei parametri del modello in base al rumore nel segnale, saranno soggetti a errore (*sottostimando* se il rumore è *più basso* della media e *sovrestimando* se il rumore è *maggiore* della media).

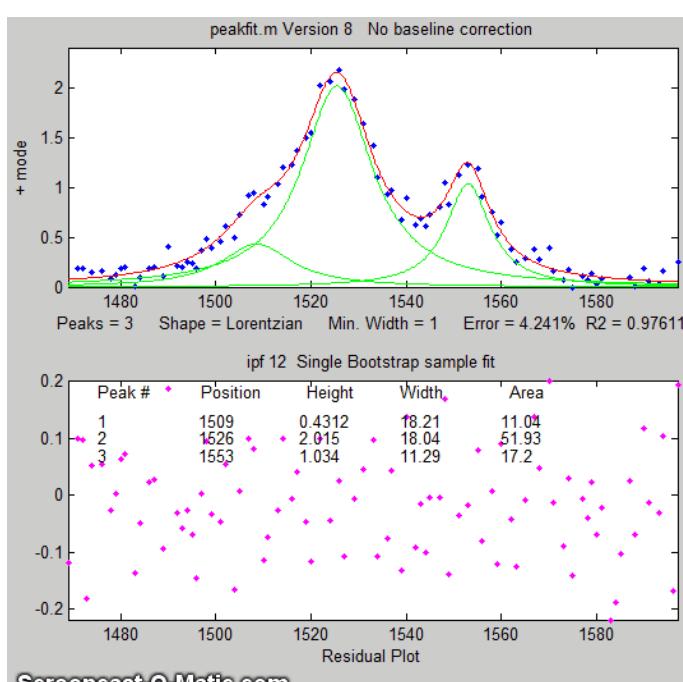
Un modo migliore per stimare gli errori dei parametri è registrare più campioni del segnale, approssimarli separatamente, calcolare i parametri da ogni approssimazione e calcolare l'errore standard di ogni parametro. Tuttavia, se ciò non risulta pratico, è possibile simulare tali misure aggiungendo del rumore casuale a un modello con dei parametri noti, quindi approssimando quel segnale rumoroso simulato per determinare i parametri, e ripetendo la procedura più volte con diversi set di rumore casuale. Questo è esattamente ciò che fa lo script [DemoPeakfit.m](#) (che richiede la funzione [peakfit.m](#)) per simulare segnali rumorosi come quelli illustrati di seguito. È facile dimostrare che, come previsto, la precisione dell'errore medio dell'approssimazione e la deviazione standard relativa dei parametri aumenta direttamente con il livello di rumore casuale nel segnale. Ma la precisione e l'accuracy dei parametri misurati dipende *anche* dal parametro (le posizioni dei picchi sono sempre misurate più accuratamente delle loro altezze, larghezze e aree) e dall'altezza del picco e dall'entità della sovrapposizione (i due picchi più a sinistra in questo esempio non solo sono più deboli ma anche più sovrapposti rispetto al picco più a destra, e quindi mostrano misure più scadenti dei parametri). In questo esempio, l'errore di approssimazione è dell'1,6% e la deviazione standard relativa percentuale dei parametri varia dallo 0,05% per la posizione del picco più grande al 12% per l'area del picco più piccolo.



La sovrapposizione conta: Gli errori nei valori dei parametri misurati dall'approssimazione della curva dipendono non solo dalle caratteristiche dei picchi in esame e dal rapporto segnale/rumore, ma anche da altri picchi che si sovrappongono. Da sinistra a destra:

(a) un singolo picco a $x=100$ con un'altezza di 1.0 e larghezza 30 viene approssimato con un modello Gaussiano, che produce un errore di approssimazione relativo del 4.9% e una deviazione standard relativa della posizione, dell'altezza e della larghezza di 0.2%, 0.95% e 1.5%, rispettivamente.

(b) Lo stesso picco, con lo stesso livello di rumore ma con un altro picco sovrapposto, in effetti riduce l'errore relativo dell'approssimazione a 2.4% (semplicemente perché l'aggiunta del secondo picco aumenta l'ampiezza complessiva del segnale). Tuttavia, **aumenta** la deviazione standard relativa della posizione, dell'altezza e della larghezza a 0.84%, 5% e 4% - un'approssimazione apparentemente migliore, ma con minore precisione per il primo picco.

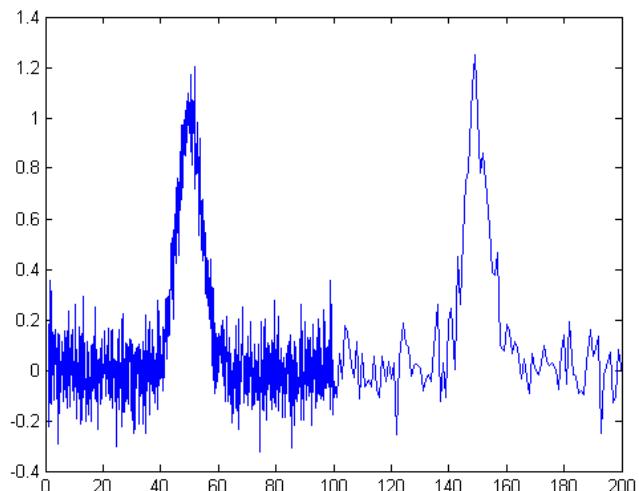


(3) L'aggiunta di un terzo picco (più grande ma non sovrapposto) riduce ulteriormente l'errore di approssimazione all'1.6%, ma la deviazione standard relativa della posizione, dell'altezza e della larghezza del primo picco sono all'incirca le stesse dei due picchi, perché **il terzo picco non è sovrapposto al primo in modo significativo**.

Se il rumore medio nel segnale non è noto o la sua distribuzione di probabilità è incerta, è possibile utilizzare il [metodo di campionamento bootstrap](#) (pagina 162), illustrato

dall'animazione a lato, per valutare le incertezza delle altezze, posizioni e larghezze dei picchi, come illustrato a lato e come descritto in dettaglio a pagina 162. La funzione [interattiva operante da tastiera, ipf.m](#), (pag. 405) ha questa funzione, che è attivata dal tasto 'N'; cliccare sulla figura per aprire un'animazione GIF mostrata a lato. Oppure premere 'V' per calcolare le deviazioni standard previste di tutti i parametri di picco utilizzando questo metodo.

Un modo per ridurre l'effetto del rumore è acquisire più dati. Se l'esperimento consente di ridurre l'intervallo tra i punti sull'asse x o di effettuare più letture per ciascun valore dell'asse x, l'aumento risultante del numero di punti in ciascun picco dovrebbe aiutare a ridurre l'effetto del rumore. A titolo dimostrativo, utilizzando lo script [DemoPeakfit.m](#) per creare il segnale di picchi sovrapposti come quello mostrato sopra a sinistra, è possibile modificare l'intervallo tra i valori x e quindi il numero totale di punti nel segnale. Con un livello di rumore dell'1% e 75 punti nel segnale, l'errore di approssimazione è 0,35 e l'errore medio del parametro è 0,8%. Con 300 punti nel segnale e lo stesso livello di rumore, l'errore di approssimazione è essenzialmente lo stesso, ma l'errore medio del parametro scende allo 0,4%, suggerendo che l'accuratezza dei parametri misurati varia inversamente con la radice quadrata del numero di punti dati sui picchi.



La figura a lato illustra l'importanza dell'intervallo di campionamento e della densità dei dati. (Si può scaricare il file "udx" dei dati in formato [TXT](#) o in Matlab [MAT](#)). Il segnale consiste in due picchi Gaussiani, uno posizionato a $x=50$ e l'altro a $x=150$. Entrambi i picchi hanno un'altezza di 1.0 e una semi-larghezza di 10. All'intero segnale è stato aggiunto un rumore bianco casuale normalmente distribuito con una deviazione standard di 0.1. L'*intervallo di campionamento sull'asse x*, tuttavia, è diverso per i due picchi; è 0.1 per il primo e 1.0 per il secondo. Ciò significa che il primo picco è caratterizzato da dieci volte più punti rispetto al secondo picco. Quando si approssimano questi picchi separatamente a un modello Gaussiano (ad esempio, utilizzando peakfit.m o ipf.m), si noterà che tutti i parametri del primo picco vengono misurati in modo più accurato del secondo, anche se l'errore di approssimazione non è molto diverso:

Primo picco: **Secondo picco:**

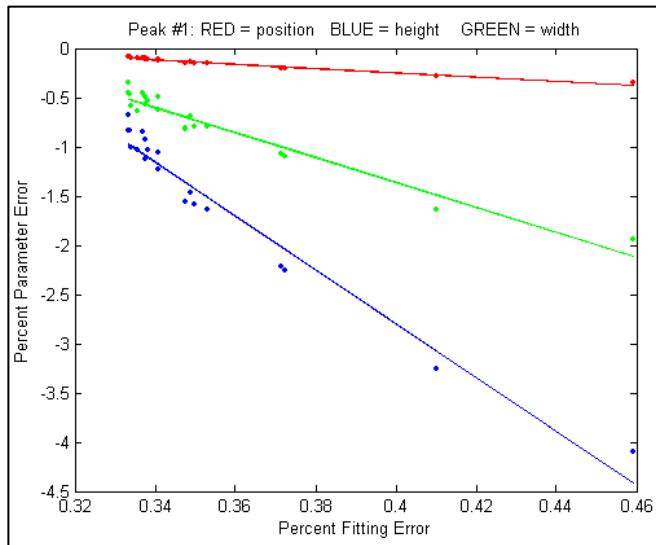
Percent Fitting Error=7.6434% **Percent Fitting Error=8.8827%**

Peak#	Position	Height	Width	Peak#	Position	Height	Width
1	49.95	1.0049	10.111	1	149.64	1.0313	9.941

Colore del rumore. Finora, questa discussione ha riguardato il rumore *bianco*. Ma altri colori di rumore (pag. 28) hanno effetti diversi. Il rumore ponderato a bassa frequenza ("rosa") ha un effetto *maggior*e sull'accuratezza delle misure dei parametri del picco nell'approssimazione, e, in una bella simmetria, il rumore "blu" alle alte frequenze ha un effetto *minore* rispetto a quanto ci si aspetterebbe sulla base della sua deviazione standard. Questo perché l'informazione in un segnale di un picco uniforme è concentrata alle basse frequenze. Un esempio di ciò si verifica quando si applica

l'approssimazione della curva a un segnale che è stato [deconvoluto](#) (pag. 106) per rimuovere un effetto di ampliamento. Questo è il motivo per cui lo [smoothing prima dell'approssimazione non è d'aiuto](#) (pagina 226) perché le informazioni sul segnale sono concentrate alle *basse* frequenze, ma lo smoothing riduce principalmente il rumore alle *alte* frequenze.

A volte si potrebbe notare che i residui in un'operazione di approssimazione sono strutturati in bande o linee piuttosto che essere completamente casuali. Ciò può verificarsi se la [variabile indipendente](#) o la [variabile dipendente](#) viene *quantizzata* a passi discreti anziché continui. Può sembrare strano, ma ha scarso effetto sui risultati se il rumore casuale è maggiore dei passi. Quando c'è rumore nei dati (in altre parole, praticamente sempre), i risultati esatti dipenderanno dalla regione selezionata per l'approssimazione - ad esempio, i risultati varieranno leggermente con l'impostazione del pan e dello zoom in ipf.m, e maggiore è il rumore, maggiore è l'effetto.



d. Errori nell'approssimazione iterativa

A differenza della regressione lineare multipla, il "curve fitting", i metodi iterativi potrebbero non convergere sempre sugli stessi parametri del modello ripetendo l'approssimazione con valori iniziali leggermente diversi (prime ipotesi). L'Interactive Peak Fitter (Approssimatore interattivo di picchi) ipf.m (pagina 405) facilita questo test, perché utilizza valori iniziali leggermente diversi ogni volta che il segnale viene approssimato (premendo il tasto **F** in ipf.m, per esempio).

Ancora meglio, premendo il tasto **X**, la funzione ipf.m calcola in modo silente 10 approssimazioni con diversi valori iniziali e prende quello con l'errore di approssimazione più basso. Un presupposto di base di qualsiasi operazione di approssimazione della curva è che l'errore di approssimazione (la differenza quadratica media tra il modello e i dati) sia ridotto al minimo; anche gli errori dei parametri (la differenza tra i parametri effettivi e i parametri del modello migliore) saranno ridotti al minimo. Questa è generalmente una buona ipotesi, come dimostrato dal grafico sopra, che mostra i tipici errori dei parametri percentuali in funzione dell'errore di approssimazione per il picco più a sinistra in un segnale campione simulato generato da DemoPeakfit.m (mostrato nella sezione precedente). La variabilità dell'errore di approssimazione qui è causata da piccole variazioni casuali nelle prime ipotesi, piuttosto che da rumore casuale nel segnale. In molti casi pratici, c'è abbastanza rumore casuale nei segnali che gli errori di approssimazione iterativo all'interno di un campione del segnale sono piccoli rispetto agli errori del rumore casuale tra i campioni. Da ricordare che la variabilità nei parametri misurati dall'approssimazione di un singolo campione del segnale non è una buona stima della precisione o dell'accuratezza di quei parametri, per il semplice motivo che quei risultati rappresentano solo un campione del segnale, del rumore e dello sfondo. Le variazioni da campione a campione tendono ad essere molto maggiori delle variazioni nel singolo campione a causa del "curve fitting" iterativo. (In questo caso, un "campione" è una singola registrazione del segnale). Per stimare il contributo del rumore casuale alla variabilità dei parametri misurati quando è disponibile un solo campione se il segnale è disponibile, è possibile utilizzare il metodo bootstrap (pagina 162).

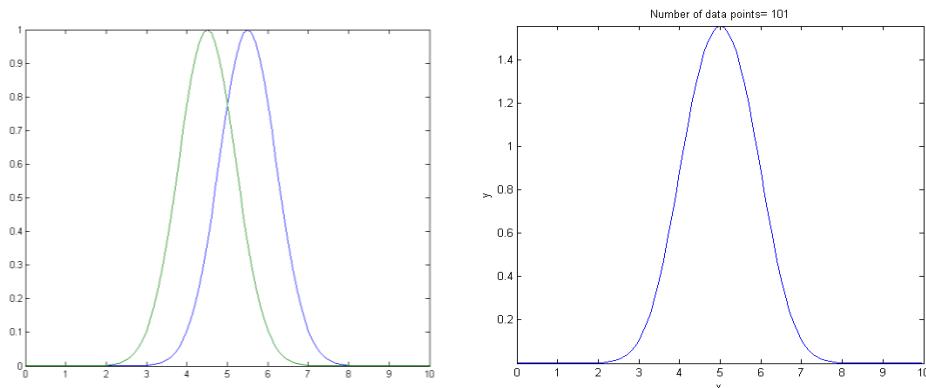
Selezione della regione di interesse ottimale. Quando si esegue un'approssimazione del picco utilizzando ipf.m (pagina 405), si ha il controllo sulla regione dei dati selezionata utilizzando il pan

e lo zoom (o, utilizzando la funzione a riga di comando `peakfit.m`, impostando gli argomenti di input "center" e "window"). La modifica di queste impostazioni di solito cambia i parametri risultanti dall'approssimazione. Se i dati fossero perfetti, diciamo, una forma di picco matematicamente perfetta senza rumore casuale, le impostazioni di pan e zoom non farebbero alcuna differenza; si otterrebbero gli stessi valori esatti per i parametri con tutte le impostazioni, assumendo solo che il modello che si sta utilizzando corrisponda alla forma effettiva. Ma ovviamente, nel mondo reale, i dati non sono mai matematicamente perfetti e senza rumore. Maggiore è la quantità di rumore casuale nei dati, o maggiore è la discrepanza tra i dati e il modello selezionato, più i parametri misurati varieranno se si approssimano regioni diverse utilizzando i controlli di pan e zoom. Questa è semplicemente un'indicazione dell'inevitabile incertezza nei parametri misurati.

Un caso difficile.

Come esempio notevole delle idee nelle sezioni precedenti, si consideri questo segnale simulato sopra. È costituito da due picchi Gaussiani di uguale altezza = 1.00, mostrati separatamente a sinistra, che si sovrappongono abbastanza, in modo che la loro somma, mostrata a destra, sia un singolo picco simmetrico che assomiglia molto ad una singola Gaussiana.

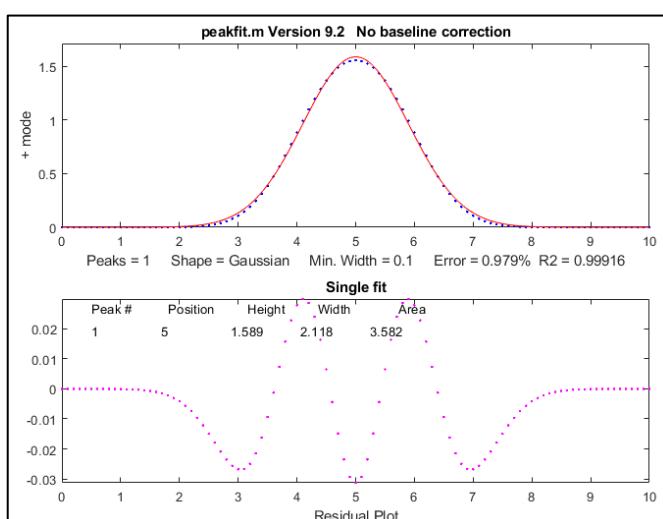
```
>> x=[0:.1:10]';
>> y=exp(-(x-5.5).^2)+exp(-(x-4.5).^2);
```



I tentativi di approssimarla con una *singola* Gaussiana (mostrati nel grafico sottostante) producono un errore di approssimazione abbastanza basso dell'1%. Ma il residuo è notevolmente ondulato e regolare, suggerendo che c'è poco o nessun rumore casuale nei dati ma il *modello non è quello giusto*.

```
>> peakfit([x y],5,19,1,1)
Peak# Position Height Width Area
1 4.5004 1.001 1.6648 1.773
2 5.5006 0.99934 1.6641 1.770
```

Se non ci fosse rumore nel segnale, le routine di approssimazione della curva iterativa (`peakfit.m` o



ipf.m) potrebbero facilmente estrarre le due componenti Gaussiane uguali con una precisione di 1 parte su 1000. Ma in presenza di rumore anche minimo (ad esempio, 1% RSD), i risultati non sono uniformi; un picco è quasi sempre significativamente più alto dell'altro:

```
>> y=exp(-(x-5.5).^2)+exp(-(x-4.5).^2)+.01*randn(size(x))
>> peakfit([x y],5,19,2,1)

Peak# Position Height Width Area
1 4.4117  0.83282  1.61   1.43
2 5.4022  1.1486   1.734   2.12
```

L'approssimazione è stabile con qualsiasi campione di rumore, se [peakfit.m](#) viene eseguito di nuovo con valori iniziali leggermente diversi, ad esempio premendo più volte il tasto F in [ipf.m](#) (pagina 405). Quindi il problema *non* sono gli errori dell'approssimazione iterativa causati da valori iniziali diversi. Il problema è il *rumore*: sebbene il segnale sia completamente simmetrico, ogni campione del rumore è leggermente asimmetrico (ad esempio, la prima metà del rumore invariabilmente è in media leggermente superiore o leggermente inferiore alla seconda metà, risultandone un'approssimazione asimmetrica). La cosa sorprendente è che l'errore nelle altezze dei picchi è molto maggiore (circa il 15% relativo, in media) rispetto al rumore casuale nei dati (1% in questo esempio). Quindi anche se *l'approssimazione sembra buona* - l'errore è basso (meno dell'1%) e i residui sono casuali e non strutturati - *i parametri del modello possono essere ancora molto lontani*. Se si dovesse simulare un'altra misura (ovvero generare un altro insieme indipendente di rumore), i risultati sarebbero diversi ma comunque imprecisi (il primo picco ha la stessa probabilità di essere maggiore o minore del secondo). Sfortunatamente, l'errore atteso non è previsto accuratamente dal *metodo bootstrap* (pagina 162), che sottostima seriamente la deviazione standard dei parametri con ripetute misure dei segnali indipendenti (perché è probabile che un sotto-campione bootstrap del rumore asimmetrico rimanga asimmetrico). In questi casi, una [simulazione Monte Carlo](#) (pag. 161) fornirebbe una stima più affidabile dell'incertezza.

È possibile ottenere risultati migliori nei casi in cui si prevede che le larghezze dei picchi siano uguali, nel qual caso è possibile utilizzare il profilo 6 (Gaussiane di uguale larghezza) invece del profilo 1:

```
peakfit([x y],5,19,2,6)
```

Aiuta anche a fornire delle ipotesi iniziali decenti (start) e impostare il numero di prove (Num-Trials) superiore a 1):

```
peakfit([x,y],5,10,2,6,0,10,[4 2 5 2],0,0)
```

Il caso migliore sarà se il profilo, la posizione e la larghezza dei due picchi sono accuratamente note e se le *sole* incognite sono le altezze. Quindi si può applicare la tecnica [Classica dei Quadrati Minimi \(regressione multipla\)](#) e i risultati saranno molto migliori.

Per un esempio simile ancora più impegnativo, in cui i due picchi strettamente sovrapposti hanno un'altezza molto diversa, vedere a pagina 315.

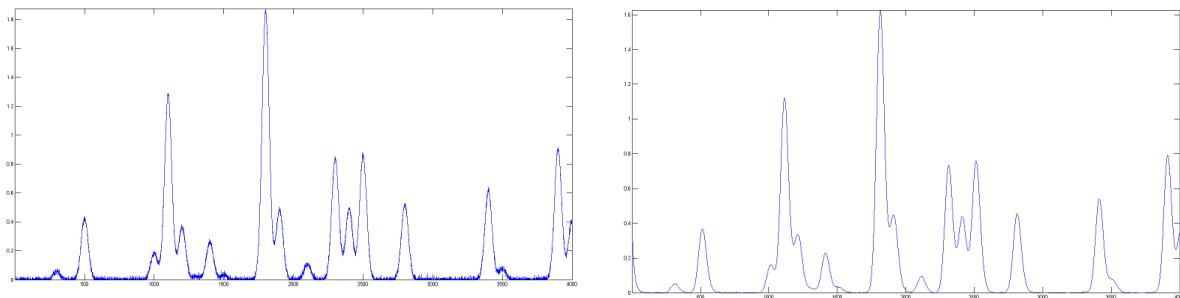
Quindi, per riassumere, si possono fare le seguenti osservazioni circa l'accuratezza dei parametri del modello:

- (1) gli errori dei parametri dipendono dall'accuratezza del modello scelto e dal numero di picchi;
- (2) gli errori dei parametri sono direttamente proporzionali al rumore nei dati (e il peggio si ha per le basse frequenze e per il rumore rosa);
- (3) a parità di condizioni, gli errori dei parametri sono proporzionali all'errore di approssimazione, ma un modello che si adatta meglio alla realtà in esame, ad es., larghezze o forme uguali o fisse, spesso fornisce errori inferiori del parametro anche se l'errore di approssimazione è maggiore;
- (4) gli errori sono tipicamente minimi per la posizione del picco e peggiori per la larghezza e l'area del picco;
- (5) gli errori dipendono dalla *densità dei dati* (numero di punti dati indipendenti nell'ampiezza di ciascun picco) e dall'*estensione della sovrapposizione del picco* (i parametri dei picchi isolati sono più facili da misurare rispetto ai picchi molto sovrapposti);
- (6) se è disponibile un solo segnale, l'effetto del rumore sulla deviazione standard dei parametri in molti casi può essere previsto approssimativamente con il [metodo bootstrap](#), ma se la sovrapposizione dei picchi è troppo grande, l'errore delle misure dei parametri può essere molto maggiore del previsto.

A volte l'approssimazione della curva è complicato se i picchi sono *asimmetrici* (più larghi da un lato rispetto all'altro). [AsymmetricalOverlappingPeaks.m](#) illustra un modo per affrontare il problema dell'eccessiva sovrapposizione dei picchi in uno script a passaggi multipli che utilizza la simmetrizzazione della derivata prima come pre-elaborazione eseguita prima dell'approssimazione iterativa dei minimi quadrati per analizzare un segnale complesso costituito da più picchi asimmetrici sovrapposti. Vedere pagina 358 per i dettagli. Oppure si può utilizzare un modello asimmetrico, come descritto nella prossima sezione.

Approssimazione dei segnali soggetti ad ampliamento [broadening] esponenziale.

[DataMatrix2](#) (in basso a sinistra) è un segnale di test generato al computer costituito da 16 picchi Gaussiani simmetrici con rumore bianco casuale aggiunto. I picchi si verificano in gruppi di 1, 2 o 3 picchi sovrapposti, ma i massimi dei picchi si trovano esattamente a valori interi di x da 300 a 3900 (sui 100) e le larghezze dei picchi sono sempre esattamente 60 unità. Le altezze dei picchi variano da 0.06 a 1.85. La deviazione standard del rumore è 0.01. È possibile utilizzare questo segnale per testare i programmi di approssimazione della curva e per determinare l'accuratezza delle misure dei



parametri dei picchi. Cliccare con il tasto destro e selezionare "Save" per scaricare questo segnale, inserirlo nel path di ricerca di Matlab, poi digitare "load [DataMatrix2](#)" al prompt dei comandi per caricarlo nello spazio di lavoro di Matlab. [DataMatrix3](#) (in alto a destra) è una versione esponenzialmente allargata di DataMatrix2, con una "costante di decadimento", detta anche "costante di tempo", di 33 punti sull'asse x. Il risultato dell'ampliamento esponenziale è che tutti i picchi in questo segnale sono asimmetrici, i loro massimi vengono spostati su valori di x più lunghi, le altezze sono inferiori e le larghezze sono maggiori dei picchi corrispondenti in DataMatrix2. Inoltre, il rumore casuale è smorzato in questo segnale rispetto all'originale e non è più a "bianco", come conseguenza dell'ampliamento. Questo tipo di effetto è comune nelle misure fisiche e proviene da qualche effetto fisico o elettrico del sistema di misura e che è a parte delle caratteristiche fondamentali del picco. In tali casi, si potrebbe voler compensare l'effetto dell'ampliamento, o mediante deconvoluzione o mediante approssimazione della curva, nel tentativo di misurare *quali sarebbero stati i parametri prima dell'ampliamento* (e anche per misurare l'ampliamento stesso). Questa operazione si può eseguire per i picchi Gaussiani ampliati in modo esponenziale utilizzando il profilo "Exp-Gaussian" in peakfit.m e ipf.m (pagina 405), o "ExpLorentzian", se i picchi sono Lorentziani. Click destro e selezionare "Save" per scaricare questo segnale, inserirlo nel percorso di ricerca di Matlab, poi digitare "load [DataMatrix3](#)" per caricarlo nell'area di lavoro di Matlab. L'esempio illustrato a lato si concentra sul singolo picco isolato la cui posizione, altezza, larghezza e area "reale" nel segnale originale non espanso sono rispettivamente 2800, 0.52, 60 e 33.2. (La deviazione standard relativa del rumore è $0.01/0.52=2\%$). Nel segnale espanso, il picco è visibilmente asimmetrico, il massimo è spostato su valori di x maggiori e ha un'altezza più corta e una larghezza maggiore, come dimostrato dal tentativo di approssimare una Gaussiana normale (simmetrica) al picco espanso. (L'*area* del picco, tuttavia, non è molto influenzato dall'ampliamento).

```
>> load DataMatrix3

>> ipf(DataMatrix3);

Peak Shape = Gaussian

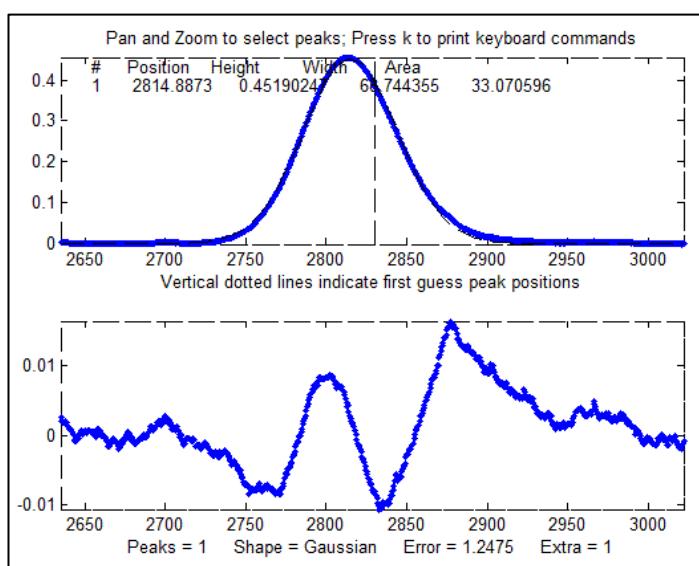
BaselineMode ON

Number of peaks = 1

Fitted range = 2640 - 2979.5 (339.5) (2809.75)
```

Percent Error = 1.2084

Peak# Position Height Width Area

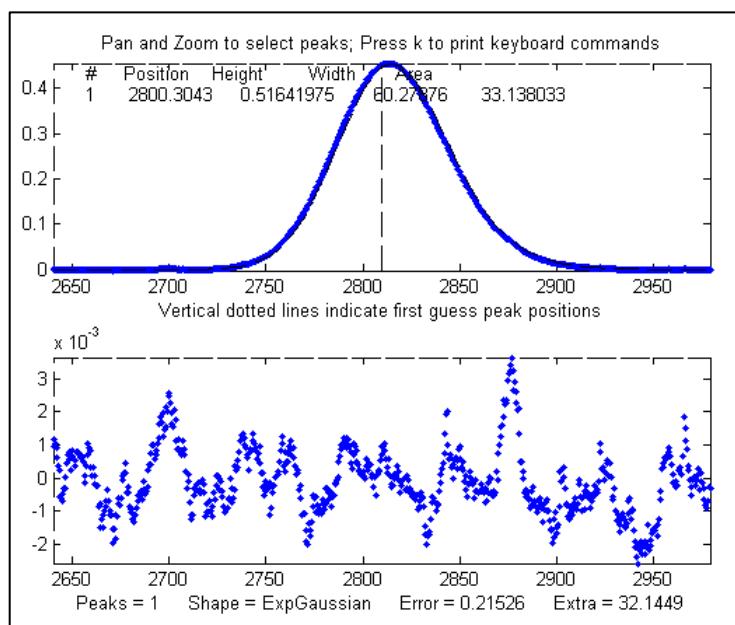


duo quasi casuale. Ma la cosa interessante è che *recupera anche la posizione, l'altezza e la larghezza originali (pre-dilatazione) del picco con una precisione di una frazione dell'1%*, se può interessare. Nell'eseguire questa approssimazione, la costante di decadimento ("extra") è stata determinata sperimentalmente dal segnale ampliato regolandolo, con i tasti A e Z, per ottenere l'errore di approssimazione più basso; ciò si traduce anche in una misura ragionevolmente buona del fattore di ampliamento (32.6, contro il valore effettivo di 33). Se il segnale originale fosse stato più rumoroso, queste misure non sarebbero state così accurate. Nota: quando si utilizza peakshape 5 (Gaussiana ampliata esponenzialmente con costante di decadimento fissa), gli si deve assegnare un valore ragionevolmente buono per la costante di decadimento ('extra'), l'argomento di input subito dopo il numero del profilo. Se il valore è troppo lontano, l'approssimazione potrebbe non riuscire completamente, restituendo tutti zeri. Qualche tentativo è sufficiente. In alternativa, si può usare la semplice *tecnica dell'aggiunta della derivata prima* (pag. 77) per ottenere una buona stima della costante di tempo prima dell'approssimazione della curva. (Inoltre, [peakfit.m versione 8.4](#) ha due forme di Gaussiana allargata esponenzialmente con costante di decadimento variabile non vincolata, profili 31 e 39, che misureranno la costante di decadimento come variabile iterata. Profilo 31 ([expgaussian.m](#)) crea il profilo eseguendo una convoluzione di Fourier di una Gaussiana specificata da un decadimento esponenziale con una costante di decadimento specifica, mentre il profilo 39 ([expgaussian2.m](#)) usa un'espressione matematica per la forma finale così prodotta. Entrambi danno come risultato lo *stesso profilo del picco* ma sono parametrizzati in modo diverso. Il profilo 31 riporta l'altezza e la posizione del picco come quella della Gaussiana *originale* prima dell'ampliamento, mentre il profilo 39 riporta l'altezza del picco del *risultato* espanso. Il profilo 31 riporta la larghezza come FWHM (larghezza intera a metà del massimo) e il 39 riporta la deviazione standard (sigma) della Gaussiana. Il profilo 31 riporta il fattore esponenzia-

1 2814.832 0.451005 68.4412
32.8594

Il grande residuo "ondulato" nel grafico sopra è un indizio che il modello non è del tutto corretto. Inoltre, l'errore di approssimazione (1.2%) è maggiore del previsto per un picco con una semi-larghezza di 60 punti e un RSD del rumore del 2% (circa $2/\sqrt{60} = 0.25\%$).

L'approssimazione a una Gaussiana esponenzialmente espansa (foto a lato) fornisce un errore di approssimazione molto più basso ("errore percentuale") e un grafico resi-



le e il *numero dei punti*, il profilo 39 riporta il *reciproco della costante di decadimento* in unità di tempo. (Vedere lo script [DemoExpgaussian.m](#) per un esempio numerico più dettagliato). Per le approssimazioni di picchi multipli, entrambi i profili solitamente richiedono un vettore ragionevole per la prima ipotesi (“start”) per ottenere i migliori risultati, che determinabili automaticamente da un'approssimazione preliminare con un semplice modello Gaussiano ([come in questo esempio](#)). Se ci si aspetta che la costante di decadimento esponenziale di ciascun picco sia diversa e se ne devono misurare i valori, utilizzare i profili 31 o 39, ma se si prevede che la costante di decadimento per tutti i picchi sia la stessa, usare il profilo 5, e determinare la costante di decadimento approssimand un picco isolato. Per esempio:

```
Peak Shape = Exponentially-broadened Gaussian

BaselineMode ON

Number of peaks = 1

Extra = 32.6327

Fitted range = 2640 - 2979.5 (339.5) (2809.75)

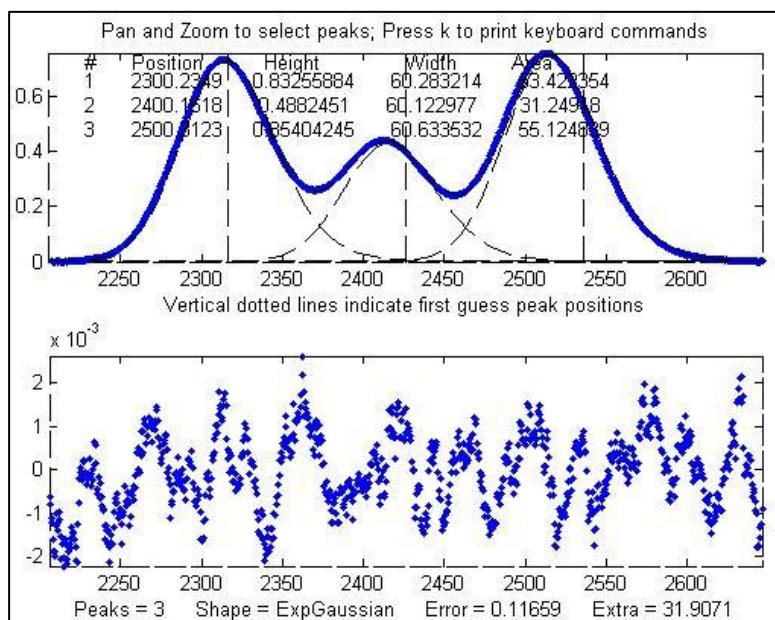
Percent Error = 0.21696

Peak# Position Height Width Area

1 2800.130 0.518299 60.08629 33.152429
```

Confrontando i due metodi, l'approssimazione di una Gaussiana ampliata esponenzialmente recupera tutti i parametri di picco sottostanti in modo abbastanza accurato:

	Posizione	Altezza	Larghezza	Area
Parametri effettivi del picco	2800	0.52	60	33.2155
Approssimazione di una Gaussiana a un segnale espanso	2814.832	0.45100549	68.441262	32.859436
Approssimazione di una ExpGaussian a un segnale espanso	2800.1302	0.51829906	60.086295	33.152429



Altri picchi nello stesso segnale, sotto l'influenza di ampliamento della stessa costante di decadimento, possono essere approssimati con impostazioni simili, ad esempio, l'insieme dei tre picchi sovrapposti vicino a $x = 2400$. Come prima, le posizioni dei picchi vengono recuperate quasi esattamente e anche le misure delle larghezze sono ragionevolmente accurate (1% o migliore). Se la costante di decadimento dell'ampliamento esponenziale *non è* la stessa per tutti i picchi nel segnale, per esempio, se aumenta gradualmen-

te all'aumentare di x , l'impostazione della costante di decadimento può essere ottimizzata per ciascun gruppo di picchi. L'errore di approssimazione più piccolo è evidente qui ed è solo un riflesso delle altezze più grandi in questo gruppo di picchi - in questo segnale il rumore è lo stesso ovunque.

Peak Shape = Exponentially-broadened Gaussian

BaselineMode OFF

Number of peaks = 3

Extra = 31.9071

Fitted range = 2206 - 2646.5 (440.5) (2426.25)

Percent Error = 0.11659

Peak# Position Height Width Area

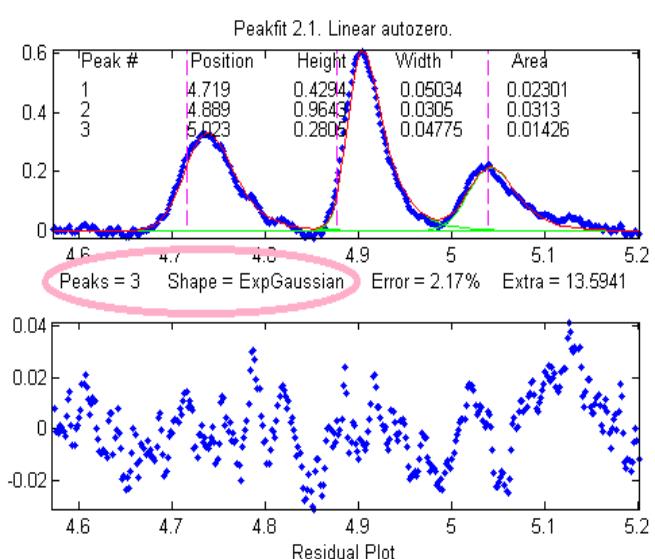
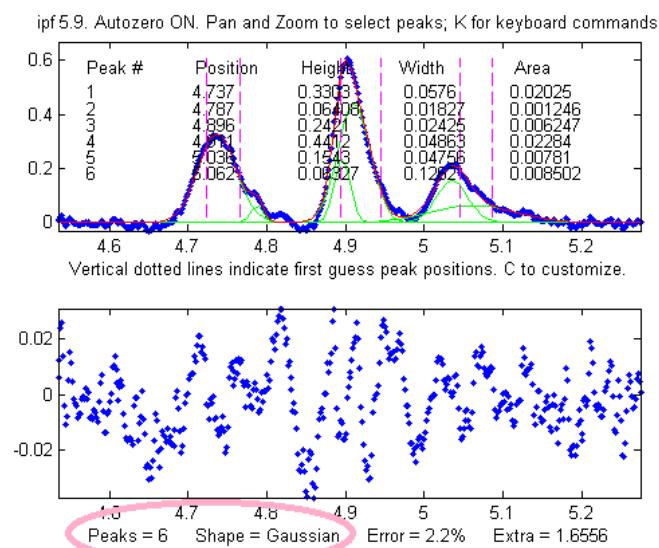
1 2300.2349 0.83255884 60.283214 53.422354

2 2400.1618 0.4882451 60.122977 31.24918

3 2500.3123 0.85404245 60.633532 55.124839

Il disegno del residuo in entrambi gli esempi continua ad avere un carattere "ondulato", anziché completamente casuale e "bianco". L'ampliamento esponenziale attenua qualsiasi rumore bianco nel segnale originale che viene introdotto *prima* dell'effetto esponenziale, agendo da filtro passa-basso nel dominio del tempo e risultando in un rumore "rosa" maggiore alla basse frequenze, che è quello che resta nei residui dopo che i picchi ampliati sono stati approssimati al meglio. D'altra parte, il rumore bianco che viene introdotto *dopo* l'effetto esponenziale continuerebbe ad apparire bianco e casuale nei residui. Nei dati sperimentali reali, entrambi i tipi di rumore possono essere presenti in quantità variabili.

Un ultimo avvertimento: l'asimmetria dei picchi come l'ampliamento esponenziale potrebbe essere il risultato di una coppia di picchi ravvicinati di altezze diverse. In effetti, un singolo picco Gaussiano espanso esponenzialmente può talvolta essere approssimato con due Gaussiane simmetriche con un errore di approssimazione almeno pari ad un'approssimazione di una singola Gaussiana espansa

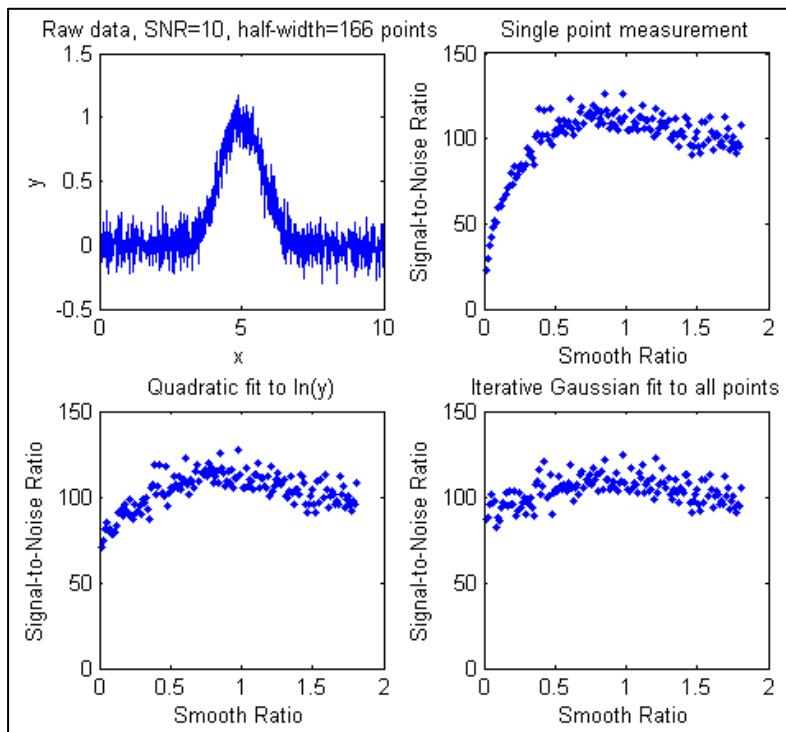


esponenzialmente. Ciò rende difficile distinguere tra questi due modelli in base al solo errore di approssimazione. Tuttavia, si può decidere basandosi sull'ispezione degli altri picchi nel segnale: nella maggior parte degli esperimenti, l'ampliamento esponenziale si applica a ogni picco nel segnale e questo è costante o cambia gradualmente lungo il segnale. Se solo uno o alcuni dei picchi mostrano asimmetria e gli altri sono simmetrici, è molto probabile che l'asimmetria sia dovuta a picchi ravvicinati di altezze diverse. Se *tutti* i picchi hanno la stessa asimmetria o una simile, è più probabile che sia un fattore di ampliamento che si applica all'intero segnale. Le due figure nella pagina precedente forniscono un esempio da *dati sperimentali reali*. A sinistra, tre picchi asimmetrici approssimati ciascuno con due Gaussiane *simmetriche* (sei picchi in totale). A destra, questi tre picchi vengono approssimati ciascuno con una Gaussiana *ampliata esponenzialmente* (tre picchi in tutto). In questo caso, i tre picchi asimmetrici hanno tutti la stessa asimmetria e possono essere approssimati con la stessa costante di decadimento ("extra"). Inoltre, l'errore di approssimazione è leggermente inferiore per quello esponenzialmente allargato a tre picchi. *Entrambe le osservazioni tendono per l'approssimazione a tre picchi esponenzialmente ampliati piuttosto che quella a sei picchi.*

Nota: se i picchi scendendo a sinistra, anziché a destra come negli esempi sopra, si usa semplicemente un valore *negativo* per la costante di decadimento; per farlo in ipf.m (pag. 405), si preme Shift-X e si inserisce un valore negativo.

Un'alternativa a questo tipo di approssimazione della curva per picchi allargati esponenzialmente [consiste nell'usare la tecnica dell'addizione della derivata prima](#) (pagina 77) per rimuovere l'asimmetria e quindi approssimare il picco risultante con un modello simmetrico. Questo è più veloce in termini di tempo di esecuzione del computer, soprattutto per segnali con molti picchi, ma richiede che la costante di tempo esponenziale sia nota o stimata sperimentalmente in anticipo.

L'Effetto dello Smoothing prima dell'analisi dei minimi



quadrati

In generale, non è consigliabile lo [smoothing](#) di un segnale prima di applicare l'approssimazione ai minimi quadrati, perché così facendo potrebbe distorcere il segnale, e può rendere difficile valutare correttamente i residui e potrebbe influenzare le stime della precisione del campionamento bootstrap, sottostimando l'effetto del rumore sulle variazioni dei parametri (pagina 162). [SmoothOptimization.m](#) è uno script Matlab/[Octave](#) che confronta l'effetto dello smoothing sulle misure dell'altezza di un picco Gaussiano con una semi-larghezza di 166 punti, più del rumore bianco con un rapporto segnale/rumore (SNR) di 10. Lo script utilizza tre metodi diversi:

- semplicemente prendendo come altezza del picco l'unico punto al centro del picco;
- utilizzando il metodo “[gaussfit](#)” per approssimare la metà superiore del picco (vedera pagina 164), e
- approssimare l'intero segnale con una Gaussiana utilizzando il metodo iterativo.

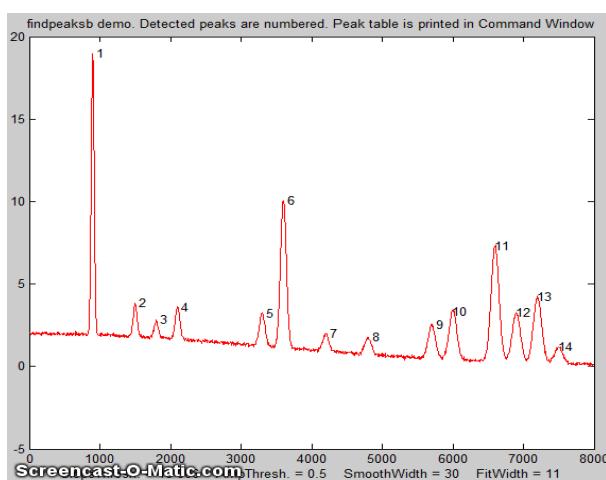
I risultati di 150 prove con campioni di rumore bianco indipendenti vengono mostrati sopra: un tipico segnale viene mostrato in alto a sinistra. Gli altri tre grafici mostrano l'effetto dell'SNR dell'altezza misurata rispetto al rapporto di smoothing (il rapporto tra la larghezza dello smoothing e la semi-larghezza del picco) per questi tre metodi di misurazione. I risultati mostrano che la semplice misura a punto singolo è effettivamente molto migliorata con lo smoothing, come previsto; tuttavia, l'SNR ottimale (che migliora di circa la radice quadrata dell'ampiezza del picco di 166 punti) si ottiene solo quando il rapporto di smoothing si avvicina a 1.0, e che più smoothing distorce notevolmente la forma del picco, riducendone l'altezza del 40% circa. I metodi di approssimazione sono molto meno influenzati dallo smoothing e il metodo iterativo quasi per niente. Quindi la conclusione è che *non* si dovrebbe eseguire lo smoothing prima dell'approssimazione, perché distorcerebbe il picco e non si otterrà alcun vantaggio significativo in termini di SNR. Le uniche situazioni in cui potrebbe essere vantaggioso lo smoothing prima dell'approssimazione sono:

- quando il rumore nel segnale è ponderato ad alta frequenza (cioè [rumore "blu"](#)), dove il filtro passa-basso renderà i [picchi più evidenti](#) allo scopo di impostare i punti di partenza per un'approssimazione iterativa, o

(b) se il segnale è contaminato da artefatti con picchi stretti di ampiezza elevata, nel qual caso un [pre-filtro basato sulla mediana](#) o un'altra [funzione di soppressione degli spike](#), può rimuovere i picchi senza un eccessivo cambiamento nel resto del segnale. Oppure, in situazione totalmente diversa, se si desidera approssimare una curva che unisce i picchi successivi di un'onda modulata, chiamata "inviluppo", allora è possibile eseguire lo smoothing del valore assoluto dell'onda prima di approssimare l'inviluppo).

Ricerca e Misura dei Picchi

Un requisito nell'elaborazione dei dati scientifici è rilevare i picchi in un segnale e misurarne posizioni, altezze, larghezze e/o aree. Un modo per farlo è utilizzare il fatto che la [derivata](#) prima di



un picco ha un [zero-crossing](#) discendente al massimo del picco (pagina 63). Tuttavia, la presenza di rumore casuale nel segnale sperimentale reale causerà molti falsi passaggi per lo zero semplicemente a causa del rumore. Per evitare questo problema, la tecnica descritta esegue lo [smoothing](#) della derivata prima del segnale, poi cerca i passaggi per lo zero discendenti, e poi prende solo quei passaggi la cui pendenza è maggiore di un certo minimo predeterminato (la "soglia della pendenza") in un punto in cui il segnale originale supera un certo minimo (la "soglia dell'ampiezza").

Regolando la larghezza dello smoothing, la soglia della pendenza e la soglia dell'ampiezza, è possibile rilevare solo i picchi desiderati e ignorare la maggior parte di quelli troppo piccoli, troppo larghi o troppo stretti. Inoltre, questa tecnica può essere estesa per stimare la posizione, l'altezza e la larghezza di ciascun picco mediante l'[approssimazione dei quadrati minimi](#) di un segmento del *segnalet originale senza smoothing* in prossimità del passaggio per lo zero. Pertanto, anche se è necessario un forte smoothing della derivata prima per fornire una discriminazione affidabile contro i picchi di rumore, i parametri estratti dall'approssimazione *non vengono distorti dallo smoothing* e l'effetto del rumore casuale nel segnale viene ridotto dall'approssimazione della curva su più punti nel picco. Questa tecnica può misurare le posizioni e le altezze in modo abbastanza accurato, ma le misure delle ampiezze e delle aree sono più accurate se i picchi sono di forma Gaussiana (o Lorentziana, nella variante `findpeaksL`). Per la misura più accurata di altri profili di picchi, o di quelli molto sovrapposti, o dei picchi sovrapposti ad una linea di base, le funzioni correlate `findpeaksb.m`, `findpeaksb3.m`, `findpeaksfit.m` utilizzano l'[approssimazione iterativa non-lineare](#) con un profilo selezionabile e una modalità di correzione della linea di base.

La routine è ora disponibile in diverse versioni descritte di seguito:

(1) un insieme di funzioni a riga di comando per Matlab o Octave, ciascuna collegata alla sua descrizione: `peaksat.m`, `findpeaksx`, `findpeaksxw`, `findpeaksG`, `findpeaksGw`, (riduzione del rumore basato sulle Wavelet, richiede la funzione "wdenoise" che si trova nel Wavelet Toolbox), `findvalleys`, `findpeaksL`, `measurepeaks`, `findpeaksGd`, `findpeaksb`, `findpeaksb3`, `findpeaksplot`, `findpeaksplotL`, `peakstats`, `findpeaksE`, `findpeaksGSS`, `findpeaksLSS`, `findpeaksT`, `findpeaksfit`, `autofindpeaks`, e `autopeaks`. Queste si possono usare come componenti nella creazione dei propri script e funzioni. Non confondere la funzione "[findpeaks](#)" nel *Signal Processing Toolbox* di Matlab; che è un algoritmo totalmente diverso.

(2) una [funzione interattiva da tastiera](#), chiamata **iPeak** ([ipeak.m](#) o [ipeakoctave](#)), pagina 245, per regolare i criteri di rilevamento del picco in modo interattivo ottimizzati per qualsiasi tipo di picco particolare *iPeak* viene eseguito nella finestra Figure ed usa un semplice insieme di comandi da tastiera per ridurre l'occupazione dello schermo, minimizzare il lavoro extra e massimizzare la velocità di elaborazione.

(3) Un insieme di [spreadsheet](#), disponibili nei formati *Excel* e *OpenOffice*.

(4) Il rilevamento in *tempo reale* del picco in Matlab è discusso a pagina 339.

[Cliccare qui per scaricare il file ZIP "PeakFinder.zip"](#), che include findpeaksG.m e le sue varianti, ipeak.m e un file di dati di esempio e script demo per i test. Si può anche scaricare *iPeak* ed altri programmi da [Matlab File Exchange](#).

Semplice rilevamento dei picchi

allpeaks.m. $P = \text{allpeaks}(x, y)$ Un semplicissimo 'peak detector', per insiemi di dati x, y , che elenca *ogni* valore di y che sia più basso dei valori y più bassi da entrambi i lati. Una versione correlata, [allpeaksw.m](#), stima anche la larghezza dei picchi. [allvalleys.m](#) elenca ogni valore y che abbia valori *di y più alti* da entrambi i lati. Digitare "help allpeaks" o "help allpeaksw" per un esempio. "help allpeaks" o "help allpeaksw" per vedere un esempio di applicazione.

peaksat.m. (Peaks Above Threshold) Sintassi: $P = \text{peaksat}(x, y, \text{threshold})$. Questa funzione rileva ogni valore y con valori y inferiori su entrambi i lati *e* è maggiore di una soglia specificata. Restituisce una matrice P con i valori x e y di ciascun picco, dove n è il numero di picchi rilevati. Una versione correlata, [peaksatw.m](#), stima anche l'ampiezza dei picchi. Digitare "help peaksat" o "help peaksatw" per un esempio. Digitare "help peaksat" o "help peaksatw" per vedere un esempio di applicazione.

peaksatG.m. ("Peaks Above Threshold/Gaussian")

$P = \text{peaksatG}(x, y, \text{threshold}, \text{peakgroup})$ Questa funzione è come *peakat.m* ma in aggiunta approssima anche i minimi quadrati alla parte superiore di ogni picco rilevato con una Gaussiana per stimare la loro larghezza e area; il numero di punti in cima al picco che si approssimano è determinato dall'argomento di input "peakgroup". Restituisce una matrice P , 5 per n , con i valori x e y per ciascun picco, dove n è il numero di picchi rilevati. Digitare "help peaksatG" per gli esempi.

Queste semplici funzioni non contengono alcuno smoothing dei dati. Se i dati sono rumorosi, si può applicare preventivamente lo smoothing separatamente per evitare di rilevare i picchi di rumore (vedere pagina38). Le seguenti funzioni, invece, applicano lo smoothing prima del rilevamento del picco.

findpeaksx.m è una funzione Matlab/Octave a riga di comando per *localizzare e contare* i picchi positivi in un insieme di dati rumoroso.

```
P=findpeaksx(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, PeakGroup, smoothtype)
```

Rileva i picchi cercando gli attraversamenti per lo zero della derivata prima con smoothing che superano SlopeThreshold e le ampiezze che superano AmpThreshold restituendo un elenco (nella matrice **P**) contenente il numero del picco, la posizione e l'altezza misurata per ciascun picco (e per la variante [findpeaksxw](#), la [larghezza a metà del massimo](#), determinata chiamando la funzione [halfwidth.m](#)). Può trovare e contare oltre 10.000 picchi al secondo in segnali molto grandi. I dati vengono passati alla funzione findpeaksx nei vettori x e y (x = variabile indipendente, y = variabile dipendente). Gli altri parametri sono regolabili dall'utente:

SlopeThreshold - Pendenza della derivata prima filtrata con lo smoothing che viene presa per indicare un picco. Questo discrimina sulla base dell'ampiezza del picco. Valori più grandi di questo parametro trascureranno le caratteristiche generali del segnale. Un valore iniziale ragionevole per i picchi Gaussiani è $0.7 * \text{WidthPoints}^{-2}$, dove WidthPoints è il *numero di punti* nella semi-larghezza ([FWHM](#)) del picco.

AmpThreshold - Discrimina sulla base dell'altezza. Eventuali picchi con altezza inferiore a questo valore vengono ignorati.

SmoothWidth - Larghezza della funzione di smoothing applicata ai dati prima che venga misurata la pendenza. Più è grande SmoothWidth più verranno scartate i tratti piccoli e stretti. Un valore ragionevole è in genere circa uguale a 1/2 del *numero di punti* nella semi-larghezza dei picchi.

PeakGroup - Il numero di punti intorno alla "parte superiore" del picco (senza smoothing) che vengono presi per stimare le altezze dei picchi. Se il valore di PeakGroup è 1 o 2, il valore y massimo di 1 o 2 punti nel punto di passaggio per lo zero viene preso come valore dell'altezza del picco; se PeakGroup n è 3 o maggiore, viene presa la *media* dei successivi n punti come altezza del picco. Per gli spike o picchi strettissimi, mantenere PeakGroup=1 o 2; per picchi ampi o rumorosi, aumentare il PeakGroup per ridurre l'effetto del rumore.

Smoothtype determina l'algoritmo di smoothing (pag. 39)

Se smoothtype=1, rettangolare ((slittamento-della-media o boxcar)

Se smoothtype=2, triangolare (2 passaggi dello slittamento-della-media)

Se smoothtype=3, p-spline (3 passaggi dello slittamento-della-media)

Fondamentalmente, valori più alti producono una maggiore riduzione del rumore ad alta frequenza, a scapito di un'esecuzione più lenta. Per un confronto di questi tipi di smoothing, vedere pagina 56.

Gli script dimostrativi [demofindpeaksx.m](#) e [demofindpeaksxw.m](#) trovano, numerano, tracciano e misurano picchi rumorosi con posizioni casuali ignote. (Notare che se due picchi si sovrappongono troppo, la larghezza riportata sarà l'ampiezza dell'*unione* dei picchi; in tal caso, è meglio usare [findpeaksG.m](#).

Dimostrazione di velocità. Qui vengono confrontate le funzioni di ricerca dei picchi di cui sopra, in Matlab, su un tipico PC desktop o portatile. Nota: Le funzioni "tic" e "toc" di Matlab vengono usate per determinare il tempo trascorso.

Peaksat.m:

Il tempo impiegato è 0.025 sec. su un desktop Dell XPS i7 3.5Ghz, ovvero *523,000 picchi al secondo*.

```
findpeaksx.m: >> x=[0:.01:500]'; y=x.*sin(x.^2).^2; tic;
P=findpeaksx(x,y,0,0,3,3); toc; NumPeaks=length(P)
```

Il tempo trascorso è di 0,11 secondi sullo stesso computer, ovvero *110.000 picchi al secondo*.

Misura del picco Gaussiano

```
P=findpeaksG(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth,  
smoothtype)  
  
P=findpeaksL(...)
```

Queste funzioni Matlab/Octave individuano i picchi positivi in un insieme di dati rumoroso, eseguono un'[approssimazione dei quadrati minimi](#) di una funzione Gaussiana Lorentziana alla parte superiore del picco e calcolano la posizione, l'altezza e la larghezza ([FWHM](#)) di ciascun picco approssimato. (Il 6° argomento di input, FitWidth, è il numero dei punti intorno a ciascun massimo approssimato). Gli altri argomenti sono gli stessi di findpeaksx. Restituisce un elenco (nella matrice P) contenente il numero del picco e la *posizione*, *altezza*, *larghezza* e *area* di ciascun punto. Può trovare e approssimare oltre 1800 picchi al secondo in segnali molto grandi. (Ciò è utile principalmente per i segnali che hanno diversi punti per ciascun picco, non per i picchi che hanno solo uno o due punti, per i quali [findpeaksx](#) è migliore).

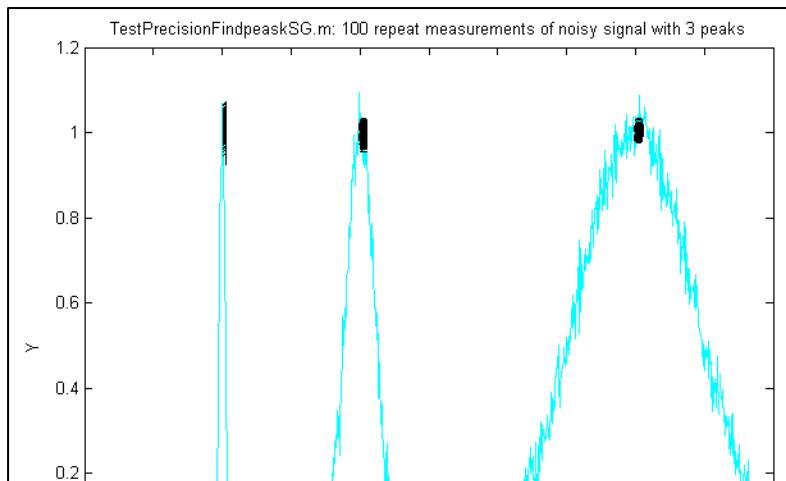
```
>> x=[0:.01:50]; y=(1+cos(x)).^2; P=findpeaksG(x,y,0,-1,5,5); plot(x,y)  
P =  
1 6.2832 4 2.3548 10.028  
2 12.566 4 2.3548 10.028  
3 18.85 4 2.3548 10.028...
```

La funzione [findpeaksplot.m](#) è una semplice variante di findpeaksG.m che in più *disegna* i dati x,y e il numero dei picchi sul grafico (se presenti). La funzione [findpeaksplotL.m](#) fa la stessa cosa ottimizzato per il picco Lorentziano.

[findpeaksSG.m](#) è una variante *segmentata* della funzione findpeaksG, con la stessa sintassi, tranne per il fatto che i quattro parametri di rilevamento del picco possono essere *vettori*, dividendo il segnale in regioni che è possibile ottimizzare per picchi di larghezze diverse. È possibile dichiarare un numero qualsiasi di segmenti, in base alla lunghezza del terzo argomento di input (SlopeThreshold). (Nota: è sufficiente inserire i vettori per quei parametri che si desiderano variare tra i segmenti; per consentire a qualsiasi altro parametro di rilevamento del picco di rimanere *lo stesso* per tutti i segmenti, si immette semplicemente un *singolo valore scalare* per quel parametro; solo SlopeThreshold dev'essere un vettore). ([FindpeaksSL.m](#) è la stessa cosa per i picchi Lorentziani). L'esempio seguente dichiara due segmenti, con AmpThreshold che rimane lo stesso in entrambi i segmenti.

```
SlopeThreshold=[0.001 .0001];  
AmpThreshold=.2;  
SmoothWidth=[5 10];  
FitWidth=[10 20];  
P=findpeaksSG(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth,3);
```

Nell'esempio grafico mostrato di seguito a lato, lo script dimostrativo [TestPrecisionFindpeaksSG.m](#) crea un segnale rumoroso con tre picchi di diverse larghezze, rileva e misura le posizioni, le altezze e le larghezze di ogni picco utilizzando findpeaksSG, quindi stampa le percentuali relative delle deviazioni standard dei parametri dei tre picchi in 100 misure con rumore casuale indipendente. Con i parametri per la ricerca di 3 segmenti, findpeaksSG rileva e misura in modo affidabile tutti e tre i picchi. Al contrario, findpeaksG, quando è regolato sul picco medio (utilizzando la



riga 26 invece della 25), misura male il primo e l'ultimo picco, perché i parametri di rilevamento sono tutt'altro che ottimali per quelle larghezze. Si può vedere anche che la *precisione* delle misure delle altezze diventa progressivamente *migliore* (deviazione standard relativa minore) quanto *maggiori* sono le larghezze, semplicemente perché nei picchi più larghi ci sono *più punti*. (È possibile modificare qualsiasi variabile nelle righe 10-18).

Una funzione correlata è [**findpeaksSGw.m**](#) che è come la precedente tranne per il fatto che utilizza il *denoising wavelet* (pag. 130) anziché lo smoothing (richiede il [Wavelet Toolbox](#)). Come argomento di input accetta il "livello" della wavelet anziché la larghezza dello smoothing. Lo script [TestPrecisionFindpeaksSGvsW.m](#) confronta la precisione e l'accuratezza della misura della posizione e dell'altezza del picco, sia per la normale funzione [**findpeaksSG.m**](#) che per [**findpeaksSGw.m**](#) basata su wavelet, trovando che c'è poco da guadagnare nella maggior parte dei casi usando il denoising wavelet anziché lo smoothing. Ciò è principalmente dovuto al fatto che in entrambi i casi le misure dei parametri si basano sui minimi quadrati che si approssimano ai dati *originali*, non a quelli con *smoothing*, per ciascuna posizione rilevata, quindi il solito vantaggio del denoising wavelet di evitare distorsioni di attenuazione qui non si applica.

Un inconveniente con le suddette funzioni di ricerca dei picchi è il fastidioso dover stimare i valori dei [parametri di rilevamento](#) che si devono usare nei segnali. Un modo rapido per stimarli consiste nell'usare [**autofindpeaks.m**](#), che è simile a [**findpeaksG.m**](#) tranne per il fatto che *si possono facoltativamente omettere i parametri di rilevamento* e scrivere semplicemente "autofindpeaks(x, y)" o "autofindpeaks(x, y, n)", dove *n* è la "peak capacity", approssimativamente il numero di picchi che si approssimerebbe alla registrazione del segnale (un *n* maggiore cerca molti picchi stretti; un *n* più piccolo cerca pochi picchi più larghi e trascura i particolari della struttura).

Semplicemente, *n* consente di *regolare rapidamente tutti i parametri di rilevamento contemporaneamente* modificando semplicemente un solo numero. Inoltre, se si evita di esplicitare i parametri di rilevamento dei picchi, autofindpeaks *stamperà l'elenco numerico degli argomenti di input* che usa, nella finestra di comando, poi si possono copiare, incollare e correggere per utilizzarli con qualsiasi altra della funzioni findpeaks... Se si chiama [**autofindpeaks**](#) con gli argomenti di *output* [P,A]=autofindpeaks(x,y,n), restituisce i parametri di rilevamento del picco calcolati come un vettore di 4 righe dell'elemento A, che è possibile quindi trasferire ad altre funzioni come [**measurepeaks**](#), dando effettivamente a tale funzione la capacità di calcolare i parametri di rilevamento del picco da un singolo numero *n*. Per esempio, nel seguente segnale, una stima visiva rileva circa 20 picchi, quindi si usa 20 come 3° argomento:

```
x=[0:.1:50];
y=10+10.*sin(x).^2+randn(size(x));
[P,A]=autofindpeaks(x,y,20);
```

Poi è possibile utilizzare A come parametri di rilevamento del picco per le altre funzioni di rilevamento, come **P=findpeaksG(x,y,A(1),A(2),A(3),A(4),1)** o **P=measurepeaks(x,y,A(1),A(2),A(3),A(4),1)**. Probabilmente si vorrà mettere a punto manualmente la soglia della *larghezza* A(2) per le proprie esigenze, ma quella è più facile da conoscere.

Digitare "help autofindpeaks" ed eseguire gli esempi. (La funzione [autofindpeaksplot.m](#) è la stessa ma, in più, traccia e numera i picchi). Lo script [testautofindpeaks.m](#) esegue tutti gli esempi nel file della guida, disegna i dati e numera i picchi (come [autofindpeaksplot.m](#)), con una pausa di 1 secondo tra ogni esempio (se si sta leggendo online, cliccare per il [grafico animato](#)).

Ottimizzazione della ricerca dei picchi

La ricerca dei picchi in un segnale dipende dalla distinzione tra i picchi legittimi e altre caratteristiche come il rumore e le modifiche alla linea di base. Idealmente, un rilevatore di picchi dovrebbe rilevare tutti i picchi legittimi e ignorare tutte le altre caratteristiche. Questo richiede che un rilevatore sia “intonizzato” o ottimizzato per i picchi desiderati. Per esempio, lo script dimostrativo Matlab/Octave [OnePeakOrTwo.m](#) crea un segnale (mostrato a lato) che potrebbe essere interpretato sia come *un picco* in $x=3$ su una linea di base curva che come *due picchi* in $x=5$ e $x=3$, a seconda del contesto. Gli algoritmi di ricerca qui descritti hanno argomenti di input che consentono una certa libertà per le regolazioni. In questo script di esempio, l'argomento “SlopeThreshold” viene regolato per rilevare solo uno o entrambi i picchi. Le funzioni

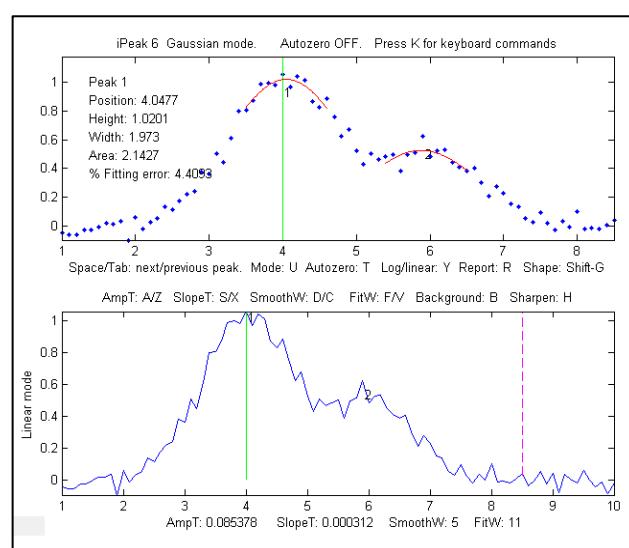
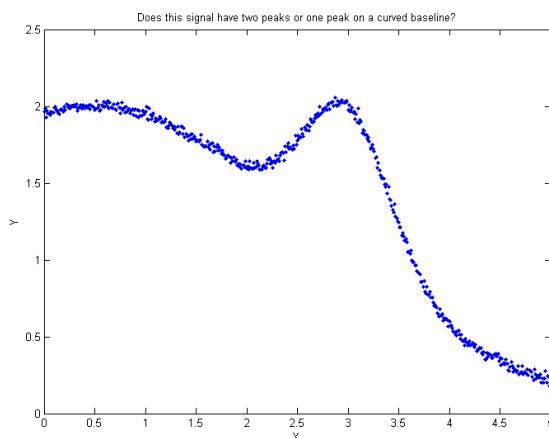
`findpeaks...` consentono *entrambe* le interpretazioni, a seconda dei parametri di rilevamento. I valori ottimali degli argomenti di input per `findpeaksG` e le relative funzioni dipendono dal segnale e da quali caratteristiche del segnale sono importanti per il proprio lavoro. I valori approssimativi di questi parametri possono essere stimati in base all'ampiezza dei picchi che si desiderano rilevare, [come descritto in precedenza](#), ma per il massimo controllo sarà meglio mettere a punto questi parametri per il proprio particolare segnale. Un modo semplice per farlo è usare [`autopeakfindplot\(x, y, n\)`](#) e regolare n finché non trova i picchi voluti; stamperà l'elenco numerico degli argomenti di input in modo da poterli copiare, incollare e modificare per utilizzarli con *qualsiasi* delle funzioni `findpeaks...` Un modo più flessibile, se si usa Matlab, è usare il *rilevatore interattivo di picchi iPeak* (pagina 245), che consente di regolare tutti questi parametri individualmente da tastiera e

visualizzare istantaneamente i risultati sul grafico.

Lo script [`FindpeaksComparison`](#) mostra come `findpeakG` si confronta con le altre funzioni di rilevamento quando si applica un segnale generato dal computer con più picchi con tipi e quantità variabili di linee di base e rumore casuale. Di per sé, `autofindpeaks.m`, `findpeaksG` e `findpeaksL` *non* correggono una linea di base diversa da zero; se i picchi sono sovrapposti ad una linea di base, è necessario sottrarla prima o utilizzare [le altre funzioni di rilevamento](#) che correggono la linea di base. Nell'esempio mostrato a lato (utilizzando il programma di rilevamento interattivo *iPeak* descritto a pagina 245), si suppone che le parti

importanti del segnale siano due ampi picchi in $x=4$ e $x=6$, il secondo con l'altezza metà del primo. Le piccole frastagliature sono solo rumore casuale. Si vogliono rilevare i due picchi e ignorare il rumore. (I picchi rilevati sono numerati 1,2,3,... nel pannello inferiore di questo grafico). Ecco come appare se *AmpThreshold* è [troppo piccolo](#) o [troppo grande](#), se *SlopeThreshold* è [troppo piccolo](#) o [troppo grande](#), se *SmoothWidth* è [troppo piccolo](#) o [troppo grande](#) e se *FitWidth* è [troppo piccolo](#) o [troppo grande](#). Se questi parametri rientrano nell'intervallo ottimale per questo obiettivo di misura, le funzioni `findpeaksG` restituiranno qualcosa di simile (sebbene i valori esatti varieranno con il rumore e con il valore di *FitWidth*):

```
Peak# Position Height Width Area
1 3.9649 0.99919 1.8237 1.94
```

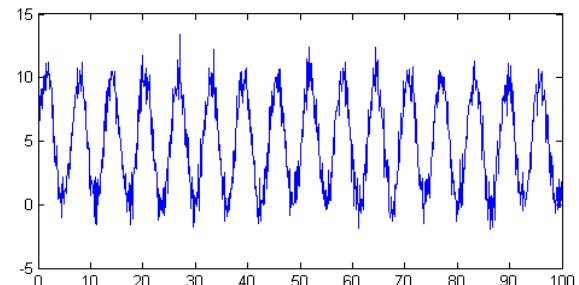


2 5.8675 0.53817 1.6671 0.955

In che modo ‘findpeaksG’ è diverso da ‘max’ in Matlab o ‘findpeaks’ nel *Signal Processing Toolkit*?

La funzione ‘max’ restituisce semplicemente il *singolo* valore più grande in un vettore. [Findpeaks](#) nel *Signal Processing Toolbox* può essere utilizzato per trovare i valori e gli indici di tutti i picchi che sono più alti di una specifica altezza e separati dai loro vicini da una certa distanza minima. La mia versione di findpeaks ([findpeaksG](#)) accetta sia una variabile indipendente (x) che una variabile dipendente (y), trova i punti in cui la curvatura media su una certa regione è concava verso il basso, esegue un'approssimazione dei quadrati minimi in tale regione e restituisce la posizione (in unità x), l'altezza, la larghezza e l'area di ogni picco che supera quella specificata. Per esempio, per creare una serie di picchi rumorosi (disegnati a lato) e si applicano entrambe le funzioni findpeaks ai dati risultanti:

```
x=[0:.1:100];
y=5+5.*sin(x)+randn(size(x));
plot(x,y)
```



Ora, la maggior parte delle persone solo guardando questo grafico conterebbe *16 picchi*, con un'altezza media di circa 10 unità. Ogni volta che vengono eseguite le istruzioni di cui sopra, il rumore casuale è diverso, ma si conterebbero comunque i 16 picchi perché il rapporto segnale/rumore è 10, che non è poi così male. Ma la funzione findpeaks nel *Signal Processing Toolbox* conta da 11 a 20 picchi, con un'altezza media (PKS) di 11.5.

```
[PKS,LOCS]=findpeaks(y,'MINPEAKHEIGHT',5,'MINPEAKDISTANCE',11)
```

Al contrario, la funzione findpeaksG [findpeaksG\(x,y,0.001,5,11,11,3\)](#) conta sempre 16 picchi, con un'altezza media di 10 ± 0.3 , che è molto più ragionevole. Misura anche la larghezza e l'area, assumendo che i picchi siano Gaussiani (o Lorentziani, nella variante findpeaksL). Per essere onesti, la funzione [findpeaks](#) nel *Signal Processing Toolbox*, o l'ancora più veloce [findpeaksx.m](#), funziona meglio per i picchi che hanno solo 1-3 punti sul picco; findpeaksG è migliore per i picchi che hanno più punti.

Lo script dimostrativo [FindpeaksSpeedTest.m](#) confronta la velocità dei quattro rilevatori di picco su uno stesso segnale di test di grandi dimensioni: Signal Processing Toolkit [findpeaks](#), [peaksat](#), [findpeaksx](#) e [findpeaksG](#):

```
Number Elapsed Peaks per
Function of peaks time, sec second
findpeaks (SPT) 160 0.012584 12715
peaksat 999 0.0012912 773699
findpeaksx 158 0.001444 109418
findpeaksG 157 0.011005 14267
```

Ricerca degli avvallamenti

Esiste anche una funzione simile per trovare *avvallamenti* (i minimi), chiamata [findvalleys.m](#), che funziona allo stesso modo di findpeaksG.m, tranne per il fatto che individua i *minimi* anziché i *massimi*. Vengono rilevati solo gli avvallamenti al di sopra di AmpThreshold (ovvero, più positivi o meno negativi); se si desidera rilevare quelli che hanno minimi negativi, allora AmpThreshold dev'essere impostato su un valore più negativo.

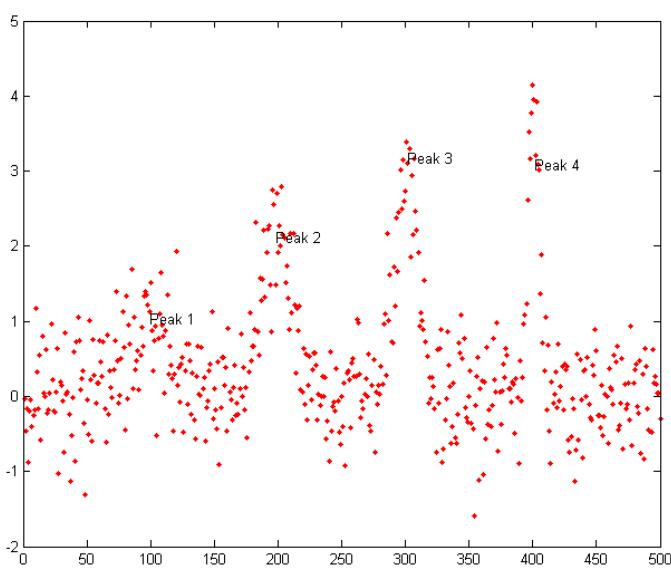
```
>> x=[0:.01:50];y=cos(x);P=findvalleys(x,y,0,-1,5,5)
```

```

P =
1.0000 3.1416 -1.0000 2.3549 0
2.0000 9.4248 -1.0000 2.3549 0
3.0000 15.7080 -1.0000 2.3549 0
4.0000 21.9911 -1.0000 2.3549 0....
....
```

Accuratezza delle misure dei picchi

L'accuratezza delle misure di posizione, altezza, larghezza e area con la funzione `findpeaksG` dipende dal profilo dei picchi, dall'entità della sovrapposizione, dall'intensità del background e dal rapporto segnale/rumore. Le misure di larghezza e area in particolare sono fortemente influenzate dalla sovrapposizione dei picchi, dal rumore e dalla scelta di `FitWidth`. I picchi isolati di forma Gaussiana vengono misurati in modo più accurato. Per i picchi *Lorentziani*, si usa invece [findpeaksL.m](#) (la sola differenza è che le altezze, le larghezze e le aree dei picchi riportati saranno più accurate se i picchi sono Lorentziani). Vedere "ipeakdemo.m" di seguito per una prova sull'accuratezza per i picchi Gaussiani. Per picchi molto sovrapposti che non mostrano massimi distinti, utilizzare [peakfit.m](#) o l'[Interactive Peak Fitter \(ipf.m\)](#) (pagina 405). Per un confronto diretto dell'accuratezza di `findpeaksG` e `peakfit`, eseguire lo script dimostrativo [peakfitVSfindpeaks.m](#).



Questo script genera quattro picchi molto rumorosi di diverse altezze e larghezze, poi li misura in due modi diversi: prima con `findpeaksG.m` (figura a lato) e poi con [peakfit.m](#), e confronta i risultati. I picchi rilevati da `findpeaksG` sono etichettati come "Peak 1", "Peak 2", ecc. Se si esegue questo script più volte, si genereranno gli stessi picchi ma ogni volta con *campioni diversi di rumore casuale*. Si scoprirà che entrambi i metodi funzionano bene per la maggior parte delle volte, con `peakfit` che fornisce errori più piccoli nella maggior parte dei casi (perché utilizza *tutti* i punti di ciascun picco, non solo la parte superiore), ma

occasionalmente `findpeaksG` perderà il primo picco (il più basso) e raramente rileverà un 5° picco che non è presente. D'altra parte, in questo caso `peakfit.m` è vincolato a contenere 4 e solo 4 picchi ogni volta.

Lo script dimostrativo [FindpeaksComparison](#) confronta l'accuratezza di `findpeaksG` e `findpeaksL` con diverse funzioni di rilevamento quando applicati a segnali con più picchi, con diverse quantità e tipologie di linee di base e di rumore casuale.

Ricerca del picco con l'approssimazione iterativa.

[findpeaksb.m](#) è una variante di `findpeaksG.m` che misura più accuratamente i parametri del picco utilizzando l'[approssimazione iterativa dei quadrati minimi](#) basato sulla funzione [peakfit.m](#). Ciò produce valori dei parametri migliori rispetto al solo `findpeaksG` per tre motivi:

- (1) può essere impostato per diversi profili con l'argomento di input 'PeakShape';
- (2) approssima *tutto* il picco, non solo la parte superiore; e
- (3) prevede la sottrazione del background (quando l'argomento di input "BaselineMode" è impostato su 1, 2 o 3: lineare, quadratico o piatto, rispettivamente).

Questa funzione lavora al meglio con i picchi isolati che non si accavallano. Per la versione 3, la sintassi è `P = findpeaksb(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, windowspan, PeakShape, extra, BASELINEMODE)`. I primi sette argomenti di input sono gli stessi della funzione `findpeaksG.m`; se è stata utilizzata `findpeaksG` o `iPeak` per cercare e misurare i picchi nel segnale, si possono usare gli stessi valori degli argomenti di input di `findpeaksb.m`. I restanti quattro argomenti di input sono per la funzione [peakfit](#):

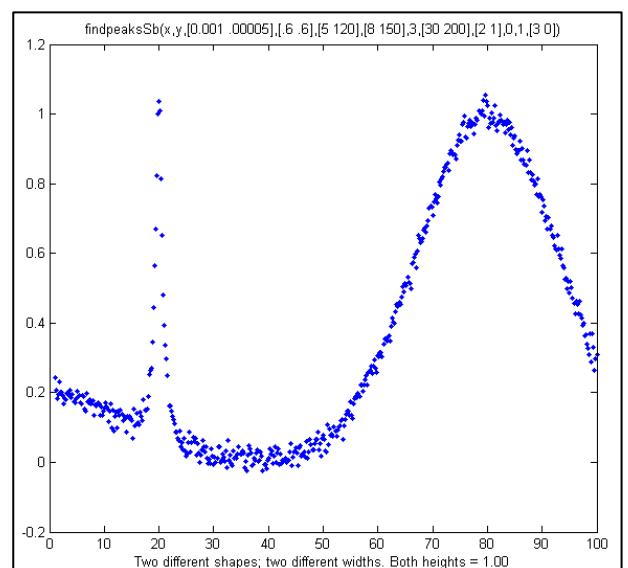
- "windowspan" specifica il numero di punti su cui ogni picco si approssima alla forma del modello (questo è il più difficile da stimare; in BaselineMode 1 e 2, 'windowspan' dev'essere abbastanza grande da coprire l'intero picco singolo e scendere in fondo su entrambi i lati del picco, ma non così grande da sovrapporsi ai picchi vicini); "PeakShape" specifica il profilo del modello di picco: 1=Gaussiana, 2=Lorentziana, ecc (digitare 'help findpeaksb' per un elenco),
- "extra" è la variabile del modificatore del profilo utilizzata per le Voigt, Pearson, Gaussiana e Lorentziana ampliate esponenzialmente, mix di Gaussiana/Lorentziana e Gaussiana e Lorentziana biforcuta, per mettere a punto la forma del picco;
- "BASELINEMODE" è 0, 1, 2, o 3 per la sottrazione di un background: nessuno, lineare, quadratico o piatto.

La tabella dei picchi restituita da questa funzione ha una sesta colonna che elenca gli errori di approssimazione percentuale per ogni picco. Ecco un semplice esempio con tre Gaussiane su un background lineare, confrontando `findpeaksG`, `findpeaksb` senza sottrazione di background (BASELINEMODE=0) e `findpeaksb` con la sottrazione del background (BASELINEMODE=1).

```
x=1:.2:100;Heights=[1 2 3];Positions=[20 50 80];Widths=[3 3 3];
y=2-(x./50)+modelpeaks(x,3,1,Heights,Positions,Widths)+.02*randn(size(x));
plot(x,y);
disp(' Peak Position Height Width Area % error')
PlainFindpeaks=findpeaksG(x,y,.00005,.5,30,20,3)
NoBackgroundSubtraction=findpeaksb(x,y,.00005,.5,30,20,3,150,1,0,0)
LinearBackgroundSubtraction=findpeaksb(x,y,.00005,.5,30,20,3,150,1,0,1)
```

Lo script dimostrativo [DemoFindPeaksb.m](#) mostra come funziona `findpeaksb` con più picchi su un background curvo, e [FindpeaksComparison](#) mostra come `findpeaksb` si confronta con le altre funzioni di rilevamento quando applicato a segnali con più picchi con tipi e quantità variabili di linee di base e rumore casuale.

Peak finder segmentato. Quali sono le larghezze dei picchi che variano notevolmente nel segnale? [findpeaksSb.m](#) è una variante segmentata di `findpeaksb.m`. Ha la stessa sintassi di `findpeaksb.m`, tranne per il fatto che gli argomenti di input SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, window, width, PeakShape, extra, NumTrials, BaselineMode e fixedparameters, possono essere tutti facoltativamente scalari o vettori con una voce per ciascun segmento, allo stesso modo di [findpeaksSG.m](#). Restituisce una matrice P che elenca il numero, la posizione, l'altezza, la larghezza, l'area, l'errore di approssimazione percentuale e lo "R2" di ciascun



picco rilevato. Nell'esempio a lato, i due picchi hanno la stessa altezza sulla linea di base (1.00) ma forme diverse (il primo Lorentziano e il secondo Gaussiano), larghezze molto diverse e linee di base diverse. Quindi, usando findpeaksG, findpeaksL o findpeaksb, sarebbe impossibile trovare un insieme di argomenti di input ottimale per entrambi i picchi. Tuttavia, utilizzando findpeaksSb.m, è possibile applicare impostazioni diverse a diverse regioni del segnale. In questo semplice esempio, ci sono solo *due* segmenti, definiti da SlopeThreshold con 2 valori diversi, gli altri argomenti di input sono gli stessi o sono diversi in quei due segmenti. Il risultato è che l'altezza del picco di entrambi i picchi viene misurata accuratamente. Innanzitutto, si definiscono i valori dei parametri di rilevamento, poi si chiama findpeaksSb.

```
>> SlopeThreshold=[.001 .00005]; AmpThreshold=.6; smoothwidth=[5 120];
peakgroup=[5 120];smoothtype=3; window=[30 200]; PeakShape=[2 1]; extra=0;
NumTrials=1; BaselineMode=[3 0];

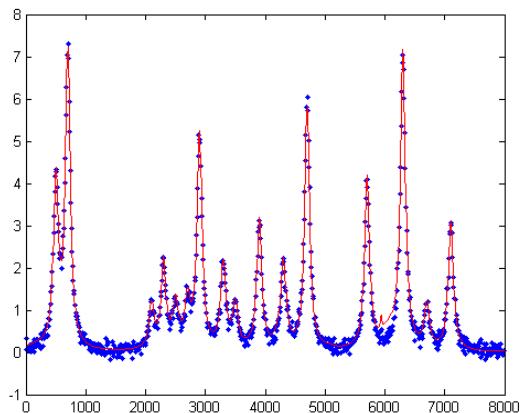
>> findpeaksSb(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup,
smoothtype, window, PeakShape, extra, NumTrials, BaselineMode)

Peak # Position Height Width Area
1 19.979 0.9882 1.487 1.565
2 79.805 1.0052 23.888 25.563
```

[DemoFindPeaksSb.m](#) mostra la funzione findpeaksSG.m creando un numero casuale di picchi Gaussiani le cui larghezze aumentano di un fattore 25 volte rispetto all'intervallo dell'asse x e che sono sovrapposti su una linea di base curva con rumore bianco casuale che aumenta gradualmente; In questo esempio vengono utilizzati quattro segmenti, modificando i valori di rilevamento e dell'approssimazione della curva in modo che tutti i picchi vengano misurati accuratamente. [Grafico. Stampa](#).

[findpeaksb3.m](#) è una variante più ambiziosa di findpeaksb.m che approssima ciascun picco rilevato assieme ai picchi precedenti e successivi trovati da findpeaksG.m, in modo da gestire meglio la sovrapposizione dei picchi adiacenti. La sintassi è

```
FPB=findpeaksb3(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup,
smoothtype, PeakShape, extra, NumTrials, BASELINEMODE, ShowPlots).
```



Lo script dimostrativo [DemoFindPeaksb3.m](#) mostra come funziona findpeaksb3 con gruppi irregolari di picchi Lorentziani sovrapposti, come nell'esempio a lato (digitare "help findpeaksb3") per ulteriori informazioni. Lo script dimostrativo [FindpeaksComparison](#) mostra come findpeaksb3 si confronta con le altre funzioni di rilevamento dei picchi quando applicato a segnali con più picchi con tipi e quantità variabili di linee di base e rumore casuale.

[findpeaksfit.m](#) è essenzialmente una combinazione seriale di findpeaksG.m e [peakfit.m](#). Utilizza il numero di picchi trovati, le posizioni e le larghezze dei picchi che sono l'output della funzione findpeaksG come input per la funzione peakfit.m, che quindi approssima l'*intero segnale* col modello specificato. Questa combinazione produce valori migliori rispetto al solo findpeaksG, perché peakfit approssima l'intero picco, non solo la parte superiore, e si occupa dei picchi non Gaussiani e sovrapposti. Tuttavia, approssima solo quei picchi trovati da findpeaksG, quindi ci si deve assicurare che tutti i picchi vengano trovati. La sintassi è:

```
function [P, FitResults, LowestError, BestStart, xi, yi] = findpeaksfit
(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype,
peakshape, extra, NumTrials, BaselineMode, fixedparameters, plots)
```

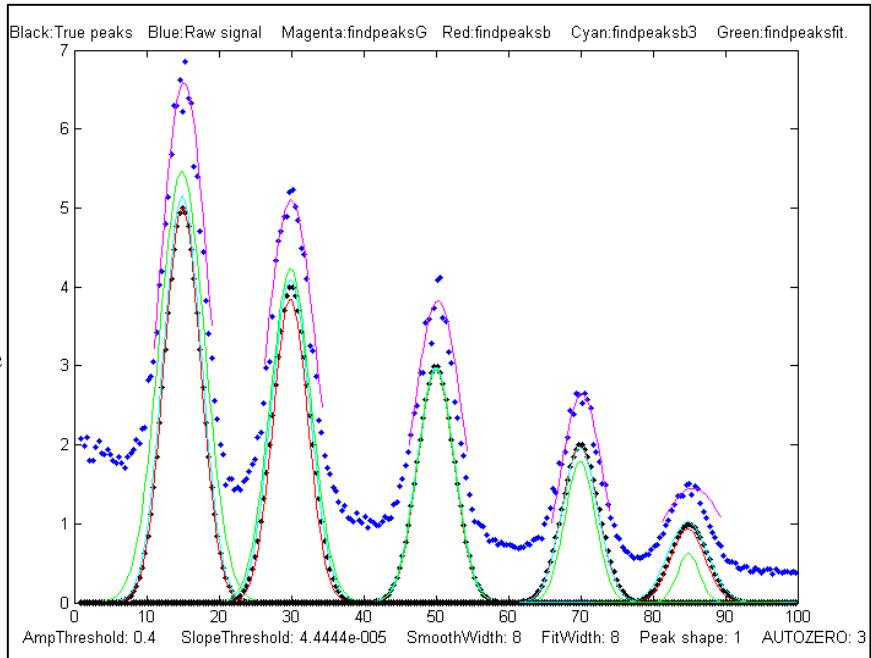
I primi sette argomenti di input sono esattamente gli stessi della funzione [findpeaksG.m](#); se è stata utilizzata [findpeaksG](#) o [iPeak](#) per cercare e misurare i picchi, si possono utilizzare gli stessi valori per gli argomenti di input di `findpeaksfit.m`. I restanti sei argomenti di `findpeaksfit.m` sono per la funzione [peakfit](#); se è stata utilizzata `peakfit.m` o [ipf.m](#) (pagina 405) per approssimare i picchi nei segnali, allora si possono utilizzare gli stessi valori per gli argomenti di input di `findpeaksfit.m`. Lo script dimostrativo [findpeaksfitdemo.m](#) è stato utilizzato per generare [l'animazione GIF](#) mostrata a lato. Mostra che `findpeaksfit` trova e approssima automaticamente i picchi in un set di 150 segnali, ognuno dei quali può avere da 1 a 3 picchi Lorentziani rumorosi in posizioni variabili, *rallentato artificialmente* con la funzione "pausa" in modo da poterlo vedere meglio. (Richiede l'installazione delle funzioni [findpeaksfit.m](#) e [lorentzian.m](#). Digitare "help `findpeaksfit`" per ulteriori informazioni).

Confronto delle funzioni per la ricerca dei picchi

Lo script dimostrativo

[FindpeaksComparison.m](#)

confronta l'accuratezza dei parametri di `findpeaksG/L`, `findpeaksb`, `findpeaksb3` e `findpeaksfit` applicati a un segnale generato dal computer con più picchi con in più tipi e quantità variabili di linee di base e rumore casuale. (Richiede queste quattro funzioni, oltre a `gaussian.m`, `lorentzian.m`, `modelpeaks.m`, `findpeaksG.m`, `findpeaksL.m`, `pinknoise.m` e `propnoise.m`, nel path di ricerca di Matlab/Octave). I risultati



vengono visualizzati graficamente nelle finestre delle figure [1](#), [2](#) e [3](#) e stampati in una tabella di accuratezza dei parametri e tempo trascorso per ciascun metodo, come mostrato di seguito. È possibile modificare le righe nello script contrassegnate da <<< per modificare il numero, il carattere e l'ampiezza dei picchi, della linea di base e del rumore. (Rendere il segnale simile al proprio per scoprire quale metodo funziona meglio per il proprio tipo di segnale). Il metodo migliore dipende principalmente dalla forma e dall'ampiezza della linea di base e dall'entità della sovrapposizione dei picchi. Digitare "[help FindpeaksComparison](#)" per i dettagli. (Tempo impiegato per Matlab 2020 su un Dell XPS i7 3.5Ghz).

Average absolute percent errors of all peaks

	Position error	Height error	Width error	Elapsed time, sec
<code>findpeaksG</code>	0.35955%	38.573%	25.797%	0.005768
<code>findpeaksb</code>	0.38828%	8.5024%	14.329%	0.069061
<code>findpeaksb3</code>	0.27187%	3.7445%	3.0474%	0.49538 <code>findpeaksfit</code>
	0.51930%	8.0417%	24.035%	0.27363

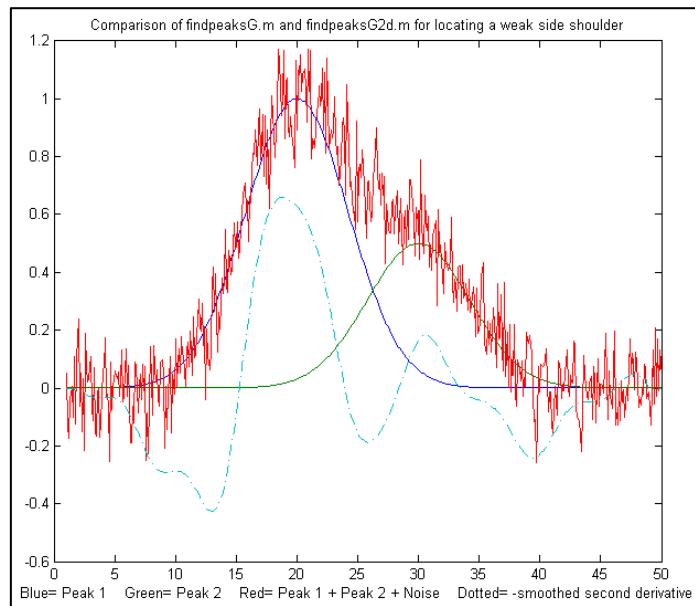
Nota: `findpeaksfit.m` differisce da `findpeaksb.m` in quanto `findpeaksfit.m` approssima contemporaneamente tutti i picchi trovati con un unico modello multi-picco, mentre `findpeaksb.m` approssima separatamente ciascun picco con un modello a picco unico e `findpeaksb3.m`

approssima ciascun picco rilevato *insieme al picco precedente e al successivo*. Di conseguenza, [findpeaksfit.m](#) funziona meglio con un numero relativamente piccolo di picchi che si sovrappongono tutti, mentre [findpeaksb.m](#) funziona meglio con un gran numero di picchi isolati non sovrapposti e [findpeaksb3.m](#) funziona per un gran numero di picchi che si sovrappongono al massimo a uno o due picchi adiacenti. [FindpeaksG/L](#) è semplice e veloce, ma non esegue la correzione della linea di base; [findpeaksfit](#) può eseguire una correzione della linea di base piatta, lineare o quadratica, ma funziona solo sull'intero segnale contemporaneamente; al contrario, [findpeaksb](#) e [findpeaksb3](#) eseguono *la correzione locale* della linea di base, che spesso funziona bene se la linea di base è curvo o irregolare.

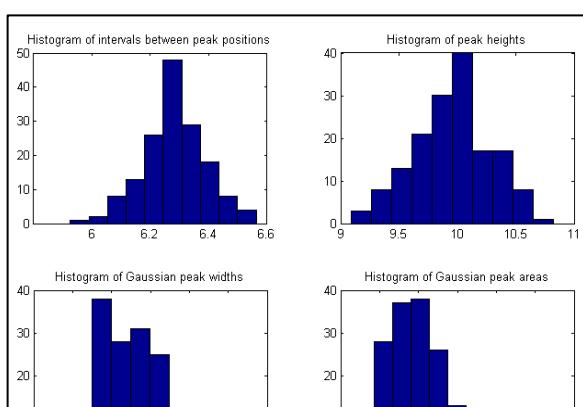
[findpeaksG2d.m](#) è una variante di [findpeaksG](#) che può essere utilizzata per individuare i picchi positivi *e le sporgenze* in un insieme rumoroso di dati temporali x-y. Rileva i picchi nel *negativo della derivata seconda* del segnale, cercando pendenze discendenti nella derivata *terza* che superano SlopeThreshold. Vedere [TestFindpeaksG2d.m](#).

[\[M,A\]=autopeaks.m](#) è un rilevatore di picchi di forma arbitraria; è fondamentalmente una combinazione di [autofindpeaks.m](#) e [measurepeaks.m](#). Ha una sintassi simile a [measurepeaks.m](#), tranne per il fatto che i parametri di rilevamento del picco

(SlopeThreshold, AmpThreshold, smoothwidth, peakgroup e smoothtype) possono essere omessi e la funzione calcolerà i valori di prova come fa [autofindpeaks.m](#). L'uso della sintassi [M,A]=autopeaks(x, y) funziona bene in alcuni casi, altrimenti provare [M,A]=autopeaks(x, y, n), utilizzando diversi valori di n (più o meno il numero di picchi che si approssimerebbero al segnale) finché non rileva i picchi che si desiderano misurare. Come [measurepeaks](#), restituisce una tabella M contenente il numero del picco, la posizione, l'altezza assoluta, la differenza picco-valle, l'area col taglio verticale (pag. 140), e quella del taglio tangente per ciascun picco rilevato (pag. 135), ma può anche restituire facoltativamente un vettore A contenente i parametri di rilevamento calcolati (per l'uso da parte di altre funzioni di rilevamento e approssimazione del picco). Per il controllo più preciso sul rilevamento, è possibile specificare tutti i parametri digitando M = autopeaks (x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup). [\[M,A\]=autopeaksplot.m](#) è lo stesso ma disegna anche il segnale e i singoli picchi come lo fa [measurepeaks.m](#) (mostrato sopra). Lo script [testautopeaks.m](#) esegue tutti gli esempi nel file della guida di [autopeaks](#), con una pausa di 1 secondo tra ciascuno di essi, stampando i risultati nella finestra di comando e inoltre disegnando e numerando i picchi (window 1) e ogni singolo picco (window 2); richiede [gaussian.m](#) e [fastsmooth.m](#) nel path di ricerca.



Statistiche dei picchi. La funzione [peakstats.m](#) usa lo stesso algoritmo di [findpeaksG](#), ma calcola e restituisce una tabella di statistiche riassuntive degli intervalli dei picchi (l'intervallo sull'asse x tra i picchi adiacenti rilevati), altezze, larghezze e aree, elencando la deviazione standard massima, minima, media e percentuale di ciascuna e, facoltativamente, traccia i dati x, y con i picchi numerati nella finestra della figura 1, stampa la tabella delle statistiche dei



picchi nella finestra di comando e disegna gli istogrammi degli intervalli dei picchi, altezze, larghezze e aree nei quattro quadranti della window 2. digitare "help peakstats". La sintassi è la stessa di findpeaksG, con l'aggiunta di un 8° argomento di input per controllare la visualizzazione e la stampa. La versione 2, marzo 2016, aggiunge mediana e modalità. Esempio:

```
x=[0:.1:1000];y=5+5.*cos(x)+randn(size(x));
PS=peakstats(x,y,0,-1,15,23,3,1);

Peak Summary Statistics
158 peaks detected
Interval Height Width Area
Maximum 6.6428 10.9101 5.6258 56.8416
Minimum 6.0035 9.1217 2.5063 28.2559
Mean 6.283 9.9973 3.3453 35.4737
% STD 1.8259 3.4265 15.1007 12.6203
Median 6.2719 10.0262 3.2468 34.6473
Mode 6.0035 9.1217 2.5063 28.2559
```

Con l'ultimo argomento di input omesso o uguale a zero, il disegno e la stampa nella finestra di comando vengono omessi; i valori numerici della tabella delle statistiche dei picchi vengono restituiti come un array 4x4, nello stesso ordine dell'esempio precedente.

tablestats.m (**PS=tablestats(P,displayit)**) è simile a peakstats.m tranne per il fatto che accetta come input una tabella dei picchi P come quella generata da findpeaksG.m, findvalleys .m, findpeaksL.m, findpeaksb.m, findpeaksplot.m, findpeaksnr.m, findpeaksGSS.m, findpeaksLSS.m o findpeaksfit.m - tutte le funzioni che restituiscono una tabella di picchi con almeno 4 colonne che elencano il numero di picco, altezza, larghezza e area. Calcola gli intervalli dei picchi (l'intervallo sull'asse x tra i picchi adiacenti rilevati) e la deviazione standard massima, minima, media e percentuale di ciascuno e, facoltativamente, visualizza gli istogrammi degli intervalli, delle altezze, delle larghezze e delle aree dei picchi nella finestra 2. Impostare l'ultimo argomento opzionale displayit = 1 se gli istogrammi devono essere visualizzati o no. Esempio:

```
x=[0:.1:1000];y=5+5.*cos(x)+.5.*randn(size(x));
figure(1);P=findpeaksplot(x,y,0,8,11,19,3);tablestats(P,1);
```

FindpeaksE.m è una variante di findpeaksG.m che stima inoltre l'errore di approssimazione relativo percentuale di ciascun picco (assumendo una forma Gaussiana del picco) e lo restituisce nella 6a colonna della tabella dei picchi.

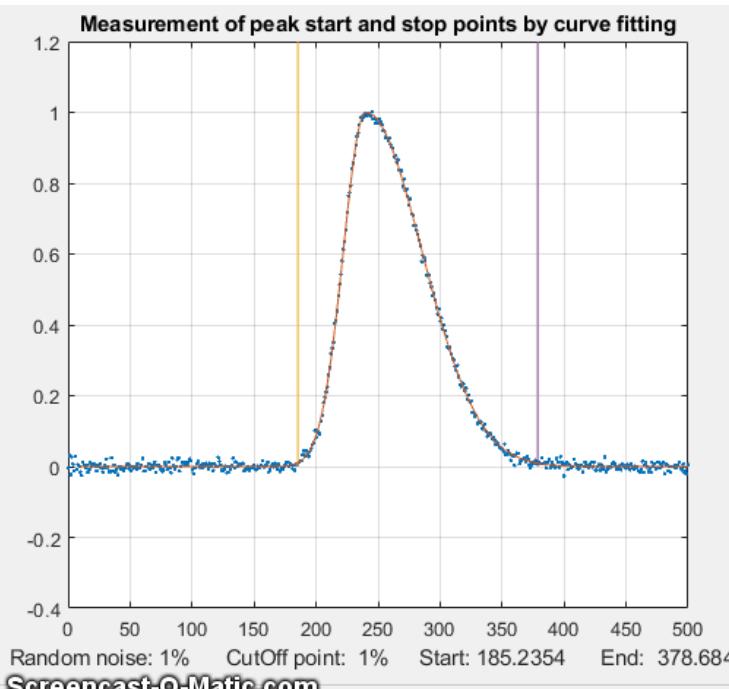
Esempio:

```
>> x=[0:.01:5];
>> y=x.*sin(x.^2).^2+.1*whitenoise(x);
>> P=findpeaksE(x,y,.0001,1,15,10)
P =
1 1.3175 1.3279 0.25511 0.36065 5.8404
2 1.4245 1.2064 0.49053 0.62998 10.476
3 2.1763 2.1516 0.65173 1.4929 3.7984
4 2.8129 2.8811 0.2291 0.70272 2.3318...
```

Inizio e fine del picco

Definire "inizio" e "fine" di un picco (il valore sulle x dove il picco inizia e dove finisce) è un po' arbitrario perché le forme tipiche dei picchi si avvicinano alla linea di base in modo *asintotico* lontano dal massimo del picco. È possibile definire i punti di inizio e fine del picco come i valori x dove il valore y è una piccola frazione, diciamo l'1%, dell'altezza del picco, ma è probabile che il rumore casuale sulla linea di base sarà una frazione più grande dell'ampiezza del segnale in quel punto. È probabile che lo smoothing per ridurre il rumore distorce e allarga i picchi, modificandone molto i punti iniziale e finale. Anche la sovrapposizione dei picchi complica notevolmente il problema. Una soluzione consiste nell'approssimare ciascun picco a un modello (pagina 165), poi

calcolare l'inizio e la fine dall'espressione del modello. Questo metodo riduce al minimo il problema del rumore approssimando i dati sull'intero picco e può gestire i quelli sovrapposti, ma funziona solo se i picchi possono essere modellati dai programmi di approssimazione disponibili. Per esempio, [si può mostrare](#) che i picchi Gaussiani raggiungono una frazione a dell'altezza in $x = p \pm \sqrt{w^2 \log(1/a)/(2 \sqrt{\log(2)})}$ dove p è la posizione del picco e w è la sua larghezza (la larghezza totale a metà del massimo). Quindi, per esempio se $a = .01$, $x = p \pm w * \sqrt{(\log(2) + \log(5))/(2 \log(2))} = 1.288784 * w$. [Si può mostrare](#) che i picchi Lorentziani raggiungono una



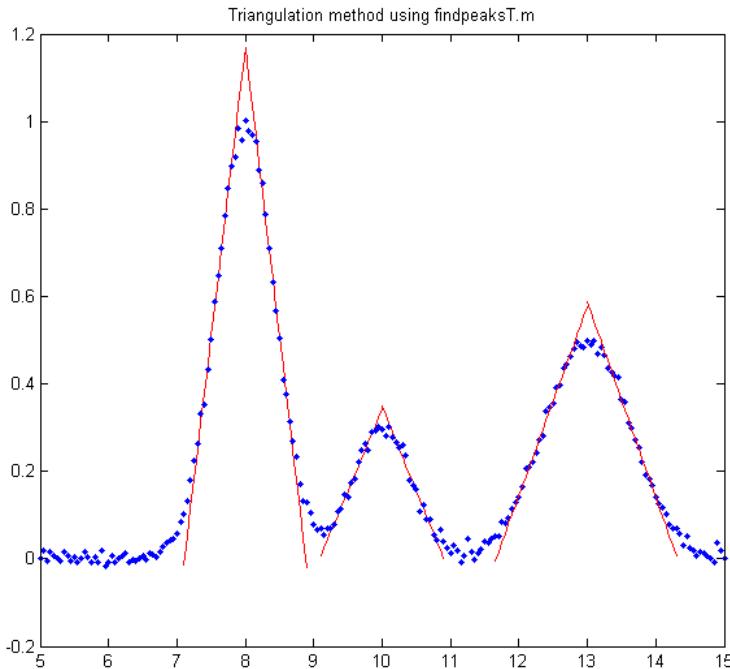
Screencast-O-Matic.com
Recorded with Windows

frazione a dell'altezza del in $x = p \pm \sqrt{[(w^2 - a w^2)/a]/2}$. Se $a = .01$, $x = p \pm (3/2 \sqrt{11}) * w = 4.97493 * w$. Le varianti di `findpeaksG` [**findpeaksGSS.m**](#) e [**findpeaksLSS.m**](#), per picchi Gaussiani e Lorentziani rispettivamente, calcolano l'inizio e la fine del picco in questo modo e li restituiscono nella 6a e 7a colonna della tabella dei picchi **P**. L'incertezza nella posizione del picco misurato p e specie della larghezza w renderà i risultati meno certi.

Il problema con questo metodo è che richiede un modello analitico del picco, definito come un'espressione in forma chiusa, che può essere risolto algebricamente per i punti di inizio e fine. Un metodo più versatile consiste nell'approssimare un modello ai dati mediante [l'approssimazione iterativa](#) (pagina 190), e poi usare il modello migliore [best-fit] per localizzare i punti di partenza e di fine per interpolazione. Per forme di picco complesse, il modello *non dev'essere limitato a un singolo picco*; le forme complesse e asimmetriche dei picchi possono spesso essere modellate come la somma di forme semplici, come le Gaussiane. Un esempio di questo metodo è mostrato nello script [**StartAndEnd.m**](#), che simula un picco rumoroso e asimmetrico e poi vi applica questo metodo utilizzando la funzione [**peakfit.m**](#) (pagina 384). Si può selezionare il punto di inizio/fine come una frazione dell'altezza del picco nella riga 8 dello script, la quantità di rumore casuale nella riga 7 e il numero di picchi nel modello nella riga 9. Nei punti di taglio, il rapporto segnale/rumore è molto scarso, quindi una misura diretta di x dove y è uguale al taglio non è pratica. Tuttavia, i punti iniziale e finale possono essere calcolati in modo sorprendentemente preciso ricavando il modello migliore con l'approssimazione dei minimi quadrati (contenuto negli argomenti di output xi e yi della funzione `peakfit`), che media il rumore sull'intero segnale (più dati ci sono meglio è). Il grafico nella pagina precedente mostra il metodo in funzione per 50 misure ripetute con diversi campioni di rumore casuali, prima con l'1% di rumore e poi col 10% di rumore. Se l'animazione non è visibile, cliccare su [questo link](#). Nonostante lo scarso rapporto segnale/rumore nei punti di taglio, la

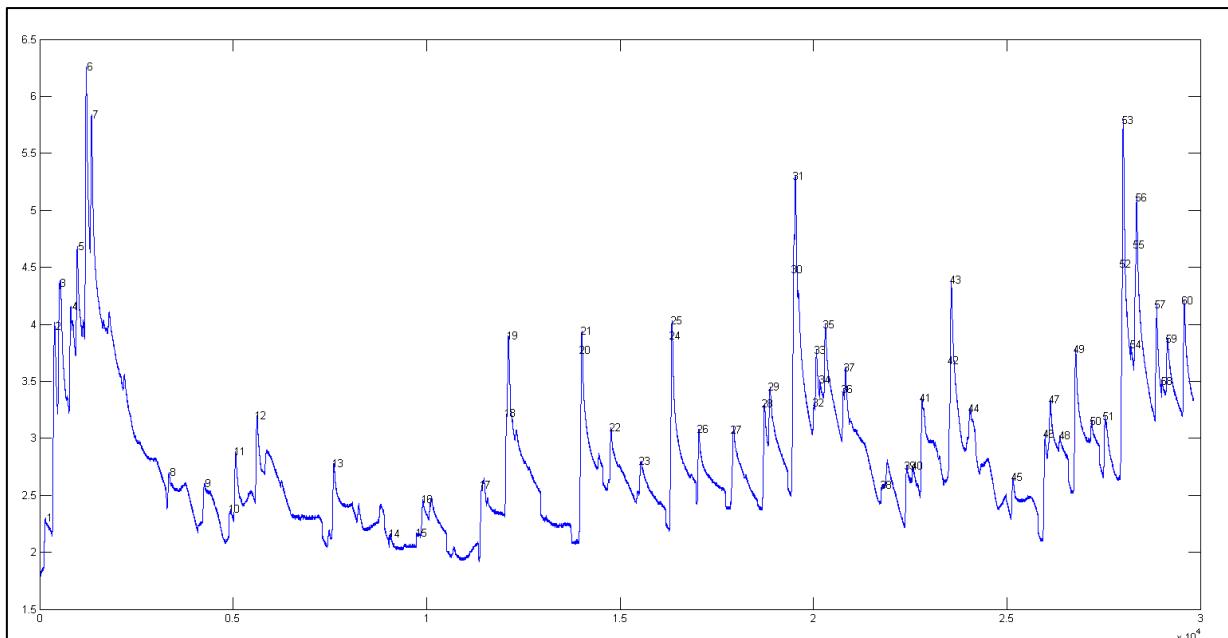
deviazione standard relativa delle misure dei punti di inizio e di fine (contrassegnati dalle linee verticali) è solo dello 0,2% circa. Anche quando il rumore è aumentato di 10 volte (riga 7), la deviazione standard relativa è ancora inferiore all'1%. (Se si dispone di un numero diverso di punti per picco, la precisione sarà inversamente proporzionale alla radice quadrata del numero di punti).

Il metodo di costruzione del triangolo. Prima dell'avvento dei computer e dell'elettronica, i parametri dei picchi venivano talvolta misurati *costruendo un triangolo attorno a ciascun picco* con i lati tangenti ai lati del picco, come mostrato di seguito. Questo vecchio metodo viene imitato dalle



funzioni [findpeaksT.m](#) e [findpeaksTplot.m](#), che sono mostrate dallo script [che ha generato questo grafico](#). In questo metodo l'altezza del picco è presa come l'apice del triangolo, che è leggermente più alto del picco della curva in esame. Risulta che le [prestazioni di questo metodo](#) sono scarse quando i segnali sono molto rumorosi o se i picchi si sovrappongono, ma in poche circostanze particolari, il metodo di costruzione del triangolo può essere più accurato per la misura dell'area rispetto al metodo Gaussiano se i picchi sono *asimmetrici* o di *forma incerta*. (Per alcuni esempi specifici, vedere la funzione demo [triangulationdemo.m](#): [cliccare per il grafico](#)).

Localizzare i transitori. La funzione [findsteps.m](#), sintassi: `P=findsteps(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, peakgroup)`, localizza i transitori positivi in serie temporali rumorose x-y, calcolando la derivata prima di y che



superà "SlopeThreshold", calcola l'altezza del transitorio [step] come la differenza tra il valore massimo e il minimo di y su un numero di punti pari a "Peakgroup" e restituisce la lista P col numero del transitorio [step], la posizione di x, quella di y, e l'altezza per ciascun rilevamento. "SlopeThreshold" e "AmpThreshold" regolano la sensibilità del passo; con valori alti si ignoreranno gradini bassi. L'aumento di "SmoothWidth" riduce i falsi transitori provocati dal rumore o da "glitch" (difetti) nei dati acquisiti. La figura sopra mostra un reale esempio di smoothing di dati sperimentali. Anche la funzione correlata [findstepsplot.m](#) disegna i dati e numera i picchi.

Gli impulsi rettangolari (onde quadre) richiedono un approccio diverso, basato sulla discriminazione dell'ampiezza piuttosto che sulla differenziazione. La funzione "[findsquarepulse.m](#)" (sintassi **S=findsquarepulse(t,y, threshold)**) individua gli impulsi rettangolari nel segnale t,y che in cui il valore di y supera quello di "threshold" determinandone il tempo iniziale, l'altezza media (relativamente alla linea di base) e la larghezza. [DemoFindsquare.m](#) crea un segnale di test (con un'altezza reale di 2636 un'altezza di 750) e chiama findsquarepulse.m per mostrarlo. Se il segnale è molto rumoroso, un po' di [smoothing](#) rettangolare preliminare (p.es. utilizzando [fastsmooth.m](#)) prima di chiamare findsquarepulse.m può aiutare ad eliminare i falsi picchi.

NumAT(m,threshold): "Numbers Above Threshold" (Numeri al di sopra della soglia): Conta il numero di elementi adiacenti nel vettore "m" che sono maggiori o uguali al valore scalare 'threshold'. Restituisce una matrice che elenca ogni gruppo di valori adiacenti, il loro indice iniziale, il numero di elementi in ogni gruppo, la somma di ogni gruppo e la media (mean) di ogni gruppo. Digitare "help NumAT" e provare l'esempio.

Utilizzo della tabella dei picchi

Tutte queste funzioni di ricerca dei picchi restituiscono una tabella come matrice, con una riga per ogni picco rilevato e con diverse colonne che elencano, ad esempio, il numero del picco, la posizione, l'altezza, la larghezza e l'area nelle colonne 1 - 5 (con colonne aggiuntive incluse per le varianti [measurepeaks.m](#), [findpeaksnr.m](#), [findpeaksGSS.m](#) e [findpeaksLSS.m](#)). È possibile assegnare questa matrice a una variabile (ad esempio **P**, negli esempi sopra) e poi utilizzare la notazione Matlab/Octave e le funzioni native per estrarne informazioni specifiche. *La potente combinazione di funzioni e la notazione "colon" (due punti ':') di Matlab permette di costruire espressioni compatte che estraggono le informazioni molto specifiche di cui necessita.* Ecco alcuni esempi:

[P(:,2) P(:,3)] è la serie temporale delle altezze (posizione del picco nella prima colonna e altezza nella seconda colonna).

mean(P(:,3)) restituisce l'altezza media di tutti i picchi (perché l'altezza è nella colonna 3). Questo funziona anche con "median".

max(P(:,3)) restituisce l'altezza massima di tutti i picchi. Anche questo funziona anche con **min**.

hist(P(:,3)) visualizza l'istogramma delle altezze dei picchi (utilizzando la funzione nativa "hist").

std(P(:,4)) ./mean(P(:,4)) restituisce la deviazione standard relativa delle larghezze dei picchi (colonna 4).

`P(:,3) ./max(P(:,3))` restituisce il rapporto tra l'altezza di ciascun picco (colonna 3) e l'altezza del picco più alto rilevato.

`100.*P(:,5) ./sum(P(:,5))` restituisce la percentuale di ciascuna area del picco (colonna 5) rispetto all'area totale di tutti i picchi rilevati.

`sortrows(P,2)` ordina **P** in base alla posizione del picco; `sort_rows(P,3)` ordina **P** in base all'altezza (dal piccolo al grande).

Per creare "d" come vettore delle differenze sull'asse x (posizione) tra i picchi adiacenti (perché la posizione del picco è nella colonna 2).

```
for n=1:length(P)-1;d(n)=max(P(n+1,2)-P(n,2));end
```

(In Matlab/Octave, è possibile inserire più istruzioni su una riga, separandole da punto e virgola).

La funzione val2ind. La semplice funzione scaricabile [val2ind.m](#) (sintassi `[index,closestval] = val2ind(v,val)`) restituisce l'indice e il valore dell'elemento del vettore 'v' che più si avvicina a 'val' (scaricare questa funzione e metterla nel path di ricerca di Matlab). Questa semplice funzione è utilissima quando si lavora con le tabelle dei picchi: `val2ind(P(:,3),7.5)` restituisce il numero del picco la cui altezza (colonna 3) è più vicina a 7.5.

`P(val2ind(P(:,2),7.5),3)` restituisce l'altezza del picco (colonna 3) la cui posizione (colonna 2) è più vicina a 7.5. `P(val2ind(P(:,3),max(P(:,3))),:)` restituisce il vettore riga dei parametri del picco più alto nella tabella dei picchi **P**. Le tre istruzioni `j=P(:,4)<5.8; k=val2ind(j,1); P(k,:)` restituisce la matrice dei parametri di tutti i picchi in **P** e cui semilarghezze (nella colonna 4) sono inferiori di un numero specifico (5,8 in questo caso). **Nota:** In Matlab e in Octave, è possibile inserire più istruzioni su una riga, separandole con un punto e virgola.

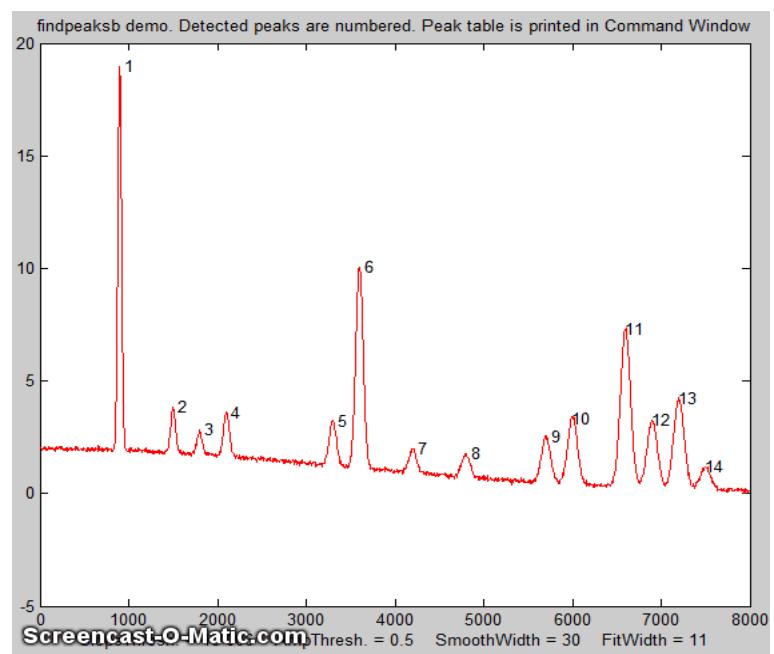
Individuare picchi nei dati multicolonna. Lo script [FindingPeaksInMultiColumnData.m](#) mostra come leggere un set di dati *multi-colonna* da un file Excel e rileva i picchi *in ciascuna colonna*, restituendo i dati dei picchi in una tabella di risultati a 3 dimensioni, **PP**. Funziona con qualsiasi delle funzioni di ricerca e/o approssimazione dei picchi di questo libro.

Script dimostrativi

[DemoFindPeak.m](#) è un semplice script dimostrativo che utilizza la funzione [findpeaksG](#) su dati sintetici rumorosi. La funzione numera i picchi e stampa la tabella dei picchi nella finestra di comando Matlab:

Peak #	Position	Height
Width	Area	
Measuredpeaks =		
1	799.95	6.9708
51.222	380.12	
2	1199.4	3.9168
50.44	210.32	
3	1600.6	2.9955
49.683	158.44	
4	1800.4	2.0039
50.779	108.33	
.....ecc.		

[DemoFindPeakSNR](#) è una variante di [DemoFindPeak.m](#) that uses [findpeaksnr.m](#)



per calcolare il rapporto segnale/rumore (SNR) di ciascun picco e lo restituisce nella 5^a colonna ([cliccare per un grafico](#)).

[DemoFindPeaksb.m](#) è uno script dimostrativo simile che utilizza la funzione [findpeaksb](#) su dati sintetici rumorosi costituiti da un numero variabile di picchi Gaussiani *sovraposti ad un background variabile curvo*. (La funzione findpeaksG non fornisce misure accurate di altezza, larghezza e area del picco per questo segnale, perché non corregge il background). [Cliccare per un'animazione](#).

Relative Percent Errors

```
Position      Height      Width      Area
-0.002246    0.54487    1.4057     1.9429
-0.02727     5.0091     8.9204     13.483
0.008429     -1.1224    -1.4923    -2.6315 ...etc.

% Root mean square errors
ans =
0.044428    2.2571     3.8253     5.850
```

Identificazione del Picco

La funzione a riga di comando [idpeaks.m](#) viene usata per identificare i picchi *in base alle posizioni dei loro massimi sull'asse x*, utile, ad esempio, quando l'identificazione di un picco dipende dalla sua posizione sull'asse x, in spettroscopia atomica e in cromatografia. La sintassi è

```
[IdentifiedPeaks, AllPeaks]=idpeaks(DataMatrix, AmpT, SlopeT, SmoothWidth,
FitWidth, maxerror, Positions, Names)
```

Cerca i picchi nel segnale "DataMatrix" (i valori x nella colonna 1 e quelli di y nella colonna 2), in base ai parametri di rilevamento "AmpT", "SlopeT", "SmoothWidth", "FitWidth" (vedere la funzione "findpeaksG" sopra), quindi confronta le posizioni dei picchi trovati (valori x) con un database di picchi noti, sotto forma di un array di posizioni note dei massimi dei picchi ('Positions') e li confronta con l'array di celle dei nomi ('Names'). Se la posizione di un picco trovato nel segnale è più vicina a uno dei picchi noti per meno dell'errore massimo specificato ('maxerror'), quel picco è considerato una corrispondenza e la sua posizione, nome, errore e ampiezza (height) vengono inseriti nella matrice di celle in output "IdentifiedPeaks". L'elenco completo dei picchi rilevati, identificati o meno, viene restituito in "AllPeaks". Utilizzare "cell2mat" per accedere agli elementi numerici di IdentifiedPeaks, p.es. `cell2mat(IdentifiedPeaks(2,1))` restituisce la posizione del primo picco identificato, `cell2mat(IdentifiedPeaks(2,2))` ne restituisce il nome, ecc. Ovviamente, per le proprie applicazioni, spetta all'utente fornire la matrice delle posizioni massime note ('Positions') e la corrispondente matrice di celle ('Names') per i propri tipi di segnali. La funzione correlata [idpeaktable.m](#) fa la stessa cosa per una tabella dei picchi P restituita da una qualsiasi delle funzioni di ricerca o approssimazione dei picchi, avendo una riga per ogni picco e le colonne per il numero, la posizione e l'altezza del picco come le prime tre colonne. La sintassi è `[IdentifiedPeaks] = idpeaktable(P, maxerror, Positions, Names)`. La funzione interattiva [iPeak](#) descritta nel seguito ha [questa funzione nativa](#) come uno dei comandi da tastiera (pagina 179).

Esempio: Scaricare [idpeaks.zip](#), scompattarlo e mettere i file estratti nel path di ricerca di Matlab o [Octave](#). Questo contiene uno spettro di emissione atomica ad alta risoluzione del rame ('spectrum', x = lunghezza d'onda in nanometri; y = ampiezza) e una tabella di righe atomiche Cu I e Cu II note ('DataTable') contenente le posizioni e i nomi di molte righe di rame. La funzione idpeaks rileva e misura le posizioni dei picchi di tutti i picchi in "spectrum", quindi cerca in 'DataTable' per vedere

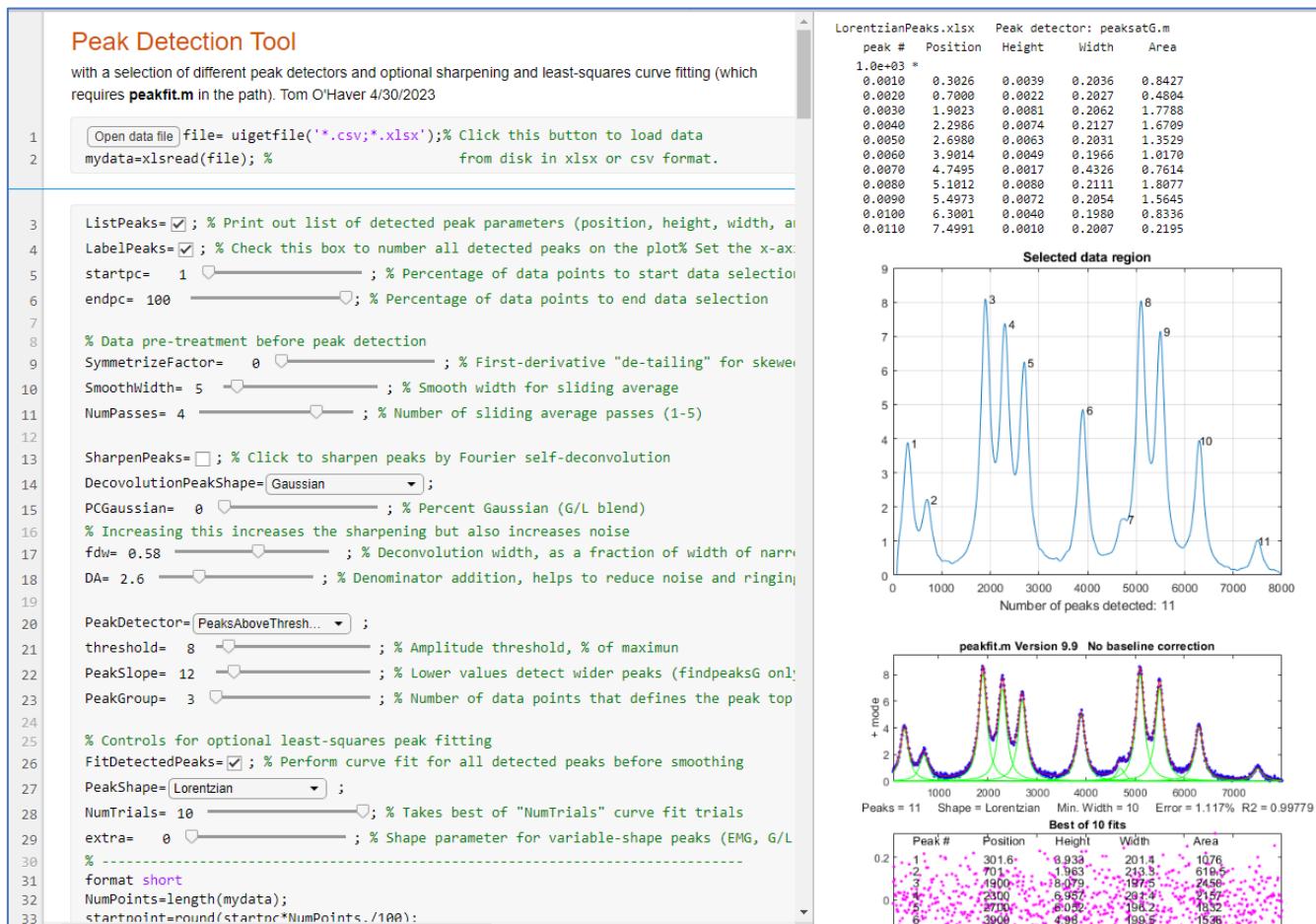
se qualcuno di quei picchi si trova entro 0,01 nm da qualsiasi voce nella tabella e stampa i picchi che corrispondono.

```
>> load DataTable
>> load spectrum
>> idpeaks(Cu,0.01,.001,5,5,.01,Positions,Names)

ans=
'Position'    'Name'          'Error'          'Amplitude'
[ 221.02]    'Cu II 221.027'  [-0.0025773]   [ 0.019536]
[ 221.46]    'Cu I 221.458'   [-0.0014301]   [ 0.4615]
[ 221.56]    'Cu I 221.565'   [-0.00093125]  [ 0.13191] ...
```

Tool Live Script per il Rilevamento del Picco

[PeakDetection.mlx](#) è un Live Script interattivo (pagina 360) per il rilevamento e la misura dei picchi. Raccoglie in uno strumento facile da usare molte delle funzioni relative ai picchi descritte in precedenza, inclusa una selezione di rilevatori di picco, smoothing dei dati, simmetrizzazione, sharpening dei picchi e approssimazione della curva, con cursori e menù a discesa per controllarli in modo interattivo.



Cliccando sul pulsante **OpenDataFile** nella riga 1 si apre un browser di file, che consente di navigare fino al file di dati (in formato .csv o .xlsx). Si inizia deselezionando la casella di controllo **FitDetectedPeaks** alla riga 26 per effettuare le regolazioni del rilevamento dei picchi più velocemente. Gli slider **startpc** e **endpc** nelle righe 5 e 6 permettono di impostare l'inizio e la fine della regione su cui concentrarsi (espressa come percentuale della lunghezza totale dei dati). È possibile impostare i controlli per lo smoothing dei dati (linee 10 e 11) o per il "de-tail" o simmetrizzare i picchi (linea 9). Si può scegliere un rilevatore di picco utilizzando il menù a discesa **PeakDetector** alla riga 20. Le caselle di controllo **ListPeaks** e **LabelPeaks** nelle righe 3 e 4

consentono di numerare i picchi sul grafico e/o di visualizzare un elenco di parametri dei picchi rilevati. Si può eventualmente provare lo [sharpening dei picchi](#), per abilitare il rilevamento del picco o delle spalle del lato debole, facendo clic sulla casella di controllo **SharpenPeaks** nella riga 13.

È possibile applicare facoltativamente l'[approssimazione dei minimi quadrati della curva](#), facendo clic sulla casella di controllo **FitDetectedPeaks** alla riga 26 e selezionando la forma della funzione di adattamento desiderata da **PeakShape** menù a discesa alla riga 27. La posizione e l'ampiezza dei picchi stimati dai rilevatori viene utilizzata come punto di partenza iniziale per l'approssimazione iterativa, quindi *solo i picchi rilevati verranno inclusi nell'approssimazione*. Questa funzione richiede che [peakfit.m](#) sia nel path di Matlab. (Normalmente, l'approssimazione della curva utilizza solo i dati senza smoothing; tuttavia, se viene applicato lo sharpening o la simmetrizzazione del picco nella riga 9 o 13, utilizza i dati elaborati). La funzione di ciascuno dei controlli è descritta nelle righe di commento associate.

Esempi della sua applicazione. Per mostrarne l'applicazione a picchi con forme e sovrapposizioni diverse, vedere il file PDF [PeakDetector.pdf](#), che fa riferimento a una serie di file di dati .csv scaricabili dallo stesso indirizzo. (Per vedere i grafici mostrati a lato dello script come sopra, cliccare con il tasto destro sul pannello di destra e selezionare "Disable synchronous scrolling").

iPeak: Rilevatore di picchi interattivo azionato tramite tastiera

iPeak ([ipeak.m](#) o [ipeakoctave.m](#)) è un rilevatore di picchi interattivo per dati di serie temporali, basato su "findpeaksG.m" e le funzioni "findpeaksL.m". L'interattività dei tasti funziona, sul proprio computer, anche se si esegue [Matlab in un browser web](#), ma non su [Matlab Mobile](#) né in Octave. Il suo funzionamento di base è simile a [iSignal](#) e [ipf.m](#). Accetta dati in un singolo vettore, una coppia di vettori o una matrice con la variabile indipendente nella prima colonna e quella dipendente nella seconda colonna. Se si chiama *iPeak* con *solo* uno o due argomenti di input, stima un valore iniziale di default per i parametri di rilevamento (AmpThreshold, SlopeThreshold, SmoothWidth e FitWidth) in base alle formule seguenti e visualizza tali valori nella parte inferiore dello schermo.

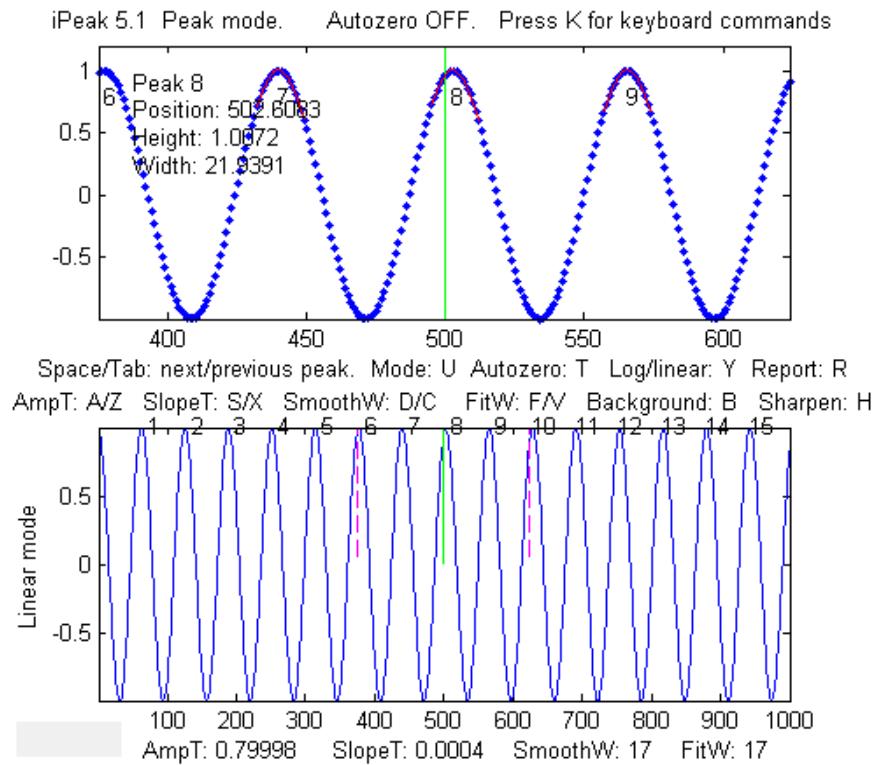
```
WidthPoints=length(y)/20;
SlopeThreshold=WidthPoints^-2;
AmpThreshold=abs(min(y)+0.1*(max(y)-min(y)));
SmoothWidth=round(WidthPoints/3);
FitWidth=round(WidthPoints/3);
```

È quindi possibile regolare con precisione il rilevamento del picco utilizzando le coppie di tasti su/giù adiacenti:

Amplitude threshold: **A/Z**
Slope threshold: **S/X**
Smooth width: **D/C**
Fit width: **F/V**.

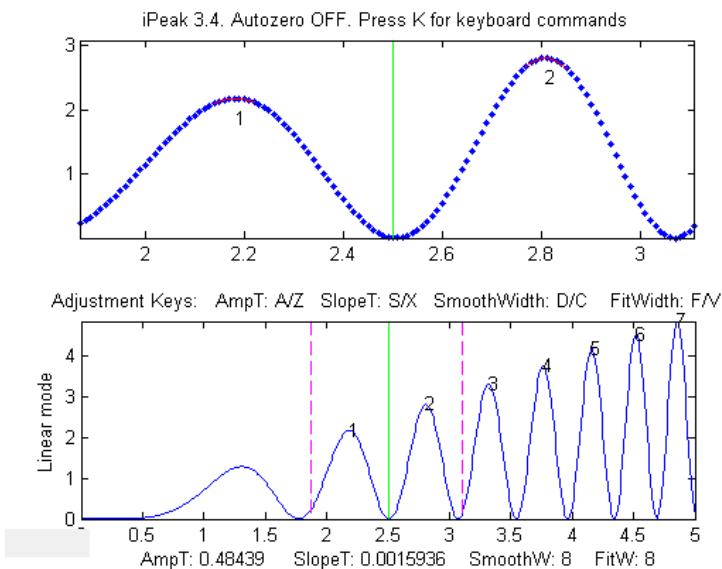
Esempio 1: Un argomento di input; dati in un unico vettore:

```
>> y=cos(.1:.1:100);
>> ipeak(y)
```



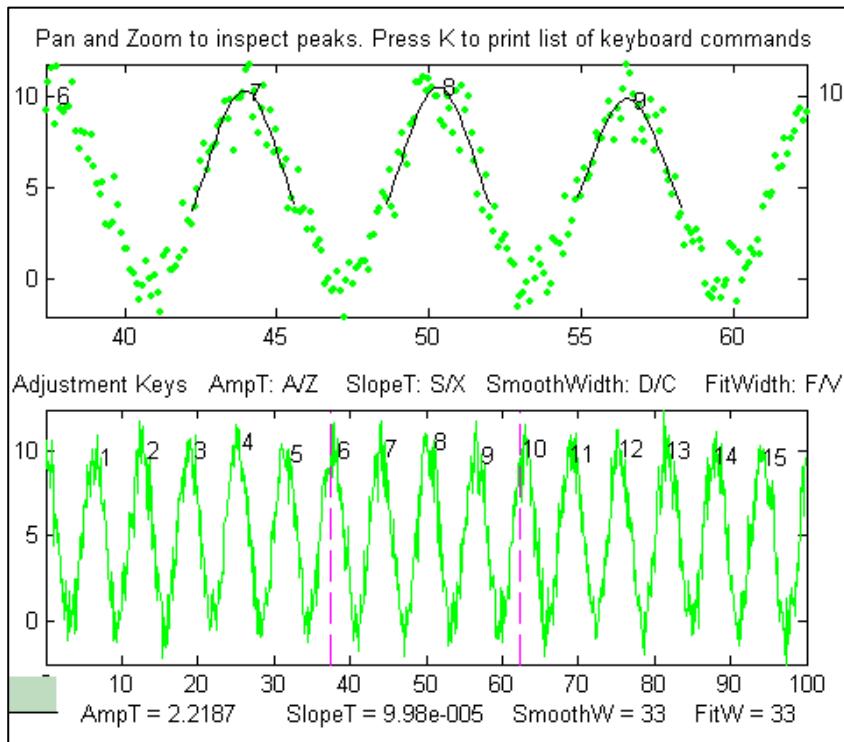
Esempio 2: Un argomento di input; dati in due colonne di una matrice:

```
>> x=[0:.01:5]';
>> y=x.*sin(x.^2).^2;M=[x y];
>> ipeak(M)
```



Esempio 3: Dati rumorosi. Due argomenti di input; i dati in vettori separati x e y:

```
>> x=[0:.1:100];
>> y=(x.*sin(x)).^2;
>> ipeak(x,y);
```



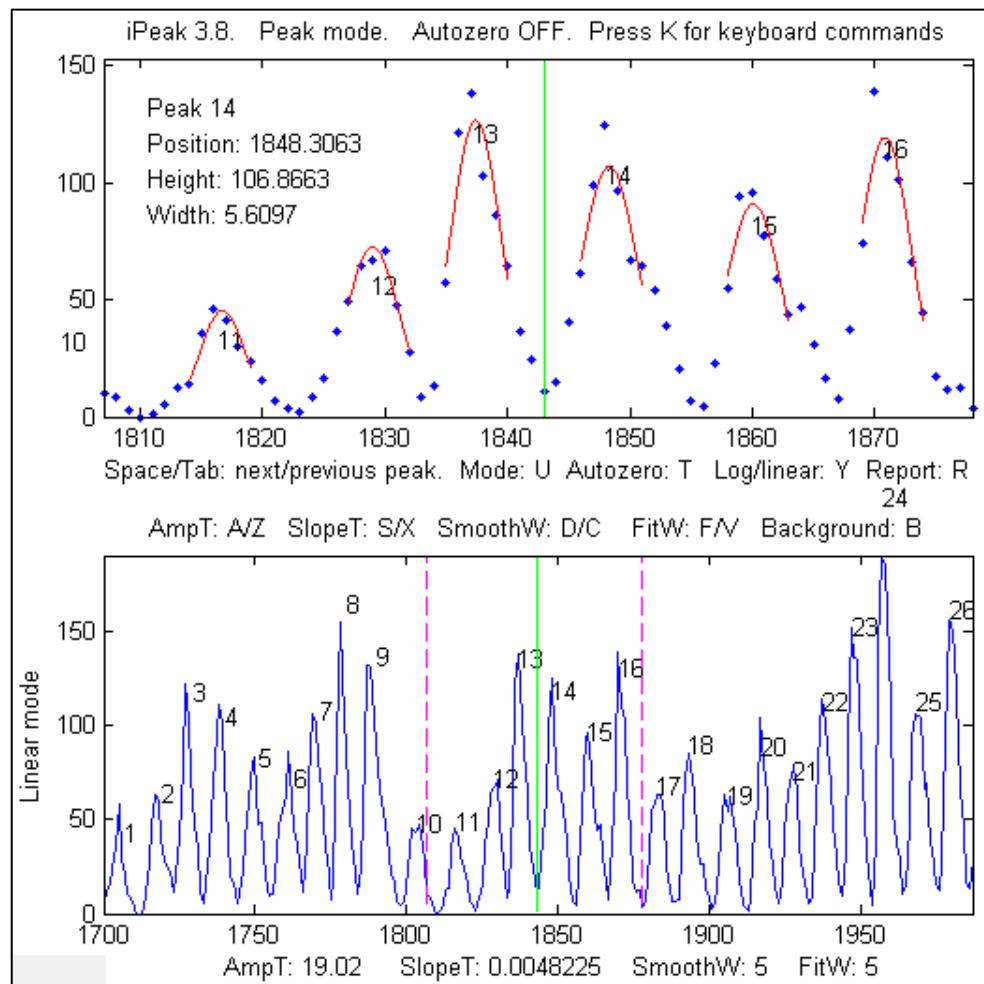
Doppio-click sulla barra del titolo della finestra Matlab per espanderla a schermo intero; di nuovo doppio-click per riportare la finestra alle dimensioni e alla posizione precedenti.

Esempio 4: Avviando *iPeak* utilizzando la semplice sintassi come illustrato sopra, i valori iniziali dei parametri di rilevamento vengono calcolati dal programma, ma se comincia a rilevare *troppi* o *troppo pochi* picchi, si può aggiungere un ulteriore argomento di input (dopo i dati) per controllare la *sensibilità del picco*.

```
>> x=[0:.1:100];y=5+5.*cos(x)+randn(size(x));ipeak(x,y,10);
o  >> ipeak([x;y],10);
o  >> ipeak(humps(0:.01:2),3)
o  >> x=[0:.1:10];y=exp(-(x-5).^2);ipeak([x' y'],1)
```

Questo ulteriore argomento numerico è una stima della *densità massima dei picchi* (PeakD), il rapporto tra la tipica larghezza del picco con la lunghezza di tutta la registrazione dei dati. Valori piccoli rilevano meno picchi; valori maggiori rilevano più picchi. Ha effetto solo sui valori *iniziali* per i parametri di rilevamento del picco. (È solo un modo rapido per impostare dei valori iniziali ragionevoli dei parametri di rilevamento, in questo modo non ci saranno molte regolazioni da fare; i parametri si possono comunque affinare singolarmente).

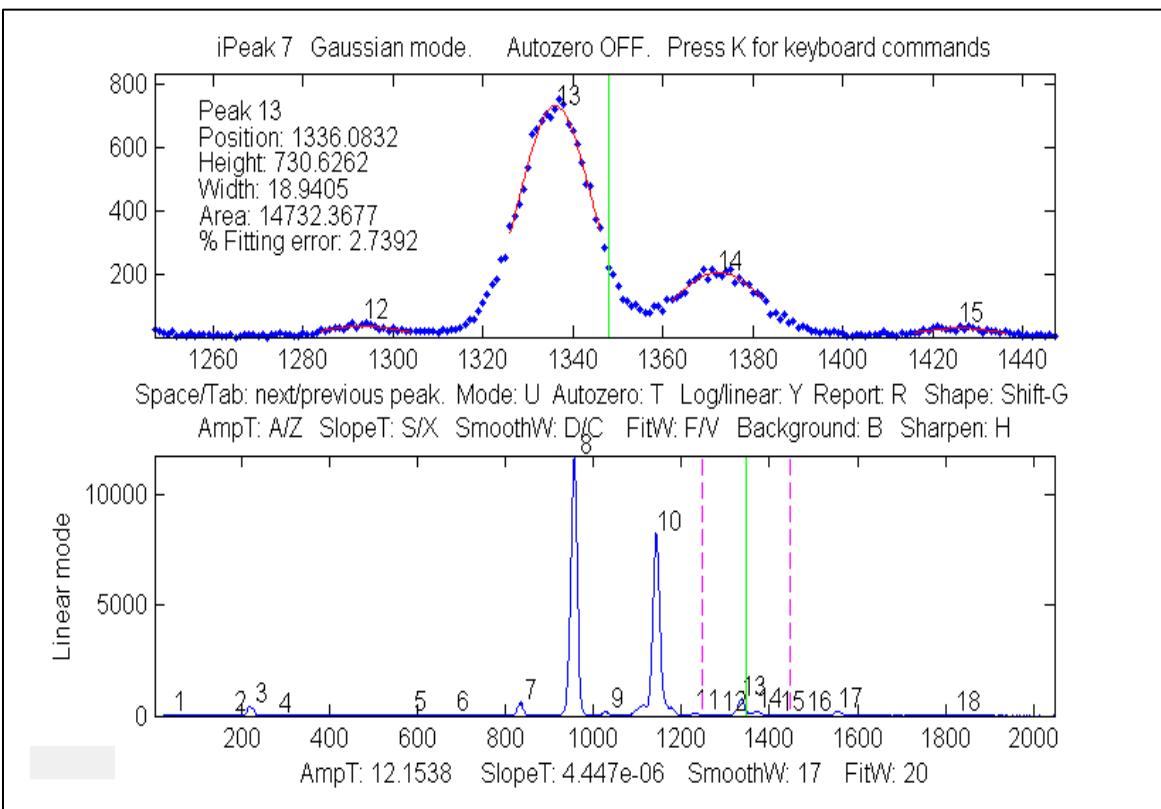
```
>> load sunspots
>> ipeak(year,number,20)
```



Picchi nel numero annuale di macchie solari dal 1700 al 2009 (scaricare il [file dei dati](#)).

Dati sulle macchie solari scaricati da [NOAA](#)

iPeak visualizza l'intero segnale nella metà inferiore della finestra e una sezione ingrandita regolabile nella finestra superiore. Eseguire il pan e lo zoom sulla porzione nella finestra in alto utilizzando i tasti freccia del cursore. Il picco più prossimo al centro della finestra superiore viene etichettato nella parte superiore sinistra e vengono elencate la posizione, l'altezza e la larghezza del picco. I tasti **Spazio/Tab** saltano al picco rilevato successivo/precedente e lo visualizzano nella finestra superiore con lo zoom corrente (utilizzare i tasti freccia cursore su e giù per regolare l'intervallo di zoom). Oppure si può premere il tasto **J** per saltare ad uno specifico numero di picco. Doppio-click sulla barra del titolo per espandere a tutto schermo per una visione ravvicinata.

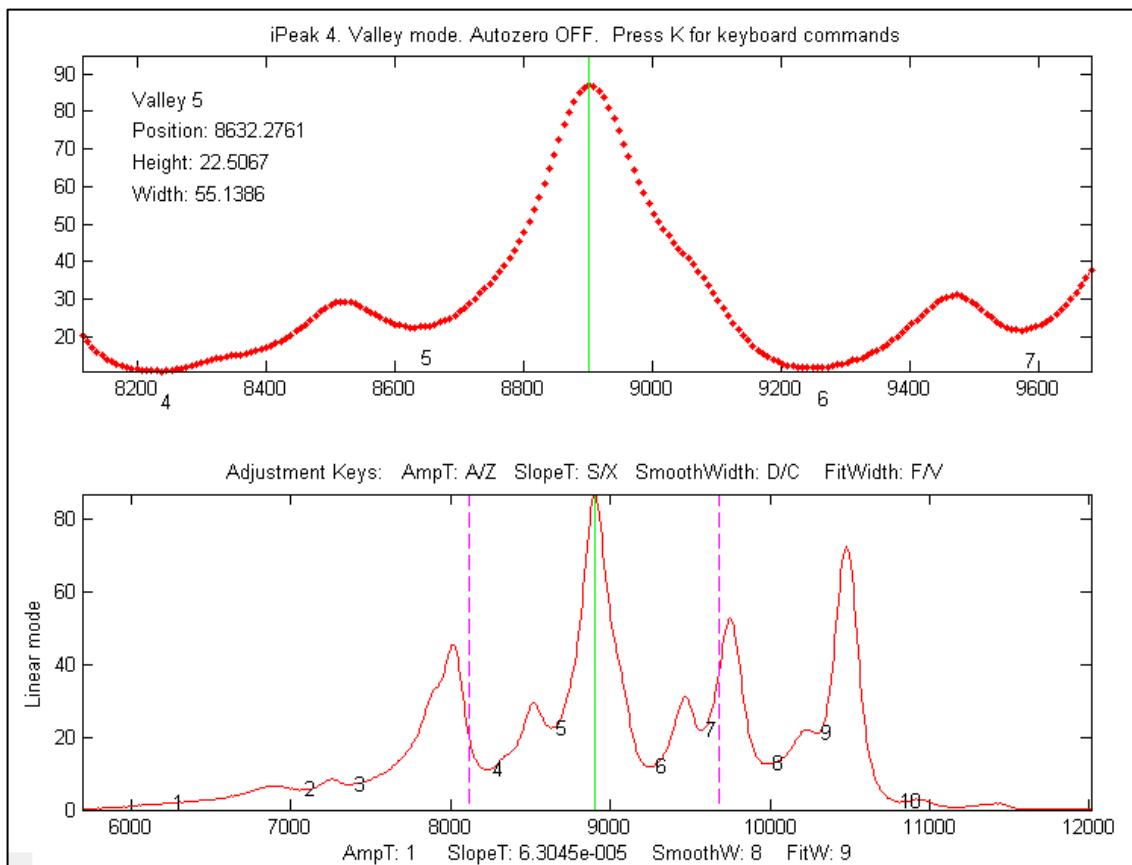


Regolare i parametri di rilevamento del picco AmpThreshold (tasti **A/Z**), SlopeThreshold (**S/X**), SmoothWidth (**D/C**), FitWidth (**F/V**) in modo da rilevare i picchi desiderati e ignorare quelli troppo piccoli, troppo ampi o troppo stretti per essere di interesse. È inoltre possibile digitare un valore specifico per AmpThreshold premendo **Shift-A** o per SlopeThreshold premendo **Shift-S**. I picchi rilevati vengono numerati da sinistra a destra.

Premere **P** per visualizzare la tabella di tutti i picchi rilevati (Peak #, Position, Height, Width, Area, e la percentuale di errore dell'approssimazione):

```
Gaussian shape mode (press Shift-G to change)
Window span: 169 units
Linear baseline subtraction
Peak# Position Height Width Area Error
1 500.93 6.0585 34.446 222.17 9.5731
2 767.75 1.8841 105.58 211.77 25.979
3 1012.8 0.20158 35.914 7.7 269.21
.....
```

Premere **Shift-G** per scorrere tra le modalità Gaussiana, Lorentziana, e "flat-top" (picco piatto). Premere **Shift-P** per salvare la tabella dei picchi in un file su disco. Premere **U** per passare dalla modalità picco a quella avvallamento. Da non dimenticare che solo gli avvallamenti *superiori a* (cioè più positive o meno negative di) AmpThreshold vengono rilevati; se si desidera rilevare avvallamenti che hanno minimi negativi, allora AmpThreshold deve essere impostato su un valore più negativo. Nota: per velocizzare l'operazione per segnali di lunghezza superiore a 100.000 punti, la finestra inferiore viene aggiornata solo quando cambia il numero dei picchi rilevati o se viene premuto il tasto **Enter**. Premere **K** per vedere tutti i comandi da tastiera.



La modalità Avvallamento. Premere il tasto U per passare dalla modalità picco a quella avvallamento.

Se la densità dei punti dati sui picchi è *tropppo bassa* - meno di 4 punti circa - i picchi potrebbero non essere rilevati in modo affidabile; è possibile migliorare utilizzando il comando di interpolazione (**Shift-I**) per ri-campionare i dati con un'interpolazione lineare ottenendo un numero *maggior*e di punti. Al contrario, se la densità dei punti sui picchi di interesse è molto alta, ad esempio più di 100 punti per picco, è possibile accelerare le operazioni di *iPeak* ri-campionando con un numero *inferiore* di punti.

Statistiche Riassuntive in iPeak. Il tasto **E** stampa una tabella con le statistiche riassuntive degli intervalli (l'intervallo sull'asse x tra picchi adiacenti rilevati), altezze, larghezze e aree dei picchi, elencando la deviazione standard massima, minima, media e percentuale, e visualizza gli *istogrammi* degli intervalli, delle altezze, delle larghezze e delle aree dei picchi nella [Window 2](#).

Peak Summary Statistics

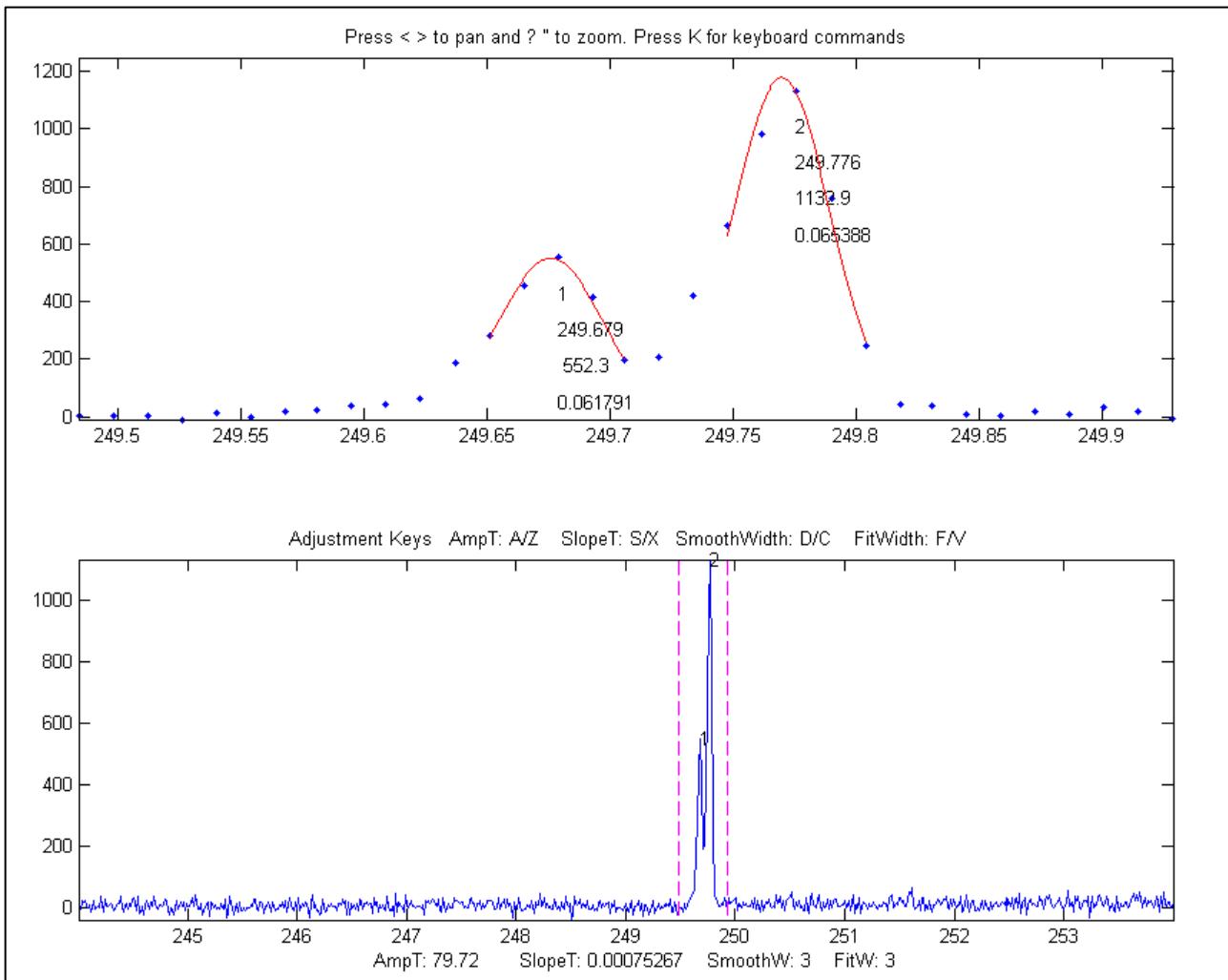
149 peaks detected

No baseline correction

Interval	Height	Width	Area
Maximum	1.3204	232.7724	0.33408 80.7861
Minimum	1.1225	208.0581	0.27146 61.6991
Mean	1.2111	223.3685	0.31313 74.4764
% STD	2.8931	1.9115	3.0915 4.0858

Esempio 5: Sei argomenti di input. Come sopra, ma gli argomenti di input dal 3° al 6° specificano direttamente i valori iniziali di AmpThreshold (AmpT), SlopeThreshold (SlopeT), SmoothWidth (SmoothW), FitWidth (FitW). PeakD viene, in questo caso, ignorato, quindi basta digitare uno '0' come secondo argomento dopo la matrice).

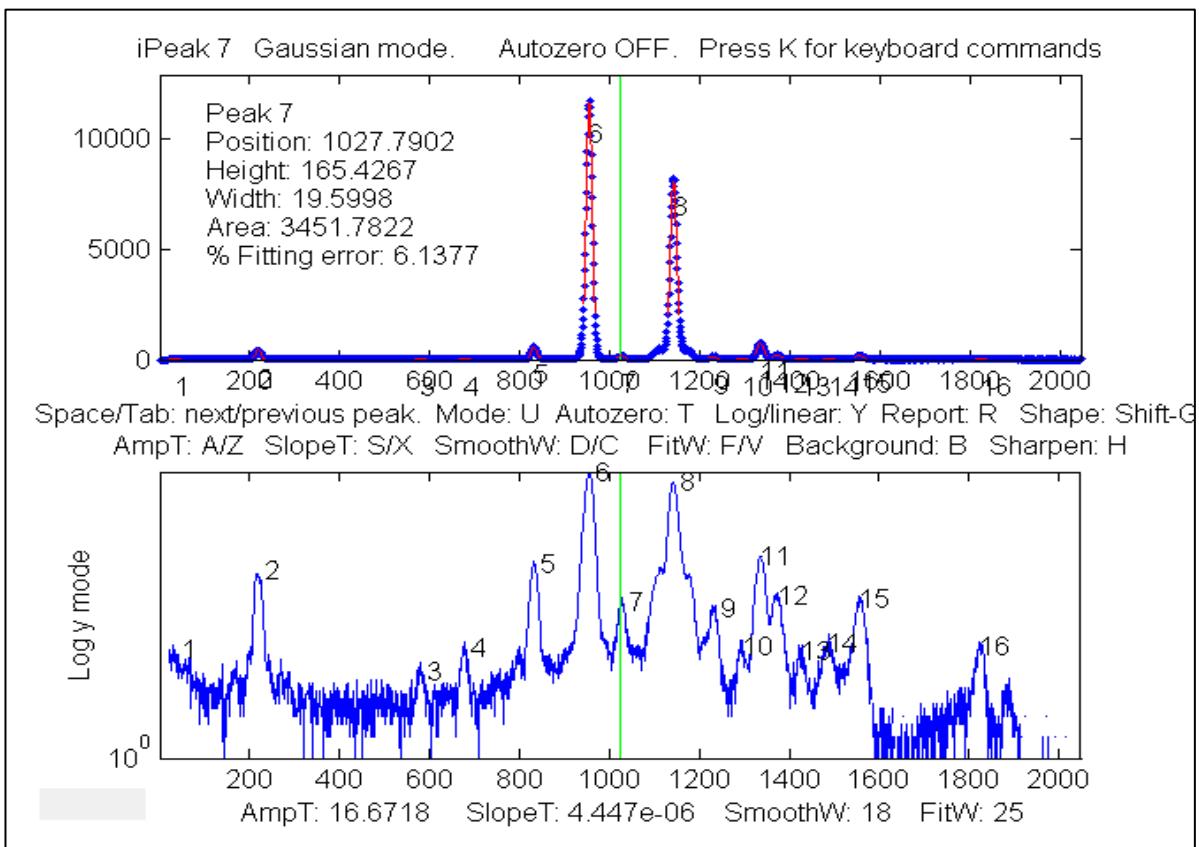
```
>> ipeak(datamatrix, 0, .5, .0001, 20, 20);
```



Premendo 'L' si attivano/disattivano le etichette dei picchi nella finestra superiore.

Dei tasti consentono di eseguire il pan e lo zoom nella finestra superiore, per ispezionare ogni picco in dettaglio se lo si desidera. È possibile impostare i valori iniziali di pan e zoom negli argomenti di input opzionali 7 ('xcenter') e 8 ('xrange'). Cfr. l'esempio 6 di seguito.

Il tasto **Y** commuta tra la scala lineare e quella logaritmica dell'asse y nella *finestra inferiore* (un asse logaritmico è utile per ispezionare i segnali con un elevato intervallo dinamico). Riguarda solo la visualizzazione nella finestra inferiore e *non ha effetti sui dati* stessi né sul rilevamento né sulle misure dei picchi.



La scala logaritmica (tasto Y) facilita la visualizzazione dei picchi più piccoli nella finestra inferiore.

Esempio 6: Otto argomenti di input. Come sopra, ma gli argomenti di input 7 e 8 specificano le impostazioni iniziali di pan e zoom, 'xcenter' e 'xrange', rispettivamente. In questo esempio, i dati sull'asse x sono lunghezze d'onda in nanometri (nm) e la finestra superiore ingrandisce una regione molto piccola di 0.4 nm centrata su 249.7 nm. (Questi dati, forniti nel [file ZIP](#), provengono da uno spettro atomico ad alta risoluzione).

```
>> load ipeakdata.mat
>> ipeak(Sample1,0,100,0.05,3,4,249.7,0.4);
```

Modalità di correzione della linea di base. Il tasto T scorre le *modalità di correzione della linea di base* da *off*, *lineare*, *quadratica*, *flat*, *linear mode(y)*, *flat mode(y)* e poi ripete da *off*. La modalità corrente viene visualizzata sopra il pannello superiore. Quando la correzione della linea di base è disattivata (OFF), le altezze dei picchi vengono misurate rispetto allo zero. (Utilizzare questa modalità quando la linea di base è zero o se in precedenza è stata sottratta la linea di base dall'intero segnale utilizzando il tasto **B** key). Nelle modalità *lineare* o nella *quadratica*, le altezze dei picchi vengono misurate automaticamente rispetto alla linea di base locale interpolata dai punti alle estremità del segmento visualizzato nel pannello superiore; utilizzare i controlli di zoom per isolare un gruppo di picchi in modo che il segnale ritorni alla linea di base locale all'inizio e alla fine del segmento visualizzato nella finestra superiore. Le altezze, le larghezze e le aree dei picchi nella tabella dei picchi (tasti **R** o **P**) verranno automaticamente corrette per la linea di base. Le modalità *lineare* o *quadratica* funzioneranno meglio se i picchi sono ben separati in modo che il segnale ritorni alla linea di base locale tra i picchi. (Se i picchi sono molto sovrapposti, o se non sono di forma Gaussiana, i migliori risultati si ottengono utilizzando la funzione di approssimazione della curva - i tasti **N** e **M**. La modalità *piatta* viene utilizzata solo per la funzione di approssimazione della curva, per tenere conto di un offset piatto della linea di base senza riferimento ai bordi del segmento del segnale da approssimare). Il metodo *mode(y)* sottrae il *valore più comune di y* da tutti i punti nella regione selezionata. Per i segnali in cui i picchi di solito ritornano sulla linea di base tra

i picchi, questa è solitamente la linea di base anche se il segnale non ritorna alla linea di base alle estremità come per le modalità 2 e 3 ([esempio grafico](#)).

Esempio 7: Nove argomenti di input. Come l'esempio 6, ma il 9º argomento di input imposta la modalità di correzione del background (equivalente alla pressione del tasto **T**)' 0=OFF; 1=lineare; 2=quadratico, 3=piatto, 4=mode(y). Se non specificato, è inizialmente OFF.

```
>> ipeak(Sample1,0,100,0.00,3,4,249.7,0.4,1);
```

Conversione in funzioni a riga di comando. Per aiutare a scrivere i propri script e funzioni per automatizzare l'elaborazione eseguita con iPeak, il tasto '**Q**' stampa i comandi `findpeaksG`, `findpeaksB` e `findpeaksfit` per il segmento di segnale nella finestra superiore e per l'intero segnale, con gli argomenti di input in posizione, copiabili e incollabili negli script. Il tasto '**W**', allo stesso modo, stampa i comandi `peakfit.m` e `ipf.m`.

Shift-Ctrl-S trasferisce il segnale corrente a `iSignal.m` (pag. 366) e **Shift-Ctrl-P** trasferisce il segnale corrente al **Interattivo Peak detector (iPeak.m)**, se tali funzioni sono installate nel percorso Matlab.

Media dell'insieme in iPeak. Per i segnali che contengono modelli di forma d'onda ripetitivi che ricorrono in un segnale continuo, con lo stesso tipo di forma tranne per il rumore, la funzione di media dell'insieme (**Shift-E**) può calcolare la media di tutte le forme d'onda ripetute. Funziona rilevando un singolo picco di riferimento in ciascuna forma d'onda per sincronizzare le ripetizioni (e quindi non richiede che le ripetizioni siano equamente distanziate o sincronizzate con un segnale di riferimento esterno). Per utilizzare questa funzione, prima si regolano i controlli di rilevamento del picco per rilevare *solo un picco in ogni schema di ripetizione*, utilizzare lo zoom per isolare uno qualsiasi di questi pattern di ripetizione ed infine premere **Shift-E**. La forma d'onda media viene visualizzata nella Figura 2 e salvata come "EnsembleAverage.mat" nella directory corrente. Vedere lo script [iPeakEnsembleAverageDemo.m](#).

Troncare i picchi con code esponenziali in iPeak. Se il segnale ha picchi che si prolungano a sinistra o a destra perché sono stati ampliati esponenzialmente, si possono rimuovere le code con la tecnica dell'addizione della derivata prima (pagina 77): premere **Shift-Y**, immettere una stima della costante di tempo esponenziale e quindi utilizzare i tasti **1** e **2** per regolarla del 10% ad ogni pressione del tasto (o **Shift-1** e **Shift-2** per regolarla dell'1% per ogni battuta). [Cliccare per l'animazione](#). Aumentare il fattore fino a quando la linea di base dopo il picco diventa negativa, quindi aumentarlo leggermente in modo che sia *il più basso possibile ma non negativo*. Ciò si traduce in picchi più stretti e più alti, ma non ha alcun effetto sulle loro aree. L'effetto è come la deconvoluzione della funzione esponenziale dal picco allargato, ma è più veloce e più semplice. (Se la coda (il prolungamento) dei picchi è a *sinistra* anziché a destra, usare un fattore *negativo*).

Approssimazione Normale e Multipla dei picchi in iPeak: Il tasto **N** applica [l'approssimazione iterativa della curva ai picchi rilevati e visualizzati nella finestra superiore](#) (qui denominato come approssimazione "Normale"). L'uso della funzione iterativa dei minimi quadrati può produrre misure dei parametri più accurate rispetto alla normale tabella dei picchi (tasti **R** o **P**), specialmente se i picchi sono di forma non Gaussiana o sono molto sovrapposti. (Se i picchi sono sovrapposti ad un background, selezionare prima la modalità di **correzione della linea di base** utilizzando il tasto **T**, poi utilizzare i tasti pan e zoom per selezionare un picco o un gruppo di picchi sovrapposti nella finestra superiore, con il segnale che scende completamente sulla linea di base locale alle estremità della finestra superiore se si utilizzano le modalità di linea di base lineare o quadratica; vedere pagina 211). Assicurarsi che AmpThreshold, Slope-Threshold, SmoothWidth siano regolati in modo che ogni picco sia numerato una volta. Vengono approssimati solo i picchi numerati. Poi si preme il tasto **N**, che visualizzerà questo **menù dei profili dei picchi** (grafico a pag. 411):

Gaussians: $y=\exp(-((x-pos) / (0.6005615.*width)) .^2)$	1
Gaussians with independent positions and widths.....	1
(default)	
Exponentially--broadened Gaussian (equal time constants).....	5
Exponentially--broadened equal-width Gaussian.....	8
Fixed-width exponentially-broadened Gaussian.....	36
Exponentially--broadened Gaussian (independent time constants)....	31
Gaussians with the same widths.....	6
Gaussians with preset fixed widths.....	11
Fixed-position Gaussians.....	16
Asymmetrical Gaussians with unequal half-widths on both sides....	14
Lorentzians: $y=ones(size(x))./(1+((x-pos) / (0.5.*width)) .^2)$	2
Lorentzians with independent positions and widths.....	2
Exponentially--broadened Lorentzian.....	18
Equal-width Lorentzians.....	7
Fixed-width Lorentzian.....	12
Fixed-position Lorentzian.....	17
Gaussian/Lorentzian blend (equal blends).....	13
Fixed-width Gaussian/Lorentzian blend.....	35
Gaussian/Lorentzian blend with independent blends).....	33
Voigt profile with equal alphas.....	20
Fixed-width Voigt profile with equal alphas.....	34
Voigt profile with independent alphas.....	30
Logistic: $n=\exp(-((x-pos) / (.477.*wid)) .^2); y=(2.*n) / (1+n)$	3
Pearson: $y=ones(size(x))./(1+((x-pos) / ((0.5.^2/m).*wid)) .^2) .^m$..	4
Fixed-width Pearson.....	37
Pearson with independent shape factors, m.....	32
Breit-Wigner-Fano.....	15
Exponential pulse: $y=(x-tau2) ./tau1.*exp(1-(x-tau2) ./tau1)$	9
Alpha function: $y=(x-spoint) ./pos.*exp(1-(x-spoint) ./pos)$;.....	19
Up Sigmoid (logistic function): $y=.5+.5*erf((x-tau1) /sqrt(2*tau2))$.10	
Down Sigmoid $y=.5-.5*erf((x-tau1) /sqrt(2*tau2))$	23
Triangular.....	21

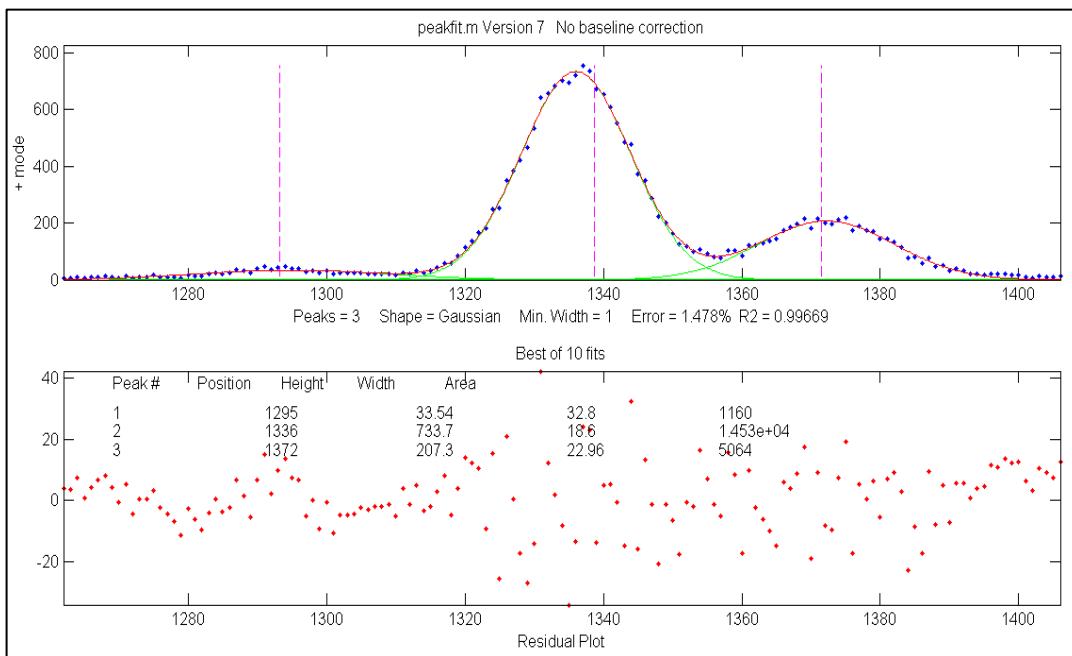
Digitare il numero per il profilo del picco desiderata da questa tabella e premere **Enter**, quindi digitare un numero di tentativi di approssimazioni e premere **Enter** (il default è 1; iniziare con quello e poi aumentare se necessario). Se è stato selezionato un picco di forma variabile (ad esempio i numeri 4, 5, 8, 13, 14, 15, 18, 20, 30-33), il programma chiederà di digitare un numero che ottimizzi la forma. Il programma eseguirà quindi l'approssimazione, visualizzerà i risultati graficamente nella finestra 2 e stamperà una tabella dei risultati nella finestra di comando, ad esempio:

```

Peak shape (1-8): 2
Number of trials: 1

Least-squares fit to Lorentzian peak model
Fitting Error 1.1581e-006%
Peak# Position Height Width Area
1 100 1 50 71.652
2 350 1 100 146.13
3 700 1 200 267.77

```

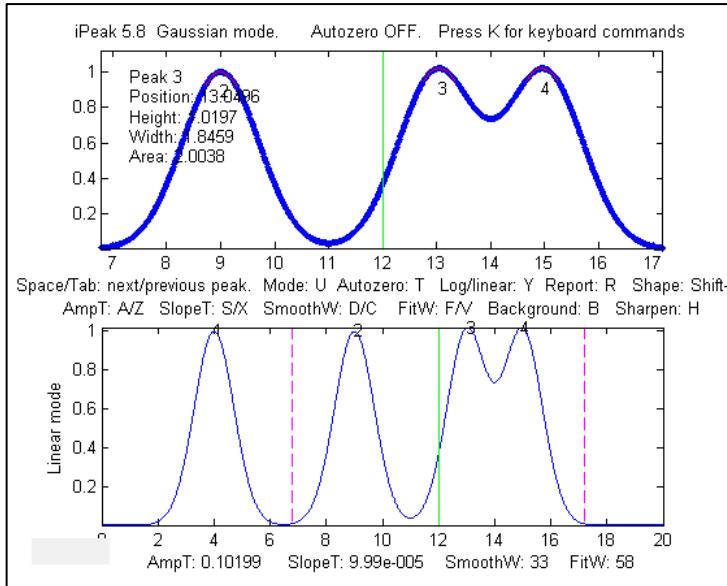


L'Approssimazione Normale del Picco (tasto N) applicata ad un gruppo di tre picchi Gaussiani sovrapposti

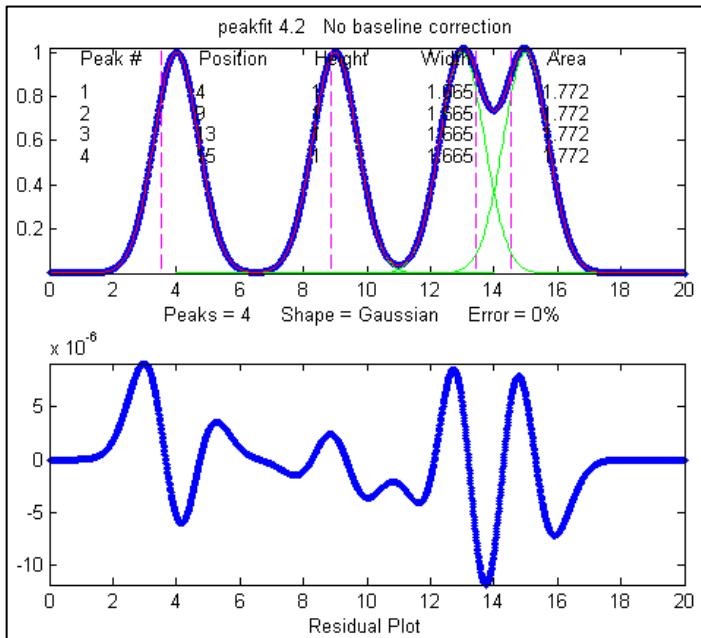
C'è anche una funzione di approssimazione a picchi "Multipli" (tasto **M**) che tenterà di applicare l'approssimazione iterativa della curva a **tutti i picchi rilevati simultaneamente**. Prima di utilizzare questa funzione, è meglio disattivare (**off**) la correzione automatica della linea di base (tasto **T**) ed usare la *funzione multi-segmento di correzione della linea di base* (tasto **B**) per rimuovere il background (perché la funzione di correzione della linea di base probabilmente non sarà in grado di sottrarre la linea di base dall'intero segnale). Poi premere **M** e procedere come per la normale approssimazione. Un'approssimazione multipla può richiedere circa un minuto per essere completata se il numero di picchi è elevato, forse più tempo di quello richiesto dalla funzione di approssimazione normale su ciascun gruppo di picchi separatamente.

I tasti **N** e **M** della funzione di approssimazione eseguono un'[approssimazione iterativa non-lineare](#) utilizzando la funzione [peakfit.m](#). Il numero di picchi e i valori iniziali delle posizioni e delle larghezze dei picchi per la funzione di approssimazione vengono forniti automaticamente dalla funzione `findpeaksG`, quindi è essenziale che le variabili di rilevamento dei picchi in *iPeak* siano regolate in modo che tutti i picchi nella regione selezionata vengano rilevati e numerati una volta. (Per un'approssimazione della curva più flessibile, utilizzare [ipf.m](#), pagina 405, che consente l'ottimizzazione manuale dei raggruppamenti dei picchi e delle posizioni iniziali).

Esempio 8. Questo esempio genera quattro picchi Gaussiani, tutti con la stessa identica altezza (1.00) e area (1.773). Il primo picco (in $x=4$) è isolato, il secondo ($x=9$) è leggermente sovrapposto al terzo, e gli ultimi due (in $x=13$ e 15) sono molto accavallati.



```
x=[0:.01:20];
y=exp(-(x-4).^2)+exp(-(x-9).^2)+exp(-(x-13).^2)+exp(-(x-15).^2);
```



ipeak(x,y)

Di per sé, *iPeak* fa un buon lavoro nel misurare le posizioni e le altezze dei picchi approssimandone solo la parte superiore, perché in questo esempio i picchi sono Gaussiani. Tuttavia, le aree e le larghezze degli ultimi due picchi (che dovrebbero essere 1.665 come gli altri) sono un po' troppo grandi a causa della sovrapposizione:

Peak#	Position	Height	Width	Area
1	4	1.6651	1.7727	
2	9	1.6651	1.7727	
3	13.049	1.02	1.8381	1.9956
4	14.951	1.02	1.8381	1.9956

In questo caso, l'approssimazione della curva (usando i tasti **N** o **M**) fa un lavoro molto migliore, anche se la sovrapposizione è ancora maggiore, ma *solo se il profilo del picco è noto*:

Peak#	Position	Height	Width	Area
1	4	1.6651	1.7724	
2	9	1.6651	1.7725	
3	13	1.6651	1.7725	
4	15	0.99999	1.6651	1.7724

Nota 1: Se i picchi sono troppo sovrapposti per essere rilevati e numerati separatamente, provare a premere il tasto **H** per attivare la funzione di sharpening con le derivate pari prima di premere **M** (solo dalla versione 4.0 in poi). Questo ha effetto solo sul rilevamento del picco, non sul segnale stesso.

Nota 2: Se si prevede di utilizzare un picco di forma variabile (numeri 4, 5, 8, 13, 14, 15, 18, o 20) per l'approssimazione Multipla, è una buona idea ottenere un valore ragionevole per il parametro "extra" del profilo eseguendo un'approssimazione Normale su un singolo picco isolato (o un piccolo gruppo di picchi parzialmente sovrapposti) con lo stesso profilo, poi usare il valore per l'approssimazione Multipla sull'intero segnale.

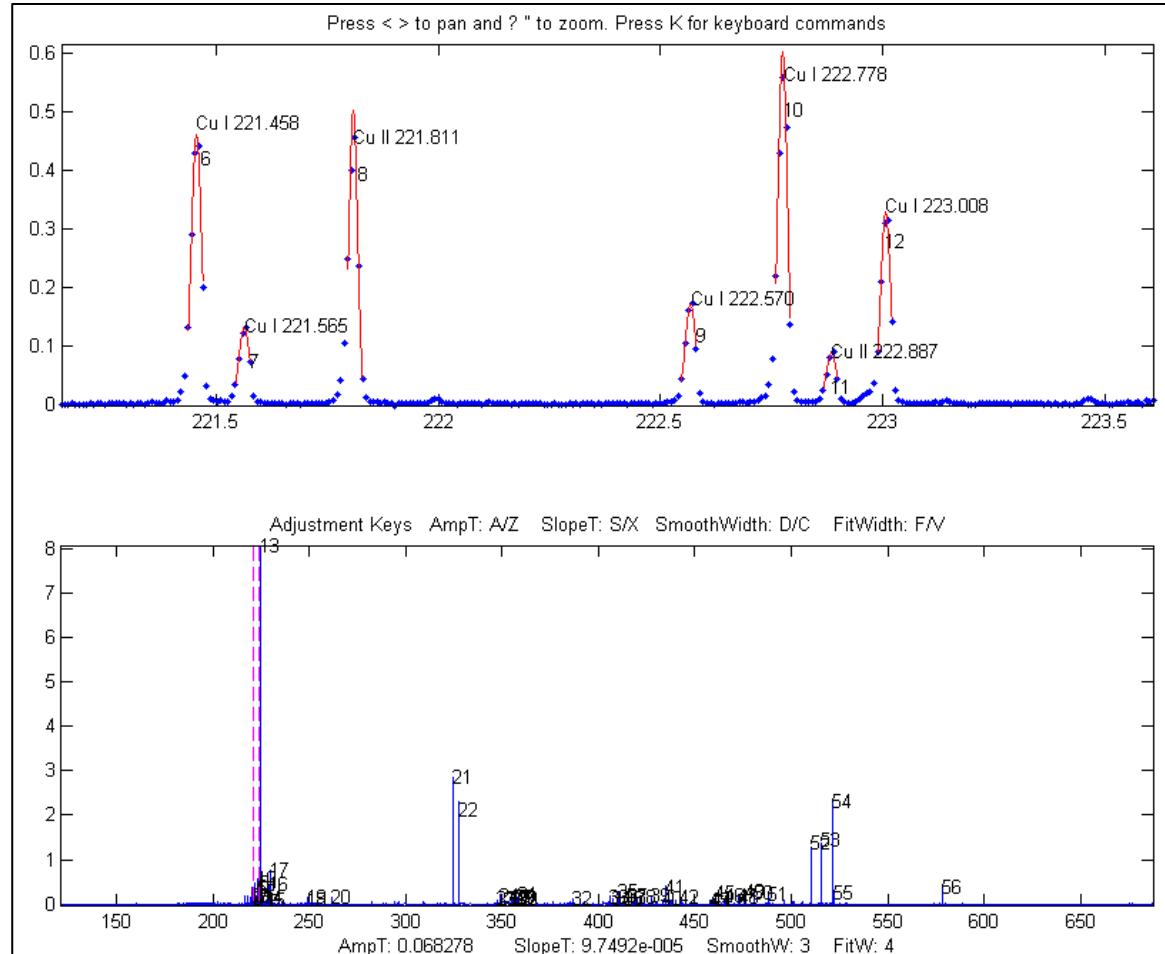
Nota 3: Se il profilo varia lungo il segnale, è possibile utilizzare o l'approssimazione Normale per ciascuna sezione con forme diverse anziché la Multipla, oppure possono usare i profili *svincolati* che approssimano individualmente la forma per ogni picco: Voigt (30), ExpGaussian (31), Pearson (32), o Gaussian/Lorentzian blend (33).

Identificazione del Picco in *iPeak*. C'è una funzione di "identificazione del picco" se

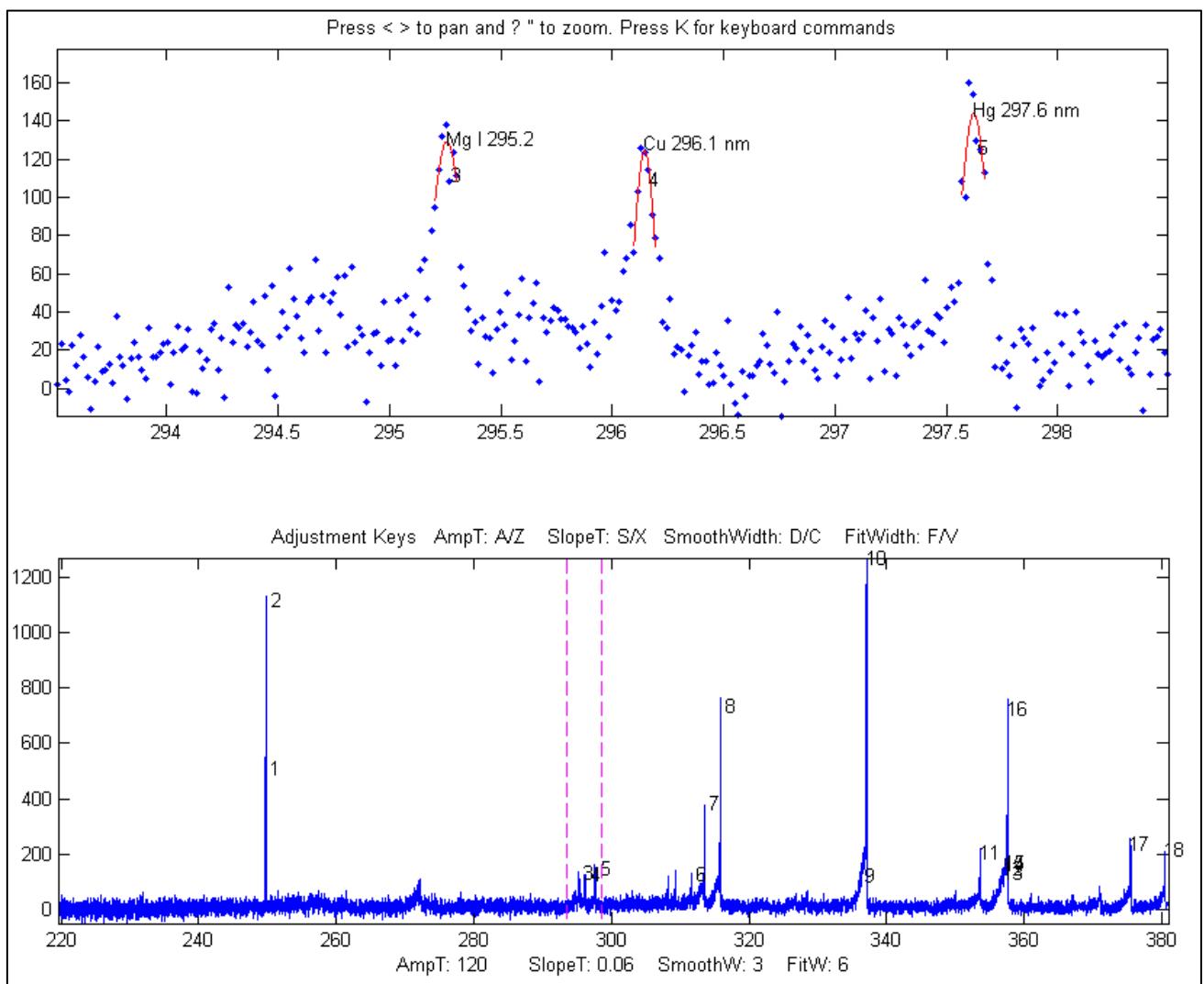
vengono inclusi gli argomenti opzionali 9 ('MaxError'), 10('Positions') e 11 ('Names'). Il tasto "I" la attiva o disattiva. Questa funzione confronta le posizioni dei picchi rilevate (valori x massimi) facendo riferimento a un *database di picchi noti*, fornito negli argomenti di input 10 e 11 sotto forma di array delle posizioni massime note dei picchi ('Positions') e l'array delle celle corrispondenti di nomi ("Names"). Se la posizione di un picco trovato nel segnale è più vicina a uno dei picchi per meno dell'errore massimo specificato ('MaxError'), allora quel picco è considerato una corrispondenza e il suo nome viene visualizzato accanto al picco in alto finestra. Quando si preme il tasto 'O' (la lettera 'O'), le posizioni, i nomi, gli errori e le ampiezze dei picchi vengono stampate nella finestra di comando in una forma tabellare.

Esempio 9: Applicazione agli spettri atomici. Undici argomenti di input. Come sopra, ma specifica anche 'MaxError', 'Positions' e 'Names' negli argomenti di input opzionali 9, 10 e 11, per la funzione di identificazione. Premendo il tasto 'T' si disattivano e si attivano le etichette di identificazione del picco nella finestra superiore. Questi dati (forniti in questo [file ZIP](#)) provengono da uno spettro atomico ad alta risoluzione).

```
>> load ipeakdata.mat
>> ipeak(Sample1,0,100,0.05,3,6,296,5,0.1,Positions,Names);
```



La funzione di identificazione del picco applicata a uno spettro di emissione atomica ad alta risoluzione del rame.



Uno spettro di emissione atomica di un campione contenente tracce di diversi elementi. iPeak viene utilizzato per ingrandire tre piccoli picchi vicino a 296 nm. iPeak ha identificato ed etichettato tre picchi in base ai dati della riga atomica in `ipeakdata.mat`. Premere il tasto **I** per visualizzare i nomi degli ID dei picchi. Doppio-click sulla barra del titolo della finestra per vedere meglio a tutto schermo.

La pressione di "O" stampa le posizioni, i nomi, gli errori e le ampiezze dei picchi nella finestra di comando in una forma tabellare.

Name	Position	Error	Amplitude
'Mg I 295.2'	[295.2]	[0.058545]	[129.27]
'Cu 296.1'	[296.1]	[0.045368]	[124.6]
'Hg 297.6'	[297.6]	[0.023142]	[143.95]

Ecco un altro esempio, da un ampio spettro di emissione atomica con oltre 10.000 punti dati e molte centinaia di picchi. La tabella di riferimento dei picchi noti in questo caso è presa dalla Tabella 1 di [ASTM C1301 - 95\(2009\)e1](#). Con le impostazioni che si stavano usando, sono stati identificati dieci picchi, mostrati nella tabella sottostante. Si può vedere che alcuni di questi elementi hanno più di una riga identificata. Ovviamente, più basse sono le impostazioni di AmpThreshold, SlopeThreshold e SmoothWidth, più picchi verranno rilevati; e maggiore è "MaxError", più picchi saranno abbastanza vicini da essere identificati. In questo esempio, i nomi degli elementi nella tabella seguente vengono collegati al volo all'immagine dello schermo del picco corrispondente rilevato e identificato da iPeak. Alcune di queste righe, in particolare Nickel 231.66nm, Silicio 288.18nm e Ferro 260.1nm, sono piuttosto deboli e quindi hanno rapporti segnale/rumore scadenti, quindi le loro identificazioni potrebbero essere dubbie (specialmente il Ferro poiché il suo errore di lunghezza d'onda è maggiore del resto). Spetta a ciascuno decidere quali picchi sono abbastanza

forti da essere significativi. In questo esempio, è stata utilizzata una *tabella delle lunghezze d'onda degli elementi pubblicata in modo indipendente*, anziché i dati acquisiti su quello stesso strumento, quindi questi risultati dipendono davvero dall'accuratezza della calibrazione della lunghezza d'onda dello strumento. I risultati suggeriscono che la calibrazione della lunghezza d'onda è in effetti eccellente, in base ai piccoli errori per le *due righe di sodio ben note e relativamente forti* a 589 e 589.59 nm. Si è impostato il requisito di corrispondenza della lunghezza d'onda (MaxError) su 0,2 nm in questo esempio. (L'identificazione del ferro a 260.1 nm potrebbe essere messa in dubbio, in base all'errore di lunghezza d'onda relativamente grande e alla bassa ampiezza).

'Name'	'Position'	'Error'	'Amplitude'
' <u>Cadmium</u> '	[226.46]	[-0.039471]	[44.603]
' <u>Nickel</u> '	[231.66]	[0.055051]	[26.381]
' <u>Silicon</u> '	[251.65]	[0.041616]	[45.275]
' <u>Iron</u> '	[260.1]	[0.156]	[38.04]
' <u>Silicon</u> '	[288.18]	[0.022458]	[27.214]
' <u>Strontium</u> '	[421.48]	[-0.068412]	[41.119]
' <u>Barium</u> '	[493.35]	[-0.057923]	[72.466]
' <u>Sodium</u> '	[589]	[0.0057964]	[405.23]
' <u>Sodium</u> '	[589.57]	[-0.015091]	[315.2]
' <u>Potassium</u> '	[766.54]	[0.051585]	[61.987]

Nota: Il [file ZIP](#) contiene l'ultima versione della funzione *iPeak* e alcuni dati di esempio per mostrare l'identificazione del picco (Esempio 8). Ovviamente, per le proprie applicazioni, spetta all'utente fornire la matrice delle posizioni massime note ('Positions') e la corrispondente matrice di celle ('Names') per i propri tipi di segnali.

iPeak keyboard Controls (version 8.1):

```

Pan signal left and right...Coarse pan: < or >
Fine pan: left or right cursor arrow keys
Nudge one point left or right: [ and ]
Zoom in and out.....Coarse zoom: / or '
Fine zoom: up or down cursor arrow keys
Resets pan and zoom.....ESC
Select entire signal.....Ctrl-A
Refresh entire plot.....Enter (Updates cursor position in lower plot)
Change plot color.....Shift-C (cycles through standard colors)
Adjust AmpThreshold.....A,Z (Larger values ignore short peaks)
Type in AmpThreshold.....Shift-A (Type value and press Enter)
Adjust SlopeThreshold.....S,X (Larger values ignore broad peaks)
Type in SlopeThreshold.....Shift-S (Type value and press Enter)
Adjust SmoothWidth.....D,C (Larger values ignore sharp peaks)
Adjust FitWidth.....F,V (Adjust to cover top part of peaks)
Toggle sharpen mode .....H Helps detect overlapped peaks.
Enter de-tailing factor.....Shift-Y. Removes exponential peak tails
Adjust de-tailing 10%.....1,2 Adjust peak de-tailing factor 10%.
Adjust de-tailing 1%.....Shift-1,Shift-2 Adjust de-tailing 1%.
Baseline correction.....B, then click baseline at multiple points
Restore original signal.....G to cancel previous background subtraction
Invert signal.....- Invert (negate) the signal (flip + and -)
Set minimum to zero.....0 (Zero) Sets minimum signal to zero
Interpolate signal.....Shift-I Interpolate (re-sample) to N points
Toggle log y mode OFF/ON....Y Plot log Y axis on lower graph
Cycles baseline modes.....T 0=none; 1=linear; 2=quad; 3=flat;
4=linear mode(y); 5=flat mode(y)
Toggle valley mode OFF/ON...U Switch to valley mode
Gaussian/Lorentzian mode....Shift-G Cycle between Gaussian,
Lorentzian, and flat-top modes
Print peak table.....P Prints Peak #, Position, Height, Width
Save peak table.....Shift-P Saves peak table as disc file

```

Step through peaks.....Space/Tab Jumps to next/previous peak
 Jump to peak number.....J Type peak number and press Enter
 Normal peak fit.....N Fit peaks in upper window with peakfit.m
 Multiple peak fit.....M Fit all peaks in signal with peakfit.m
 Ensemble average all peaks..Shift-E ([Read instructions first](#))
 Print keyboard commands.....K Prints this list
 MeasUre peak areasShift-U Areas by perp. drop and tan. skim.
 Print findpeaks arguments...Q Prints findpeaks function with arguments.
 Print ipeak arguments.....W Prints ipeak function with all arguments.
 Print report.....R Prints Peak table and parameters
 Print peak statistics.....E prints mean, std of peak intervals,
 heights, etc.
 Peak labels ON/OFF.....L Label all peaks detected in upper window.
 Peak ID ON/OFF.....I Identifies closest peaks in
 'Names' database.
 Print peak IDs.....O Prints table of peaks IDs
 Switch to ipf.m.....Shift-Ctrl-F Transfer current signal to
 Interactive Peak Fitter
 Switch to iSignal.....Shift-Ctrl-S Transfer current signal
 to iSignal.m
 Expand to full screen.....Double-click figure window title bar

[Cliccare per delle istruzioni passo-passo animate](#)

iPeak Funzioni demo

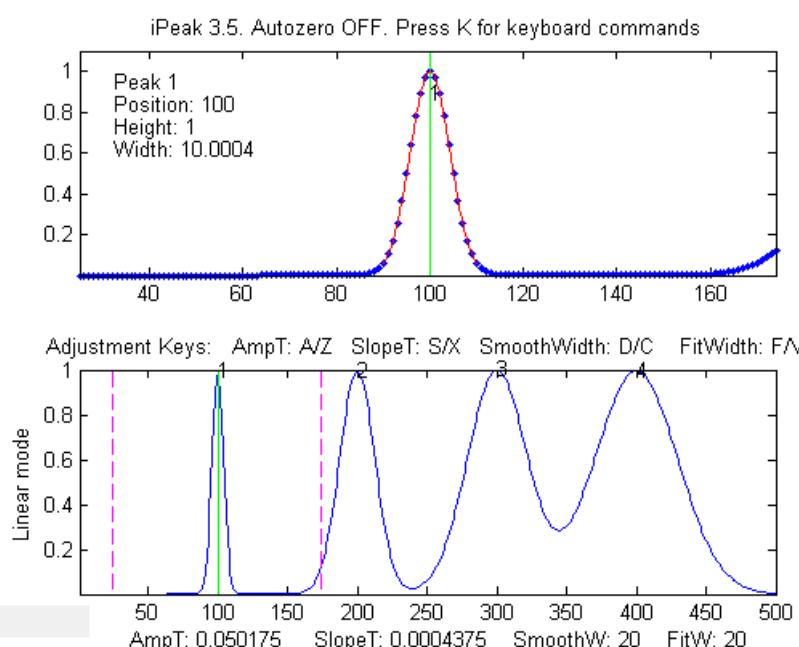
[demoipeak.m](#) (o [demoipeakoctave](#)) è una funzione dimostrativa che genera un segnale rumoroso con picchi, chiama *iPeak* e poi stampa una tabella dei parametri effettivi e un elenco dei picchi rilevati e misurati da *iPeak* per un confronto. Prima di eseguire questa demo, è necessario scaricare [ipeak.m](#) e lo si deve porre nel path di ricerca di Matlab. Il [file ZIP](#) contiene diversi script demo (ipeakdemo.m, ipeakdemo1.m, ecc.) che illustrano i vari aspetti della funzione *iPeak* e di come si possa usare efficacemente. Scaricare il file zip file, click-destro e selezionare "Extract all", quindi inserire i file risultanti nel path di Matlab ed eseguirli digitandone il nome Matlab al prompt nella finestra di comando. Per verificare la corretta installazione e funzionamento di *iPeak*, eseguire "[testipeak.m](#)".

[PeakAreaMethods.m](#) è uno script che confronta i tre metodi disponibili in *iPeak.m* per la misura dell'area dei picchi, utilizzando un segnale generato al computer di 5 picchi sovrapposti con altezze e larghezze diverse ma aree uguali. Le aree dei picchi vengono misurate mediante una stima Gaussiana (GE), taglio verticale (PD) e approssimazione iterativa della curva (Fa), che vengono presi come valori corretti. È possibile controllare la forma e la larghezza del picco nelle righe 18 e

19.

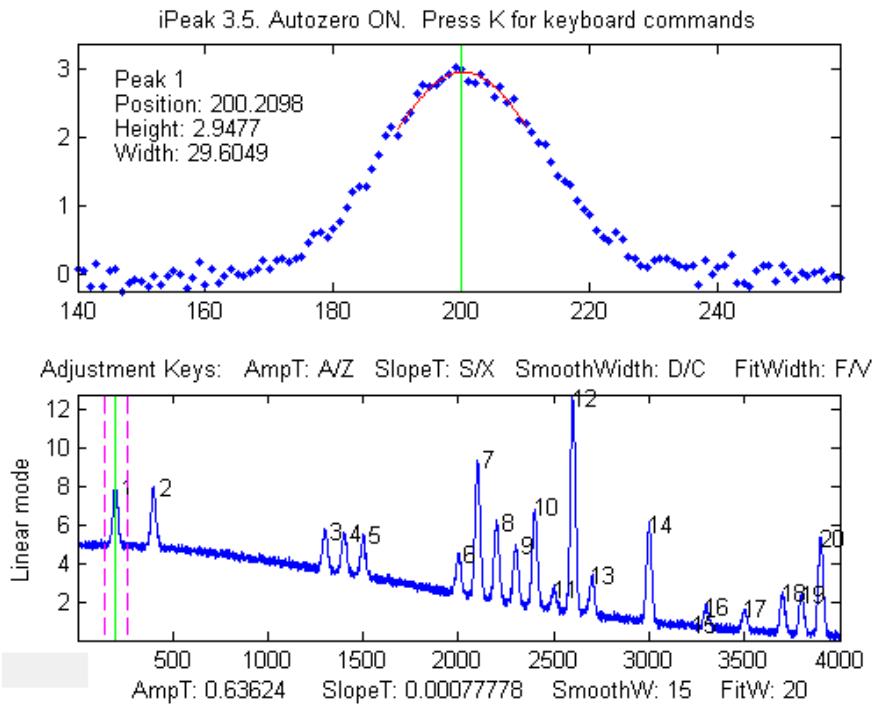
[ipeakdemo: effetto dei parametri di rilevamento del picco](#)

Quattro picchi Gaussiani con la stessa altezza ma diverse larghezze (10, 30, 50 e 70 unità). Mostra l'effetto di SlopeThreshold e SmoothWidth sul rilevamento del picco. Aumentando SlopeThreshold (tasto S) si discrimineranno i picchi più



larghi. Aumentando SmoothWidth (tasto D) si discrimineranno i picchi più stretti e il rumore. FitWidth (regolato dai tasti F e V) controlla il numero di punti intorno alla "parte superiore" del picco (senza smoothing) che vengono presi per stimare le altezze, le posizioni e le larghezze dei picchi. Un valore ragionevole è normalmente circa uguale a 1/2 del numero di punti nella semi-larghezza dei picchi. In questo caso, dove le larghezze dei picchi sono abbastanza diverse, lo si imposta a circa 1/2 del numero di punti nel picco più stretto.

ipeakdemo1: la modalità di correzione della linea di base. Dimostrazione della correzione del

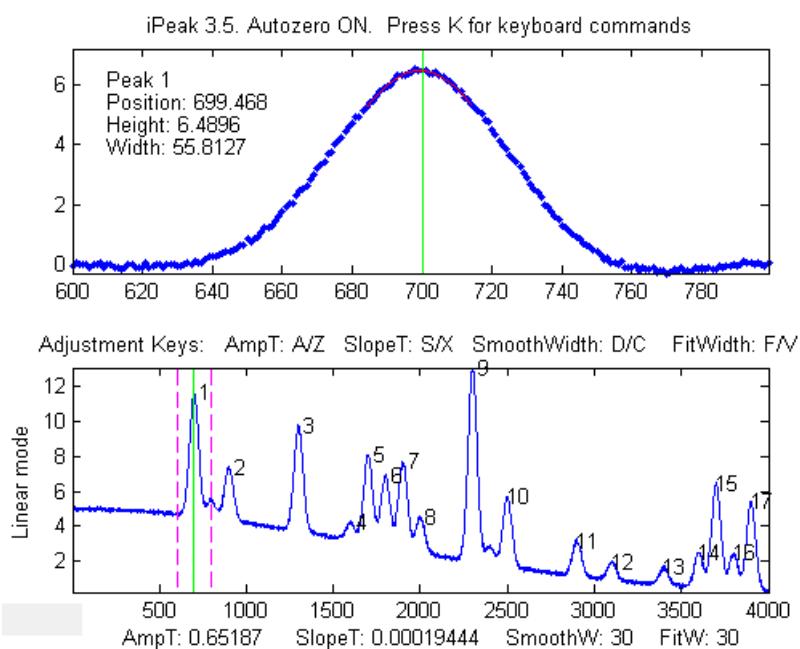


background, con picchi stretti e separati su un'ampia linea di base. Ogni volta che si esegue questa demo, si otterranno una serie diversa di picchi e rumore. Nella finestra di comando viene stampata una tabella delle posizioni, delle altezze, delle larghezze e delle aree effettive dei picchi. Si salta al picco successivo/precedente con i tasti **Spacebar/Tab**.

Suggerimento: Selezionare la modalità di correzione lineare della linea di base (tasto **T**), regolare lo zoom in modo che i picchi vengano visualizzati

uno alla volta nella finestra superiore, quindi premere il tasto **P** per visualizzare la tabella dei picchi. Per segnali con picchi come questi, sono spesso utili le modalità di correzione "mode(y)" 4 e 5, in quanto sottraggono il *valore y più comune* (la modalità statistica) dai punti nella regione selezionata, che di solito rimuove la linea di base e non è necessario che il segnale ritorni alla linea di base alle estremità della regione selezionata come le modalità 2 e 3.

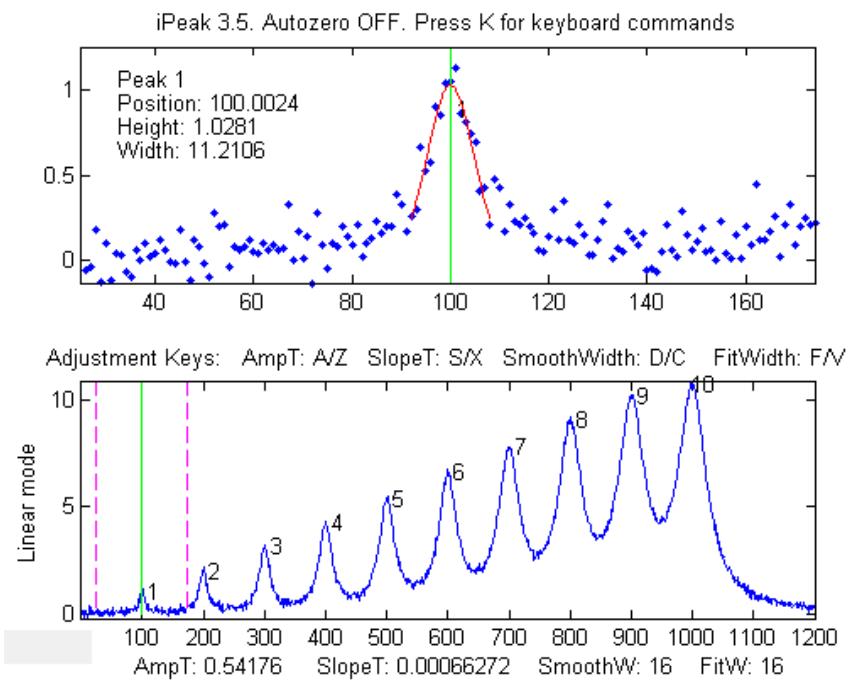
ipeakdemo2: picchi sovrapposti e funzioni per l'approssimazione.



Dimostrazione di un errore causato da picchi sovrapposti su un'ampia linea di base. Ogni volta che si esegue questa demo, si otterranno una serie diversa di picchi e rumore. Nella finestra di comando viene stampata una tabella delle posizioni, delle altezze, delle larghezze e delle aree effettive dei picchi. Si passa al picco successivo/precedente con i tasti Spazio/Tab. Suggerimento: Usare il tasto **B** e cliccare sui punti della linea di base, poi premere il tasto **P** per visualizzare la tabella dei picchi. Oppure utilizzare le modalità 2-4 di correzione del

background e l'approssimazione normale della curva (tasto **N**) col profilo 1 (Gaussiano).

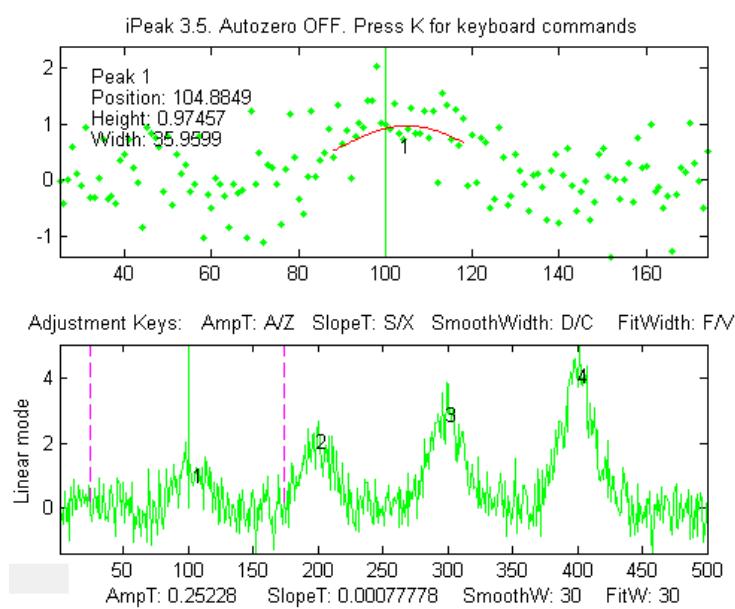
ipeakdemo3: Spostamento della linea di base causato da picchi sovrapposti



Dimostrazione di picchi Lorentziani sovrapposti, senza un background aggiunto. Nella finestra di comando viene stampata una tabella delle posizioni, delle altezze, delle larghezze e delle aree effettive dei picchi; in questo esempio, le vere altezze dei picchi sono 1,2,3,...,10. La sovrapposizione dei picchi può causare errori significativi nella misura dei parametri, specialmente per i picchi Lorentziani, perché hanno lati leggermente inclinati che contribuiscono alla linea di base di eventuali altri picchi nella regione.

Suggerimento: disattivare la modalità di correzione del background (tasto **T**) ed usare l'approssimazione Normale (tasto **N**) per approssimare piccoli gruppi di 2-5 picchi numerati nella finestra superiore, col profilo 2 (Lorentziano). Per la massima precisione nella misurazione di un particolare picco, includere uno o due picchi aggiuntivi su entrambi i lati, per aiutare a tenere conto dello spostamento della linea di base causato dai picchi vicini. In alternativa, se il numero totale di picchi non è troppo grande, è possibile utilizzare l'approssimazione Multipla (tasto **M**) per approssimare l'intero segnale nella finestra inferiore.

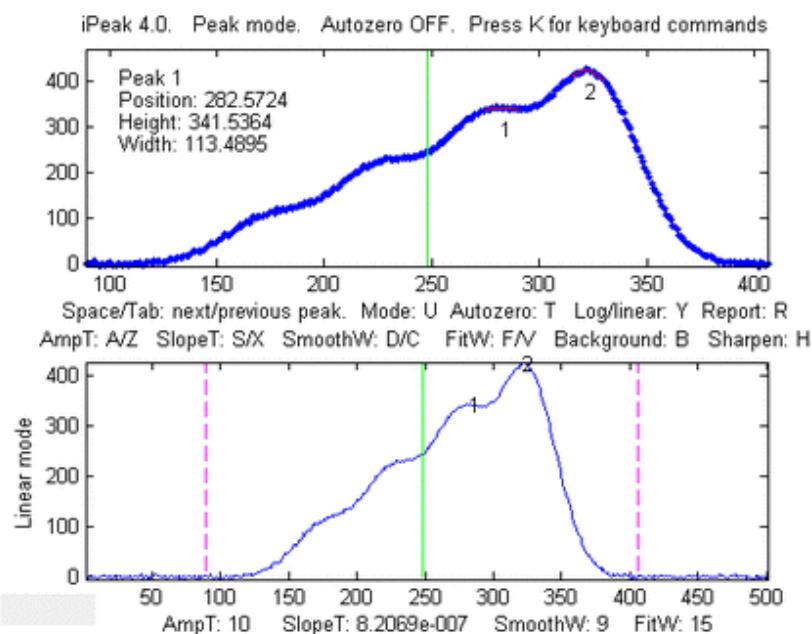
ipeakdemo4: gestire segnali molto rumorosi



Rilevamento e misurazione di quattro picchi in un segnale molto rumoroso. Il rapporto segnale/rumore del primo picco è 2. Ogni volta che si esegue questo demo, si avrà un diverso set di rumore. Nella finestra di comando viene stampata una tabella delle posizioni, delle altezze, delle larghezze e delle aree effettive dei picchi. Si passa al picco successivo/precedente con i tasti **Spazio/Tab**. Il picco in $x=100$ viene solitamente rilevato, ma la precisione della misura del parametro del picco è scarsa a causa del basso rapporto segnale/rumore. **ipeakdemo6** è simile ma ha si possono scegliere diversi tipi di rumore (bianco, rosa, proporzionale, ecc.)

Suggerimento: Con segnali molto rumorosi è solitamente meglio aumentare *SmoothWidth* e *FitWidth* per ridurre l'effetto del rumore.

ipeakdemo5: per gestire picchi molto sovrapposti



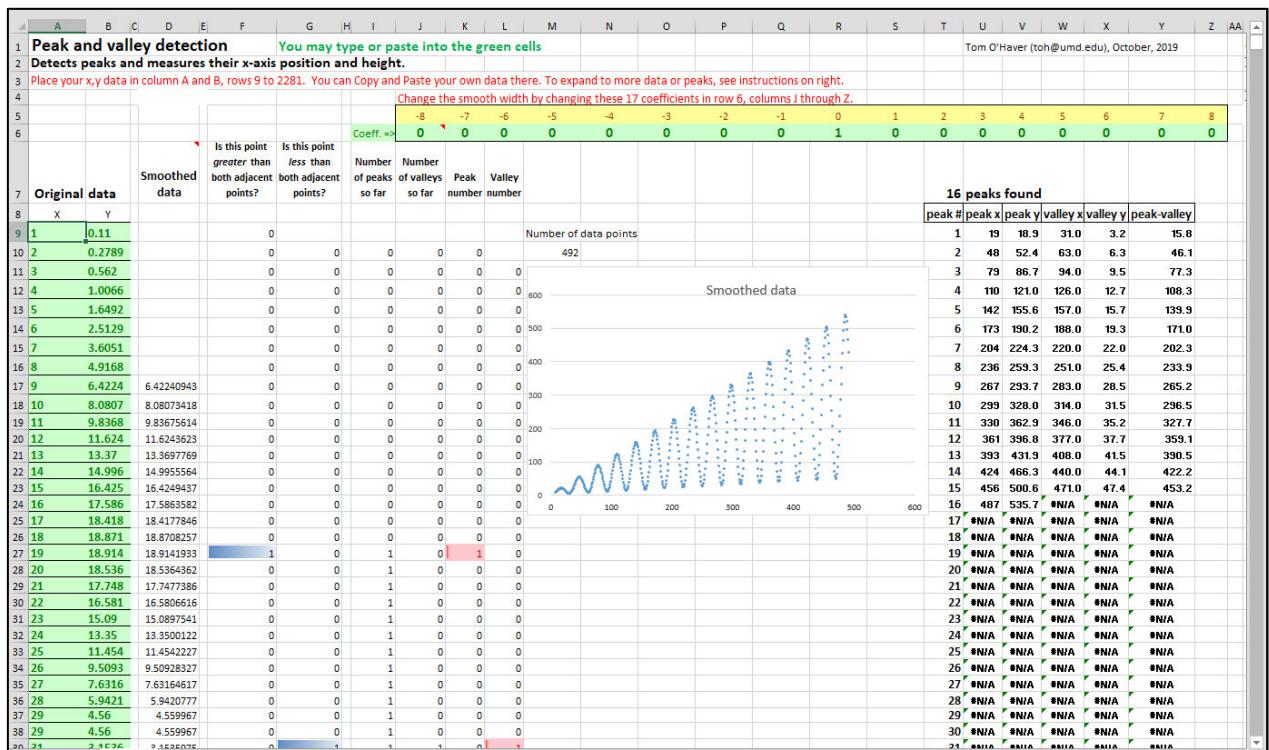
ti.

Per aiutare in questo caso, premendo il tasto 'H' si attiverà la funzione di sharpening del picco che diminuisce l'ampiezza e aumenta l'altezza di tutti i picchi, rendendo più facile rilevare e numerare tutti i picchi ad uso della funzione peakfit (tasto N). Nota: peakfit approssima i picchi originali non modificati; lo sharpening viene utilizzato solo per contribuire a localizzare i picchi per fornire a peakfit dei valori iniziali adeguati.

In questa demo, i picchi sono così sovrapposti che solo uno o due dei picchi tra i più alti producono massimi distinti che vengono rilevati da iPeak. Le stime di altezza, larghezza e area sono molto imprecise a causa di questa sovrapposizione. La normale funzione di approssimazione dei picchi (tasto N') sarebbe utile in questo caso, ma dipende da iPeak per il numero di picchi e per le ipotesi iniziali, quindi approssimerebbe solo i picchi trovati e numerati.

Se si usa **Python**, un algoritmo di base per la ricerca dei picchi è descritto a pagina 432 e su <https://terpconnect.umd.edu/~toh/spectrum/Python.html>

Peak Finder con Template di Fogli di Calcolo



Semplice rilevamento di picchi e avvallamenti. Il foglio di calcolo illustrato sopra, [PeakAndValleyDetectionTemplate.xlsx](#) (o [PeakAndValleyDetectionExample.xlsx](#) con i dati di esempio), è un semplice rilevatore di picchi e valli che *definisce un picco come qualsiasi punto con dei punti più bassi su entrambi i lati e una valle come qualsiasi punto con punti più alti su entrambi i lati*. I picchi e le valli sono contrassegnati da celle colorate nelle colonne da F a L e messi nelle colonne da T a Y con i loro valori misurati x e y, in base all'uso delle funzioni INDIRECT, ADDRESS e MATCH come descritto a pagina 345. I dati originali possono essere opzionalmente filtrati con smoothing inserendo una larghezza di smoothing (un numero intero dispari positivo) nella cella E6 per sopprimere il falso rilevamento causato dal rumore casuale. Sono incluse le istruzioni per espandere il modello.

Rilevamento selettivo dei picchi. Lo spreadsheet [PeakDetectionTemplate.xls](#) (o [PeakDetectionExample.xls](#) con i dati di esempio) implementa il metodo di rilevamento dei picchi col passaggio per lo zero della derivata prima con soglie di ampiezza e pendenza come descritto a pagina 228. I dati di input x, y sono contenuti in Sheet1, nelle colonne **A** e **B**, righe dalla 9 alla 1200. (Qui, si possono digitare o incollare i propri dati). La soglia dell'ampiezza e quella della pendenza vengono impostate nelle celle **B4** ed **E4**, rispettivamente. Lo smoothing e la differenziazione vengono eseguite dalla tecnica della convoluzione usata dagli spreadsheet [DerivativeSmoothing.xls](#) descritti precedentemente a pagina 70. Lo Smooth Width e il Fit Width vengono controllati dal numero dei coefficienti della convoluzione diversi da zero nella riga 6, dalle colonne **J** alla **Z**. (Per calcolare una derivata prima simmetrica, i coefficienti nelle colonne da **J** a **Q** devono essere i negativi di quelli positivi nelle colonne da **S** a **Z**). I dati originali e la derivata con smoothing vengono mostrati nei due grafici. Per rilevare i picchi nei dati, viene testata una serie di tre condizioni per ciascun punto nelle colonne **F**, **G** e **H**, corrispondenti ai tre cicli annidati in [findpeaksG.m](#):

1. Il segnale è maggiore della soglia di ampiezza (Amplitude Threshold)? (La riga 45 di `findpeaksG.m`; colonna **F** nello spreadsheet)
2. C'è un attraversamento dello zero verso il basso (pagina 63) nella derivata prima con smoothing? (riga 43 di `findpeaksG.m`; colonna **G** nello spreadsheet)
3. La pendenza della derivata in quel punto è maggiore della Soglia di pendenza (Slope Threshold)? (riga 44 di `findpeaksG.m`; colonna **H** nello spreadsheet)

Se la risposta a *tutte e tre* le domande è *sì* (evidenziate dal colore blu della cella), in quel punto viene registrato un picco (colonna **I**), contato nella colonna **J** e assegnato un numero d'indice nella colonna **K**. I numeri di indice, le posizioni dell'asse X e le altezze dei picchi sono elencati nelle colonne da **AC** a **AF**. Le altezze dei picchi vengono calcolate in *due* modi: "Height" è basata sui valori Y con un leggero smoothing (più accurato se i picchi sono ampi e rumorosi, come in [PeakDetection-Demo2b.xls](#)) e "Max" è il valore individuale di Y più alto vicino al picco (più accurato se i dati subiscono uno smoothing o se i picchi sono molto stretti, come in [PeakDetectionDemo2a.xls](#)). [Peak-DetectionDemo2.xls/xlsx](#) è una dimostrazione con una serie generata al computer, controllata dall'utente, di quattro picchi Gaussiani rumorosi con parametri noti. [PeakDetectionSineExample.xls](#) è una demo che genera un segnale sinusoidale con un numero di picchi regolabile.

Questi spreadsheet si possono espandere con *colonne di dati più lunghe* trascinando l'ultima riga delle colonne dalla **A** alla **K** secondo le necessità, quindi si seleziona e si modificano i dati nei grafici per includere tutti i punti nei dati (Click-destro, Si seleziona dati, Modifica). Si può estendere lo spreadsheet con un *numero maggiore di picchi* trascinando l'ultima riga delle colonne dalla **AC** alla **AD** secondo le necessità. Si modifica **R7** e l'intervallo di dati nelle equazioni delle celle nella riga 9, colonne **U**, **V**, **W**, **X**, **AE** e **AF** per includere tutte le righe contenenti dati, quindi si copia e si trascinano verso il basso per coprire tutti i picchi previsti.

Rilevamento dei picchi con la misura dei quadrati minimi. Un'estensione di tale metodo è realizzata nel template [PeakDetectionAndMeasurement.xlsx](#) (schermata), che presuppone che i picchi siano Gaussiani e ne misura l'altezza, la posizione e larghezza più precisamente utilizzando una *tecnica dei quadrati minimi*, proprio come "[findpeaksG.m](#)". Per i primi 10 picchi trovati, i dati originali x, y senza smoothing vengono **copiati** negli Sheet dal 2 all'11, rispettivamente, dove quel segmento di dati è soggetto a un'approssimazione Gaussiana dei minimi quadrati, utilizzando la stessa tecnica di [GaussianLeastSquares.xls](#). I risultati dei parametri dell'approssimazione Gaussiana migliore, vengono ricopiatati nello Sheet1, nelle colonne **AH-AK**. (Nella sua forma attuale, lo spreadsheet è limitato a misurare 10 picchi, sebbene ne possa rilevare un numero qualsiasi. Inoltre, ha dei valori di larghezza dello smoothing [Smooth Width] e larghezza dell'approssimazione [Fit Width] limitati dai coefficienti della convoluzione da 17 punti).

Lo spreadsheet è disponibile nei formati OpenOffice ([ods](#)) ed Excel ([xls](#) e [xlsx](#)). Sono funzionalmente equivalenti e differiscono solo per dei particolari cosmetici. Ci sono due spreadsheet di esempio ([A](#) e [B](#)) con forme d'onda calcolate rumorose, modificabili. *Nota: Per inserire dati nelle versioni .xls e .xlsx, cliccare sul pulsante "Enable Editing" nella barra gialla in alto.*

Se i picchi nei dati sono troppo sovrapposti, i massimi potrebbero non essere sufficientemente distinti per un rilevamento affidabile. Se il livello di rumore è basso, i picchi possono essere eviden-

ziati con sharpening artificialmente con la [tecnica descritta](#). Questi viene implementato da [PeakDetectionAndMeasurementPS.xlsx](#) e la sua versione demo con dati di esempio già inseriti [PeakDetectionAndMeasurementDemoPS.xlsx](#).

Espandere il foglio di calcolo PeakDetectionAndMeasurement.xlsx con *un numero maggiore di picchi misurati* è più difficile. Si dovrà trascinare verso il basso la riga 17, colonne da **AC** ad **AK** ed aggiustare le formule in quelle righe per il numero di ulteriori picchi, poi copiare tutto lo Sheet11 e incollarlo in una serie di nuovi fogli (Sheet12, Sheet13, ecc.), uno per ciascun picco aggiuntivo, e infine aggiustare le formule nelle colonne **B** e **C** in ciascuno di questi fogli aggiuntivi affinché facciano riferimento alla giusta riga di Sheet1. Modificare queste equazioni aggiuntive secondo lo stesso schema di quello per i picchi 1-10. (A differenza di questi fogli di calcolo, le funzioni di ricerca dei picchi Matlab/Octave si adattano automaticamente a *qualsiasi* lunghezza dei dati e *qualsiasi* numero di picchi).

Spreadsheet contro Matlab/Octave. Un confronto tra questo foglio di calcolo e l'equivalente Matlab/Octave [findpeaksplot.m](#) è istruttivo. Sul lato positivo, il foglio di calcolo letteralmente "distribuisce" i dati e i calcoli spazialmente su molte celle e fogli, suddividendo i passaggi discreti in modo molto grafico. In particolare, l'uso della [formattazione condizionale](#) nelle colonne da **F** a **K** rende il processo di rilevamento del picco più evidente per ciascun picco, e i fogli per i quadrati minimi dal 2 all'11 presentano tutti i dettagli per quei calcoli. Ciò ha un valore *istruttivo* significativo. I fogli di calcolo hanno molte flessibili e facili opzioni per la formattazione utilizzabili per rendere le visualizzazioni più attraenti. Di negativo, c'è che uno spreadsheet complicato come PeakDetectionAndMeasurement.xlsx è molto più difficile da costruire rispetto al suo equivalente Matlab/Octave. Molto più seriamente, il foglio di calcolo è *meno flessibile* e più difficile da espandere a segnali più grandi e ad un numero maggiore di picchi. Al contrario, l'equivalente Matlab/Octave è facile da usare, più veloce nell'esecuzione, molto più flessibile e può facilmente gestire segnali ed eseguire smoothing e approssimazione di larghezze di *qualsiasi dimensione*. Gli spreadsheet diventano ingombranti con set di dati molto grandi. Inoltre, una funzione Matlab/Octave può essere facilmente utilizzata come elemento nei programmi Matlab/Octave personalizzati per eseguire compiti ancora più grandi. È più difficile farlo in un foglio di calcolo.

Per confrontare la velocità di calcolo di questo foglio di calcolo per la ricerca dei picchi con l'equivalente Matlab/Octave, si può prendere come esempio lo spreadsheet [PeakDetectionExample2.xls](#) o [PeakDetectionExample2.ods](#), che calcola e disegna un segnale di prova costituito da una forma d'onda sinusoidale rumorosa con 300 punti e poi rileva e misura 10 picchi in quella forma d'onda visualizzando una tabella dei parametri dei picchi. Questo è equivalente allo script Matlab/Octave:

```
tic
x=1:300;
y(1:99)=randn(size(1:99));
y(100:300)=10.*sin(.16.*x(100:300)).^2. + randn(size(x(100:300)));
P=findpeaksplot(x,y,0.005,5,7,7,3);
disp(P)
drawnow
toc
```

La tabella seguente confronta i tempi impiegati misurati per Matlab 2020 e per Octave 6.1.0 su un Dell XPS i7 3.5Ghz. Il vantaggio, in termini di velocità, di Matlab è chiaro. Python (pagina 432) può fare altrettanto bene di Matlab e alla stessa velocità.

Metodo	Tempo impiegato
Excel	~ 1 sec
Calc	~ 1 sec
Matlab o Python	0.035 sec
Octave	0.5 sec

Questo è un test piuttosto piccolo; molte applicazioni reali hanno molti più punti e molti più picchi, in cui il vantaggio in termini di velocità di Matlab/Python sarebbe più significativo. Inoltre, questi sarebbero gli strumenti da scegliere se si hanno molti set di dati separati a cui è necessario applicare un algoritmo di rilevamento/misura dei picchi in modo completamente automatico (pagina 337).

Vedere anche [Excel VS Matlab](#).

Spettrofotometria di Assorbimento Quantitativo Iperlineare

La conoscenza specifica della strumentazione è spesso utile nella progettazione di un efficace sistema di signal-processing. Questo esempio, proveniente dalla mia ricerca del 1980, mostra come si possa costruire una procedura personalizzata di elaborazione del segnale per un sistema di misura ottica combinando diversi metodi e concetti introdotti in questo libro. Il risultato è una tecnica che *espande i limiti classici della misura* nella spettroscopia ottica di assorbimento. Questo è un metodo computazionale alternativo per l'analisi quantitativa mediante [spettroscopia di assorbimento a lunghezze d'onda multiple](#), chiamato metodo di approssimazione della trasmissione o "TFit". Si basa sull'[approssimazione di un modello](#) dello spettro di trasmissione ampliato strumentalmente ai dati di trasmissione osservati, ed è un'alternativa al metodo consueto di calcolo dell'assorbanza mediante la semplice definizione "classica" di $\log_{10}(I_0/I)$. Il metodo è descritto in T. C. O'Haver and J. Kindervater, *J. of Analytical Atomic Spectroscopy* **1**, 89 (1986); T. C. O'Haver and Jing-Chyi Chang, *Spectrochim. Acta* **44B**, 795-809 (1989); T. C. O'Haver, *Anal. Chem.* **68**, 164-169 (1991). È stato incluso qui come esempio di combinazione di diverse tecniche di elaborazione del segnale perché mette assieme molti dei concetti importanti che sono stati trattati in questo libro, vale a dire: rapporto segnale-rumore (pag. 22), Convoluzione di Fourier (pag. 103), spettroscopia multicomponente (pag. 179), approssimazione dei minimi quadrati iterativa (pag. 190) e calibrazione per l'analisi quantitativa (pag. 329).

I vantaggi del metodo TFit rispetto ai metodi convenzionali sono:

(a) un *intervallo dinamico* molto più ampio cioè l'intervallo della concentrazione su cui ci si può aspettare che una curva di calibrazione dia buoni risultati),

(b) [linearità della calibrazione](#) molto migliore ("iper-linearmemente"), che riduce la manodopera e il costo della preparazione e della esecuzione di un gran numero di soluzioni standard e un sicuro smaltimento successivo, e

(c) la capacità di operare in condizioni ottimizzate per rapporto segnale/rumore anziché per l'ideale Legge di Beer, ovvero, utilizzando piccoli spettrometri con lunghezza focale inferiore, dispersione inferiore e larghezze della fenditura maggiori per *aumentare il flusso luminoso* e ridurre l'effetto fotonico e del rumore del rivelatore (supponendo, ovviamente, che il rivelatore non sia saturo o sovraccarico).

Proprio come i classici metodi di regressione multi-lineare (quadrati minimi classico) che vengono usati convenzionalmente nella spettroscopia di assorbimento, il metodo Tfit

(a) richiede uno spettro di riferimento accurato di ogni analita,

(b) utilizza dati delle lunghezza d'onda multiple come sarebbero acquisiti dagli array di diodi, dalla trasformata di Fourier o dagli spettrometri a scansione automatica,

(c) si applica sia all'analisi di miscele monocomponente che quelle multi-componente, e

(d) richiede che le assorbanze dei componenti *varino con la lunghezza d'onda*, e, per l'analisi multicomponente, che gli spettri di assorbimento dei componenti siano sufficientemente diversi. Gli assorbenti neri o grigi non funzionano con questo metodo.

Gli svantaggi del metodo Tfit sono:

(a) Fa lavorare molto di più il *computer* rispetto ai metodi di regressione multilineare (ma, su un tipico personal computer, i calcoli richiedono solo una frazione di secondo, anche per l'analisi di una miscela con più componenti).

(b) Richiede la conoscenza della funzione dello strumento, cioè la funzione fenditura o la funzione di risoluzione di uno spettrometro ottico. (Questa, tuttavia, è una *caratteristica fisica* dello strumento e può essere misurata in anticipo scansionando lo spettro di una sorgente a riga atomica stretta come una lampada a catodo cavo). Cambia solo se si modifica la larghezza della fenditura dello spettrometro.

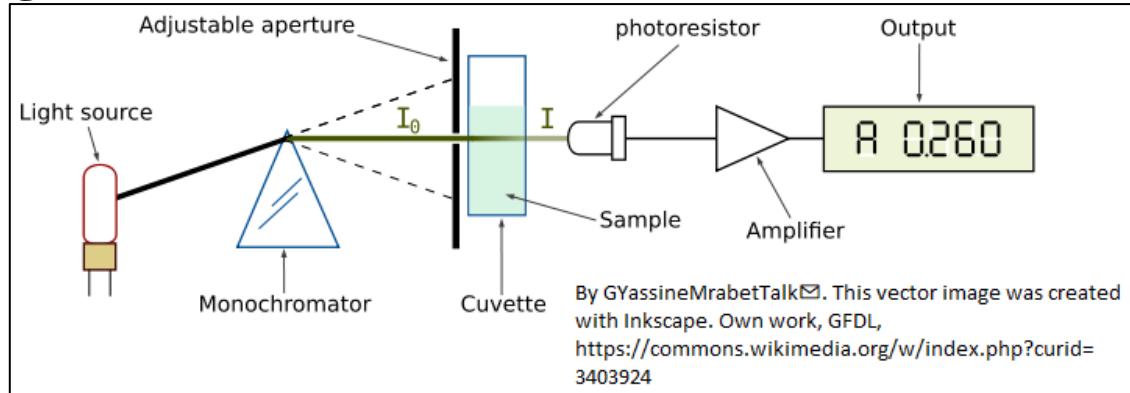
(c) È un metodo iterativo che, in circostanze sfavorevoli, può convergere su un errato ottimo locale, ma questo viene gestito mediante una corretta selezione del valore iniziale calcolato con il metodo logaritmico tradizionale $\log(I_0/I)$, e

(d) Non funzionerà per sostanze di colore grigio i cui spettri di assorbimento non variano nella regione spettrale misurata.

È possibile eseguire i calcoli richiesti in un foglio di calcolo o in Matlab o Octave, utilizzando il software descritto di seguito.

Le sezioni seguenti forniscono il [background del metodo](#) e una descrizione della [funzione principale, programmi dimostrativi](#) e [template](#):

Background



La figura precedente mostra la [spettroscopia di assorbimento ottica](#), dove l'intensità “I” della luce monocromatica che passa attraverso un campione assorbente è data (in notazione Matlab) dalla [Legge di Beer-Lambert](#):

$$I = I_0 \cdot 10^{-\alpha \cdot L \cdot c}$$

dove “ I_0 ” (si pronuncia "eye-zero") è l'intensità della luce incidente del campione, “ α ” è il coefficiente di assorbimento dell'assorbente, “ L ” è la distanza percorsa dalla luce attraverso il materiale (la lunghezza del percorso ottico) e “ c ” è la concentrazione dell'assorbente nel campione. Le variabili I , I_0 e α sono tutte in funzione della lunghezza d'onda; L e c sono scalari.

Tradizionalmente, i valori misurati di I e I_0 vengono utilizzati per calcolare una quantità chiamata ["assorbanza"](#), definita come

$$A = \log_{10}(I_0/I)$$

L'assorbanza è definita così in modo che, quando si combina questa definizione con la legge di Beer-Lambert, si ottiene:

$$A = \alpha \cdot L \cdot c$$

In questo modo, l'assorbanza è, idealmente, proporzionale alla concentrazione il che semplifica la calibrazione analitica. Questo funziona per fasci di luce monocromatici. Tuttavia, qualsiasi spettrometro reale ha una *risoluzione spettrale finita*, il che significa che il raggio di luce che passa attraverso il campione non è veramente monocromatico, col risultato che una lettura dell'intensità a una lunghezza d'onda impostata è una media su un piccolo intervallo spettrale, che è determinata dalla "apertura regolabile" nel diagramma sopra, detta anche "larghezza della fenditura". Più esattamente, ciò che viene misurato è una [convoluzione](#) del vero spettro dell'assorbente e della *funzione dello strumento*, la risposta dello strumento in funzione della lunghezza d'onda della luce. Idealmente, la funzione dello strumento è infinitamente stretta (una funzione "delta"), ma in pratica gli spettrometri hanno una *non nulla larghezza della fenditura*, la larghezza dell'apertura regolabile nello schema

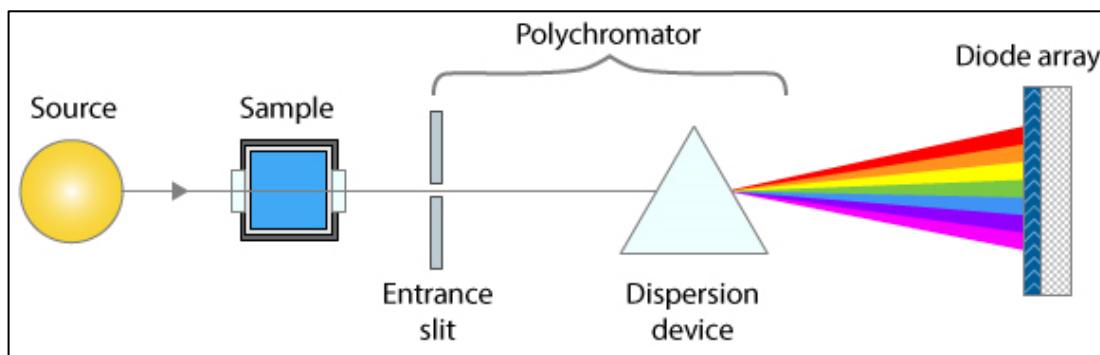
sopra, in cui passa un piccolo intervallo spettrale di lunghezze d'onda della luce dall'elemento dispersante (prisma) sul campione e sul rivelatore. Se il coefficiente di assorbimento "alpha" varia su quell'intervallo spettrale, l'assorbanza calcolata tradizionalmente non sarà più linearmente proporzionale alla concentrazione (questo è detto errore di "policromicità"). L'effetto è più evidente ad assorbimenti elevati. In pratica, molti strumenti diventeranno non lineari a partire da un'assorbanza di 2 (~1% Trasmissione). All'aumentare dell'assorbanza, l'effetto della luce diffusa non assorbita e del rumore dello strumento diventa più significativo.

Si può dimostrare che il miglior rapporto segnale/rumore teorico e la precisione di assorbanza per uno strumento di assorbimento ottico limitato dal rumore fotonico si avvicinano a un'assorbanza prossima a 1.0 (vedere "Esiste un'assorbanza ottimale per il miglior rapporto segnale/rumore?"). L'intervallo di assorbanze *inferiori a* 1.0 è facilmente accessibile fino ad almeno 0.001, ma quello *superiore a* 1.0 è limitato. Anche un'assorbanza di 10 è irraggiungibile sulla maggior parte degli strumenti e la misura diretta di un'assorbanza di 100 è impensabile, poiché implica la misura dell'attenuazione della luce di dieci elevato a 100 - nessun sistema di misura reale ha una gamma dinamica che almeno ci si avvicini. In pratica, è difficile ottenere un intervallo dinamico anche fino a 5 o 6 unità di assorbanza, quindi gran parte dell'intervallo di assorbanza teoricamente ottimale *superiore a* 1.0 è inutilizzabile. (vedere <http://en.wikipedia.org/wiki/Absorbance>). Pertanto, nella pratica convenzionale, sono necessarie diluizioni dei campioni maggiori e percorsi più brevi per forzare l'intervallo dell'assorbanza a valori più bassi, *anche se questo significa un rapporto segnale/rumore più scarso e una precisione della misura nell'estremità inferiore*.

È vero che la non linearità causata dalla policromicità può essere ridotta azionando lo strumento alla massima risoluzione (riducendo la larghezza strumentale della fenditura). Tuttavia, questo provoca un grave effetto collaterale indesiderato: negli strumenti dispersivi, la riduzione della larghezza della fenditura per aumentare la risoluzione spettrale degrada sostanzialmente il rapporto segnale/rumore. Riduce anche il numero di atomi o molecole misurati. Ecco perché: la spettroscopia di assorbimento UV/visibile si basa sull'assorbimento di fotoni di luce da parte di molecole o atomi risultanti dalle transizioni tra stati energetici elettronici. È noto che i picchi di assorbimento delle molecole sono *bande*, non *linee* monocromatiche, perché le molecole stanno subendo transizioni vibrazionali e rotazionali oltre a quelle elettroniche e sono anche sotto l'influenza perturbatrice del loro ambiente. Questo è anche il caso della spettroscopia di assorbimento atomica: le "righe" di assorbimento degli atomi liberi in fase gassosa, sebbene molto più strette delle bande molecolari, sebbene hanno una larghezza finita diversa da zero, principalmente a causa della loro velocità (espansione per *temperatura* o per effetto *Doppler*) e delle collisioni con la matrice di gas (espansione per la *pressione*). Una raccolta macroscopica di molecole o atomi, quindi, presenta al fascio di luce incidente una *distribuzione* di stati energetici e lunghezze d'onda di assorbimento. L'assorbimento è il risultato dell'interazione collettiva di molti *singoli* atomi o molecole con *singoli* fotoni. Un fascio di luce incidente puramente *monocromatico* avrebbe fotoni tutti della *stessa* energia, idealmente corrispondente al massimo della distribuzione dell'energia dell'insieme di atomi o molecole misurati. Ma, infatti, molti degli atomi o delle molecole lungo il percorso della luce avrebbero un'energia leggermente *maggior o minore della media* e *non verrebbe quindi misurata*. Se la larghezza di banda del fascio incidente viene aumentata, più di quegli atomi o molecole al di fuori della media sarebbero disponibili per essere misurati e più fotoni trasmessi raggiungerebbero il rivelatore, ma

poi il semplice calcolo dell'assorbanza come $\log_{10}(I_0/I)$ non si traduce più in una risposta lineare alla concentrazione.

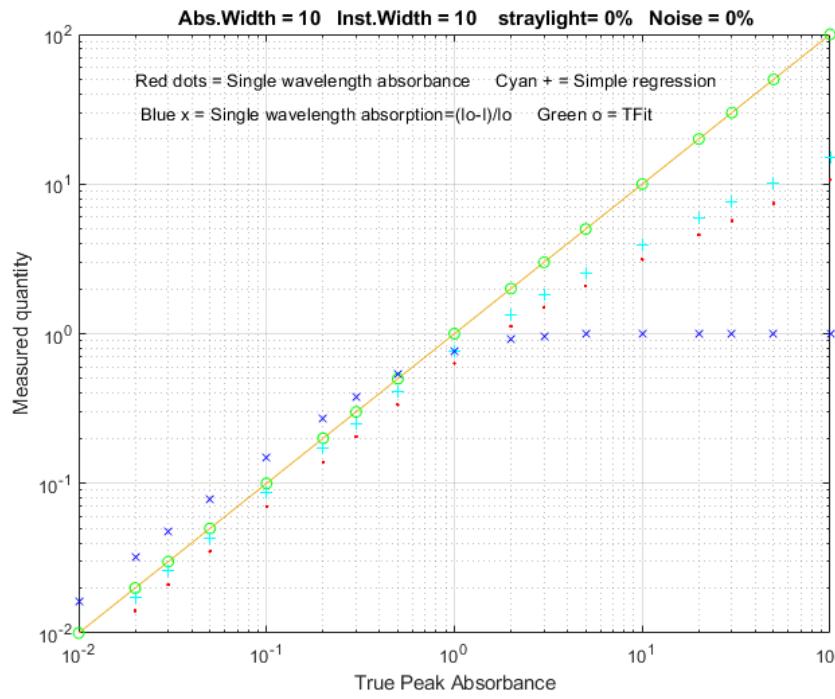
Le simulazioni numeriche mostrano che il rapporto segnale/rumore ottimale si ottiene tipicamente quando *la risoluzione spettrale dello strumento corrisponde all'ampiezza dell'assorbimento dell'analita*, ma in queste condizioni utilizzando il convenzionale $\log_{10}(I_0/I)$ l'assorbanza risulterebbe in una non linearità molto sostanziale nell'intervallo più alto dei suoi valori a causa dell'errore di "polichromicità". Questa non linearità ha origine nel *dominio spettrale* (intensità rispetto alla lunghezza d'onda), non nel *dominio della calibrazione* (assorbanza rispetto alla concentrazione). Pertanto non dovrebbe sorprendere che l'approssimazione della curva nel dominio della calibrazione, ad esempio l'approssimazione dei dati di calibrazione con un'approssimazione quadratica o cubica, non sarebbe



la soluzione migliore, perché non esiste una teoria che dice che le deviazioni dalla linearità dovrebbero essere esattamente quadratiche o cubiche. Un approccio più rigoroso basato sulla teoria sarebbe quello di eseguire l'approssimazione della curva nel *dominio spettrale*, dove c'è la sorgente della non linearità. Ciò è possibile con i *moderni* spettrofotometri ad assorbimento che utilizzano *array di rilevatori*, che hanno molti piccoli elementi rivelatori che suddividono lo spettro del raggio trasmesso in molti piccoli segmenti di lunghezza d'onda, piuttosto che rilevare la somma di tutti quei segmenti con un grande rivelatore fotovoltaico, come fanno gli strumenti più vecchi. Uno strumento con un array di rivelatori utilizza tipicamente una disposizione ottica leggermente diversa, come mostrato dal diagramma semplificato sopra: la luce viaggia *prima* attraverso il campione e *poi* verso il policromatore e l'array di rilevatori. La risoluzione spettrale è determinata sia dalla larghezza della fenditura d'ingresso sia dalla larghezza degli elementi del rivelatore ottico (o dal numero di quegli elementi che vengono sommati per determinare l'intensità trasmessa).

Il metodo TFit elude i problemi di cui sopra calcolando l'assorbanza in un modo completamente diverso. Inizia con gli spettri di riferimento (uno spettro di assorbimento accurato per ciascun analita, richiesto anche dai metodi di regressione multilineare). Normalizza gli spettri di riferimento all'altezza unitaria, moltiplica ciascuno per un coefficiente regolabile - di solito iniziando con l'assorbanza convenzionale $\log_{10}(I_0/I)$ come prima ipotesi - li somma, calcola l'antilog e fa la convoluzione con la funzione di fenditura misurata in precedenza. Il risultato, che rappresenta lo spettro di trasmissione ampliato strumentalmente, viene confrontato con lo spettro di trasmissione osservato. Il programma regola i coefficienti (uno per ogni componente ignota nella miscela) fino a quando il modello di trasmissione calcolato si l'approssimazione ai minimi quadrati si adatta allo spettro di trasmissione osservato. I coefficienti dell'approssimazione sono quindi uguali alle assorbanze determinate in condizioni ottiche ideali. Il programma compensa anche la luce diffusa non assorbita e le variazioni dell'intensità del background (assorbimento del background). Questi calcoli vengono eseguiti dalla funzione [fitM](#), che viene utilizzata come funzione di approssimazione per la funzione

di approssimazione iterativa non lineare di Matlab [fminsearch](#). Sembra complicato ma, in realtà, richiede solo una frazione di secondo per il calcolo. Il metodo TFit fornisce le misure dell'assorbanza che sono molto più vicine al "vero" picco di assorbanza che sarebbe stato misurato in assenza di luce diffusa ed errori di luce policromatica. Ancora più importante, consente di effettuare misure lineari e ad ampio range dinamico anche se la larghezza della fenditura dello strumento viene aumentata ottimizzando il rapporto segnale/rumore. Da una prospettiva storica, quando Pierre Bouguer formulò quella che divenne nota come legge di *Beer-Lambert* nel 1729, il *logaritmo* era già ben noto,



essendo stato introdotto da [John Napier](#) nel 1614. Il lavoro matematico aggiuntivo necessario per calcolare l'*assorbanza*, $\log(I_o/I)$, piuttosto che il più semplice *assorbimento*, $(I_o - I)/I_o$, è stato giustificato dalla migliore linearità dell'assorbanza rispetto alla concentrazione e alla lunghezza del percorso ottico. Inoltre, il calcolo del log poteva essere facilmente eseguito anche in quel periodo semplicemente anche con un [regolo calcolatore](#). Certamente, per gli standard odierni, il calcolo di un logaritmo è considerato completamente di routine. Al contrario, il metodo TFit presentato qui è certamente matematicamente più complesso del calcolo di un logaritmo e non può essere eseguito senza l'ausilio del software (almeno di un [foglio di calcolo](#)) e almeno [qualche hardware computazionale miniaturizzato basilare](#). Tuttavia, offre un ulteriore miglioramento della linearità oltre a quello ottenuto dal calcolo logaritmico dell'assorbanza, e inoltre consente di *allentare la piccola limitazione della larghezza della fenditura*. La figura sopra a destra confronta l'intervallo dinamico di un semplice assorbimento relativo (x blu), dell'assorbanza logaritmica a lunghezza d'onda singola (punti rossi), della regressione multilineare o del metodo CLS (+ in ciano) basato sull'assorbanza e del metodo TFit (o in verde) su un intervallo di assorbanze di 10^4 . Questo grafico è stato creato dallo script Matlab/Octave [TFitCalCurveAbs.m](#).

Conclusion: È importante capire che *il metodo TFit non rinnega la legge di Beer-Lambert*; infatti; è *basato* su di essa. Il metodo TFit semplicemente *calcola l'assorbanza in un modo diverso* che non richiede l'assunzione che gli effetti della luce diffusa e della radiazione policromatica siano nulli. Poiché consente di utilizzare larghezze di fenditura maggiori e lunghezze focali più corte, produce rapporti segnale/rumore maggiori pur ottenendo una gamma dinamica lineare molto più ampia rispetto ai metodi precedenti, richiedendo quindi meno standard per definire correttamente la curva di calibrazione ed evitare la necessità di modelli di calibrazione non lineari. Si tenga presente che il calcolo dell'assorbanza $\log(I_o/I)$ è un'[esemplificazione vecchia di 165 anni](#) guidata dalla necessità della *convenienza matematica* (e limitato anche dalle capacità matematiche degli studenti universitari a cui questa materia viene tipicamente presentata per la prima volta), *non* dal desiderio di ottimizzare la linearità e il rapporto segnale/rumore. Risale al tempo prima dell'elettronica e dei computer, quando gli unici strumenti di calcolo erano carta, penna e regoli, e quando un metodo come

Conclusione: È importante capire che *il metodo TFit non rinnega la legge di Beer-Lambert*; infatti; è *basato* su di essa. Il metodo TFit semplicemente *calcola l'assorbanza in un modo diverso* che non richiede l'assunzione che gli effetti della luce diffusa e della radiazione policromatica siano nulli. Poiché consente di utilizzare larghezze di fenditura maggiori e lunghezze focali più corte, produce rapporti segnale/rumore maggiori pur ottenendo una gamma dinamica lineare molto più ampia rispetto ai metodi precedenti, richiedendo quindi meno standard per definire correttamente la curva di calibrazione ed evitare la necessità di modelli di calibrazione non lineari. Si tenga presente che il calcolo dell'assorbanza $\log(I_o/I)$ è un'[esemplificazione vecchia di 165 anni](#) guidata dalla necessità della *convenienza matematica* (e limitato anche dalle capacità matematiche degli studenti universitari a cui questa materia viene tipicamente presentata per la prima volta), *non* dal desiderio di ottimizzare la linearità e il rapporto segnale/rumore. Risale al tempo prima dell'elettronica e dei computer, quando gli unici strumenti di calcolo erano carta, penna e regoli, e quando un metodo come

quello descritto qui sarebbe stato completamente inimmaginabile. Quello era allora; questo è ora.
Tfit è il metodo del 21° secolo per eseguire la spettrofotometria di assorbimento quantitativa.

Nota 1: Il metodo TFit compensa la non linearità causata dalla luce diffusa non assorbita e dall'effetto della luce policromatica, ma rimangono altre potenziali fonti di non linearità, specificatamente, gli *effetti chimici*, come fotolisi, variazioni di equilibrio, effetti della temperatura e del pH, legami, dimerizzazione, polimerizzazione, fototropismo molecolare, fluorescenza, ecc. Un metodo analitico quantitativo ben progettato lo è per ridurre al minimo questi effetti.

Nota 2: Nelle applicazioni pratiche, le informazioni richieste dal metodo Tfit potrebbero non essere esattamente note, ma una stima ragionevole è migliore di niente. Per esempio, anche una stima imperfetta della larghezza di banda strumentale e/o della luce diffusa è migliore della semplice supposizione che siano *zero*.

Implementazione dello spreadsheet

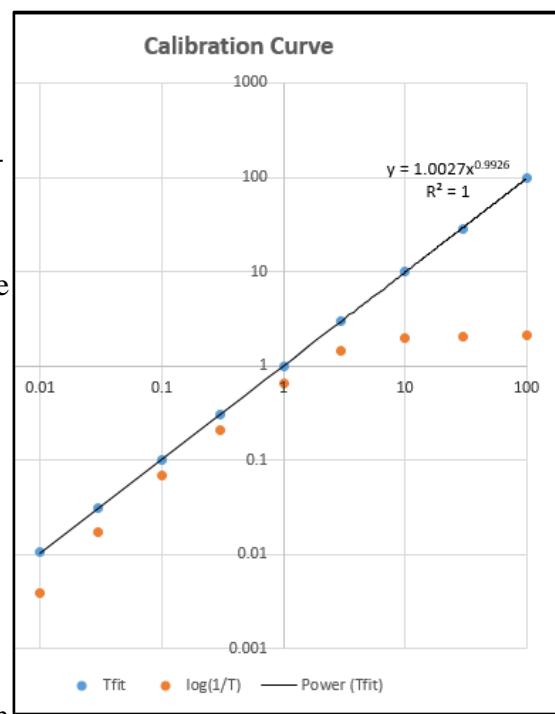
Il metodo Tfit può anche essere implementato in un foglio di calcolo *Excel* o *Calc*; è un po' più macchinoso dell'[implementazione in Matlab/Octave](#), ma funziona. Il metodo trasla-e-moltiplica viene utilizzato per la convoluzione dello spettro di riferimento con la funzione di fenditura e il l'add-in "Solver" per Excel e Calc viene utilizzato per l'approssimazione iterativa del modello allo spettro di trasmissione osservato. È molto utile, ma non essenziale, disporre di una "macro" per automatizzare il processo. (Vedere <http://peltiertech.com/Excel/SolverVBA.html#Solver2> per informazioni sulla configurazione delle macro e del solver per la propria versione di Excel).

[TransmissionFittingTemplate.xls \(schermata\)](#) è un template vuoto; è sufficiente inserire i dati nelle celle contrassegnate da uno sfondo grigio: lunghezza d'onda (Colonna A), assorbanza osservata del campione (Colonna B), lo spettro di assorbanza di riferimento ad alta risoluzione (Colonna D), la luce diffusa (A6) e la funzione di fenditura utilizzata per l'assorbanza osservata del campione (**M6-A6**). ([TransmissionFittingTemplateExample.xls \(schermata\)](#)) è lo stesso template con dei dati di esempio inseriti.

[TransmissionFittingDemoGaussian.xls \(schermata\)](#) è una dimostrazione con un picco di assorbimento Gaussiano simulato con una posizione, larghezza e altezza del picco variabili, con in più della luce parassita e del rumore foto-nico, e il rumore del rilevatore, visto da uno spettrometro con funzione di fenditura triangolare. È possibile variare tutti i parametri e confrontare la migliore approssimazione dell'assorbanza con l'altezza del picco reale e all'assorbanza convenzionale $\log(1/T)$.

Tutti questi fogli di calcolo includono una [macro](#), attivata premendo **Ctrl-f**, che utilizza la funzione Solver per eseguire il calcolo iterativo dei minimi quadrati (vedere pagina 308). Tuttavia, se si preferisce non utilizzare le macro, lo si può fare manualmente cliccando sulla scheda **Data, Solver, Solve** e poi **OK**.

[TransmissionFittingCalibrationCurve.xls \(schermata\)](#) è un



foglio di calcolo dimostrativo che include un'altra [macro](#) Excel che costruisce curve di calibrazione confrontando TFit e il metodo convenzionale $\log(1/T)$ per una serie di 9 concentrazioni standard che è possibile specificare. Per creare una curva di calibrazione, si inseriscono le concentrazioni standard in AF10 - AF18 (o si usa semplicemente quelle già presenti, che coprono un intervallo di concentrazione di 10.000 volte da 0,01 a 100), poi si preme **Ctrl-f** per eseguire la macro. In questo foglio di calcolo la macro fa molto di più rispetto all'esempio precedente: scorre automaticamente la prima riga della tabellina in AF10 - AH18, estrae a turno ogni valore di concentrazione, lo inserisce nella cella di concentrazione A6, ricalcola il foglio di calcolo, prende l'assorbanza convenzionale risultante (cella J6) e la inserisce come prima ipotesi nella cella I6, richiama il Solver per calcolare l'assorbanza più adatta per quell'altezza del picco, inserisce sia l'assorbanza convenzionale che l'assorbanza più adatta nella tabella in AF10 - AH18, poi passa alla concentrazione successiva e si ripete per ogni valore di concentrazione. Poi costruisce e disegna la curva di calibrazione log-log (monstrata a lato) sia per il metodo TFit (punti blu) che per quello convenzionale (punti rossi) e calcola l'equazione della linea di tendenza e il valore di R^2 per il metodo TFit, nell'angolo in alto a destra del grafico. Ogni volta che si preme **Ctrl-f** viene ricostruita l'intera curva di calibrazione con un'altra serie di campioni di rumore casuali. (Nota: è anche possibile utilizzare questo spreadsheet per confrontare la precisione e la riproducibilità dei due metodi inserendo la *stessa* concentrazione 9 volte in AF10 - AF18. Il risultato dovrebbe idealmente essere una linea retta piatta con pendenza zero).

Implementazione Matlab/Octave: La funzione [fitM.m](#)

```
function err = fitM(lam, yobsd, Spectra, InstFun, StrayLight)
```

[fitM](#) è una funzione di approssimazione per il metodo Tfit, da utilizzare con la funzione, Matlab o di [Octave](#), [fminsearch](#). Gli argomenti di input per fitM sono:

absorbance= vettore delle assorbanze calcolate per approssimare al meglio lo spettro di trasmissione.

yobsd = spettro di trasmissione osservato del campione di miscela sull'intervallo spettrale (vettore colonna)

Spectra = spettri di riferimento per ciascun componente, sullo stesso intervallo spettrale, una colonna/componente, normalizzati a 1,00.

InstFun = Funzione dello strumento centrato sullo zero o funzione fenditura (vettore colonna)

StrayLight = luce parassita frazionaria (vettore scalare o colonna, se varia con la lunghezza d'onda)
Nota: **yobsd**, **Spectra** e **InstFun** devono avere lo stesso numero di righe (lunghezze d'onda).

Spectra deve avere una colonna per ogni componente assorbente.

Uso tipico:

```
absorbance = fminsearch(@(lambda)(fitM(lambda, yobsd, TrueSpectrum,  
InstFun, StrayLight)), start);
```

dove "start" è la prima ipotesi (o le ipotesi) per la/e assorbanza/e e lo/gli analita/i; è conveniente utilizzare la stima convenzionale $\log_{10}(I_0/I)$ dell'assorbanza per start. Gli altri argomenti (descritti sopra) vengono passati a FitM. In questo esempio, la funzione fminsearch restituisce il valore di assorbanza che sarebbe stato misurato in assenza di luce diffusa e di errori di luce policromatica (che è un valore singolo o un vettore di assorbanze, se si tratta di un'analisi multicomponente). L'assorbanza può quindi essere convertita in concentrazione mediante una qualsiasi delle [soltre procedure](#) (Legge di Beer, [standard esterni](#), [aggiunta di standard](#), ecc.).

Ecco un esempio numerico molto semplice, per una **misura mono-componente** dove la vera assorbanza è 1.00, utilizzando solo uno *spettro di 4 punti* per semplicità illustrativa (naturalmente, i sistemi con array di rilevatori acquisirebbero *molte* più lunghezze d'onda di questa, ma il principio è lo stesso). In questo caso la larghezza dello strumento ("InstFun") è *due volte* la larghezza di assorbimento, la luce diffusa è costante a 0.01 (1% relativo). La convenzionale stima dell'assorbanza a lunghezza d'onda singola è troppo bassa:

$\log_{10}(1/3.8696)=0.4123$. Al contrario, il metodo TFit che utilizza fitM è impostato in questo modo:

```
yobsd=[0.56529 0.38696 0.56529 0.73496]';

TrueSpectrum=[0.2 1 0.2 0.058824]';

InstFun=[1 0.5 0.0625 0.5]';

straylight=.01;

start=.4;

absorbance=fminsearch(@(lambda) (fitM(lambda,yobsd,TrueSpectrum,InstFun,
straylight)),start)
```

Ciò restituisce il valore di assorbanza corretto di 1.000. Il valore di "start" non è critico in questo caso e può essere praticamente qualsiasi valore si voglia, ma mi piace usare l'assorbanza convenzionale $\log_{10}(I_0/I)$, che è facilmente calcolabile e utile come *stima approssimativa ma ragionevole* del valore corretto.

Per una **misura multi-componente**, l'unica differenza è che la variabile "TrueSpectrum" è una *matrice* anziché un *vettore*, con una colonna per ogni componente assorbente. La risultante "assorbanza" sarebbe quindi un *vettore* anziché un *singolo numero*, con un valore di assorbanza per ogni componente. (Vedere [TFit3.m](#) di seguito per un esempio di una miscela a 3 componenti).

I metodi iterativi dei minimi quadrati sono generalmente considerati più difficili e meno affidabili dei più comuni metodi di regressione multilineare non iterativi. Questo può essere vero se c'è più di una variabile non lineare che dev'essere iterata, specialmente se queste variabili sono correlate. Tuttavia, nel metodo TFit, esiste solo *una* variabile iterata (assorbanza) per ogni componente misurato e le prime ipotesi ragionevoli sono prontamente disponibili dal calcolo dell'assorbanza a lunghezza d'onda singola convenzionale o dai metodi standard della regressione a lunghezza d'onda multipla. Di conseguenza, il metodo iterativo dei minimi quadrati funziona molto bene in questo caso. L'espressione di assorbanza fornita sopra per il metodo TFit può essere confrontata con quella per *il metodo di regressione ponderata*:

```
absorbance=([weight weight].*[Background RefSpec])\(-
log10(yobsd).*weight)
```

dove RefSpec è la matrice degli spettri di riferimento di tutti i componenti puri. Si può vedere che, oltre alla RefSpec e allo spettro di trasmissione osservato (yobsd), il metodo TFit richiede anche una misura della funzione Strumento (passa banda spettrale) e della luce parassita (che il metodo

della regressione lineare assume essere zero), ma queste sono caratteristiche dello *spettrometro* e devono essere eseguite solo una volta per un dato spettrometro. Infine, sebbene il metodo TFit faccia lavorare di più il *computer*, il tempo di calcolo su un tipico personal computer di laboratorio è solo una frazione di secondo (circa 25 μ sec per ogni punto spettrale per ogni componente analizzato), utilizzando Matlab come ambiente di calcolo. Il costo dell'hardware computazionale non dev'essere gravoso; il metodo può essere eseguito in Python, o in *Octave* (con una certa perdita di velocità), o anche su un [computer single-board da \\$35](#) (vedere pag. 335) che viene venduto con Python installato.

Funzione demo per Octave o Matlab

La funzione [tfit.m](#) è una funzione dimostrativa Matlab a riga di comando autonoma che confronta il metodo TFit con i metodi a lunghezza d'onda singola (SingleW), la regressione semplice (SimpleR) e la regressione pesata (WeightR). La sintassi è **tfit(absorbance)**, dove 'absorbance' è la *vera assorbanza del picco sottostante* (La A vera) di un assorbente con un profilo spettrale Lorentziano di larghezza 'width' (riga 29), misurato con uno spettrometro con una banda passante spettrale Gaussiana di larghezza 'InstWidth' (riga 30), livello di luce parassita frazionaria non assorbita di 'straylight' (riga 32), livello di rumore fotonico di 'noise' (riga 31) e uno spostamento casuale Io di 'IzeroShift' (riga 33). Disegna i profili spettrali e stampa le assorbanze misurate di ciascun metodo nella finestra di comando. Esempi:

```
>> tfit(1)

width = 10
InstWidth = 20
noise = 0.01
straylight = 0.01
IzeroShift = 0.01

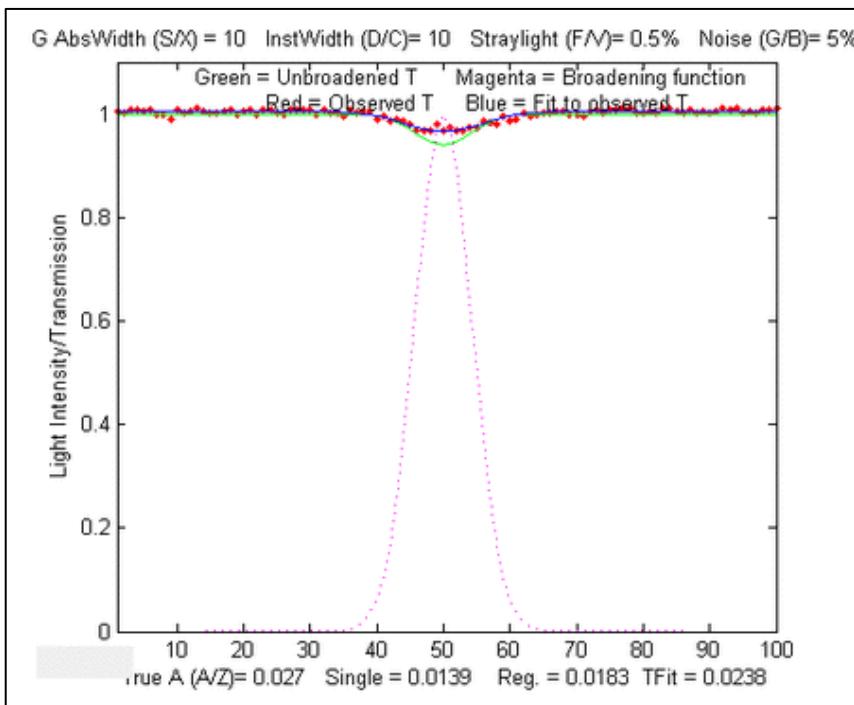
True A SingleW SimpleR WeightR TFit
1  0.38292  0.54536  0.86839  1.0002
>> tfit(10)
10  1.4858  2.2244  9.5123  9.9979

>> tfit(100)
100  2.0011  3.6962  57.123  99.951

>> tfit(200)
200  2.0049  3.7836  56.137  200.01

>> tfit(.001)
0.001  0.00327    0.00633    0.000520  0.000976
```

TFitDemo.m: Demo interattiva per il metodo Tfit



[TFitDemo.m](#) è un esploratore interattivo a riga di comando per il metodo Tfit (per Matlab o Octave), applicato alla misura di un singolo componente con picco di assorbimento Lorentziano (o Gaussiano), con controlli che consentono di regolare l'assorbanza reale (Peak A), l'ampiezza spettrale del picco di assorbimento (AbsWidth), la larghezza spettrale della funzione strumento (InstWidth), la luce diffusa e il livello di rumore fotonico (Noise) osservando continuamente gli effetti graficamente e numericamente. Se l'animazione non è visibile, cliccare su [questo link](#)). La demo simula l'effetto del rumore dei fotoni, della luce diffusa non assorbita e delle variazioni casuali dell'intensità dello sfondo (sfarfallio della sorgente luminosa). Confronta le assorbanze osservate mediante la regressione multilineare ponderata a lunghezza d'onda singola (talvolta chiamata dei minimi quadrati classici nella letteratura chemiometrica) e i metodi TFit. Per eseguire questo file, click destro su [TFitDemo.m](#) click su "Save link as...", salvarlo in una cartelle nel path di ricerca di Matlab, poi digitare "TFitDemo" nel propt dei comandi di Matlab. Con la finestra della figura in alto sullo schermo, premere **K** per ottenere un elenco delle funzioni da tastiera. La versione 2.1, Novembre 2011, aggiunge il calcolo del rapporto segnale-rumore e utilizza il tasto **W** per passare dalla visualizzazione Trasmissione alla Assorbanza.

Nell'esempio mostrato sopra, l'assorbanza del picco reale è mostrata variando da 0,0027 a 57 unità di assorbanza, le larghezze di assorbimento e le larghezze delle funzioni dello strumento sono uguali ([ne risulta un rapporto segnale/rumore ottimale](#)), la luce parassita non assorbita è dello 0.5% e il rumore fotonico è del 5%. (A scopo dimostrativo, le 6 forme di picco di assorbimento più basse sono *Gaussiane* e le 3 più alte sono *Lorentziane*). I risultati sotto i grafici mostrano che, ad ogni assorbanza e per una forma di picco Gaussiana o Lorentziana, il metodo TFit fornisce misure molto più accurate rispetto al metodo a lunghezza d'onda singola o alla regressione multilineare ponderata.

KEYBOARD COMMANDS

Peak shape....Q Toggles between Gaussian and Lorentzian
del picco di assorbimento
True peak A...A/Z True absorbance of analyte at peak center, without
instrumental broadening, stray light, or noise.
AbsWidth.....S/X Width of the absorption peak

```

SlitWidth.....D/C  Width of instrument function (spectral bandpass)
Straylight....F/V  Fractional unabsorbed stray light.
Noise.....G/B  Random noise level
Re-measure....Spacebar Re-measure signal with another random noise sample
Switch mode...W  Switch between Transmission and Absorbance display
Statistics....Tab Prints table of statistics of 50 repeats
Cal. Curve....M  Displays analytical calibration curve in Fig. window 2
Keys.....K  Print this list of keyboard commands

```

Perché il rumore sul grafico cambia se si cambia la funzione dello strumento (larghezza della fenditura o InstWidth)? In un comune spettrometro ad assorbimento, utilizzando una sorgente di luce continua e uno spettrometro dispersivo, ci sono *due* aperture regolabili o fenditure, una prima dell'elemento di dispersione, che controlla l'ampiezza fisica del fascio di luce, e una dopo, che controlla la gamma di lunghezze d'onda della luce misurata (e che, in un array di rivelatori, è controllato dal software che legge gli elementi dell'array). La larghezza di banda spettrale di uno spettrometro ("InstWidth") viene modificata cambiando entrambi, il che influenza anche sull'intensità della luce misurata dal rilevatore e quindi sul [rapporto segnale/rumore](#). Pertanto, in tutti questi programmi, quando si modifica *InstWidth*, il rumore fotonico viene automaticamente modificato di conseguenza proprio come farebbe in un vero spettrofotometro. Il rumore del rilevatore, al contrario, rimane lo stesso. Si sta anche assumendo che il rilevatore non si saturi o si sovraccarichi se la larghezza della fenditura viene aumentata.

Statistiche dei metodi a confronto ([TFitStats.m](#), per Matlab o Octave)

Si tratta di un semplice script che calcola le statistiche del metodo TFit rispetto ai metodi a lunghezza d'onda singola (SingleW), regressione semplice (SimpleR) e regressione ponderata (WeightR). Simula il rumore fotonico, la luce dispersa non assorbita e le variazioni casuali dell'intensità del background. Stima la precisione e l'accuratezza dei quattro metodi ripetendo i calcoli 50 volte con diversi campioni di rumore casuali. Calcola la media, la deviazione standard percentuale relativa e la deviazione percentuale relativa dall'assorbanza reale. È possibile modificare facilmente i parametri nelle righe 19 - 26. Il programma visualizza i risultati nella finestra dei comandi MATLAB.

Nell'output di esempio mostrato di seguito, il programma ha calcolato i risultati per le assorbanze reali di 0,001 e 100, dimostrando che l'accuratezza e la precisione del metodo TFit sono superiori agli altri metodi su un intervallo di 10.000 volte.

Questa funzione statistica è inclusa come comando da tastiera (tasto **Tab**) in [TFitDemo.m](#).

Risultati per assorbanze reali di 0,001

True A	SingleW	SimpleR	WeightR	TFit
MeanResult =				
0.0010	0.0004	0.0005	0.0006	0.0010
PercentRelativeStandardDeviation =				
1.0e+003 *				
0.0000	1.0318	1.4230	0.0152	0.0140
PercentAccuracy =				
0.0000	-60.1090	-45.1035	-38.6300	0.4898

Risultati per assorbanze reali di 100

```
MeanResult =
100.0000   2.0038   3.7013   57.1530   99.9967

PercentRelativeStandardDeviation =
0   0.2252   0.2318   0.0784   0.0682

PercentAccuracy =
0  -97.9962  -96.2987  -42.8470  -0.0033
```

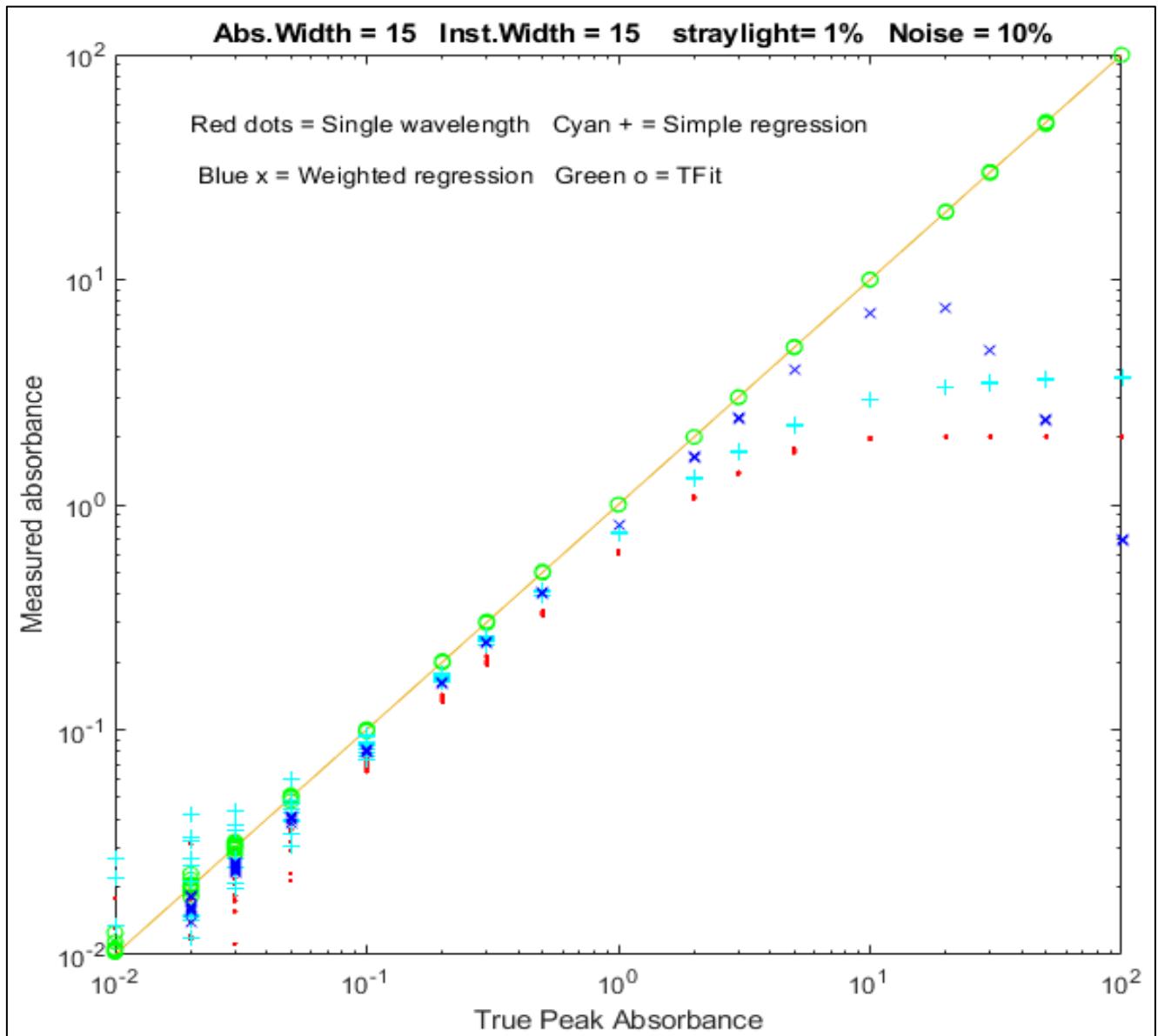
Come si vede, il metodo Tfit offre una maggiore accuratezza e precisione.

Confronto delle curve analitiche ([TFitCalDemo.m](#), per Matlab o Octave)

TFitDemo.m è una funzione dimostrativa che confronta le curve analitiche per la lunghezza d'onda singola, la regressione semplice, la regressione ponderata e il metodo TFit su qualsiasi intervallo di assorbanza specificato (dal vettore "assorbancelist" nella riga 20). Simula il rumore fotonico, la luce dispersa non assorbita e le variazioni casuali dell'intensità del background. Disegna un grafico a dispersione log-log con ciascuna misura ripetuta tracciata come un punto separato (in modo da poter vedere la dispersione dei punti alle basse assorbanze). I parametri possono essere modificati nelle righe 20 - 27.

Nel risultato del campione mostrato di seguito, le curve analitiche per i quattro metodi vengono calcolate su un intervallo di 10.000 volte, fino a un'assorbanza di picco di 100, dimostrando che il metodo TFit (i cerchi verdi) è molto più lineare nell'intero range rispetto ai metodi a lunghezza d'onda singola, regressione semplice o regressione ponderata. *L'ampio intervallo di linearità di Tfit è particolarmente importante nei laboratori regolamentati dove sono sconsigliate tutte le approssimazioni ad esclusione di quella dei minimi quadrati.*

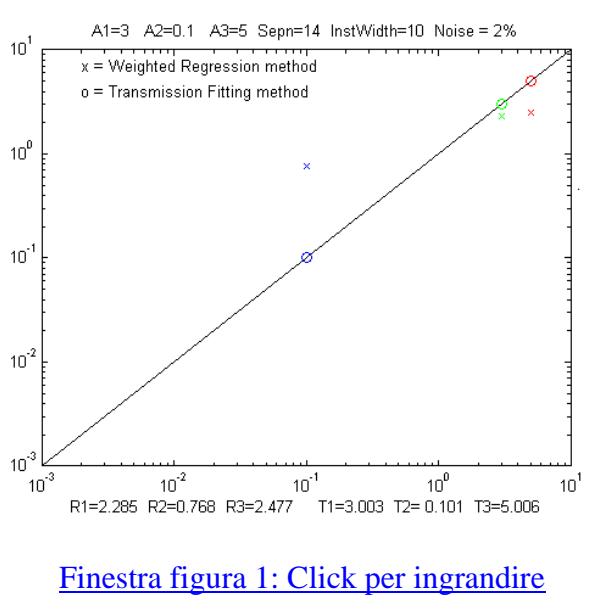
Questa funzione della curva di calibrazione è inclusa come comando da tastiera (tasto **M**) in [TFitDemo.m](#).



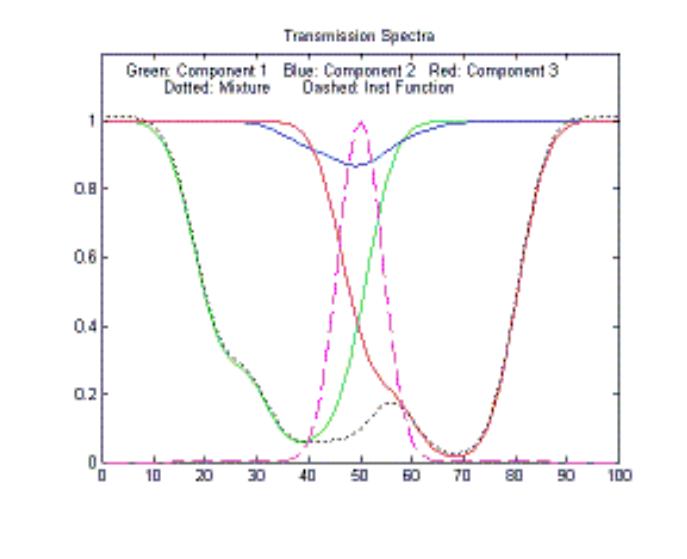
Confronto delle curve analitiche simulate per i metodi a lunghezza d'onda singola, regressione semplice, regressione ponderata e TFit su un intervallo di assorbanza di 10.000 volte, creato da [TFitCalDemo.m](#).

Applicazione su una miscela a tre componenti

L'applicazione della spettroscopia di assorbimento a *miscele* di componenti assorbenti richiede l'adozione di un ulteriore presupposto: quello dell'additività delle assorbanze, nel senso che l'assorbanza misurata di una miscela è uguale alla somma delle assorbanze dei componenti separati. In pratica, ciò richiede che gli assorbenti non interagiscano chimicamente; vale a dire che non reagiscono con se stessi o con gli altri componenti né modificano alcuna proprietà della soluzione (p.es., pH, forza ionica, densità, ecc.) che potrebbe influenzare gli spettri degli altri componenti. Questi requisiti si applicano sia ai metodi multicomponente convenzionali (pag. 179) che al metodo T-Fit.



[Finestra figura 1: Click per ingrandire](#)



[Finestra figura 2: Click per ingrandire](#)

[TFit3.m](#) è una dimostrazione Matlab/Octave del metodo T-Fit applicato alla [spettroscopia di assorbimento multicomponente](#) di una miscela di tre assorbenti. I parametri regolabili sono: le assorbanze dei tre componenti (A_1 , A_2 e A_3), la sovrapposizione spettrale tra gli spettri dei componenti ("Sepn"), l'ampiezza della funzione dello strumento ("InstWidth") e il livello di rumore ("Noise"). TFit3 confronta la misura quantitativa mediante regressione ponderata e i metodi TFit e simula il rumore dei fotoni, la luce parassita non assorbita e le variazioni casuali dell'intensità del background. Nota: Dopo aver eseguito questo file m, posizionare le finestre "Figure No. 1" e "Figure No.2" fianco a fianco in modo che non si sovrappongano. La Figure 1 mostra un grafico a dispersione log-log delle assorbanze reali rispetto a quelle misurate, con i tre assorbenti disegnati in colori e simboli diversi. La finestra 'Figure 2' mostra gli spettri di trasmissione dei tre assorbenti disegnati con i colori corrispondenti. Quando si utilizzano i comandi da tastiera (sotto) nella finestra "Figure No. 1", entrambi i grafici cambiano di conseguenza.

Nel calcolo del campione mostrato sopra, la componente 2 (il piccolo avallamento mostrato in blu) è quasi completamente sepolta dalle bande di assorbimento più forti delle componenti 1 e 3 su entrambi i lati, dando un'assorbanza molto più debole (0,1) rispetto alle altre due componenti (3 e 5, rispettivamente). Anche in questo caso difficile il metodo TFit fornisce un risultato ($T_2 = 0.101$) entro l'1% del valore corretto ($A_2 = 0.1$). In effetti, nella maggior parte delle combinazioni delle tre concentrazioni, il metodo TFit funziona meglio (anche se, ovviamente, niente funziona se la differenza spettrale tra i componenti è troppo piccola).

Nota: in questo programma, come in tutto quanto sopra, quando si modifica InstWidth, il rumore fotonico cambia automaticamente di conseguenza proprio come farebbe in un vero spettrofotometro dispersivo a fenditura variabile con una sorgente di luce bianca. Si può anche scaricare la [versione unica più recente gestita da tastiera](#) ([TFit3Demo.m](#)) che funziona in Matlab o nelle recenti versioni di Octave:

TASTO COMANDI (Matlab e Octave)

A1 **A/Z** Aumenta/diminuisce l'assorbanza reale della componente 1

A2 **S/X** Aumenta/diminuisce l'assorbanza reale della componente 2

A3 **D/C** Aumenta/diminuisce l'assorbanza reale della componente 3

Sepn **F/V** Aumenta/diminuisce la separazione spettrale delle

componenti

InstWidth **G/B** Aumenta/diminuisce la larghezza della funzione strumento (passa-banda spettrale)

Noise **H/N** Aumenta/diminuisce il livello del rumore casuale quando

InstWidth = 1

Peak shape **Q** Commuta tra il profilo Gaussiano e Lorentziano del picco di assorbimento

Table **Tab** Stampa la tabella dei risultati

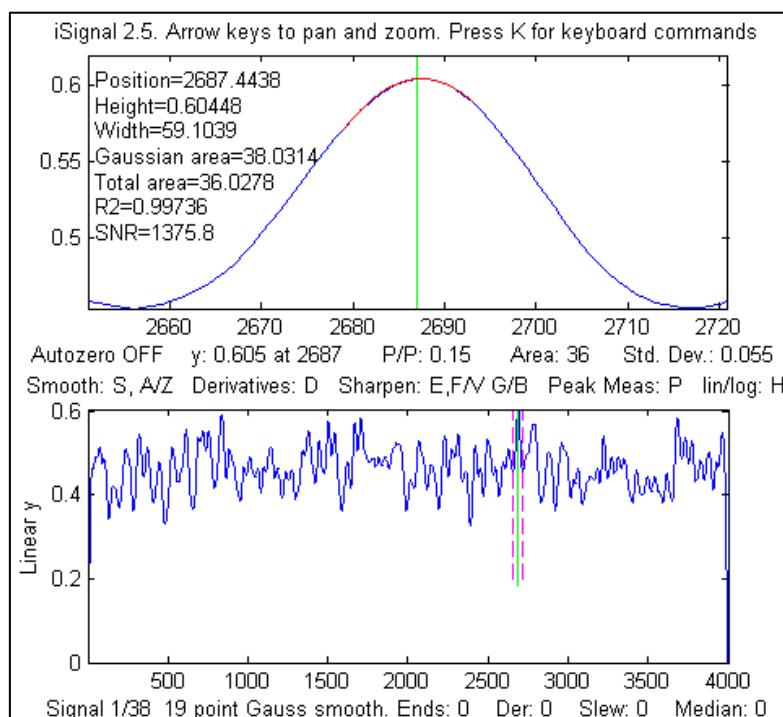
K Print this list of keyboard commands

Esempio di tabella dei risultati tipici (visualizzata premendo il tasto **Tab**):

```
-----  
True Weighted TFit  
  
Absorbance Regression method  
  
Component 1 3           2.06      3.001  
Component 2 0.1          0.4316    0.09829  
Component 3 5           2.464     4.998
```

Tutorial, Casi di Studio e Simulazioni.

Il rumore con smoothing può essere confuso con l'effettivo segnale?



Ecco due esempi che dimostrano che la risposta a questa domanda è sì. Il primo esempio è illustrato a sinistra. Questo mostra iSignal (pagina 366) che visualizza un segnale di 4000 punti generato dal computer costituito da puro rumore casuale filtrato con uno smoothing P-spline da 19 punti. La finestra superiore mostra una piccola porzione di questo segnale che sembra un picco Gaussiano con un SNR calcolato superiore a

1000. Solo guardando l'intero segnale (finestra inferiore) si vede la vera immagine; quel "picco" è solo una parte del rumore, filtrato con smoothing per apparire piacevole. Non ci si faccia ingannare.

Il secondo esempio è una semplice serie di tre comandi Matlab che utilizzano la funzione 'randn' per generare un set di dati di 10000 punti contenente solo rumore bianco normalmente distribuito.

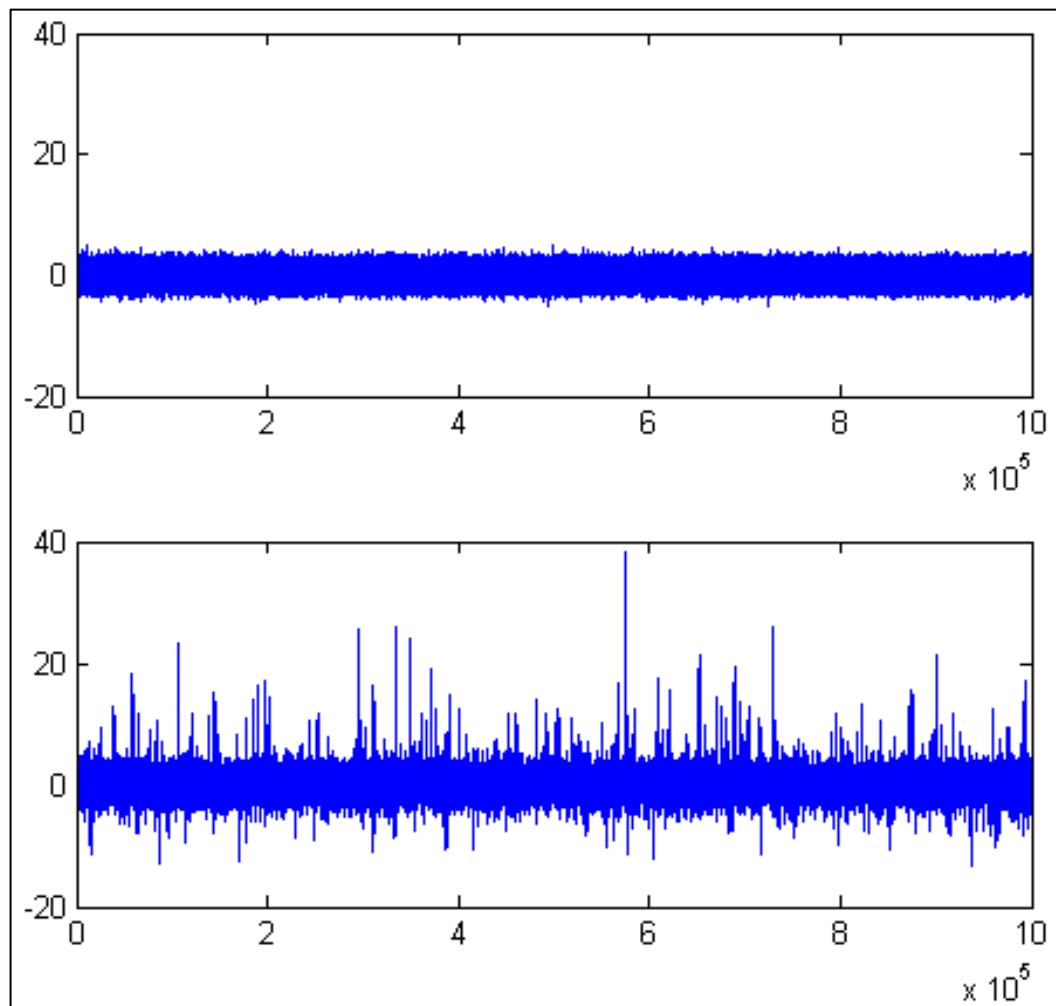
Quindi utilizza 'fastsmooth.m' lo smoothing del rumore, ottenendo un 'segnale' con una deviazione standard di circa 0,3 e un valore massimo intorno a 1,0. Il segnale viene quindi inviato a *iPeak* (pagina 245). Se i criteri di rilevamento dei picchi (ad esempio, AmpThreshold e SmoothWidth) sono impostati su un valore troppo basso, verranno trovati molti picchi. Ma impostando AmpThreshold a 3 volte la deviazione standard ($3 \times 0,3 = 0,9$) si ridurrà notevolmente l'incidenza di questi falsi picchi.

```
>> noise=randn(1,10000);  
  
>> signal=fastsmooth(noise,13);  
  
>> ipeak([1:10000;signal],0,0.6,1e-006,17,17)
```

La funzione di identificazione del picco, che li identifica in base alla loro esatta posizione sull'asse x e a una tabella memorizzata delle posizioni dei picchi precedentemente identificati, è *ancora meno* probabile che venga ingannata dal rumore casuale, perché oltre ai criteri di rilevamento dell'algoritmo findpeaks, qualsiasi picco rilevato deve *anche* corrispondere strettamente a una posizione nella tabella dei picchi noti.

Segnale o Rumore?

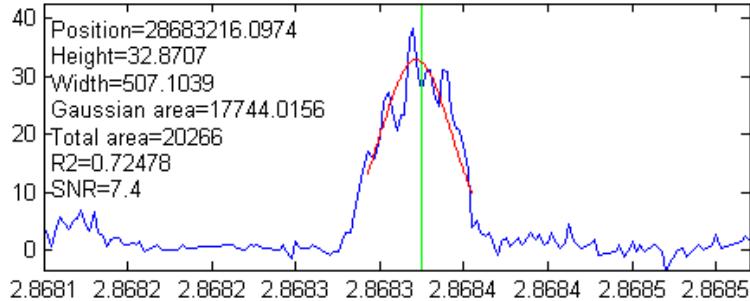
Il segnale sperimentale del cliente in questo caso di studio era insolito perché non sembrava un tipico segnale una volta disegnato; in effetti, a prima vista assomigliava molto al rumore. La figura seguente confronta il segnale sperimentale originale (in basso) con lo stesso numero di punti di rumore bianco distribuito normalmente (in alto) con una media di zero e una deviazione standard di 1.0 (ottenuta dalla funzione Matlab/Octave 'randn').



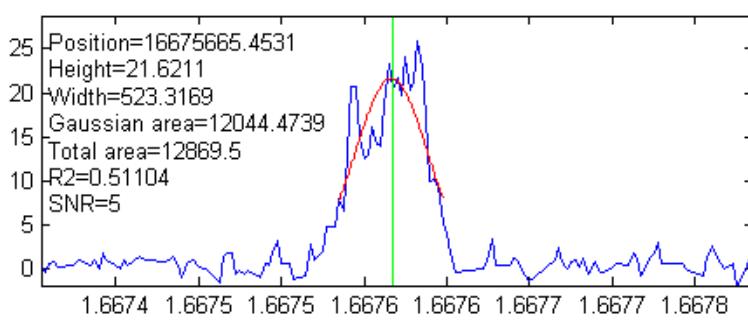
Come si vede, la differenza principale è che il segnale ha "picchi" più grandi, soprattutto in direzione positiva. Questa differenza è evidente quando si guardano le [statistiche descrittive](#) del segnale e della funzione randn:

STATISTICHE DESCRIPTIVE	Segnale originale	rumore casuale (funzione randn)
Media	0.4	0
Massimo	38	circa 5 - 6
Deviazione Standard (STD)	1.05	1.0
Scarto Inter-Quartile (IQR)	1.04	1.3489
Curtosi	38	3
Indice di asimmetria	1.64	0

Si può vedere che le *deviazioni standard* di questi due sono quasi le stesse, ma le altre statistiche (specialmente la [curtosi e la simmetria](#)) indicano che la [distribuzione della probabilità](#) del segnale è *tutt'altro che normale*; nel segnale ci sono molti più picchi positivi del previsto per il rumore puro. La maggior parte di questi si è rivelata essere i picchi di interesse per questo segnale; sembrano spi-

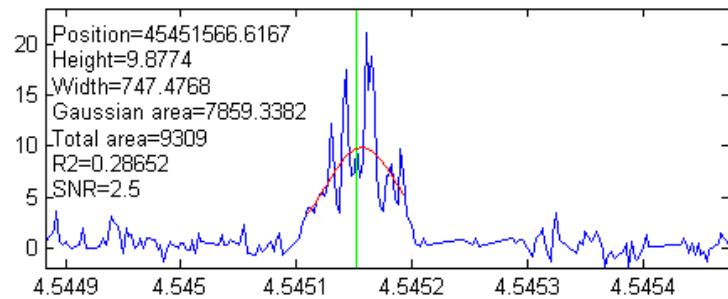


ke solo perché la lunghezza del segnale (più di 1,000,000 di punti) fa sì che i picchi vengano compressi in un pixel dello schermo o meno quando l'intero segnale viene disegnato sullo schermo. Nelle figure a lato, [iSignal](#) (pagina 366) viene usato per "ingrandire" alcuni di questi picchi più grandi (utilizzando i tasti frecce). I picchi sono molto poca separati (da una media di 1000 semi-larghezze tra i picchi) e sono ben al di sopra del livello del rumore di fondo (che ha una deviazione standard di circa 0,9 in tutto il segnale).



Il ricercatore che ha ottenuto questo segnale ha detto che un picco 'buono' era 'a forma di campana', con un'ampiezza superiore a 5 e una larghezza di 500-1000 unità dell'asse x. Quindi ciò significa

che ci si può aspettare che il rapporto segnale/rumore del background sia almeno $5/0,9 = 5,5$. Si possono vedere nei tre picchi di esempio a lato che le larghezze soddisfano effettivamente quelle aspettative. L'intervallo tra i punti adiacenti dell'asse x è 25, quindi significa che ci si può aspettare che i picchi abbiano una larghezza compresa tra 20 e 40 punti. Sulla base di ciò, ci si aspetta che le posizioni, le altezze e le larghezze dei picchi possano essere misurate abbastanza accuratamente usando i metodi dei minimi quadrati (che riducono l'incertezza dei parametri misurati di circa la radice quadrata del numero di punti utilizzati - circa di un fattore 5 in questo caso). Tuttavia, *il rumore sembra essere dipendente dal segnale*; cioè il rumore sulla sommità dei picchi è nettamente maggiore di quello sulla linea di base. Il risultato è che l'effettivo rapporto segnale/rumore (S/N) della misura del parametro per i picchi più grandi non sarà buona come ci si potrebbe aspettare in base al rapporto tra l'altezza del picco e il rumore sul background. Molto probabilmente, il rumore totale in questo segnale è la somma di due componenti principali, una con una deviazione standard fissa di 0,9 e l'altra approssimativamente uguale al 10% dell'altezza del picco.



Per automatizzare il rilevamento di un gran numero di picchi, si possono utilizzare le funzioni [findpeaksG](#) e [iPeak](#) (pag. 405). Valori ragionevoli degli argomenti di input *AmplitudeThreshold*, *SlopeThreshold*, *SmoothWidth* e *FitWidth* per queste funzioni possono essere stimati in base alle presunte altezza (5) ed ampiezza (da 20 a 40 punti) dei picchi "buoni". Ad esempio, utilizzando *AmplitudeThreshold=5*, *SlopeThreshold=.001*, *SmoothWidth=25* e *FitWidth=25*, queste funzioni rilevano e misurano 76 picchi al di sopra di un'ampiezza di 5 e con una larghezza media di 523. La funzione interattiva [iPeak](#) (pagina 405) è particolarmente comoda per esplorare l'effetto di questi parametri di rilevamento e per ispezionare graficamente i picchi trovati. Idealmente, l'obiettivo è

le presunte altezza (5) ed ampiezza (da 20 a 40 punti) dei picchi "buoni". Ad esempio, utilizzando *AmplitudeThreshold=5*, *SlopeThreshold=.001*, *SmoothWidth=25* e *FitWidth=25*, queste funzioni rilevano e misurano 76 picchi al di sopra di un'ampiezza di 5 e con una larghezza media di 523. La funzione interattiva [iPeak](#) (pagina 405) è particolarmente comoda per esplorare l'effetto di questi parametri di rilevamento e per ispezionare graficamente i picchi trovati. Idealmente, l'obiettivo è

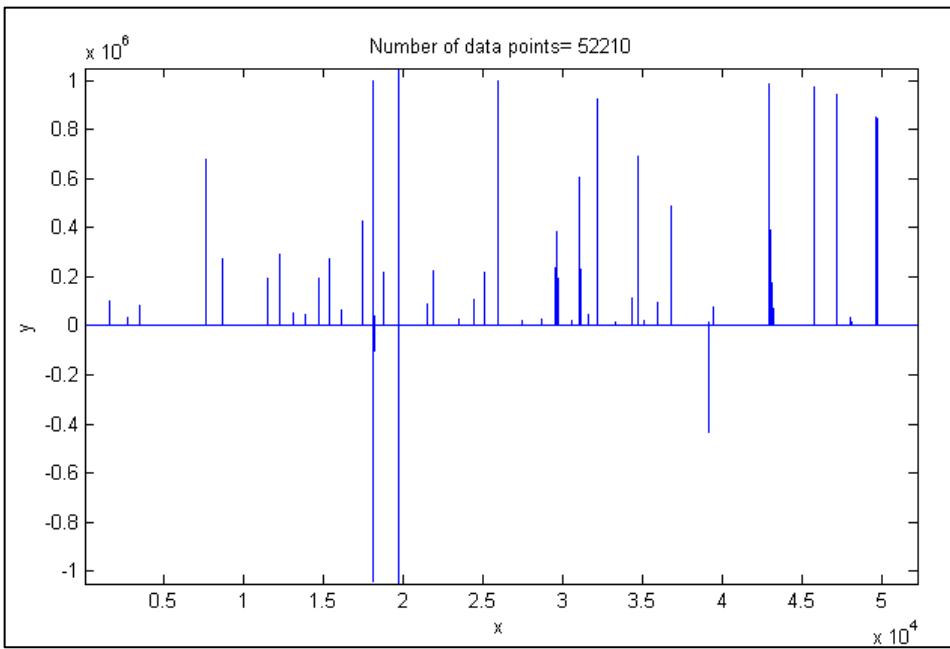
trovare un set di argomenti di rilevamento che trovano e misurano accuratamente tutti i picchi che si considererebbero "buoni" saltando quelli "cattivi". Ma in realtà i criteri per i picchi buoni e cattivi sono almeno in parte soggettivi, quindi di solito è meglio sbagliare per eccesso di cautela ed evitare di saltare i picchi "buoni" con il rischio di includere alcuni picchi "cattivi" nel mix, che si possono eliminare manualmente in base a posizione, altezza, larghezza o aspetto insolito.

Ovviamente, ci si deve aspettare che i valori di posizione, altezza e larghezza del picco forniti dalle funzioni [findpeaksG](#) o [iPeak](#) saranno solo approssimativi e varieranno a seconda sull'esatta impostazione degli argomenti di rilevamento; più rumorosi sono i dati, maggiore è l'incertezza nei parametri. A questo proposito, le funzioni di approssimazione del picco [peakfit.m](#) e [ipf.m](#) di solito danno risultati più accurati, perché fanno uso di *tutti* i dati nel picco, non solo la cima come in `findpeaksG` e in `iPeak`. Ad esempio, confrontare i risultati del picco vicino a $x = 3035200$ misurato con [iPeak \(cliccare per visualizzare\)](#) e con [peakfit \(cliccare per visualizzare\)](#). Inoltre, le funzioni di approssimazione dei picchi sono migliori per gestire i picchi sovrapposti e per stimare l'incertezza dei parametri, utilizzando le opzioni [bootstrap](#) di queste funzioni. Ad esempio, il picco più grande in questo segnale ha una posizione sull'asse x di $2.8683\text{e}+007$, un'altezza di 32 e una larghezza di 500. Il metodo bootstrap determina che le deviazioni standard sono rispettivamente 4, 0.92 e 9.3.

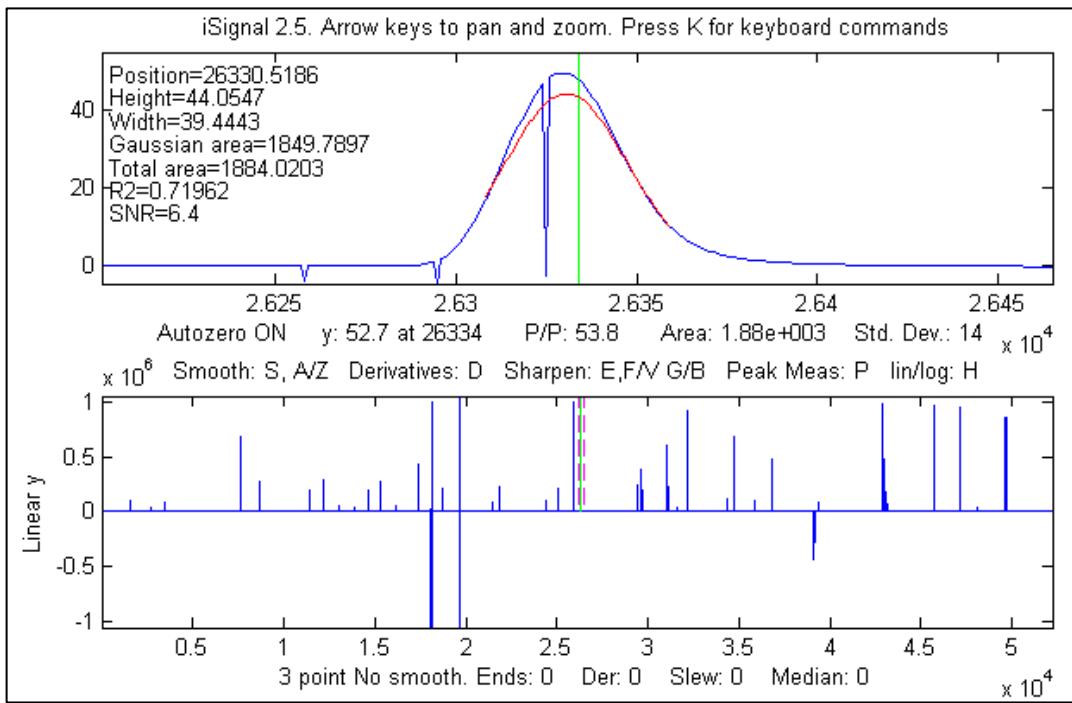
Poiché il segnale nel caso di studio era così grande (oltre 1.000.000 di punti), i programmi interattivi come [iPeak](#), [iSignal](#) e [ipf](#) potrebbe risultare lenti nel funzionamento, specialmente se il computer non è veloce dal punto di vista computazionale o grafico. Se questo è un problema serio, potrebbe essere meglio suddividere il segnale in due o più segmenti e trattare ogni segmento separatamente, quindi combinare i risultati. In alternativa, è possibile utilizzare la funzione [condense](#) per calcolare la media dell'intero segnale in un numero inferiore di punti di un fattore 2 o 3 (a rischio di ridurre leggermente le altezze dei picchi e aumentare le ampiezze dei picchi) quindi si dovrebbero ridurre `SmoothWidth` e `FitWidth` dello stesso fattore per compensare il numero ridotto di punti sui picchi. Eseguire [testcondense.m](#) per una dimostrazione della funzione 'condense'.

Il tesoro sepolto

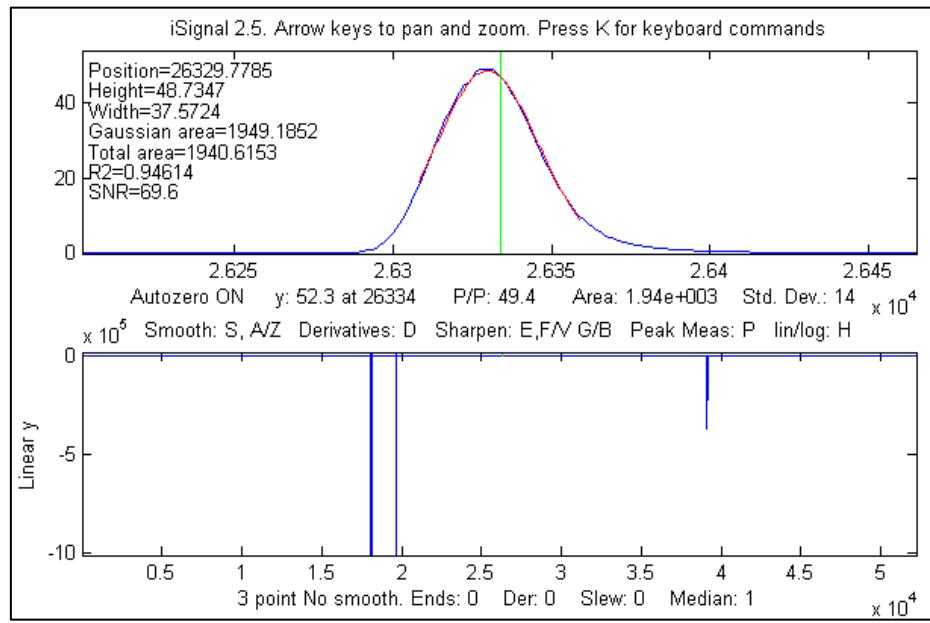
Il segnale sperimentale in questo caso di studio presentava diversi spike stretti che superavano una linea di base apparentemente piatta.



Utilizzando [iSignal](#) (pagina 366) per indagare sul segnale, si è scoperto che i picchi positivi visibili erano *punti singoli* di ampiezza molto grande, mentre le regioni *tra* gli spike non erano realmente piatte ma contenevano molti picchi a forma di campana che erano tanto più piccoli (di un fattore 1000) da non essere subito visibili. Ad esempio, utilizzando [iSignal](#) per ingrandire la regione intorno a $x=26300$, si può vedere uno di quei picchi a forma di campana con un piccolo spike negativo a punto singolo in prossimità del suo apice.

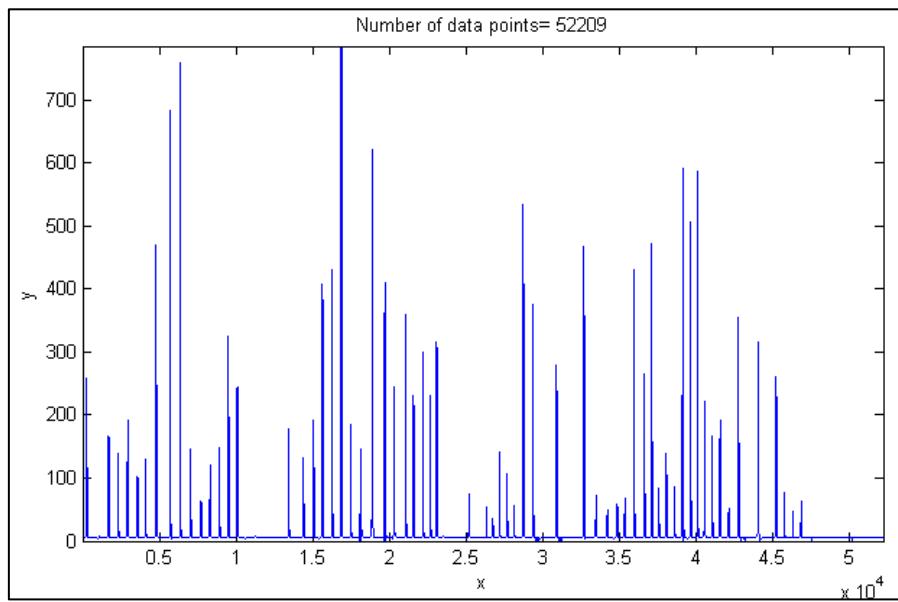


La maggior parte dei picchi in questo segnale aveva degli spike stretti come questo; tali artefatti sono comuni in alcuni segnali sperimentali. Sono facili da eliminare utilizzando un [filtro mediano](#). La funzione [iSignal](#) (pagina 366) ha questo filtro, attivato dal tasto “M”. Il risultato (nella pagina successiva) mostra che gli spike a punto singolo sono stati eliminati, con un effetto minimo sul carattere del picco a campana.

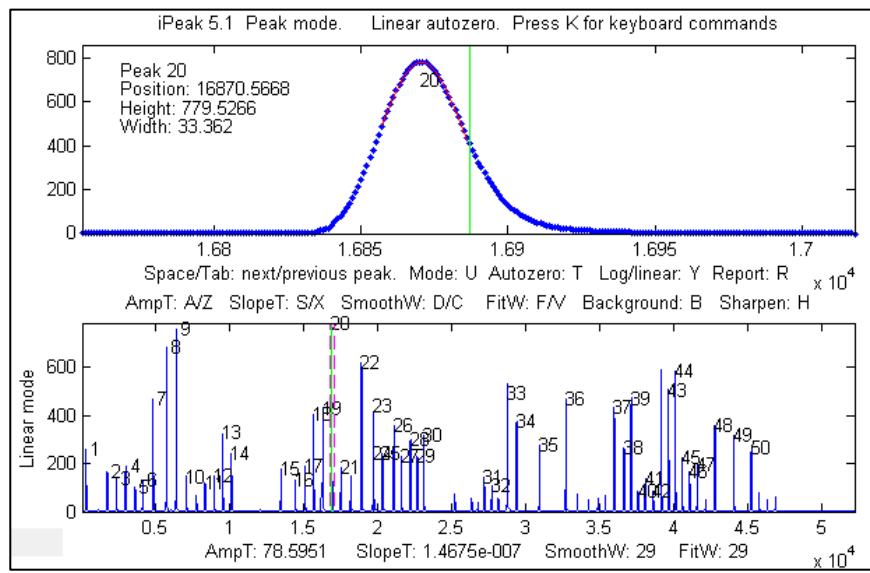


Altri tipi di filtri, come la maggior parte delle forme di [smoothing](#) (pag. 38), sarebbero molto meno efficaci di un filtro mediano per questo tipo di artefatto e distorcono i picchi.

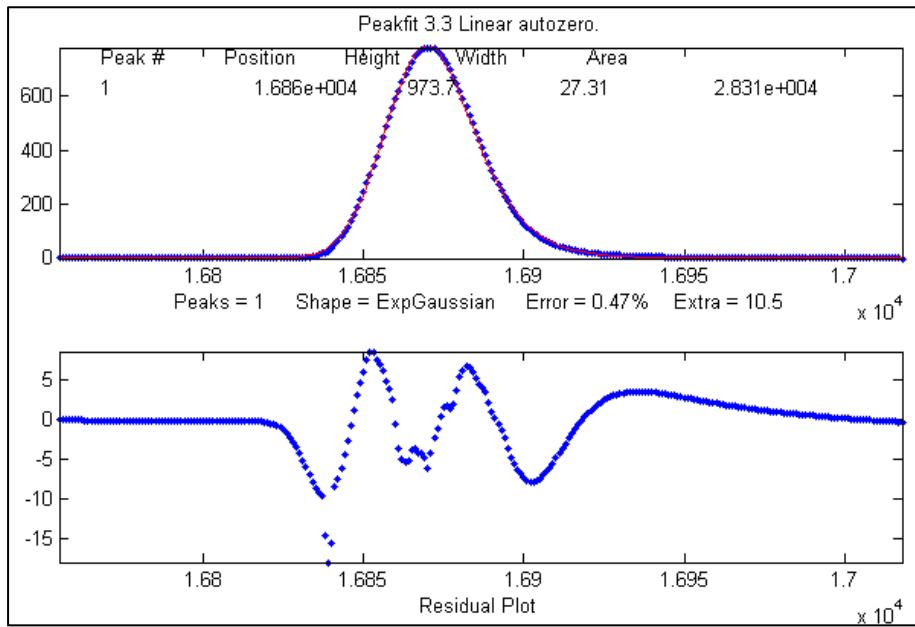
Gli spikes negativi in questo segnale si sono rivelati essere ripidi *gradini*, che possono essere ridotti utilizzando la funzione [limite della velocità di variazione \[slew-rate\]](#) di iSignal (il tasto `) o eliminati manualmente utilizzando il tasto punto e virgola (;) per impostare a zero l'area selezionata tra le linee rosse tratteggiate del cursore. Utilizzando quest'ultimo approccio, l'intero segnale ripulito è mostrato di seguito. I picchi rimanenti sono tutti positivi, a forma di campana e hanno ampiezze da circa 6 a circa 750.



iPeak (pag. 405) può automatizzare le misure delle posizioni e delle altezze dei picchi per l'intero segnale, utilizzando le impostazioni per il rilevamento dei picchi mostrate nella parte inferiore della schermata seguente.



Se necessario, i singoli picchi possono essere misurati in modo più accurato approssimando in *iPeak* l'intero picco col tasto “N” (pagina 405) o con *peakfit.m* o con *ipf.m* (pagina 405). I picchi sono tutti leggermente asimmetrici; si approssimano a un [modello Gaussiano esponenzialmente allargato](#) (pag. 220) con un errore di approssimazione inferiore allo 0.5% circa, come mostrato di seguito. I grafici residui uniformi suggeriscono che il segnale ha subito uno smoothing *prima* che venissero introdotti gli spikes e che il rumore aumenta con l'ampiezza del segnale (perché sono pochi o rumorosi sulla linea di base).



Si noti che l'approssimazione con un modello Gaussiano espanso esponzialmente fornisce i [parametri del picco Gaussiano prima dell'espansione](#). *iSignal* (pagina 366) e *iPeak* (pagina 405) stima i parametri del picco espanso. Come in precedenza, l'effetto dell'ampliamento è quello di spostare la posizione del picco su valori maggiori, ridurne l'altezza e aumentarne l'ampiezza.

```

Position  Height  Width  Area   error
isignal  16871  788.88  32.881  27612      S/N Ratio = 172
ipeak    16871  785.34  33.525  28029
peakfit  16871  777.9   33.488  27729  1.68% Gaussian model
  
```

Il Torneo: un confronto dei metodi

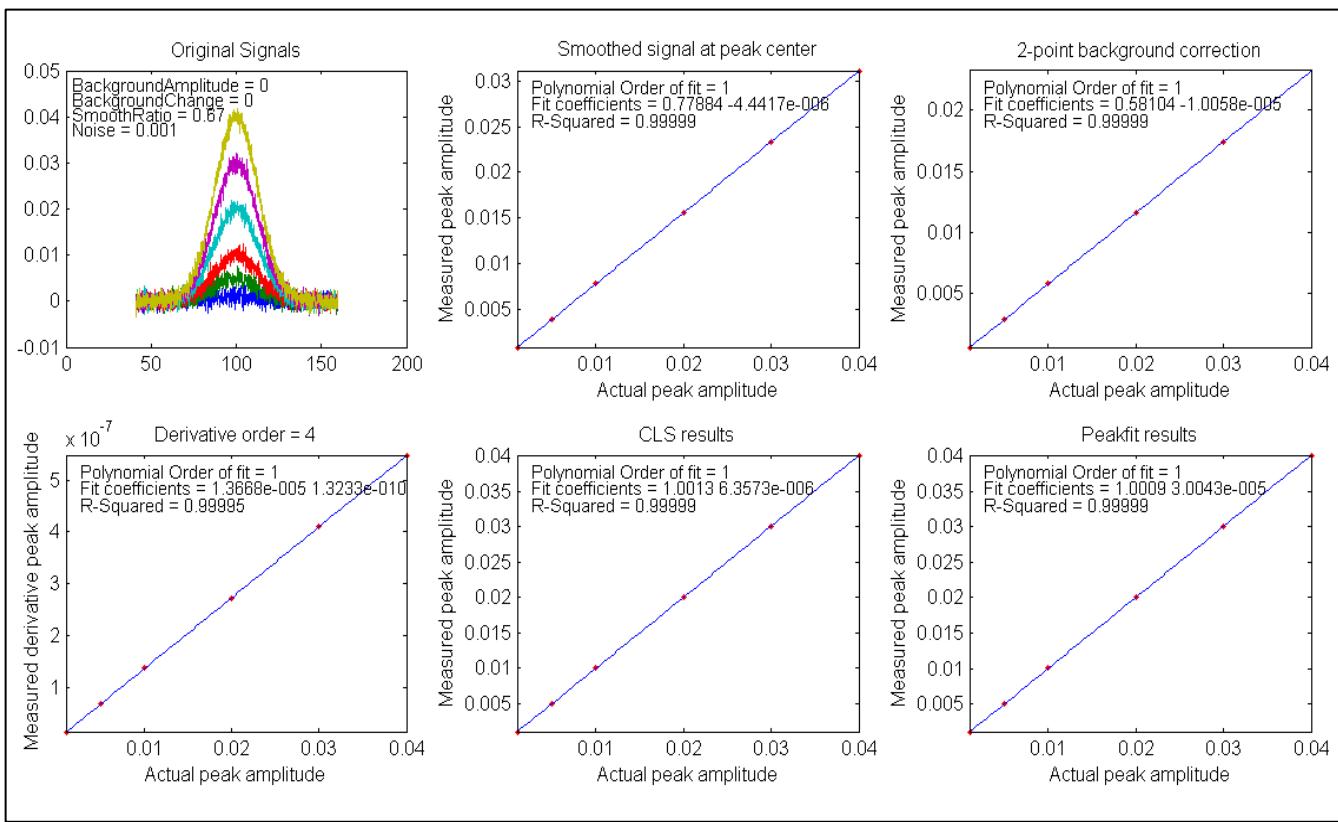
Questa simulazione mostra l'applicazione di diverse tecniche descritte in questo documento per la misura quantitativa di un picco sepolto in un background instabile, una situazione che si verificarsi spesso nelle applicazioni di analisi quantitativa di varie forme di spettroscopia, monitoraggio di processo e telerilevamento. L'obiettivo è ricavare una misura dell'ampiezza del picco che varia linearmente con la l'ampiezza effettiva del picco ma che è *minimamente influenzata dai cambiamenti nel background e per il rumore casuale*. In questo esempio, il picco da misurare si trova in una posizione fissa al centro del segnale registrato, in $x=100$ e ha una forma (Gaussiana) e una larghezza fisso (30). Il background, invece, è molto variabile, sia in ampiezza *che* nel profilo. La simulazione mostra sei registrazioni sovrapposte del segnale con sei diverse ampiezze di picco e con ampiezze e forme variabili in modo casuale del background (riga in alto a sinistra nelle figure seguenti). I metodi che vengono messi a confronto qui includono lo [smoothing](#) (pag. 38), la [differenziazione](#) (pag. 58), [il metodo classico dei quadrati minimi multi-componente](#) (pag. 180) e [l'approssimazione iterativa non-lineare](#) (pag. 190).

[CaseStudyC.m](#) è una funzione autonoma dimostrativa Matlab/Octave che mostra questo caso. Per eseguirlo, scaricarlo, inserirlo nel path di ricerca di, e digitare “CaseStudyC” al prompt dei comandi. Ogni volta che lo si esegue, si otterrà la stessa serie di vere ampiezze di picco (impostate dal vettore “SignalAmplitudes”, nella riga 12) ma un diverso insieme di forme e ampiezze del background. Il background è modellato come un picco Gaussiano di ampiezza, posizione e larghezza variabili in modo casuale; è possibile controllare l' *ampiezza media* del background modificando la variabile “BackgroundAmplitude” e la *variazione media* nel background modificando la variabile “BackgroundChange”.

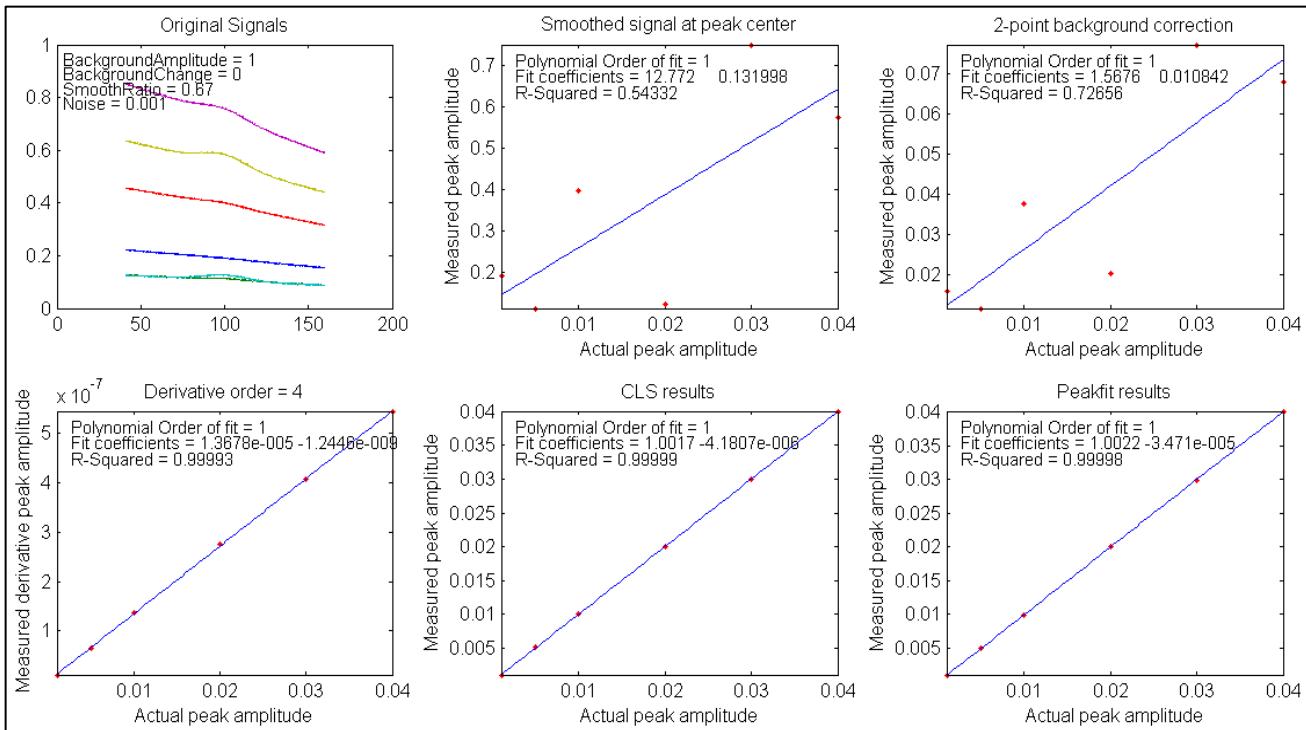
I cinque metodi confrontati nelle figure seguenti sono:

- 1: Fila in alto al centro. Una semplice misura da zero a picco [segnale con smoothing](#), che presuppone che lo sfondo sia *zero*.
- 2: Fila in alto a destra. La differenza tra il segnale del picco e il background medio su entrambi i lati del picco (entrambi con smoothing), che presuppone che lo sfondo sia *piatto*.
- 3: Fila in basso a sinistra. Un [metodo basato sulla derivata](#), che presuppone che il background sia *molto largo* rispetto al picco misurato.
- 4: Fila in basso al centro. [Classico dei Minimi quadrati](#) (CLS), che presuppone che il background sia un picco di *forma, larghezza e posizione note* (l'unica incognita è *l'altezza*).
- 5: Fila in basso a destra. [approssimazione iterativa non-lineare](#) (INLS), che presuppone che lo sfondo sia un picco di *forma nota* ma larghezza e posizione ignote. Questo metodo può tenere traccia dei cambiamenti nella posizione e nell'ampiezza del background del picco (entro certi limiti), se il picco misurato e i *profili* del background sono indipendenti dalla concentrazione dell'incognita.

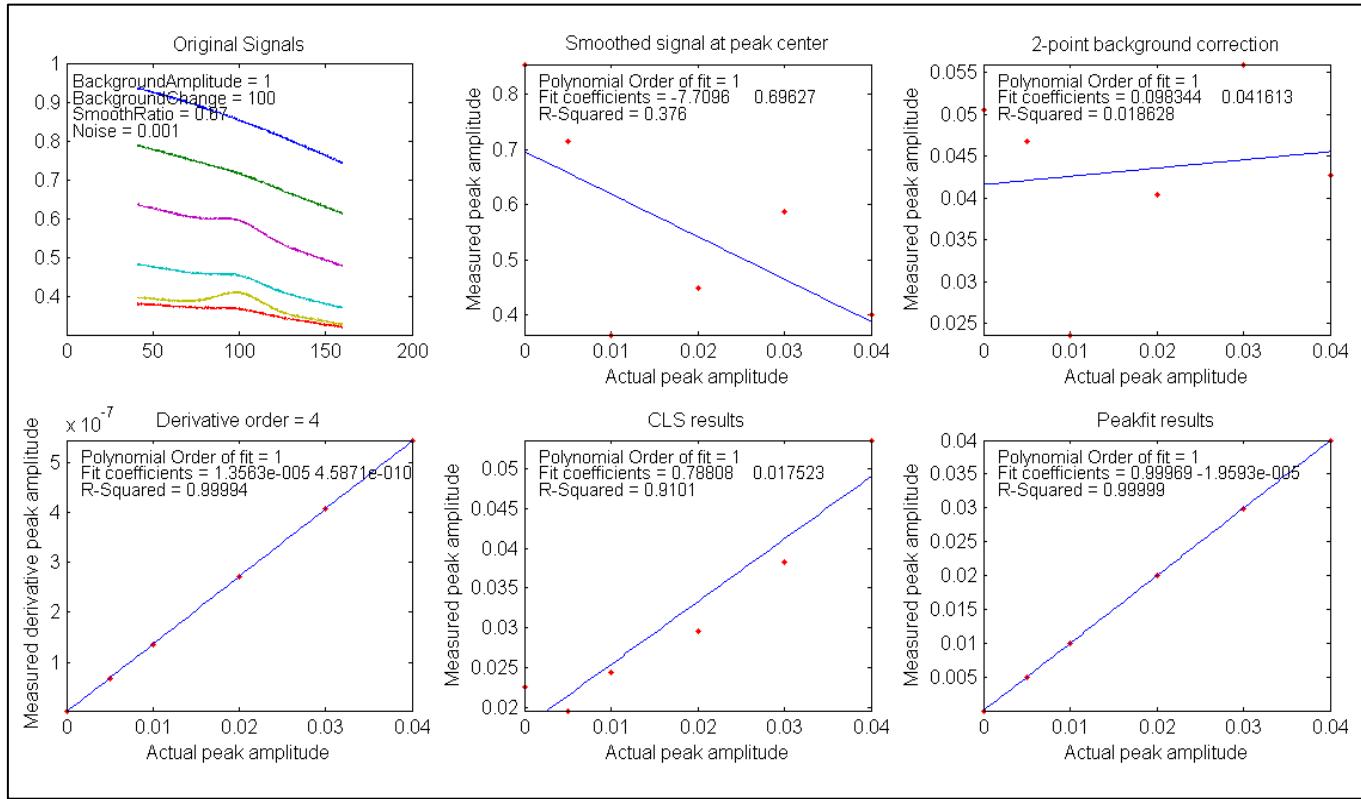
Questi cinque metodi sono elencati approssimativamente in ordine crescente di complessità matematica e geometrica. Vengono confrontati di seguito tracciando le altezze effettive dei picchi (impostate dal vettore "SignalAmplitudes") rispetto alla misura derivata da quel metodo, approssimando i dati a una linea retta e calcolando il *coefficiente di determinazione*, R^2 , che è 1.0000 per un grafico perfettamente lineare.



Per il primo test (mostrato nella figura sopra), sia “BackgroundAmplitude” che “BackgroundChange” sono impostati a zero, in modo che sia presente solo il rumore casuale. In tal caso, tutti i metodi funzionano bene, con i valori di R^2 tutti molto prossimi a 0.9999. Con un livello di rumore 10 volte più alto ([cliccare per visualizzare](#)), tutti i metodi funzionano ugualmente bene, ma con un coefficiente di determinazione R^2 più basso, come ci si potrebbe aspettare.



Per il secondo test (mostrato nella figura immediatamente sopra), "BackgroundAmplitude"=1 e "BackgroundChange"=0, quindi il background ha una variazione di ampiezza significativa ma forma, posizione e larghezza fisse. In questo caso, i primi due metodi falliscono, ma i metodi derivativi, CLS e INLS funzionano bene.



Per il terzo test, mostrato nella figura sopra, "BackgroundAmplitude"=1 e "BackgroundChange"=100, quindi in questo caso il background varia in posizione, larghezza e ampiezza (ma rimane ampio rispetto al segnale). Qui, il metodo CLS fallisce, perché presume che lo sfondo vari solo in ampiezza. Tuttavia, se facciamo un ulteriore passo avanti ([click per visualizzare](#)) e si imposta "BackgroundChange"=1000, la forma del background è ora così instabile che anche il metodo INLS fallisce, ma il metodo derivativo rimane comunque efficace fintanto che il background è più ampio del picco in esame, indipendentemente dalla sua forma. D'altra parte, se la larghezza e la posizione del picco *misurato* cambiano da campione a campione, il metodo derivativo fallirà e il metodo INLS è più efficace ([click per visualizzare](#)), purché sia nota la forma fondamentale sia del picco misurato che del background (p.es. Gaussiano, Lorentziano, ecc.)

Non sorprende che i metodi più matematicamente complessi funzionino mediamente meglio. Fortunatamente, il software può "nascondere" quella complessità, nello stesso modo, ad esempio, in cui una calcolatrice portatile nasconde la complessità di una lunga divisione o calcolando le radici quadrate.

Schemi di media di insieme in un segnale continuo

[La media d'insieme](#) è un potente metodo per ridurre l'effetto del rumore casuale nei segnali sperimentali quando è applicabile. L'idea è che il segnale è ripetuto, preferibilmente molte volte, e poi si mediano tutte le ripetizioni. Il segnale si accumula, ed il rumore gradualmente scende a zero, man mano che aumentano le ripetizioni.

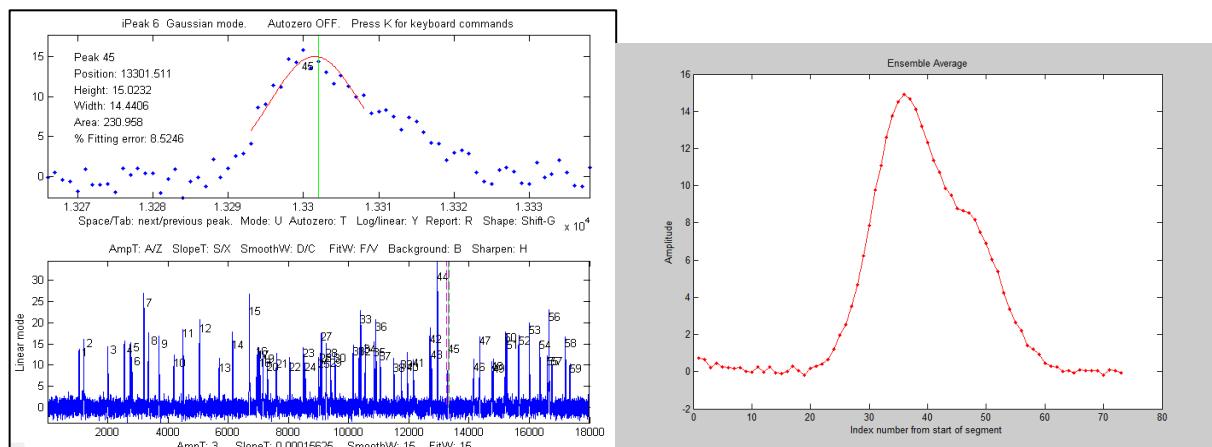
Un requisito importante è che le ripetizioni siano allineate o sincronizzate in modo che, in assenza di rumore casuale, i segnali ripetuti si allineino esattamente. Ci sono due modi per gestirlo:

(a) le ripetizioni del segnale sono attivate da un evento esterno e l'acquisizione dei dati può utilizzare tale trigger per sincronizzare i segnali, o

(b) il segnale stesso ha alcune caratteristiche che possono essere utilizzate per rilevare ogni ripetizione, ogni volta che si verifica.

Il primo metodo (a) ha il vantaggio che il rapporto segnale/rumore (S/N) può essere arbitrariamente basso e il segnale medio emergerà comunque gradualmente dal rumore se il numero di ripetizioni è abbastanza grande. Tuttavia, non tutti gli esperimenti hanno un sincronismo esterno affidabile.

Il secondo metodo (b) può essere utilizzato per mediare pattern ripetuti in un segnale continuo senza un trigger esterno che corrisponde a ciascuna ripetizione, ma il segnale deve quindi contenere qualche caratteristica (ad esempio, un picco) con un rapporto segnale-rumore abbastanza grande da essere rilevabile in modo affidabile in ogni ripetizione. Questo metodo può essere utilizzato anche quando i pattern del segnale si verificano a intervalli casuali quando la tempistica delle ripetizioni non è di interesse. Il rilevatore di picchi interattivo *iPeak 6* (pag. 405) ha una funzione di calcolo della media dell'insieme (Shift-E) che può calcolare la media di tutte le forme d'onda ripetute. Funziona rilevando un singolo picco in ogni ripetizione per sincronizzarla.

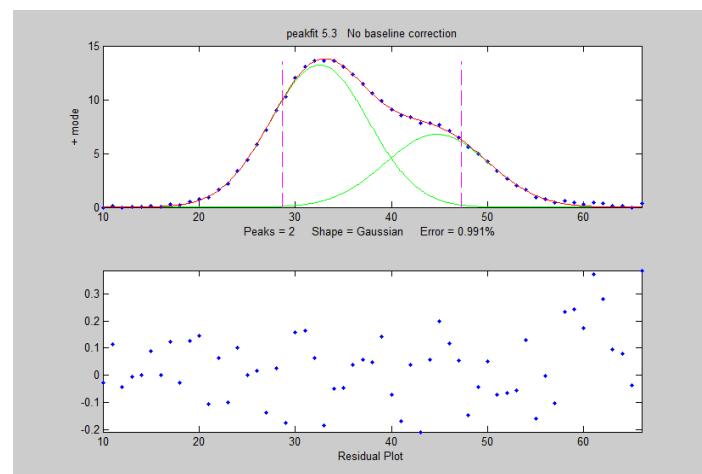


Lo script Matlab [iPeakEnsembleAverageDemo.m](#) (su <http://tinyurl.com/cey8rwh>) mostra questa idea, con un segnale che contiene un pattern sottostante ripetuto di due picchi Gaussiani sovrapposti, a 12 punti di distanza, con un rapporto di altezza 2: 1, entrambi di larghezza 12. Questi schemi si verificano a intervalli casuali e il livello di rumore è circa il 10% dell'altezza media del picco. Usando *iPeak* (pag. 405) mostrato sopra a sinistra), si aggiustano i controlli affinché si rilevi un solo un picco in ogni schema di ripetizione, si ingrandisce e si isola uno qualsiasi di questi schemi e si preme Shift-E. In questo caso, ci sono circa 60 ripetizioni, quindi il miglioramento del rapporto segnale/rumore (S/N) atteso è $\sqrt{60} = 7.7$. È possibile salvare il pattern medio (in alto a destra) nell'area di lavoro Matlab come "EA" digitando

```
>> load EnsembleAverage; EA=EnsembleAverage;
```

quindi approssimare questo pattern medio a un modello di 2-Gaussiane utilizzando la [funzione peakfit.m](#) (figura a destra):

```
peakfit([1:length(EA);EA],40,60,2,1,0,10)
```



```

Position Height Width Area
32.54 13.255 12.003 169.36
44.72 6.7916 12.677 91.69

```

Si vedrà un grande miglioramento nella precisione della separazione dei picchi, del rapporto di altezza e della larghezza, rispetto all'approssimazione di un *singolo* pattern nel segnale originale x,y:
`>> peakfit([x;y],16352,60,2,1,0,10)`

Analisi delle Armoniche dell'Effetto Doppler

Il file wav “[horngoby.wav](#)” (Ctrl-click per aprirlo) è una registrazione di 2 secondi del suono emesso dal clacson di un'automobile di passaggio, per esporre il familiare [effetto Doppler](#). La frequenza di campionamento è di 22000 Hz. Scaricare questo file e metterlo nel path di Matlab. È quindi possibile caricarlo nello spazio di lavoro Matlab come variabile “doppler” e visualizzarlo utilizzando [iSignal](#) (pagina 366):

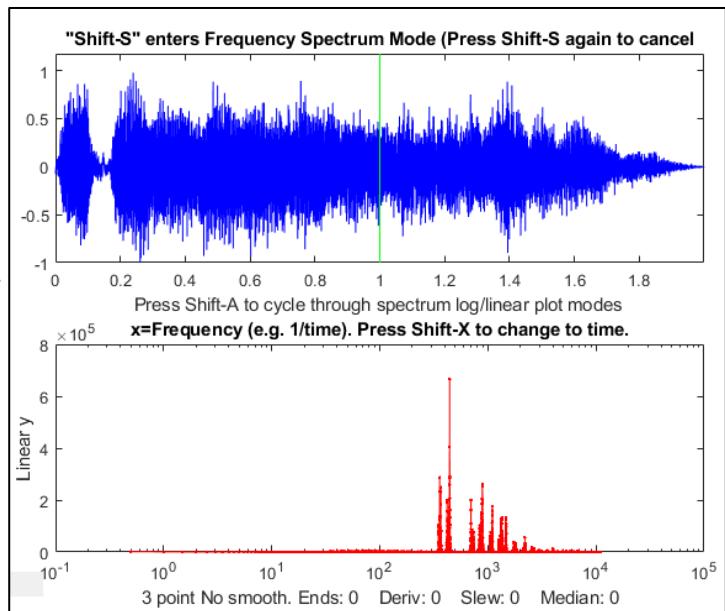
```

t=0:1/21920:2;

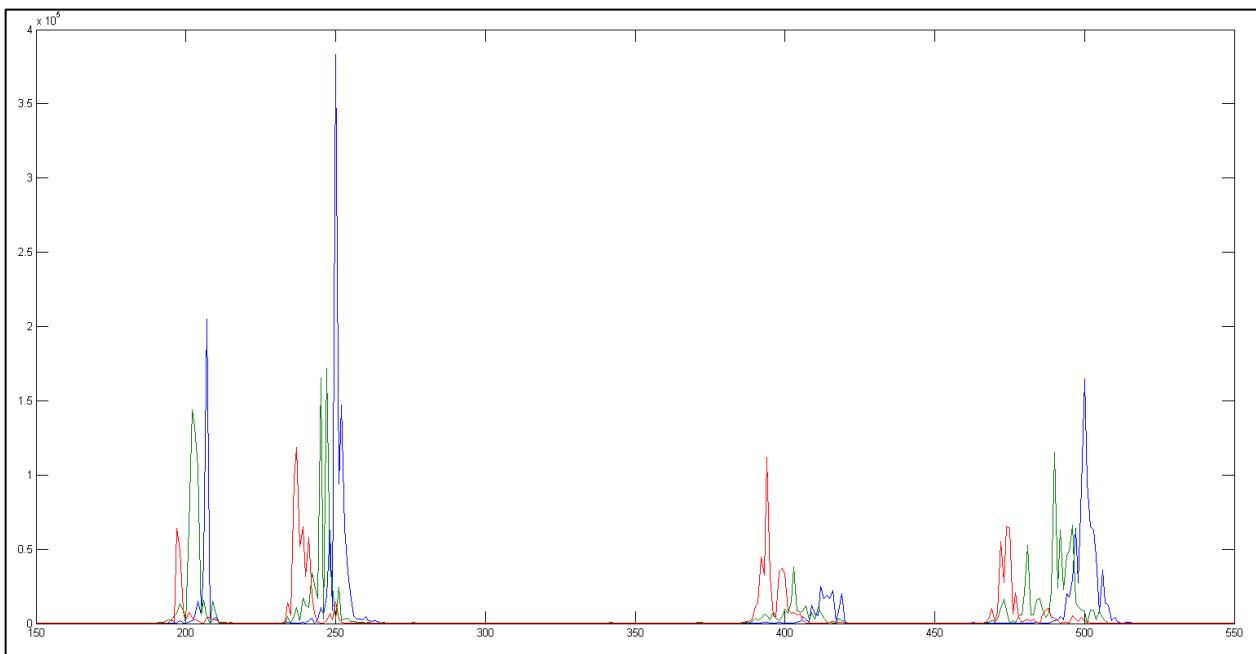
load horngoby.mat

isignal(t,doppler);

```



All'interno di iSignal, è possibile passare alla [modalità spettro di frequenza](#) premendo **Shift-S** e ingrandendo le diverse parti della forma d'onda utilizzando i tasti cursore, in modo da poter osservare lo *spostamento di frequenza verso il basso* e misurarlo quantitativamente. (In realtà è molto più facile *ascoltare* lo spostamento in frequenza - premere **Shift-P** per riprodurre il suono - anziché *vederlo* graficamente. Lo spostamento è piuttosto modesto percentualmente, ma [l'orecchio umano è molto sensibile alle piccole variazioni \(di frequenza\)](#)). È più facile vedere se si ri-disegnano i dati per allungare la regione di frequenza attorno a quella fondamentale o a una delle armoniche. Si è usato [iSignal](#) per ingrandire tre sezioni di questa forma d'onda e poi è stato disegnato lo spettro della frequenza (**Shift-S**) vicino *all'inizio* (in blu), *al centro* (verde) e alla *fine* (rosso) del suono. La regione di frequenza tra 150 Hz e 550 Hz è disegnata nella figura seguente:



Il gruppo di picchi vicino a 200 è la [frequenza fondamentale](#) della nota più bassa del clacson e il gruppo vicino a 400 è la [seconda armonica](#). (I suoni intonati hanno una struttura armonica di 1, 2, 3... volte la frequenza fondamentale). Il gruppo di picchi vicino a 250 è la frequenza fondamentale della nota successiva più alta del clacson e il gruppo vicino a 500 è la sua seconda armonica. (I clacson delle auto e dei treni hanno spesso due o tre [note armoniose](#) suonate insieme). In ciascuno di questi gruppi di armoniche, si può vedere chiaramente che il picco blu (lo spettro misurato all'*inizio* del suono) ha una frequenza *più alta* del picco rosso (lo spettro misurato alla *fine* del suono). Il picco verde, preso al centro, ha una frequenza intermedia. *I picchi sono irregolari perché l'ampiezza e la frequenza variano nell'intervallo di campionamento*, ma è comunque possibile ottenere buone misure quantitative della frequenza di ciascun componente con l'[approssimazione ad un modello Gaussiano](#) utilizzando [peakfit.m](#) o [ipf.m](#) (pagina 405):

Peak	Position	Height	Width	Area
Beginning	206.69	3.0191e+005	0.81866	2.4636e+005
Middle	202.65	1.5481e+005	2.911	4.797e+005
End	197.42	81906	1.3785	1.1994e+005

La precisione stimata delle misure della posizione (cioè della frequenza) è di circa lo 0.2% relativo, in base al [metodo bootstrap](#), abbastanza buona da consentire un calcolo accurato dello spostamento della frequenza (circa il 4.2%) e della [velocità del veicolo](#) e per dimostrare che il rapporto misurato tra la seconda armonica e la fondamentale per questi dati è 2.0023, che è molto vicino al valore teorico di 2.

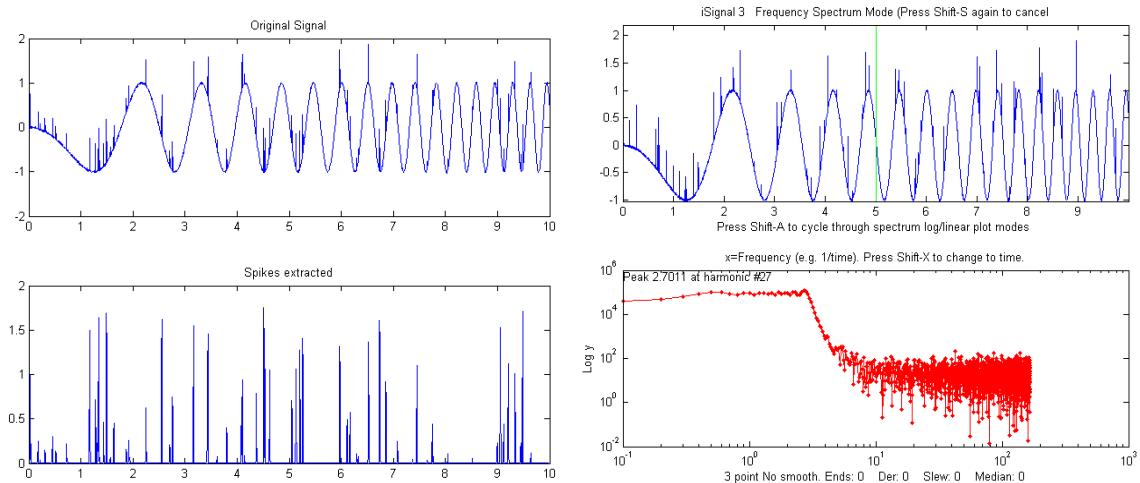
Misura degli spike

Gli spike, stretti impulsi, larghi solo uno o due punti, si incontrano talvolta nei segnali a causa di un “glitch” elettronico o un disturbo parassita proveniente da apparecchiature vicine e possono essere facilmente eliminati utilizzando un [filtro “mediano”](#).

Ma è possibile che in alcuni esperimenti gli *stessi spike* possano essere la parte importante del segnale e che sia necessario contarli o misurarli. Questa situazione è stata riscontrata in un'applicazione di ricerca da uno dei miei clienti, e porta ad alcune interessanti svolte sulle procedure solite. In

quella applicazione i picchi erano causati da granelli di sabbia della spiaggia portati dal vento che colpivano il diaframma di un microfono, accompagnati da forme d'onda più o meno sinusoidali, forse causate dal fischio del vento attraverso l'apparecchiatura o dai richiami degli uccelli costieri. L'obiettivo era contare gli impatti dei granelli e ignorare gli altri suoni, eventualmente più forti.

Per simulare questa situazione, lo script Matlab/Octave [SpikeDemo1.m](#) crea una forma d'onda (riquadro superiore della figura seguente) in cui una serie di picchi sono distribuiti casualmente nel tempo, contaminato da due tipi di rumore: rumore bianco e un'interferenza oscillatoria di grande ampiezza simulata da un'onda sinusoidale a frequenza variabile. L'obiettivo è contare gli spike e individuare la loro posizione sull'asse x (tempo). L'applicazione diretta di `findpeaks` o di `iPeak` (pag. 405) al segnale originale non funziona bene.



Uno spike da un singolo punto, chiamato *funzione delta* in matematica, ha uno spettro di potenza piatto; cioè, ha *la stessa potenza a tutte le frequenze*, proprio come il rumore bianco. Ma l'interferenza oscillatoria, in questo caso, è uno *specifico intervallo di frequenze*, il che apre alcune interessanti possibilità. Un approccio potrebbe essere quello di utilizzare un [filtro di Fourier](#), ad esempio, un filtro notch o escludi-banda, per [rimuovere le oscillazioni fastidiose](#) selettivamente. Ma se l'obiettivo della misura è solo quello di contare gli spike e misurarne i tempi, un approccio più semplice sarebbe (1) calcolare la [derivata seconda](#) (che amplifica notevolmente gli spike relativi alle oscillazioni), (2) eseguire lo [smoothing](#) del risultato (per limitare l'[amplificazione del rumore bianco causata dalla differenziazione](#)), quindi (3) prendere il [valore assoluto](#) (per ottenere spike positivi). Questo può essere fatto in una *singola riga* di codice Matlab/Octave annidato:

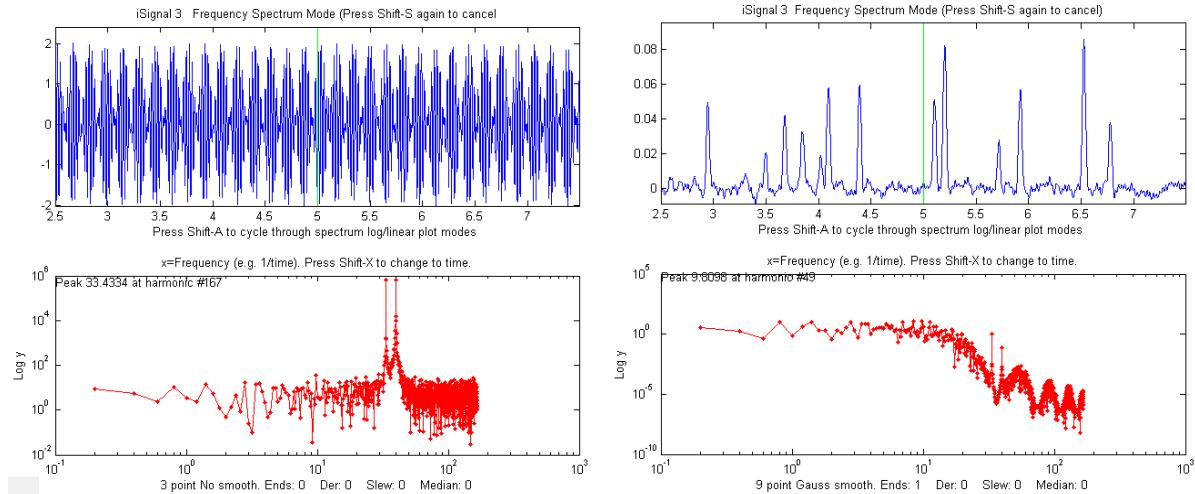
```
y1=abs(fastsmooth((deriv2(y)).^2,3,2));
```

Il risultato, mostrato nel pannello inferiore della figura in alto a sinistra, è un'estrazione quasi completa degli spike, che possono quindi essere contati con [findpeaksG.m](#) o [peakstats.m](#) o [iPeak.m](#) (pag. 405):

```
P=ipeak([x;y1],0,0.1,2e-005,1,3,3,0.2,0);
```

Il secondo esempio, [SpikeDemo2.m](#), è ancora più difficile. In questo caso l'interferenza oscillatoria è causata da *due* onde sinusoidali a una frequenza fissa più alta, che *oscura completamente* gli spike nel segnale originale (pannello superiore della figura a sinistra in basso). Nello [spettro di potenza](#) (pannello inferiore, in rosso), l'interferenza oscillatoria si manifesta come due stretti picchi che dominano lo spettro e raggiungono $y=10^6$, mentre gli spike appaiono come un appiattimento più basso e largo a circa $y=10$. In questo caso, si può utilizzare un'interessante proprietà degli smoothing a

slittamento della media, come gli [smoothing](#) boxcar, triangolari e Gaussiani; le loro risposte in frequenza mostrano una serie di profonde [cuspidi](#) a frequenze inversamente proporzionali alla larghezza del loro filtro. Questo apre la possibilità di sopprimere specifiche frequenze di interferenza oscillatorie regolando la larghezza del filtro fino a quando le cuspidi si verificano in prossimità o alla frequenza o delle oscillazioni. Poiché il segnale, in questo caso, è costituito da spike che hanno uno spettro di potenza piatto, vengono semplicemente appiattiti da questa operazione, che ridurrà le loro altezze e aumenterà la loro larghezza, ma avrà poco o nessun effetto sul loro numero o sulle posizioni sull'asse x. In questo caso, un smoothing P-spline a 9 punti colloca la prima cuspide (frequenza più bassa) proprio tra le due frequenze oscillatorie.



Nella figura a destra, si può vedere l'effetto di tale filtro; gli spike, che *non erano nemmeno visibili* nel segnale originale, ora vengono estratti in modo pulito (pannello superiore), e si può vedere nello spettro di potenza (pannello inferiore, in rosso) che i due sottili picchi dell'interferenza oscillatoria sono *ridotti di un fattore di circa 1,000,000!* (Nota: gli spettri di frequenza sono disegnati su una *scala log-log*). Questa operazione può essere eseguita da una singola funzione a riga di comando, regolando la larghezza dello smoothing (il secondo argomento di input, qui un 9) per tentativi per ridurre al minimo l'interferenza oscillatoria:

```
y1=fastsmooth(y,9,3);
```

Se l'interferenza varia in frequenza nel segnale, è possibile utilizzare uno [smoothing segmentato](#) anziché la fastsmooth standard. In alternativa, lo [Spettro di Fourier segmentato](#) (pag. 98) potrebbe essere utilizzato per [visualizzare questo segnale](#) e un [filtro di Fourier](#) (pag. 122) in modalità “notch” potrebbe essere utilizzato specificamente [per eliminare le frequenze di interferenza](#). I picchi estratti potrebbero quindi essere contati utilizzando una qualsiasi delle [funzioni di ricerca dei picchi](#), come ad esempio:

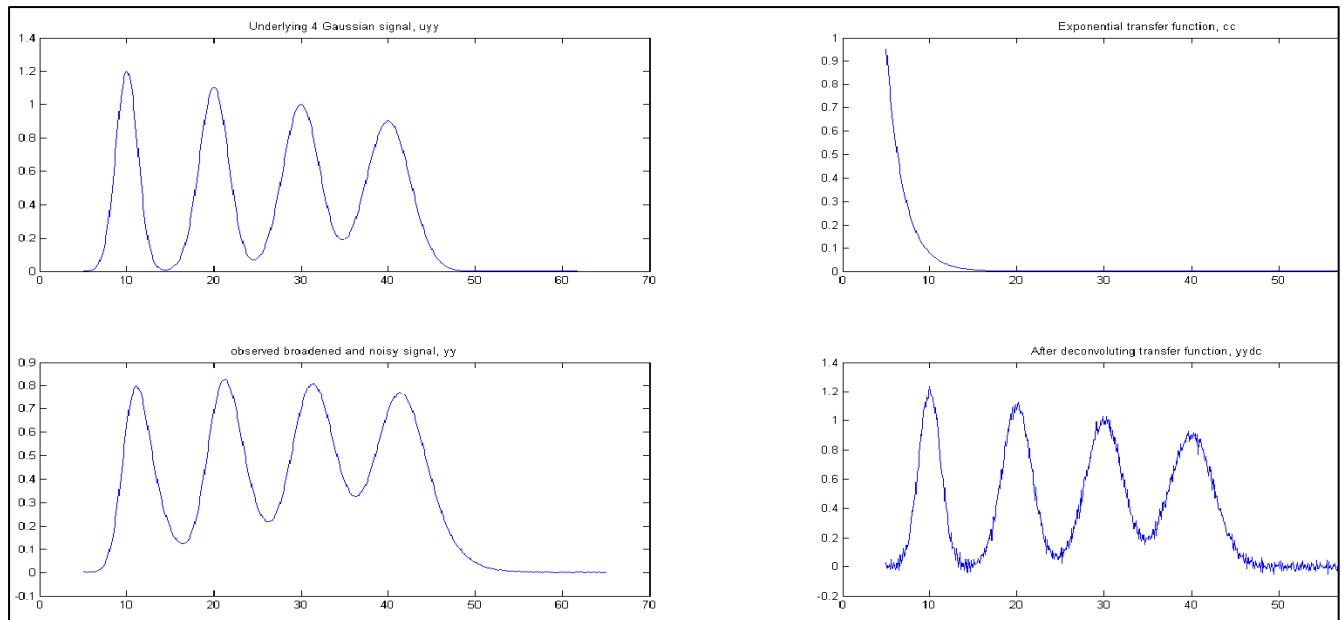
```
P=findpeaksG(x,y1,2e-005,0.01,2,5,3);
0
P=findpeaksplot(x,y1,2e-005,0.01,2,5,3);
0
PS=peakstats(x,y1,2e-005,0.01,2,5,3,1);
```

Il semplice script “[iSignalDeltaTest](#)” mostra lo spettro di potenza delle funzioni di smoothing e differenziazione di iSignal (pagina 366) applicandole a una [funzione delta](#). Modificare il tipo, la larghezza e l'ordine di derivazione dello smoothing e altre funzioni per vedere come cambia lo spettro di potenza.

Deconvoluzione di Fourier rispetto all'approssimazione della curva [curve fitting] (non è la stessa cosa)

Alcuni esperimenti producono picchi che vengono distorti subendo una convoluzione da processi che rendono i picchi meno distinti e modificandone la posizione, l'altezza e la larghezza.

L'[ampliamento esponenziale](#) è uno dei più comuni di questi processi. La [deconvoluzione di Fourier](#) e l'[approssimazione iterativa](#) sono due metodi che possono aiutare a misurare *i veri parametri dei picchi*. Questi due metodi sono concettualmente distinti perché, nella deconvoluzione di Fourier, il profilo del picco in esame è sconosciuto ma si presume che siano note la natura e l'ampiezza della funzione di ampliamento (ad esempio esponenziale); mentre nell'approssimazione iterativa dei minimi quadrati è esattamente il contrario: il *profilo* del picco (p. es. Gaussiano, Lorentziano, ecc.) dev'essere noto, ma la larghezza del processo di ampliamento è inizialmente sconosciuto.



Nello script mostrato sotto e il grafico risultante mostrato a sopra ([Scaricare questo script](#)), il segnale sottostante (uyy) è un insieme di quattro Gaussiane con altezze di picco di 1.2, 1.1, 1, 0.9 situate a $x = 10, 20, 30, 40$ e larghezze di picco di 3, 4, 5, 6, ma nel segnale *osservato* (yy) questi picchi sono ampliati esponenzialmente dalla funzione esponenziale cc, ne risultano dei [picchi spostati, più corti e più larghi](#), e poi viene aggiunto un po' di rumore bianco costante *dopo* la dilatazione. La deconvoluzione di cc da yy rimuove con successo la dilatazione (yydc), ma a scapito di un [sostanziale aumento del rumore](#). Tuttavia, il rumore extra nel segnale deconvoluto è più caricato alle alte frequenze ("blu") e quindi può essere facilmente ridotto dallo [smoothing](#) e *ha meno effetto sulle approssimazioni dei minimi quadrati rispetto al rumore bianco*. (Per un'impresa più difficile, provare ad aggiungere più rumore nella riga 6 o utilizzare una stima errata della costante di tempo nella riga 10). Per disegnare il segnale recuperato sovrapposto a quello originale : [`plot\(xx, uyy, xx, yydc\)`](#). Per disegnare il segnale osservato sovrapposto a quello originale: [`plot\(xx, uyy, xx, yy\)`](#). Eccellenti valori per le posizioni, le altezze e le larghezze dei picchi originali in esame si possono ottenere [approssimando la curva](#) del segnale recuperato a quattro Gaussiane: [`\[FitResults, FitError\] = peakfit\(\[xx;yydc\], 26, 42, 4, 1, 0, 10\)`](#). Con *dieci volte* il livello di rumore precedente (Noise=.01), i valori dei parametri determinati dall'approssimazione sono [ancora abbastanza buoni](#), e anche con *100 volte più rumore* (Noise=0.1) i parametri sono [più accurati di quanto ci si potrebbe aspettare](#) per quella quantità di rumore (perché quel rumore è *blu*). Visivamente, il rumore è così grande che la situazione sembra senza speranza, ma il 'curve fitting' della curva funziona bene.

```

xx=5:.1:65;
% Underlying Gaussian peaks with unknown heights, positions, and widths.
uyy=modelpeaks2(xx,[1 1 1 1],[1.2 1.1 1 .9],[10 20 30 40],[3 4 5 6],...
[0 0 0 0]);
% Observed signal yy, with noise added AFTER the broadening convolution
Noise=.001; % <---Try more noise to see how this method handles it.
yy=modelpeaks2(xx,[5 5 5 5],[1.2 1.1 1 .9],[10 20 30 40],[3 4 5 6],...
[-40 -40 -40 -40])+Noise.*randn(size(xx));
% Compute transfer function, cc,
cc=exp(-(1:length(yy))./40); % <---Change exponential factor here
% Attempt to recover original signal uyy by deconvoluting cc from yy
% It is necessary to zero-pad the observed signal yy as shown here.
yydc=deconv([yy zeros(1,length(yy)-1)],cc).*sum(cc);
% Plot and label everything
subplot(2,2,1);plot(xx,uyy);title('Underlying signal, uyy');
subplot(2,2,2);plot(xx,cc);title('Exponential transfer function, cc')
subplot(2,2,3);plot(xx,yy);title('observed broadened, noisy signal, yy');
subplot(2,2,4);plot(xx,yydc);title('Recovered signal, yydc')

```

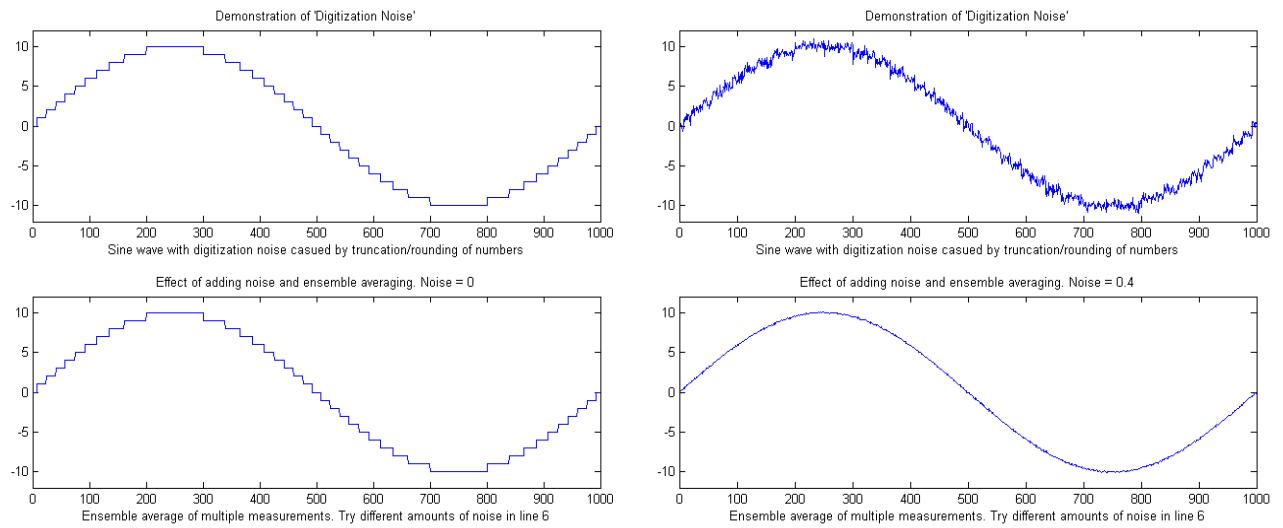
Un'alternativa all'approccio dell.deconvoluzione di cui sopra è quello di approssimare la curva del segnale osservato direttamente con una [Gaussiana ampliata esponenzialmente](#) (profilo numero 5):

[\[FitResults,FitError\] = peakfit\(\[xx;yy\],26,50,4,5,40,10\)](#). Entrambi i metodi danno buoni valori dei parametri, ma il metodo della deconvoluzione è notevolmente più veloce perché l'approssimazione della curva con un semplice modello Gaussiano è più veloce di quella con un modello di picco allargato esponenzialmente, specialmente se il numero di picchi è grande. Inoltre, se il fattore esponenziale non è noto, può essere determinato approssimando la curva a uno o due dei picchi nel segnale osservato, utilizzando ipf.m (pag. 405), regolando interattivamente il fattore esponenziale per ottenere la corrispondenza migliore. Notare che *si deve dare a peakfit un valore ragionevolmente buono della costante di tempo ('extra')*, l'argomento di input subito dopo il numero di peakshape. Se il valore è troppo lontano, l'approssimazione potrebbe non riuscire completamente, restituendo tutti zeri. Qualche tentativo è sufficiente. In alternativa, si può provare a utilizzare [peakfit.m versione 7](#) con la Gaussiana allargata esponenzialmente con la *variabile indipendente* profilo numero 31 o 39, per misurare la costante di tempo come una variabile iterata (per capire la differenza, vedere l'[esempio 39](#)). Se si prevede che la costante di tempo sia *la stessa* per tutti i picchi, si otterranno risultati migliori utilizzando inizialmente il profilo numero 31 o il 39 per misurarla su un picco isolato (preferibilmente uno con un buon rapporto S/N), poi si applica quella costante di tempo fissa nel profilo 5 a tutti gli altri gruppi di picchi sovrapposti.

Rumore di digitalizzazione - l'aggiunta di rumore può davvero aiutare?

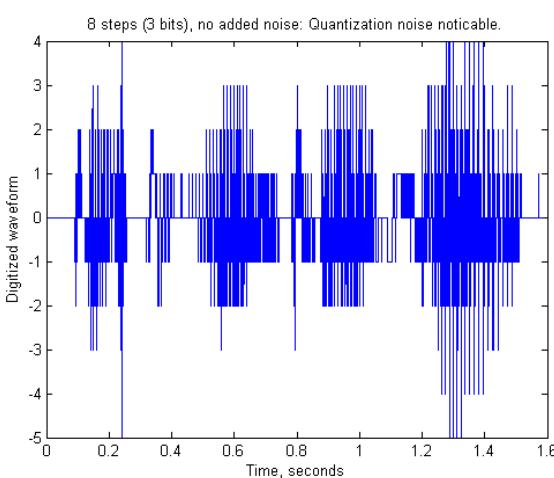
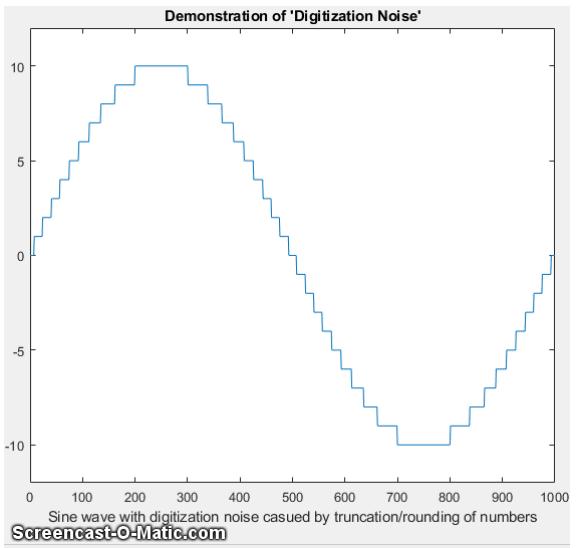
Il rumore di digitalizzazione, chiamato anche [rumore di quantizzazione](#), è un artefatto causato dall'arrotondamento o dal troncamento di numeri a un numero fisso di cifre. Può avere origine nel [convertitore analogico-digitale](#) che converte un segnale analogico in uno digitale, o nei circuiti o nel software coinvolti nella trasmissione del segnale digitale a un computer, o anche nel processo di tra-

sferimento dei dati da un programma a un altro, come nel copiare e incollare dati in e da un foglio di calcolo. Il risultato è una serie di salti non casuali di uguale altezza. La distribuzione delle frequenze è bianca, a causa della ripidità dei gradini, come si può vedere osservando lo [spettro di potenza](#).



La figura a lato, pannello superiore, mostra l'effetto della digitalizzazione intera su un'onda sinusoidale con un'ampiezza di +/- 10. La media dell'insieme, che di solito è la più efficace delle tecniche di riduzione del rumore, non riduce questo tipo di rumore (pannello inferiore) perché non è casuale.

È interessante notare che, se nel segnale è presente ulteriore rumore casuale, la media dell'insieme diventa efficace nel ridurre sia il rumore casuale, che il rumore della digitalizzazione. In sostanza, il rumore aggiunto *randomizza la digitalizzazione*, consentendone la riduzione



mediante la media dell'insieme. Inoltre, se il rumore già presente nel segnale è troppo basso, può essere utile *aggiungere artificialmente ulteriore rumore!* Sul serio. Lo script [RoundingError.m](#) illustra questo effetto, come mostrato nell'[animazione](#) in alto a destra, che mostra l'onda sinusoidale digitalizzata con quantità gradualmente crescenti di rumore casuale aggiunto in linea 8 (generato dalla funzione randn.m) seguito da una media d'insieme di 100 ripetizioni (nelle righe 17-20). Si osservi attentamente la forma d'onda in questa animazione mentre cambia in risposta dell'aggiunta di rumore casuale mostrato nel titolo. È possibile vedere chiaramente come il rumore inizia principalmente come rumore di quantizzazione, ma poi diminuisce rapidamente man mano che vengono aggiunte quantità piccole ma crescenti di rumore casuale prima del passaggio della media dell'insieme, quin-

Pagina | 301

di alla fine aumenta quando viene aggiunto troppo rumore. La deviazione standard ottimale del rumore casuale è circa 0,36 volte la dimensione di quantizzazione, come si può dimostrare aggiungendo quantità minori o maggiori tramite la variabile *Noise* nella riga 6 di questo script. Si noti che questo funziona solo per i segnali con insieme della media (ensembled averaged) in cui il rumore casuale viene aggiunto *prima* della quantizzazione.

Un esempio *udibile* di questa idea è illustrato dallo script Matlab/Octave [DigitizedSpeech.m](#), che inizia con una registrazione audio della frase parlata "Testing, one, two, three", precedentemente registrata a 44000 Hz e salvata sia in formato WAV ([TestingOneTwoThree.wav](#)) che in [.mat\(testing123.mat\)](#), arrotonda progressivamente l'ampiezza dei dati a 8 bit (256 passi; [link al suono](#)), mostrato nella pagina precedente, 4 bit (16 passi; [link al suono](#)) e a 1 bit (2 passi; [link al suono](#)), e poi ancora il caso a 2 passi *con l'aggiunta di rumore bianco casuale* prima dell'arrotondamento (2 passi + rumore; [link al suono](#)), disegna le forme d'onda e riproduce i suoni risultanti, mostrando sia l'effetto degradante dell'arrotondamento che il notevole miglioramento causato dall'aggiunta di rumore. (Fare clic su questi collegamenti audio per ascoltare i suoni sul computer). Sebbene il programma per computer, in questo caso *non* esegua una *esplicita* operazione di media d'insieme come fa [RoundingError.m](#), è probabile che i neuroni del centro uditivo del cervello forniscono quella funzione in virtù del loro tempo di risposta e dell'effetto memoria.

Quanto in basso si può scendere? Prestazioni con rapporti segnale/rumore molto bassi.

Questa è una simulazione di diverse tecniche descritte qui applicate alla misura quantitativa di un picco che è *sepolti in un eccessivo rumore casuale*, dove il rapporto segnale/rumore (S/N) è *inferiore a 2*. (Normalmente, è desiderabile un rapporto S/N di 3 o maggiore per un rilevamento affidabile).

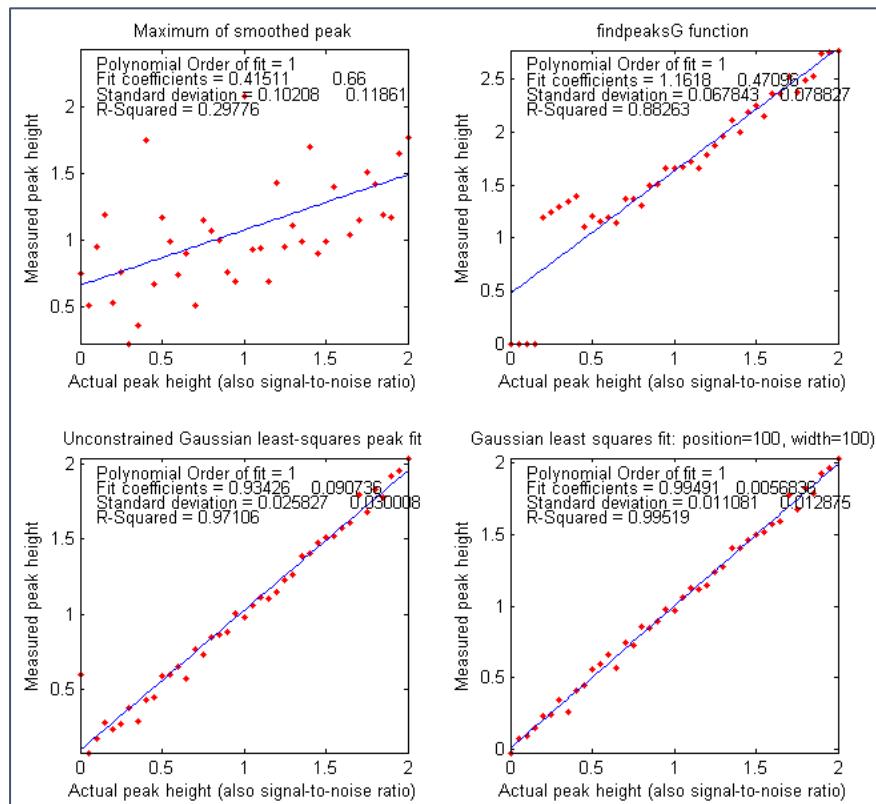
Lo script Matlab/Octave [LowSNRdemo.m](#) esegue le simulazioni e i calcoli e confronta i risultati graficamente, concentrandosi sul comportamento di ciascun metodo quando il rapporto S/N si approssima a zero. Vengono confrontati quattro metodi:

- (1) [smoothing](#), seguito dalla misura picco-picco del segnale filtrato (pag. 38);
- (2) un metodo di ricerca dei picchi basato su [findpeakG](#) (pagina 230);
- (3) [approssimazione iterativa dei quadrati minimi non vincolata](#) (INLS) basata su [peakfit.m](#) (pagina 190);
- (4) [i quadrati minimi classici vincolati](#) (CLS) basati sulla funzione [cls2.m](#) (pagina 180);

Le misure vengono eseguite su un range di altezze dei picchi per le quali il rapporto S/N varia da 0 a 2. Il [rumore](#) è casuale, costante e bianco. Ogni volta che si esegue lo script, si ottiene lo stesso set di segnali ma campioni indipendenti del rumore casuale.

I risultati per i valori iniziali nello script sono mostrati nei grafici e nella tabella della pagina successiva, entrambi creati dallo script [LowSNRdemo.m](#). I grafici a lato mostrano i grafici di correlazione tra l'altezza misurata del picco e quella reale, che idealmente dovrebbe essere una linea retta con una pendenza di 1, una intercetta di zero e un R-quadro di 1. Come si può vedere, il metodo dello smoothing più semplice (in alto a sinistra) è completamente inadeguato, con una pendenza bassa (perché lo smoothing riduce l'altezza del picco) e un'intercetta alta (perché anche il rumore filtrato ha un valore da picco a picco diverso da zero). La funzione findpeaks (in alto a destra) funzio-

na bene per altezze di picco più alte ma fallisce completamente al di sotto di un rapporto S/N di 0,5 perché l'altezza del picco scende al di sotto del valore della soglia di ampiezza. In confronto, le due tecniche dei minimi quadrati funzionano molto meglio, riportando valori molto migliori per la pendenza, l'intercetta dello zero e l'R-squared. Ma se si osserva attentamente l'estremità inferiore dell'intervallo di altezza del picco, vicino a $x = \text{zero}$, si può vedere che i valori riportati dall'approssimazione non vincolata (in basso a sinistra) occasionalmente si allontanano dalla linea, mentre l'approssimazione vincolata (in basso a destra) diminuisce gradualmente fino allo zero ogni volta che si esegue lo script. Essenzialmente il motivo per cui è anche possibile effettuare misure con rapporti S/N così bassi è che *la densità dei dati è molto alta*: cioè, ci sono molti punti in ogni segnale (circa 1000 punti attraverso la semi-larghezza del picco con i valori iniziali dello script). I risultati sono



riassunti nella tabella sottostante.

Number of points in half-width of peak: 1000

Method	Height Error	Position Error
Smoothed peak	21.2359%	120.688%
findpeaksG.m	32.3709%	33.363%
peakfit.m	2.7542%	4.6466%
cls2.m	1.6565%	

Gli errori delle altezze sono riportati come percentuale dell'altezza massima (inizialmente 2). (Per i primi tre metodi, viene misurata anche la posizione del picco e ne viene riportata l'accuratezza relativa. L'approssimazione vincolata classica dei quadrati minimi non misura la posizione del picco, ma assume piuttosto che rimanga fisso al valore iniziale di 100). Si può vedere che il metodo CLS ha un leggero vantaggio nella precisione, ma si deve anche considerare che questo metodo funziona bene solo se la forma, la posizione e la larghezza del picco sono note. Il metodo iterativo senza vincoli può tenere traccia dei cambi nella posizione e nella larghezza del picco.

È possibile modificare diversi fattori in questa simulazione per testare la robustezza dei metodi in esame. Cercare la parola 'change' nei commenti per i valori che possono essere modificati. Ridurre

MaxPeakHeight (riga 8) per rendere il problema più difficile. Modificare la posizione e/o la larghezza del picco (righe 9 e 10) per mostrare come il metodo CLS fallisce. Come al solito, più si sa, migliori saranno i risultati. Modificare l'incremento (riga 4) per modificare la densità dei dati; più dati è sempre meglio.

(Sorprendentemente, come si vedrà a pagina 329, [Calibrazione della Misura](#), non è nemmeno necessario disporre di un profilo accurato del picco per ottenere una buona correlazione tra l'altezza misurata e quella reale).

LowSNRdemo.m calcola anche lo [spettro di potenza](#) del segnale e l'ampiezza (radice quadrata della potenza) della fondamentale, dove cade la maggior parte della potenza di un ampio picco Gaussiano, e la disegna nella [Window\(2\)](#). La correlazione con l'altezza del picco è come il metodo CLS, ma l'intercetta è maggiore perché c'è una quantità di rumore diversa da zero anche in quella fetta di frequenza dello spettro di potenza.

Ora, siamo nel 21° secolo, nell'era dei "big data", in cui sistemi di dati automatizzati ad alta velocità possono acquisire, archiviare ed elaborare quantità di dati maggiori che mai. Come mostra questo piccolo esempio, maggiori quantità di dati consentono ai ricercatori di sondare più a fondo e misurare effetti minori rispetto al passato.

Elaborazione dei segnali nella ricerca dell'intelligenza extraterrestre

I problemi di rilevamento del segnale che devono affrontare coloro che cercano nel cosmo prove di civiltà extraterrestri o interessanti fenomeni naturali sono enormi. Tra questi problemi c'è il fatto che non sappiamo molto su cosa aspettarci: non sappiamo esattamente dove guardare nel cielo, o quali frequenze potrebbero essere utilizzate, o le possibili forme di trasmissioni. Inoltre, gli astronomi vogliono distinguere le molte potenti sorgenti di segnali di interferenza *terrestri* con le autentiche sorgenti *extraterrestri*. C'è anche l'enorme potenza del computer richiesta, che ha guidato lo sviluppo di hardware e software specializzati, così come, fino a poco tempo fa, il calcolo distribuito su migliaia di personal computer connessi a Internet in tutto il mondo utilizzando il vecchio screensaver computazionale [SETI@home](#). Sebbene alcune delle [tecniche computazionali](#) usate in questa ricerca siano molto più sofisticate di quelle trattate in questo libro, iniziano con i concetti di base qui trattati.

Uno dei temi ricorrenti di questo libro è stato che più si conoscono i dati, più è probabile che si ottenga una misura affidabile. Nel caso di possibili segnali extraterrestri, non sappiamo molto, ma sappiamo alcune cose.

Sappiamo che la radiazione elettromagnetica su un'ampia gamma di frequenze viene utilizzata per la trasmissione a lunga distanza sulla terra e tra la terra e i satelliti e le sonde lontane dalla terra. Gli astronomi usano già radiotelescopi per ricevere la radiazione naturale da grandi distanze. Per guardare contemporaneamente frequenze diverse, le [trasformate di Fourier](#) dei segnali originali del telescopio possono essere calcolate su più segmenti temporali. A pagina 91, è stata mostrata una [simulazione](#) che ha dimostrato quanto sia difficile vedere una componente periodica in presenza di una pari quantità di rumore casuale e tuttavia quanto sia facile è selezionarlo nello spettro delle frequenze.

Inoltre, le trasmissioni da civiltà extraterrestri potrebbero essere sotto forma di gruppi di impulsi, quindi anche la loro rilevazione e verifica fanno parte dell'elaborazione del segnale SETI. È interessante notare che le terzine e altri gruppi di impulsi equidistanti compaiono nelle trasformate di Fourier delle onde portanti ad alta frequenza che sono [modulate in ampiezza o frequenza](#) (come la [radio AM o FM](#)). Naturalmente, non c'è motivo di presumere, né di rifiutare, che le civiltà extraterrestri potrebbero usare gli stessi metodi di comunicazione dei nostri.

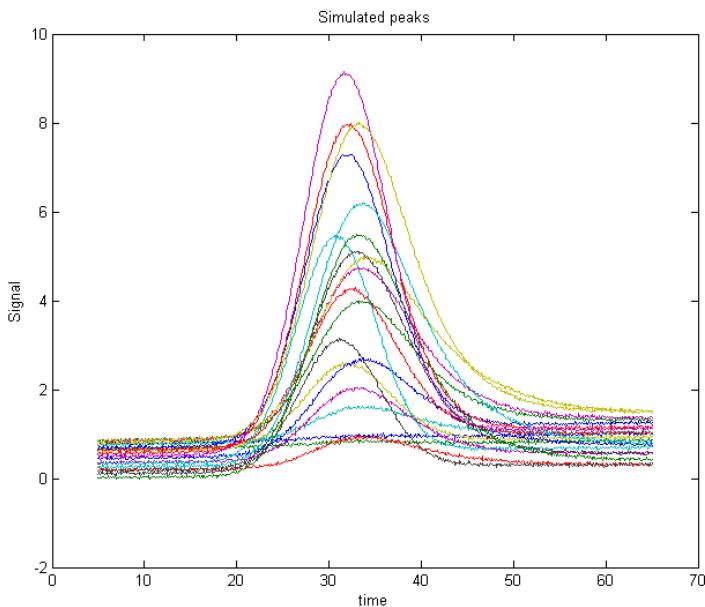
Una cosa che sappiamo per certo è che la terra ruota attorno al suo asse una volta al giorno e che ruota attorno al sole una volta all'anno. Quindi, se guardiamo in una direzione fissa fuori dalla Terra, le stelle lontane sembreranno muoversi secondo uno schema prevedibile, mentre le sorgenti terrestri rimarranno fisse sulla Terra. L'enorme parabola dell'[Osservatorio di Arecibo](#) a Porto Rico ([purtroppo non più operativa](#)) era fissata in posizione ed è stata spesso utilizzata per guardare in una data direzione per lunghi periodi di tempo. Il campo visivo di questo telescopio è tale che una sorgente puntiforme a una distanza impiega 12 secondi per passare, mentre la terra ruota. [Come dice SETI:](#)

"I segnali radio da un trasmettitore distante dovrebbero diventare più forti e poi più deboli mentre il punto focale del telescopio si sposta su quell'area del cielo. Nello specifico, la potenza dovrebbe aumentare e poi diminuire con una curva a campana ([una curva Gaussiana](#)). L'[approssimazione a una Gaussiana](#) è un test eccellente per determinare se un'onda radio è stata generata -là fuori- piuttosto che una semplice fonte di interferenza proveniente da qualche parte sulla Terra, poiché i segnali provenienti dalla Terra mostrano tipicamente schemi di potenza costanti piuttosto che curve".
is an excellent test to determine if a radio wave was generated 'out there' rather than a simple source of interference somewhere here on Earth since signals originating from Earth will typically show constant power patterns rather than curves".

Inoltre, qualsiasi picco di 12 secondi osservato può essere riesaminato con un altro punto focale spostato verso ovest per vedere se si ripete con il tempo e la durata previsti.

Sappiamo anche che ci sarà uno [spostamento Doppler](#) nelle frequenze osservate se la sorgente si muove rispetto al ricevitore; questo si osserva con le [onde sonore](#) così come con le [onde elettromagnetiche come la radio o la luce](#). Poiché la Terra ruota e gira a una velocità nota e costante, possiamo prevedere e compensare con precisione lo spostamento Doppler causato dal movimento terrestre (questo è chiamato "[de-chirping](#)" dei dati).

Perché misurare l'area anziché l'altezza del picco?



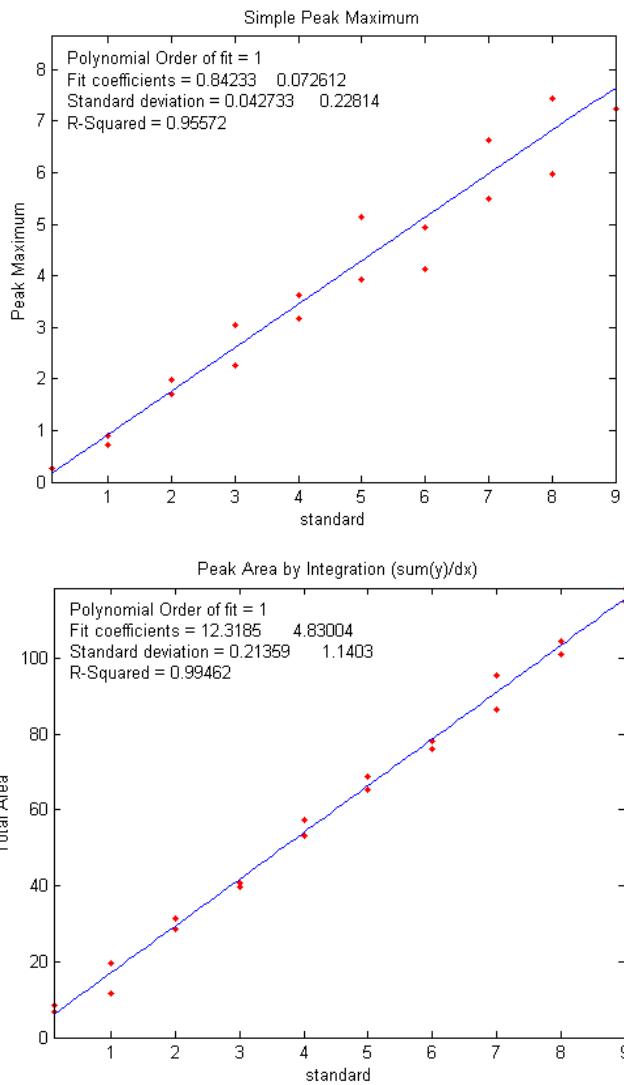
Questa simulazione esamina più da vicino la questione della misura dell'area piuttosto che dell'altezza del picco per ridurre l'effetto dell'ampliamento, che si verifica comunemente in cromatografia, per ragioni discusse [precedentemente](#), e anche in alcune forme di spettroscopia. In quali condizioni la misura dell'area del picco potrebbe essere migliore della sua altezza?

Lo script Matlab/Octave "[HeightVsArea.m](#)" simula la misura di una serie di campioni standard le cui concentrazioni sono date dal vettore 'standards'. Ogni standard produce un picco

isolato la cui altezza è direttamente proporzionale al valore corrispondente in 'standards' e il cui *profilo* è una Gaussiana con una posizione ('pos') e una larghezza ('wid') costanti. Per simulare la misura di questi campioni in condizioni tipiche, lo script modifica la forma dei picchi (mediante dilatazione esponenziale) e aggiunge una linea di base variabile e rumore casuale. È possibile controllare, tramite le definizioni delle variabili nelle prime righe dello script, l'inizio e la fine del picco, la frequenza di campionamento 'deltaX' (incremento tra i valori x), la posizione e la larghezza del picco ('pos' e 'wid'), la sequenza delle altezze dei picchi ('standards'), l'ampiezza della linea di base ('baseline') e il suo grado di variabilità ('vba'), l'entità del cambiamento di forma ('vbr') e la quantità di rumore casuale aggiunto al segnale finale ('noise').

I picchi risultanti sono mostrati nella figura sopra. Lo script prepara una serie di "[curve di calibrazione](#)" disegnando i valori di 'standard' rispetto alle altezze o alle aree dei picchi misurate per ciascun metodo di misura. I metodi di misura includono l'altezza del picco nella Window 2, l'area del picco nella Window 3 e l'altezza e l'area [approssimate](#) nelle [Window 4](#) e [5](#), rispettivamente. Questi grafici dovrebbero idealmente avere un'intercetta pari a zero e un R^2 di 1.000, ma la *pendenza* è maggiore per le misure dell'area perché questa ha unità diverse ed è numericamente maggiore dell'altezza del picco. Tutti i metodi di misura hanno la correzione della linea di base; ovvero, includono codice che tenta di compensare i cambiamenti nella linea di base (controllata dalla variabile 'baseline').

Con i valori iniziali di 'baseline', 'noise', 'vba' e 'vbr', è possibile vedere chiaramente il vantaggio delle misure dell'area (figura 3) rispetto all'altezza del picco (figura 2). Ciò è dovuto principalmente alla variabilità dell'ampliamento del profilo del picco ('vbr') e alla media del rumore casuale nel calcolo dell'area.



Window 2: misura dell'altezza del picco

Window 3: misura dell'area

Se si impostano 'baseline', 'noise', 'vba' e 'vbr' tutti a zero, si simula un mondo perfetto in cui tutti i metodi funzionano perfettamente.

L'approssimazione [Curve fitting] può misurare sia l'altezza che l'area del picco; non è nemmeno assolutamente necessario utilizzare un modello accurato della forma del picco. Utilizzando un semplice modello Gaussiano questo esempio funziona meglio per l'area ([Finestra 5](#)) che per l'altezza ([Finestra 4](#)) ma è non significativamente migliore di una semplice misura dell'area del picco (Figura 3). I migliori risultati si ottengono se si utilizza un modello Gaussiano *espanso esponenzialmente* (profilo 31 o 39) utilizzando il codice nella riga 30, ma il calcolo richiede più tempo. Inoltre, se il picco misurato si *sovrappone* ad un altro picco in modo significativo, l'approssimazione della curva di entrambi i picchi insieme può fornire risultati molto più accurati rispetto ad altri metodi di misura dell'area del picco.

Utilizzo delle macro per estendere le capacità degli spreadsheet

Sia Microsoft *Excel* che OpenOffice *Calc* possono automatizzare attività ripetitive utilizzando le "macro", sequenze di comandi o sequenze di tasti che vengono memorizzate per un uso successivo. Le macro possono essere create più facilmente utilizzando il "Registratore di macro" nativo, che guarderà letteralmente tutti i clic, selezioni e sequenze di tasti e li registrerà per riprodurli in seguito. Oppure si può scrivere o modificare le macro nel linguaggio di quel foglio di calcolo (VBA in *Excel*; Python o JavaScript in *Calc*). Oppure si possono fare entrambe le cose: si usa prima il registratore, poi si modifica manualmente il codice risultante per adattarlo. Che è quello che si fa di solito.

Per abilitare le macro in Excel, andare alla scheda **File tab > Options**. Nel riquadro di sinistra, selezionare **Trust Center**, e poi click su **Trust Center Settings**. Nella finestra di dialogo Trust Center, cliccare su **Macro Settings** a sinistra, selezionare **Enable all macros** e cliccare su OK. Quindi eseguire le operazioni sullo spreadsheet e, al termine, cliccare su **Stop Recording** e salvare lo spreadsheet. Successivamente, premendo semplicemente il tasto di scelta rapida Ctrl verrà eseguita la macro e tutte le operazioni registrate nel foglio di calcolo.

Qui si mostreranno due applicazioni in Excel utilizzando macro con la funzione Solver. (Vedere <http://peltiertech.com/Excel/SolverVBA.html#Solver2> per informazioni sulla configurazione delle macro e del solver per la propria versione di Excel).

In una sezione precedente (pagina 192) si descrive l'uso della funzione "Solver" applicata all'approssimazione iterativa di picchi sovrapposti in uno spreadsheet. I passaggi elencati possono essere facilmente acquisiti con il registratore delle macro e salvati con lo spreadsheet. Tuttavia, sarà necessaria una macro diversa per ogni diverso numero di picchi, poiché il blocco di celle che rappresenta il "Proposed Model" sarà diverso per ogni numero di picchi. Per esempio, il template ["CurveFitter2Gaussian.xlsxm"](#) include una macro chiamata 'fit' per un'approssimazione di 2 picchi, attivata premendo **Ctrl-f**. Ecco il testo di quella macro:

```
Sub fit()
'
' fit Macro
'
' Keyboard Shortcut: Ctrl+f
'

SolverOk
SetCell:="$C$12", MaxMinVal:=2, ValueOf:=0, ByChange:="$C$8:$D$9",
Engine:=1, EngineDesc:="GRG Nonlinear"
SolverSolve
End Sub
```

Si può vedere che il testo della macro utilizza solo due istruzioni: "SolverOK" e "SolverSolve". SolverOK specifica tutte le informazioni nella finestra di dialogo "Solver Parameters" nei suoi argomenti di input: 'SetCell' imposta l'obiettivo come errore di adattamento percentuale nella cella

C12, 'MaxMinVal' è impostato sulla seconda scelta (Minimum) e 'ByChange' specifica la tabella di celle che rappresenta il modello proposto (C8:D9) i cui valori devono essere modificati per ridurre al minimo l'obiettivo nella cella C12. L'ultimo argomento imposta il motore del Solver su 'GRG Nonlinear', il migliore per l'approssimazione iterativa dei picchi. Infine, "SolverSolve" avvia l'esecuzione del motore del Solver. È possibile modificare facilmente questa macro per template per approssimazioni con altri numeri di picchi semplicemente cambiando le celle a cui si fa riferimento nell'argomento 'ByChange', p.es., C8:E9 per un'approssimazione a 3 picchi. In questo caso, tuttavia, è probabilmente altrettanto facile utilizzare il registratore per una macro valida per ogni tipo di approssimazione.

Un esempio più elaborato di un foglio di calcolo che utilizza una macro è [TransmissionFittingCalibrationCurve.xls](#) ([schermata](#)) che crea una curva di calibrazione per una serie di concentrazioni standard nel metodo TFit, precedentemente descritto a [pagina 274](#). Ecco una porzione di quella macro:

```
Range("AF10").Select
Application.CutCopyMode = False
Selection.CopySelection.Copy
Range("A6").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone,
SkipBlanks _
:=False, Transpose:=False
Calculate
Range("J6").Select
Selection.CopySelection.Copy
Range("I6").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone,
SkipBlanks _
:=False, Transpose:=False
Calculate
SolverOk SetCell:="$H$6", MaxMinVal:=2, ValueOf:=0, ByChange:="$I$6", Engine:=1 _
, EngineDesc:="GRG Nonlinear"
SolverOk SetCell:="$H$6", MaxMinVal:=2, ValueOf:=0, ByChange:="$I$6", Engine:=1 _
, EngineDesc:="GRG Nonlinear"
SolverSolve userFinish:=True
SolverSolve userFinish:=True
SolverSolve userFinish:=True
Range("I6:J6").Select
Selection.CopySelection.Copy
Range("AG10").Select
```

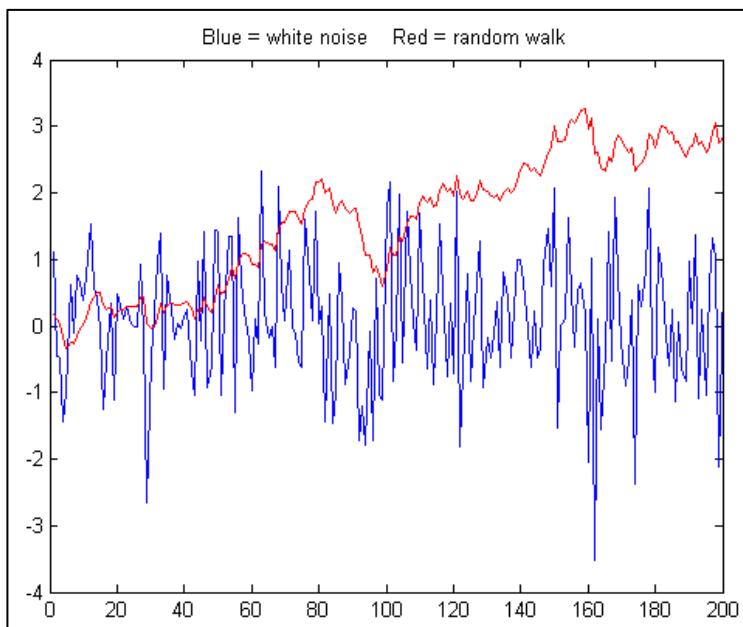
```
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone,  
SkipBlanks _  
:=False, Transpose:=False
```

La macro in questo foglio di calcolo ripete questo pezzo di codice più volte, una per ogni concentrazione nella curva di calibrazione (cambiando solo "AF10" nella prima riga per raccogliere una concentrazione diversa dalla "Results table" nella colonna AF). Questa macro utilizza diverse istruzioni aggiuntive per selezionare gli intervalli ("Range...Select"), copiare ("Selection.Copy") e incollare ("Selection.PasteSpecial Paste:=xlPasteValues") i valori da un posto all'altro e ri-calcolare lo spreadsheet ("Calculate"). Ogni clic, selezione di menù o ogni pressione di un tasto, si creano una o più righe di testo nella macro. La sintassi è prolissa ma abbastanza esplicita e chiara; si può imparare un bel po' semplicemente registrando varie azioni e guardando il testo della macro risultante. O almeno questo è un modo per farlo.

Vale la pena notare che da dicembre 2022 è disponibile un ulteriore modo per creare funzioni e macro di Excel: utilizzando il chatbot conversazionale di intelligenza artificiale ChatGPT (pagina 434). Una volta ottenuto un account gratuito su <https://chat.openai.com/>, si può semplicemente chiedere "Scrivi una macro Excel in..." e descrivere chiaramente ciò che si desidera, facendo riferimento alle celle e a range nel solito modo e ChatGPT digiterà una soluzione che puoi semplicemente copiare e incollare nel foglio di lavoro Excel. [Un aiuto ulteriore](#).

Passeggiate aleatorie (random walk) e correzione della linea di base

La *random walk* è stata citata nella sezione su [Segnali e rumore](#) (pag. 22) come un tipo di rumore a bassa frequenza ("rosa"). [Wikipedia](#) recita: "una passeggiata aleatoria (random walk) è la formalizzazione dell'idea di prendere passi successivi in direzioni casuali. Per esempio, il percorso di una molecola mentre viaggia in un liquido o in un gas, il percorso di un animale in cerca di cibo, il comportamento delle superstringhe, il prezzo fluttuante di un'azione e la situazione finanziaria di uno speculatore, possono essere tutti modellati come passeggiate aleatorie, anche se in realtà potrebbero non essere veramente casuali". Le passeggiate aleatorie descrivono e fungono da modello per molti tipi di comportamento instabile. Mentre i rumori bianco, $1/f$ e blu sono legati a un valore medio a cui tendono a tornare, le passeggiate aleatorie sono per lo più senza meta e spesso si allontanano in una o nell'altra direzione, forse per non tornare mai più. Matematicamente, una passeggiata aleatoria può essere modellata come la somma cumulativa di un processo casuale, ad esempio la funzione 'randn'. Il grafico seguente confronta un campione di 200 punti di rumore bianco (calcolato



come 'randn' e mostrato in blu) con una passeggiata aleatoria (calcolata come somma cumulativa, 'cumsum', e mostrato in rosso). Entrambi i campioni sono scalati per avere la stessa deviazione standard, ma il loro comportamento è molto diverso. La passeggiata aleatoria ha un comportamento molto più a bassa frequenza e in questo caso si allontana oltre la gamma di ampiezza del rumore bianco. Questo tipo di comportamento casuale è molto distruttivo per il processo di misurazione, distorcendo le forme dei picchi e causando lo spostamento delle linee di base, rendendole difficili da definire. Peggio ancora, non può essere ridotto in modo significativo mediante lo smoothing (come mostrato da [NoiseColorTest.m](#)). In questo particolare esempio, la 'random walk' ha una pendenza complessivamente positiva e una "protuberanza" vicino al centro che potrebbe essere confusa per un vero picco di segnale (non lo è; è solo rumore casuale). Ma un altro campione potrebbe avere un comportamento molto diverso. Sfortunatamente, questo comportamento viene talvolta osservato nei segnali sperimentali.

Per mostrare le difficoltà delle misure, lo script [RandomWalkBaseline.m](#) simula un picco Gaussiano con posizione e larghezza variabili casualmente, su una linea di base di tipo 'passeggiata aleatoria', con un rapporto segnale-rumore relativamente buono di 15. Il picco viene misurato con i metodi di approssimazione dei minimi quadrati utilizzando [peakfit.m](#) con due diversi metodi di correzione della linea di base per gestire la passeggiata aleatoria:

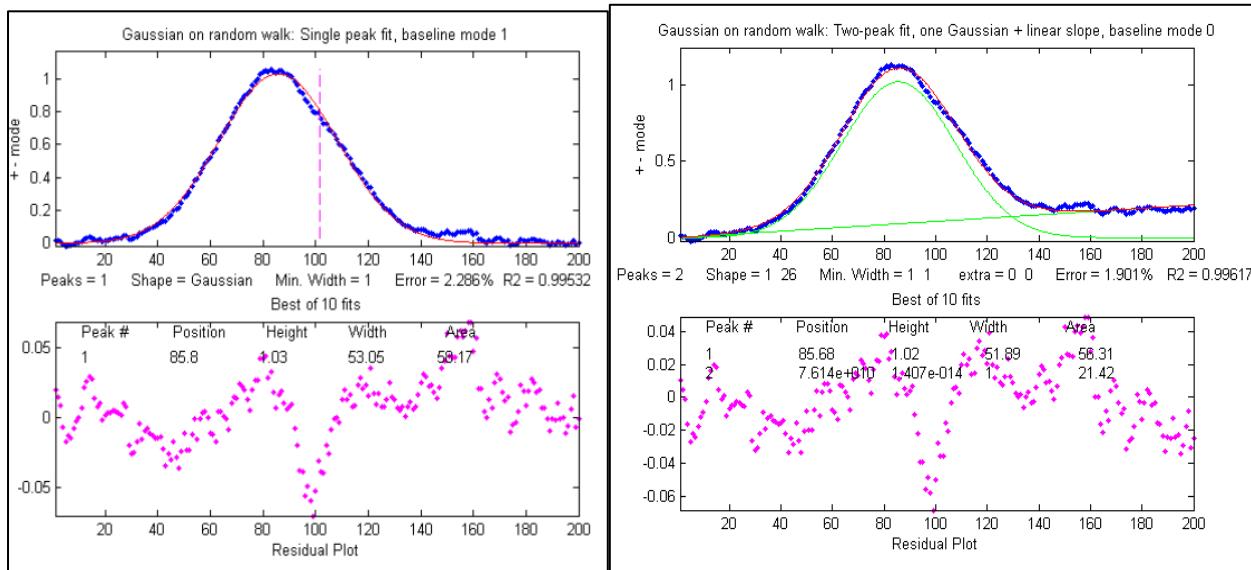
(Metodo *a*) un modello Gaussiano mono-componente (profilo 1) con BaselineMode impostato a 1 (il che significa che una linea di base lineare viene prima interpolata dai bordi

del segmento di dati e poi sottratta dal segnale): `peakfit([x;y],0,0,1,1,0,10,1);`

(Metodo *b*) un modello a 2 componenti, il primo una Gaussiana (profilo 1) e il secondo una pendenza lineare (profilo 26), con BaselineMode impostato a 1 :

`peakfit([x;y],0,0,2,[1 26],[0 0],10,0).`

In questo caso, l'errore di approssimazione è inferiore per il secondo metodo, specialmente se il picco cade vicino ai bordi dell'intervallo dei dati.



Ma gli errori percentuali relativi dei parametri mostrano che il *primo metodo* fornisce un errore inferiore per posizione e la larghezza, almeno in questo caso. In media, i parametri del picco sono più o meno gli stessi.

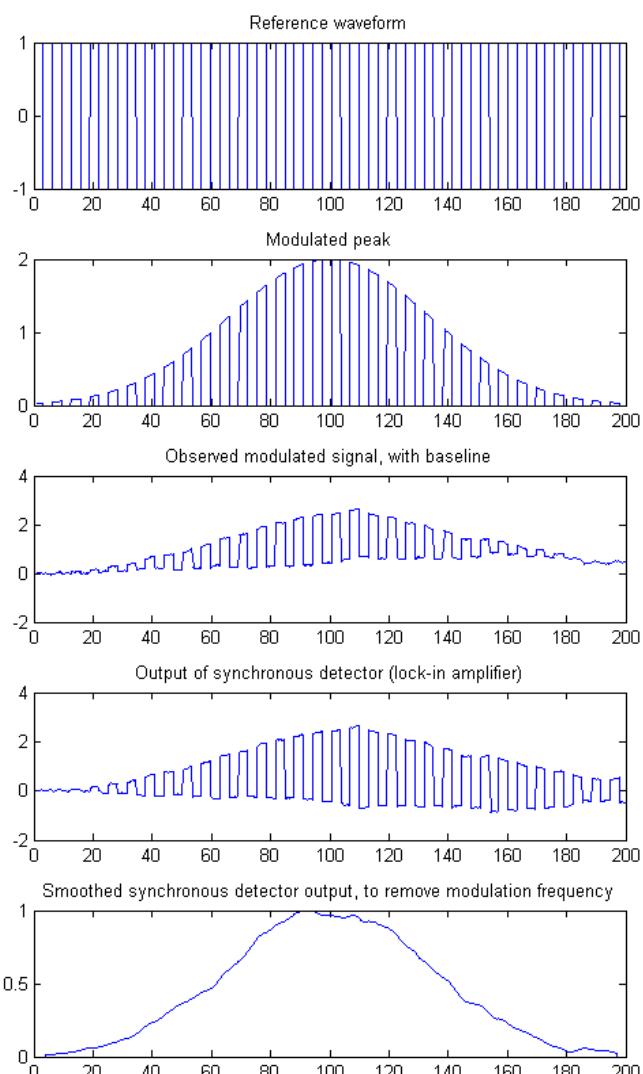
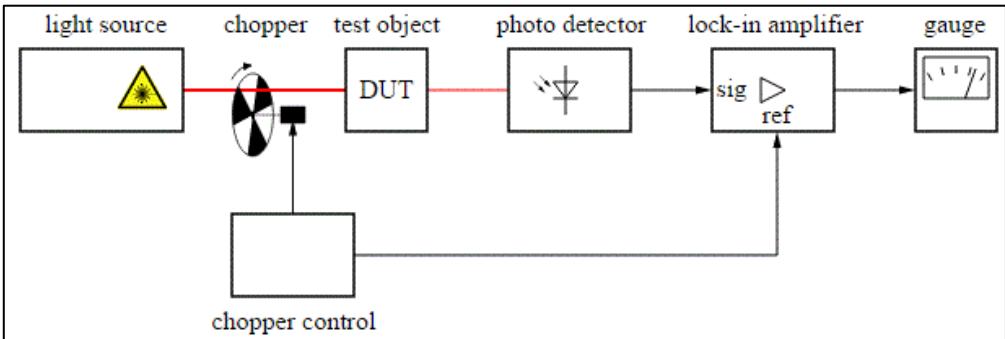
Position Error	Height Error	Width Error
Method a: 0.2772	3.0306	0.0125
Method b: 0.4938	2.3085	1.5418

Lo si può confrontare con [WhiteNoiseBaseline.m](#) che ha un segnale e un rapporto S/N simili, tranne per il fatto che il rumore è *bianco*. È interessante notare che *l'errore di approssimazione* col rumore bianco è *maggior*, ma gli *errori dei parametri* (posizione, altezza, larghezza e area del picco) sono *più bassi*, e i residui sono più casuali e hanno meno probabilità di produrre falsi picchi di rumore. Questo perché il rumore della 'random walk' è [molto concentrato alle basse frequenze](#) dove normalmente si trovano le frequenze del segnale, mentre il rumore bianco ha anche una notevole potenza alle *frequenze più alte*, che *aumenta l'errore di approssimazione* ma danneggia relativamente poco l'accuratezza della misura del segnale. Questo può essere contro intuitivo, ma è importante rendersi conto che l'errore di approssimazione non è *sempre* correlato all'errore del parametro del picco. In conclusione: la passeggiata aleatoria è fastidiosa.

A seconda del tipo di esperimento, un progetto strumentale basato su [tecniche di modulazione](#) può aiutare e una misura multipla con la [media di insieme](#) può aiutare con qualsiasi tipo di rumore casuale imprevedibile, discusso nella sezione successiva.

Modulazione e rilevamento sincrono.

In alcuni progetti sperimentali può essere utile applicare la tecnica della modulazione, in cui una delle variabili indipendenti controllate viene fatta oscillare in modo periodico, per poi rilevare l'oscillazione risultante nel segnale misurato. Uno strumento così congegnato può ridurre o eliminare alcuni tipi di rumore e derive.



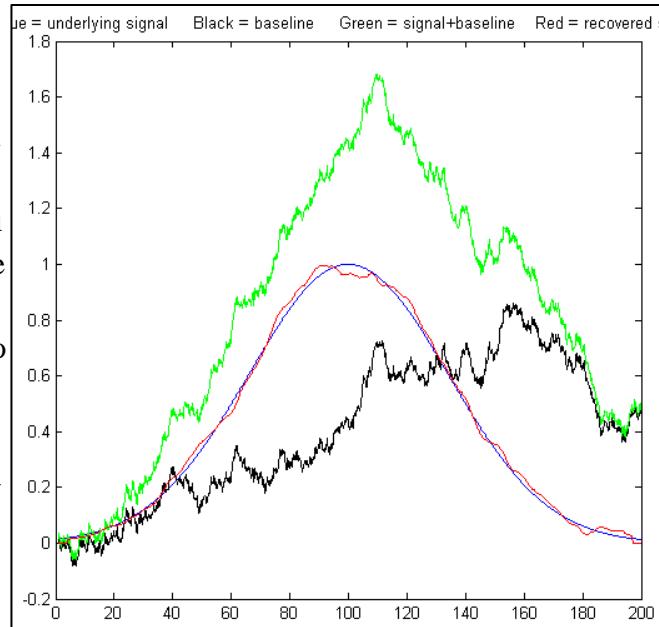
dì riferimento di sincronizzazione direttamente dal chopper per garantire la sincronizzazione anche se la frequenza dovesse variare (Cfr. L'amplificatore lock-in è talvolta visto come una "scatola nera" con possibilità quasi magiche, ma in realtà esegue un'operazione piuttosto semplice (ma molto utile), come mostrato in questa simulazione. (vedere la simulazione interattiva su <https://terpconnect.umd.edu/~toh/models/lockin.html> e descritto in T. C. O'Haver, "Lock-in Amplifiers," J. Chem. Ed. 49, March and April (1972).

AmplitudeModulation.m è uno script Matlab/Octave che simula la modulazione e il rilevamento sincrono, in cui il segnale creato quando il raggio di luce scansiona il campione in esame viene modellato con una banda Gaussiana ('y'), i cui parametri sono definiti nelle prime righe dello script. Il grafico della pagina precedente confronta il segnale nei diversi punti dell'apparato. Quando la lun-

Un semplice esempio sono i sistemi di misurazione ottica come quello nella foto sopra. Una sorgente di luce illumina un oggetto di prova (DUT = "Device Under Test") e la luce proveniente dall'oggetto di prova viene misurata dal fotorilevatore. A seconda dell'obiettivo dell'esperimento e della disposizione delle parti, il rilevatore potrebbe misurare la luce *trasmessa, riflessa, diffusa o eccitata* dal raggio. Un otturatore [chopper] ottico interrompe rapidamente e ripetutamente il raggio di luce che cade sull'oggetto di prova in modo che il fotorilevatore veda un segnale oscillante e il successivo sistema elettronico è progettato per misurare *solo* la componente oscillante ignorando la componente costante (corrente continua). Il vantaggio di questa disposizione è che qualsiasi interferenza introdotta *dopo* il chopper - come la luce costante emessa dallo stesso oggetto in esame o quella ambientale che proviene dall'esterno o qualsiasi segnale di fondo costante generati dal fotorilevatore stesso - *non oscillano in sincronia con il chopper* e quindi vengono rifiutati. Funziona meglio se l'elettronica è *sincronizzata* con la frequenza dell'otturatore. Questa è esattamente la funzione dell'amplificatore lock-in, che riceve un segnale

ghezza d'onda del raggio di luce che colpisce l'oggetto sotto test viene scansionata lentamente, il raggio di luce viene periodicamente interrotto dall'elica rotante, rappresentato come un'onda quadra definita dal 'riferimento' del vettore bipolare, che passa tra +1 e -1, mostrato nel pannello in alto a sinistra. Questa è detta *modulazione d'ampiezza*. La frequenza di modulazione è molte volte più veloce della velocità con cui viene scansionata la lunghezza d'onda. La luce che emerge dal campione, quindi, mostra una Gaussiana finemente 'tagliuzzata' ('my'), mostrata nel secondo pannello a sinistra. Ma il *segnaletotale* visto dal rivelatore include anche un background instabile introdotto *dopo* la modulazione ('omy'), come la luce emessa dal campione stesso o il background del rivelatore, che in questa simulazione è modellato come una 'random walk' (pag. 310), che distorce gravemente il segnale, mostrato nel terzo pannello. Il segnale del rilevatore viene quindi inviato a un amplificatore lock-in sincronizzato con la forma d'onda di riferimento. L'azione essenziale del lock-in è moltiplicare il segnale per la forma d'onda bipolare di riferimento, *invertendo il segnale* quando la luce è *spenta* e *facendolo passare invariato* quando la luce è *accesa*. Questo fa sì che il segnale di fondo non modulato venga convertito in un'onda quadra bipolare, mentre il segnale modulato non viene influenzato perché è "spento" quando il segnale di riferimento è negativo. Il risultato ('dy') è mostrato nel 4° pannello. Infine, questo segnale viene [filtrato con un passa-basso](#) dall'ultimo stadio dell'amplificatore lock-in per rimuovere la frequenza di modulazione, risultando nel picco del segnale recuperato "sdy" mostrato nel pannello inferiore. In effetti, la modulazione trasforma il segnale in una frequenza più alta ('frequency' nella riga 44) dove il rumore ponderato a bassa frequenza sulla linea di base (riga 50) è meno intenso.

Questi diversi segnali vengono confrontati nella figura a lato. Il segnale del picco Gaussiano ('y') in esame è mostrato con una linea blu, e il background contaminante ('baseline') in nero, in questo caso, modellato come una 'random walk'. Il segnale totale ('oy') che sarebbe stato visto dal rilevatore se non fosse stata utilizzata la modulazione è mostrato in verde. La distorsione del segnale è evidente e qualsiasi tentativo di misurare il picco in quel segnale sarebbe notevolmente errato. Il segnale 'sdy' recuperato dal sistema di modulazione e lock-in viene mostrato in rosso e sovrapposto al picco del segnale originale 'y' in blu per confronto. Il fatto che la linea blu e quella rossa siano così vicine l'una all'altra indica fino a che punto questo metodo ha successo. Per fare un confronto più quantitativo, questo script utilizza anche la funzione [peakfit.m](#), che impiega un metodo dei minimi quadrati per misurare i parametri nel segnale totale non modulato originale (linea verde) e nel segnale recuperato modulato (verde) e nel segnale recuperato modulato (blu) e per calcolare l'errore percentuale relativo di posizione, larghezza e larghezza del picco con entrambi i metodi:



SignalToNoiseRatio = 4

Relative % Error: Position Height Width

Original: 8.07 23.1 13.7

Modulated: 0.11 0.22 1.01

Ogni volta che lo si esegue si otterrà lo *stesso segnale del picco* ma un background di tipo 'random walk' molto *diverso*. Il rapporto S/N varierà da circa 4 a 9. Non è raro vedere un *miglioramento di 100 volte* nella precisione dell'altezza con la modulazione, come nell'esempio mostrato qui. (Volendo, si possono modificare i parametri del segnale e il livello di rumore nelle prime righe del codice di questa simulazione. Per una sfida ancora maggiore, cambiare la riga 47 in "baseLine=10.*noise + cumsum(noise); " per rendere il rumore un mix di bianco e una deriva random walk, che si traduce in un pessimo segnale; si può vedere che il rumore bianco passa attraverso il rilevatore sincrono ma viene ridotto dal filtro passa basso di smoothing nell'ultimo stadio). In effetti, il filtro passa-basso determina la larghezza di banda della frequenza del sistema lock-in, ma aumenta anche il tempo di risposta alle variazioni del gradino (come nell'eSEMPIO DEL CODICE MORSE).

Questo miglioramento della precisione della misura funziona solo perché l'errore casuale dominante, in questo caso, è:

- (a) introdotto *dopo* la modulazione, e
- (b) un rumore prevalentemente a *bassa* frequenza.

Se il rumore fosse *bianco*, non ci sarebbe *alcun* miglioramento, perché il rumore bianco è lo stesso a tutte le frequenze; infatti ci sarebbe una leggera riduzione della precisione perché il chopper blocca mediamente metà della luce. Se il campione (dispositivo sotto test) genera un picco di "assorbimento" che inizia con un valore positivo e poi scende a un valore inferiore prima di tornare, l'uscita demodulata sarà un picco negativo anziché positivo (vedere AmplitudeModulationAbsorption.m).

In un sistema sperimentale interfacciato con un computer, si potrebbe non aver bisogno di un amplificatore lock-in fisico. È possibile simulare l'effetto via software, come viene fatto in questa simulazione. È sufficiente digitalizzare sia il segnale di campionamento modulato che quello di riferimento di modulazione, quindi invertire il segnale totale ogni volta che il segnale di riferimento è "spento".

In alcune applicazioni spettroscopiche un altro tipo di modulazione utile è la "modulazione a lunghezza d'onda", in cui la *lunghezza d'onda* della luce oscilla sulla regione della lunghezza d'onda di un picco di emissione o picco di assorbimento nello spettro (riferimento 32); questo è stato spesso utilizzato nell'emissione atomica e nella spettroscopia di assorbimento (riferimenti 25,26) e nella spettroscopia laser a diodi sintonizzabili applicata alla misura di gas come metano, vapore acqueo e anidride carbonica, specialmente nel telerilevamento, dove il campione può essere lontano dal rivelatore. Meno comunemente tecniche di modulazione sono applicate anche in elettrochimica e in spettroelettrochimica in "AC" (corrente alternata).

La misura di un picco sepolto

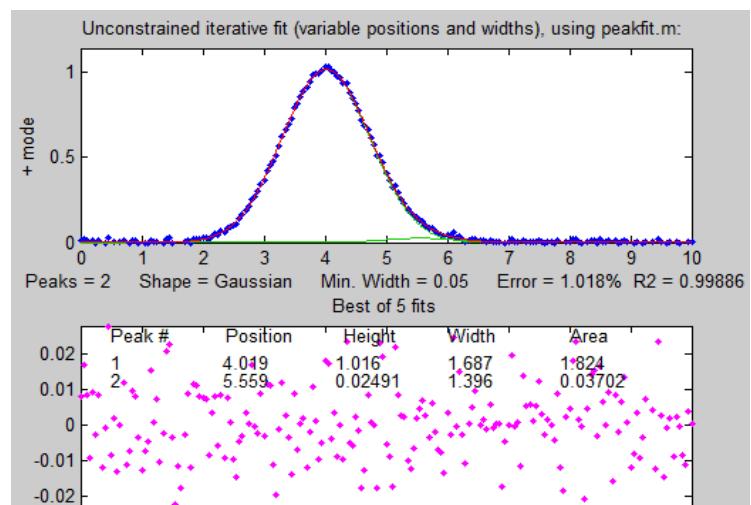
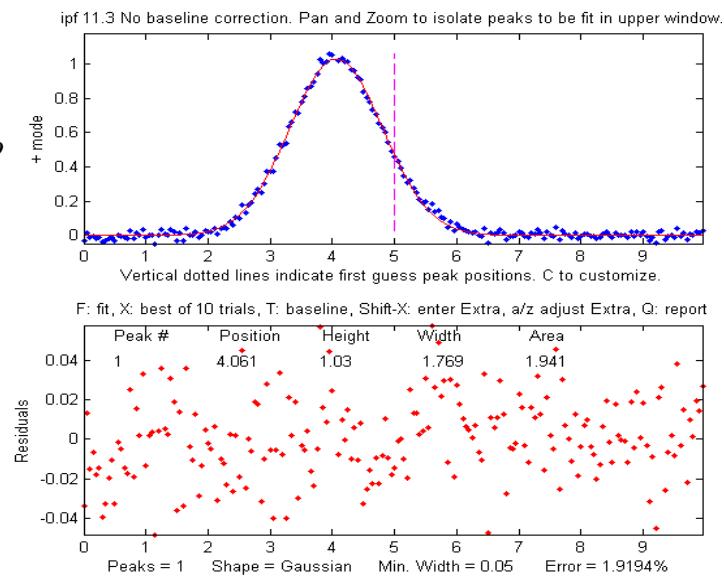
Questa simulazione esplora il problema della misura dell'altezza di un picco piccolo (un "picco figlio") che è sepolto nella coda di un picco sovrapposto molto più forte (un "picco genitore"), nel

caso particolarmente impegnativo che il picco più piccolo *non sia nemmeno visibile* a occhio nudo. Verranno esplorati tre diversi strumenti di misura: [iterativo dei quadrati minimi](#), [regressione classica dei quadrati minimi](#) e [rilevamento del picco](#), utilizzando gli strumenti Matlab/Octave [peakfit.m](#), [cls.m](#) e [findpeaksG.m](#), rispettivamente. (In alternativa, si possono utilizzare gli [spreadsheet](#) corrispondenti).

In questo esempio il picco più grande si trova in $x = 4$ e ha un'altezza di 1.0 e una larghezza di 1.66; il picco misurato più piccolo si trova in $x = 5$ e ha un'altezza di 0,1; entrambi hanno una larghezza di 1,66. Naturalmente, ai fini di questa simulazione, pretendiamo di non conoscere necessariamente tutti questi fatti e cercheremo di trovare metodi che estraggano tali informazioni il più possibile dai dati, anche se il segnale è rumoroso. Il picco misurato è sufficientemente piccolo e abbastanza vicino al picco sovrapposto più forte (separato da un valore inferiore alla larghezza dei picchi) in modo da *non formare mai un massimo* nel segnale totale. Quindi *sembra* che ci sia *un solo* picco, come mostrato nella figura a lato. Per questo motivo, la funzione `findpeaks.m` (che trova automaticamente i massimi dei picchi) non sarà utile da sola per individuare il picco più piccolo. Metodi più semplici per rilevare il secondo picco non riescono nemmeno a fornire un modo per misurare il secondo picco più piccolo, come l'ispezione delle derivate del segnale (lo smoothing della derivata quarta mostra [alcune prove di asimmetria](#), ma ciò potrebbe semplicemente essere dovuto alla forma del picco più grande), o l'[auto-deconvoluzione di Fourier](#) per restringere i picchi in modo che siano distinguibili, ma è improbabile che abbia successo con così tanto rumore. I metodi dei minimi quadrati funzionano meglio quando il rapporto segnale/rumore è scarso e possono essere ottimizzati per utilizzare le informazioni o i vincoli disponibili, come verrà dimostrato di seguito.

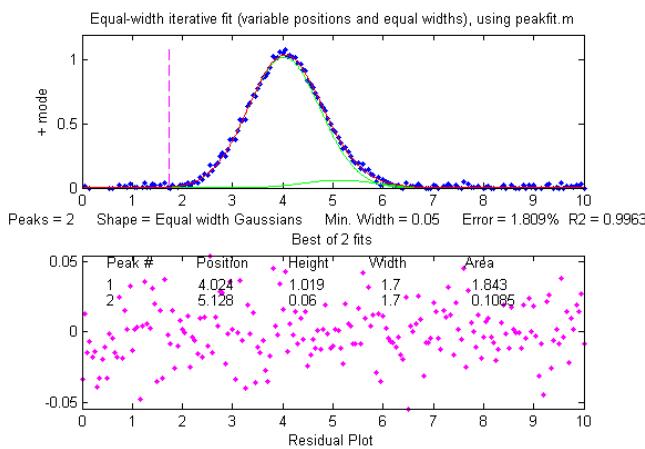
La scelta del metodo migliore dipenderà da quanto si conosce del segnale e dai vincoli che possono essere imposti; questo dipenderà dalla conoscenza del segnale sperimentale. In questa simulazione (eseguita dallo script Matlab/Octave [SmallPeak.m](#)), il segnale è composto da due picchi Gaussiani (anche se, volendo, si possono modificare nella riga 26). La prima domanda è: c'è più di un picco lì? Se si esegue un'approssimazione iterativa non vincolata di una *singola* Gaussiana dei dati, come nella figura a lato, che mostra poca o nessuna evidenza di un secondo picco - i residui sembrano casuali. (Se si potesse ridurre il rumore, o se si calcola la [media dell'insieme](#) anche di appena 10 ripetizioni del segnale, il rumore sarebbe abbastanza basso da vedere la [prova di un secondo picco](#)). Tuttavia, così com'è, non c'è nulla che salti fuori per suggerire un secondo picco.

Ma supponiamo di sospettare che *ci sia* un altro picco della stessa forma Gaussiana appena sul lato destro del picco più grande. Possiamo provare ad approssimare una *coppia* di Gaussiane ai dati (figura a lato), ma in questo caso, *c'è così tanto rumore casuale che l'approssimazione non è stabile*. Quando si esegue [SmallPeak.m](#), lo script esegue 20 approssimazioni ripetute ("NumSignals" nella riga 20) con gli stessi

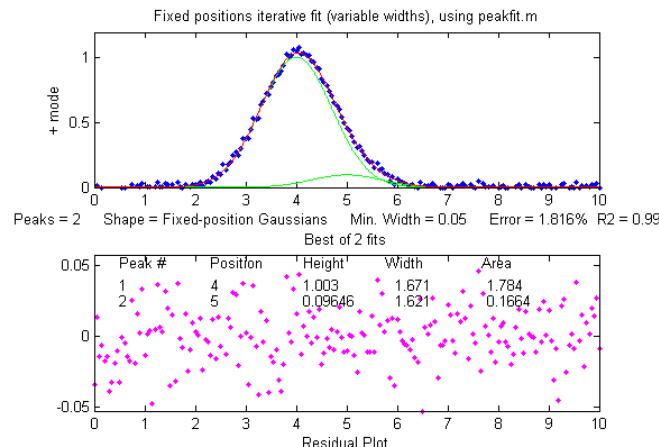


picchi in esame ma con 20 diversi campioni di rumore casuali, rivelando la stabilità (o instabilità) di ciascun metodo di misura. I picchi approssimati nella Window 1 rimbalzano ovunque durante l'esecuzione dello script. (Se l'animazione non è visibile, cliccare su [questo link](#)). L'errore di approssimazione è in media *inferiore* rispetto all'approssimazione con una singola Gaussiana, ma ciò di per sé non significa che i parametri così misurati saranno affidabili; potrebbe semplicemente che si sta "approssimando il rumore". *Se fosse isolato da solo*, il picco piccolo avrebbe un [rapporto S/N di circa 5](#) e se ne potrebbe essere misurare l'altezza con una precisione di circa il 3%, ma la presenza del picco disturbatore maggiore rende la misura molto più difficile. (Suggerimento: Dopo aver eseguito SmallPeak.m la prima volta, allargare tutte le finestre delle figure in modo che si possano vedere singolarmente senza sovrapporsi. In questo modo si può confrontare più facilmente la stabilità dei diversi metodi).

Ma supponiamo di avere motivo di aspettarci che i *due picchi abbiano la stessa larghezza*, ma non sappiamo quale potrebbe essere la larghezza. Potremmo provare un'approssimazione di Gaussiane *a larghezze uguali* (profilo #6, mostrato nella Window 2 di Matlab/Octave); l'approssimazione risultante è molto più stabile e mostra che un piccolo picco si trova a circa $x = 5$ a destra del picco più grande, mostrato in basso a sinistra. D'altra parte, se conosciamo in anticipo le *posizioni* del picco, ma non le larghezze, possiamo usare un'approssimazione Gaussiana *a posizione fissa* (profilo #16) mostrata a destra (Window 3). Nella situazione molto comune in cui l'obiettivo è misurare una *concentrazione ignota* di un componente *sconosciuto*, è possibile preparare campioni standard in cui la concentrazione del componente ricercato è abbastanza alta da poter determinare con certezza la sua posizione o la larghezza.

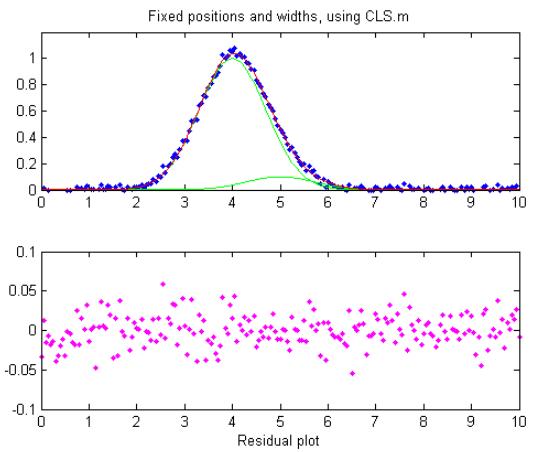


<https://terpconnect.umd.edu/~toh/spectrum/FixedPositions.png>



Finora tutti questi esempi hanno utilizzato l'approssimazione iterativa del picco con almeno un parametro (posizione e/o larghezza) sconosciuto e determinato dalla misura. Se, d'altra parte, *tutti* i parametri del picco sono noti *eccetto* l'altezza, allora può essere impiegata la [regressione dei minimi quadrati classica](#) (CLS) più veloce e più diretta

(Window 4). In questo caso, è necessario conoscere la posizione e l'ampiezza sia dei picchi di interferenza misurati che di quelli più grandi (il computer calcolerà le loro altezze). Se le posizioni e le altezze sono davvero costanti e note, questo metodo offre la migliore stabilità e precisione della misura. È anche computazionalmente più veloce, il che potrebbe essere importante se si hanno molti dati da elaborare automaticamente.



Il problema col CLS è che non riesce a fornire misure accurate se la posizione del picco e/o la larghezza cambia senza preavviso, mentre due dei metodi iterativi (approssimazioni Gaussiane non vincolate e Gaussiane di uguale larghezza) possono adattarsi a tali cambiamenti. Alcuni esperimenti sono abbastanza comuni per avere piccoli, inaspettati cambiamenti nella posizione del picco, specialmente nella cromatografia o in altre misure basate sul flusso, causati da cambiamenti inaspettati di temperatura, pressione, portata o altri fattori strumentali. In SmallPeaks.m, tali spostamenti dell'asse x possono essere simulati utilizzando la variabile "xshift" nella riga 18. Inizialmente è zero, ma se lo si imposta a qualcosa di più grande (ad esempio 0,2) si scoprirà che l'approssimazione Gaussiana di uguale larghezza (Window 2) funziona meglio, perché può tenere il passo con i cambiamenti negli spostamenti dell'asse x.

Ma con uno spostamento dell'asse x maggiore (`xshift = 1.0`) anche l'approssimazione a 'larghezza uguale' ha dei problemi. Tuttavia, se conosciamo la *separazione* tra i due picchi, è possibile utilizzare la funzione `findpeaksG` per cercare e individuare il picco *più grande* e utilizzarlo per calcolare la posizione di quello *più piccolo*. Quindi il metodo CLS, con le posizioni dei picchi così determinate per ogni segnale separato, mostrato nella Window 5 ed etichettato "findpeaksP" nella tabella sottostante, funziona meglio. In alternativa, un altro modo per utilizzare i risultati di `findpeaks` è una variazione del metodo di approssimazione iterativa di uguale larghezza in cui le prime posizioni dei picchi ipotizzati (riga 82) vengono derivate dai risultati di `findpeaks`, mostrati nella Window 6 ed etichettati "findpeaksP2" nel tabella che segue. Questo metodo non dipende dalla conoscenza accurata delle larghezze, ma solo dalla loro uguaglianza.

Ogni volta che si esegue SmallPeaks.m, *tutti questi metodi vengono calcolati* più volte ("NumSignals", impostato nella riga 20) e confrontati in una tabella che fornisce l'accuratezza media dell'altezza del picco di tutte le ripetizioni:

```
xshift=0

Unconstr. EqualW FixedP FixedP&W findpeaksP findpeaksP2

35.607 16.849 5.1375 4.4437 13.384 16.849
```

```
xshift=1

Unconstr. EqualW FixedP FixedP&W findpeaksP findpeaksP2

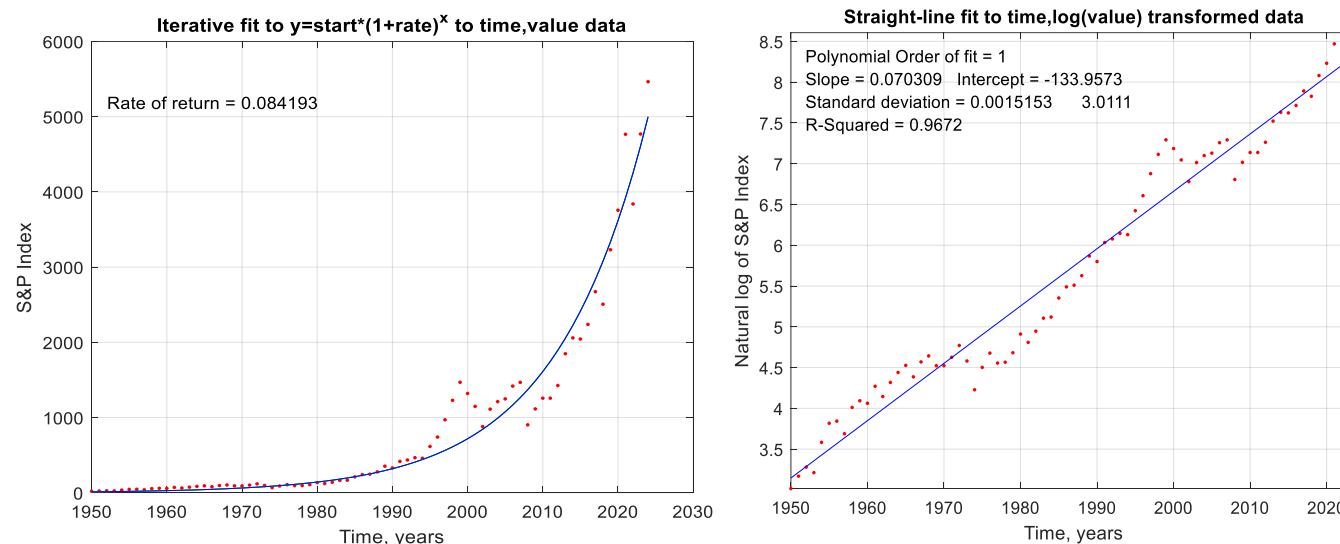
31.263 44.107 22.794 46.18 10.607 10.808
```

La conclusione è questa: più si conoscono i segnali in esame, meglio si possono misurare. Un segnale stabile con posizioni e ampiezze *note* è quello misurabile con più precisione in presenza di rumore casuale ("FixedP&W"), ma se le posizioni o le larghezze variano da misura a misura, si devono utilizzare metodi diversi e la precisione viene ridotta perché la maggior parte delle informazioni disponibili vengono utilizzate per tenere conto dei modifiche *diverse* da quelle che si vogliono misurare.

Segnale e Rumore nel Mercato Azionario (aggiornato)

Dal punto di vista del rapporto segnale/rumore, il mercato azionario è un esempio interessante. Un mercato azionario nazionale o globale è un'aggregazione di un gran numero di acquirenti e venditori di azioni di società quotate in borsa. Sono descritti dagli *indici* del mercato azionario, calcolati come media ponderata di molte azioni selezionate. Per esempio, l'[indice S&P 500](#) viene calcolato dalle valutazioni delle azioni di 500 grandi società statunitensi. Milioni di individui e organizzazioni partecipano all'acquisto e alla vendita di azioni quotidianamente, quindi l'indice S&P 500 è un prototipo di conglomerato di "big data", che riflette il valore complessivo di 500 delle più grandi società nel più grande mercato azionario della terra. Le singole azioni possono fallire o diminuire drasticamente di valore, ma gli indici di mercato fanno la media delle prestazioni di centinaia di società.

Un grafico del valore annuale, V, dell'[indice S&P 500 rispetto al tempo](#), T, per il periodo di 75 anni



dal 1950 al 2024 è mostrato nei grafici seguenti.

Ogni grafico contiene 75 punti, uno per ogni anno, mostrati in rosso. Il grafico a sinistra traccia il valore V su coordinate lineari e quello a destra traccia il *logaritmo naturale* di V, ln(V). Esistono notevoli fluttuazioni su e giù nel valore nel corso del tempo che possono essere collegate a eventi storici: la crisi petrolifera degli anni '70, il boom e il crollo tecnologico del 2000, la crisi dei mutui subprime del 2008, le guerre commerciali del 2019 e la pandemia di coronavirus del 2020. Tuttavia, la tendenza *a lungo termine* del valore è al rialzo: nel 2024 il valore era *oltre 250 volte maggiore* rispetto al suo valore nel 1950. Questo è fondamentalmente il motivo per cui le persone investono nel mercato azionario, perché in media, *nel lungo periodo*, i valori delle azioni alla fine salgono, quasi sempre più dell'inflazione (che è stata di circa il 3,5% annuo dal 1950).

Il modo più comune per modellare questo aumento complessivo a lungo termine nel tempo si basa sull'[equazione dell'interesse composto](#) che prevede la crescita degli investimenti che hanno un tasso di rendimento costante, come conti di risparmio o certificati di deposito:

$$V = S * (1 + R)^T$$

dove V è il valore, S è il valore iniziale, R è il tasso di rendimento annuale e T è il tempo. Di per sé, questa espressione produrrebbe una curva liscia esponenziale verso l'alto, senza picchi e avvallamenti. Possiamo utilizzare l'approssimazione della curva (pag. 154 e 190) per determinare quanto bene questo modello si adatti all'S&P effettivo. Questo può essere fatto in due modi:

- (1) utilizzando direttamente il [metodo iterativo dell' approssimazione](#), mostrato a destra sopra, oppure
- (2) prendendo il [logaritmo dei valori](#) e approssimando una *linea retta* ai dati trasformati, mostrati in alto a destra.

Le approssimazioni migliori ai dati S&P appaiono nei grafici dalle linee blu filtrate con smoothing. [FitSandPto2024.m](#) è uno script Matlab/Octave che esegue entrambi i calcoli utilizzando i dati in [SandPfrom1950.mat](#) o in [SandPfrom1950.xlsx](#). Quando applicato ai dati dell'indice S&P 500, il tasso di rendimento annuo R è di circa 0,08 (o 8%), ma è interessante notare che questi due metodi danno risultati leggermente *differenti*, anche se per entrambi vengono utilizzati *gli stessi dati*. Inoltre, *entrambi* i metodi producono lo *stesso* tasso se applicati a *dati sintetici* senza rumore calcolati dall'espressione precedente. Come può essere? Questa differenza tra i metodi è causata dalle irregolarità nei dati azionari che si discostano da una linea uniforme - in altre parole, il *rumore* - ed è aggravata dall'ampio intervallo dei valori di V nel tempo e dal fatto che il rendimento medio dal 1950 al 1983 è inferiore a quello dal 1983 al 2024.

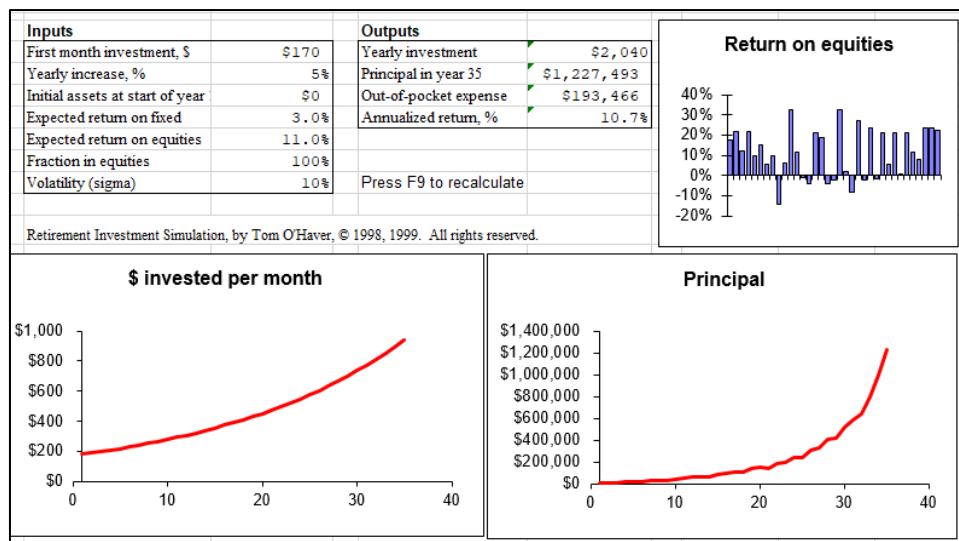
Ci si starà chiedendo quanto siano stati buoni quei dati nel prevedere le tendenze del mercato azionario. Nel breve termine, tali previsioni spesso non sono molto accurate. Ad esempio, la linea di tendenza (in blu) nel grafico a sinistra a pagina 322 prevede che il valore dell'S&P nel 2024 dovrebbe essere 5000. Infatti, entro il 4 luglio 2024, il valore era 5600, ovvero circa il 12% in più rispetto alla previsione. Ciò, tuttavia, non è fuori dall'ordinario; uno sguardo ai grafici qui sopra mostrerà che in passato i valori individuali sono scesi occasionalmente almeno tanto al di sopra (o al di sotto) della linea dell'approssimazione migliore.

Dal punto di vista dell'approssimazione della curva, la deviazione da una curva regolare descritta dall'espressione dell'interesse composto è semplicemente *rumore*. Ma dal punto di vista dell'investitore del mercato azionario, quelle deviazioni possono essere un'*opportunità* e un *warning*. Naturalmente, la maggior parte degli investitori vorrebbe sapere come si comporterà il mercato azionario in futuro, ma ciò richiede *un'estrapolazione oltre la gamma dei dati disponibili*, che è sempre incerta e pericolosa. Tuttavia, è *molto probabile* (ma non certo) che il comportamento *a lungo termine* del mercato (diciamo, su un periodo di 10 anni o più) sarà come il passato, cioè crescente in modo esponenziale all'incirca alla stessa velocità di prima, ma con fluttuazioni imprevedibili simili a ciò che è accaduto in passato.

Ci sono molte caratteristiche notevoli di questo "rumore". In primo luogo, *le deviazioni sono approssimativamente proporzionali a V* e quindi relativamente uguali quando tracciate su una scala logaritmica. In secondo luogo, nel corso degli anni si sono verificati numerosi casi in cui si è avuto un forte calo seguito da una ripresa più lenta in prossimità del valore precedente. E viceversa, ogni

picco è alla fine seguito da un calo. Il consiglio convenzionale per investire è di "comprare al ribasso" (sui cali) e "vendere al rialzo" (sui picchi). Ma ovviamente il problema è che non è possibile determinare *in anticipo* in modo affidabile esattamente dove cadranno i picchi e le cadute; abbiamo solo il passato a guidarci. Tuttavia, se il valore di mercato corrente è molto *più alto* della tendenza a lungo termine, probabilmente diminuirà e se il valore di mercato è molto *inferiore* alla tendenza a lungo termine, alla fine probabilmente aumenterà. L'unica cosa di cui si può essere certi è che, a lungo termine, il mercato alla fine salirà. Questo è il motivo per cui è così importante risparmiare per la pensione investendo in borsa, e *iniziarlo il prima possibile*: in una vita lavorativa di 30 anni, è quasi garantito che il mercato aumenti sostanzialmente. Il modo più indolore per farlo è con il piano di prelievo automatico delle buste paga 401k o 403b del proprio datore di lavoro. Non si può investire nel mercato azionario *nel suo insieme*, ma si può investire in *fondi comuni di investimento indicizzati* o *fondi negoziati in borsa* (ETF), che sono raccolte di azioni costruite per abbinare o tracciare i componenti di un indice di mercato. Tali fondi hanno tipicamente *commissioni di gestione molto basse*, un fattore importante nella selezione di un investimento. Altri fondi comuni di investimento tentano di "battere il mercato" acquistando e vendendo azioni con un'attenzione nel creare un rendimento superiore agli indici di mercato complessivi; alcuni riescono *temporaneamente* a farlo, ma applicano commissioni di gestione più elevate. I fondi comuni di investimento e gli ETF sono investimenti molto meno rischiosi rispetto alle singole azioni.

Alcune società distribuiscono periodicamente pagamenti, agli investitori, chiamati "dividendi". Questi dividendi sono indipendenti dalle variazioni giornaliere del prezzo delle azioni, quindi anche



se il valore delle azioni scende temporaneamente, si ottiene comunque lo stesso dividendo. Per questo motivo è importante impostare il proprio conto di investimento su "reinvestire automaticamente i dividendi", quindi quando il prezzo delle azioni scende, i dividendi stanno acquistando azioni al *prezzo più basso*. I valori dell'indice S&P 500 utilizzati sopra, chiamati *rendimenti dei prezzi*, non includevano il reinvestimento dei dividendi. Se si calcolassero i *rendimenti totali* con i dividendi reinvestiti, questi sarebbero stati sostanzialmente più alti, più vicini all'11% (https://en.wikipedia.org/wiki/S%26P_500_Index#Versions). Con un rendimento annuo totale medio dell'11% (assumendo il reinvestimento dei dividendi) e iniziando con un investimento di \$170 il primo mese - meno di \$6 al giorno - e aumentando tale importo del 5% ogni anno, si potrebbero accumulare oltre \$600.000 in 30 anni di vita lavorativa, o \$1.000.000 se si continua ad investire per altri 5 anni o se si inizia 5 anni prima, come mostrato dal grafico sopra dello spreadsheet. Lo si potrebbe definire anche schema per un "lento arricchimento". E a partire da soli \$6 al giorno, circa il costo di un elegante caffè da Starbucks. Da rifletterci, la prossima volta che si vede una fila di giovani in attesa di ordinare il loro caffè quotidiano. La parte difficile non è tanto

rinunciare al caffè quanto *trovare un lavoro stabile* che consenta di versare contributi automatici di routine al proprio conto pensione nel lungo periodo. Diventare milionario quando si andrà in pensione è possibile, ma non è entusiasmante; piuttosto, è lento e faticoso.

Per illustrare quanta influenza ha la fluttuazione della volatilità del mercato azionario ("rumore") sui guadagni del mercato, lo script Matlab/Octave [SnPsimulation.m](#) aggiunge del **rumore proporzionale** (pag. 28) al calcolo dell'interesse composto per imitare i dati S&P, esegue i due metodi di approssimazione della curva descritti sopra, ripete le allocazioni più e più volte con campioni indipendenti di rumore proporzionale, quindi calcola la media e la relativa deviazione standard (RSD) dei tassi di rendimento. Un risultato tipico è:

```
TrueRateOfReturn = 0.08  
Measured Rate RSD  
Coordinate transformation: 0.078      3%  
Iterative curve fitting: 0.077      6%
```

Come si può vedere, i due metodi non vanno perfettamente d'accordo. Il rendimento calcolato con il metodo iterativo in questo caso è inferiore, ma *avrebbe potuto facilmente essere il contrario*. Il fatto è che le deviazioni standard sono grandi, e il metodo iterativo ha sempre una deviazione standard più alta, perché pesa maggiormente i valori più alti, dove le deviazioni dalla linea sono maggiori, mentre il metodo di trasformazione logaritmica pesa i dati in modo più uniforme. Anche con questa incertezza, investire in un fondo indicizzato del mercato azionario ottiene quasi sempre risultati migliori *nel lungo periodo* rispetto a investimenti più prevedibili come i conti di risparmio o certificati di deposito (CD), che hanno tassi di rendimento molto più bassi. Quando le azioni scendono, anche per ragioni ben note, alcuni investitori acquistano azioni a prezzi ridotti, mentre quando le azioni salgono, soprattutto quando raggiungono i massimi storici, alcuni vendono azioni per "bloccare i propri guadagni". Questo comportamento è stato costante nel corso dei decenni e funge da freno naturale alle fluttuazioni del mercato.

Nell'investire in borsa, è importante concentrarsi sulle tendenze a lungo termine e non lasciarsi spaventare dalle fluttuazioni al rialzo o al ribasso a breve termine, anche se la maggior parte della copertura delle notizie sottolinea comprensibilmente il breve termine. È simile alla distinzione tra [meteo](#) e [clima](#); le ampie e drammatiche variazioni del *tempo* a breve termine sono importanti e tendono a mascherare il riscaldamento *climatico* a lungo termine che sta lentamente sciogliendo le calotte polari e [innalzando il livello del mare](#) (sia che sia causato dall'attività umana che dalle sole cause naturali o da una combinazione di entrambi). Tutti parlano di cambiamenti del tempo, ma il clima cambia così lentamente che è facile concludere che rimane lo stesso La lancetta delle ore dell'orologio non si vede mai muoversi. Da giovani e con molti anni davanti, si mantiene l'investimento in fondi azionari, che hanno i rendimenti migliori. Man mano che si invecchia, è possibile passare gradualmente a investimenti a basso rischio ma a rendimento inferiore, come i conti di risparmio ad alto rendimento, certificati di deposito (CD), conti del mercato monetario, titoli del Tesoro e altri fondi obbligazionari.

Per un modello di foglio di calcolo che consente di calcolare i possibili ritorni sugli investimenti a lungo termine in fondi comuni di investimento del mercato azionario, vedere <https://terpconnect.umd.edu/~toh/simulations/Investment.html>.

Misura del rapporto segnale/rumore nei segnali complessi

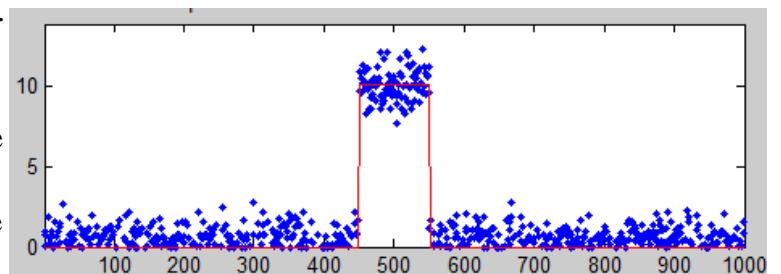
Nella sezione “[Segnali e Rumore](#)” a pagina 22, si dice: “La qualità di un segnale è spesso espressa quantitativamente come il rapporto segnale/rumore (S/N), che è il rapporto dell'ampiezza dell'effettivo segnale ... con la deviazione standard del rumore”. Questa è un'affermazione abbastanza semplice, ma automatizzare la misura del segnale e del rumore nei segnali reali non è sempre semplice. A volte è difficile separare o distinguere tra il segnale e il rumore, perché dipende non solo dalla natura numerica dei dati, ma anche dagli obiettivi della misura.

Per un semplice segnale CC (corrente continua), ad esempio, che misura una tensione fluttuante, il segnale è solo il valore di tensione medio e il rumore è la sua deviazione standard. Questo è facilmente calcolabile in un foglio di calcolo o in Matlab/Octave:

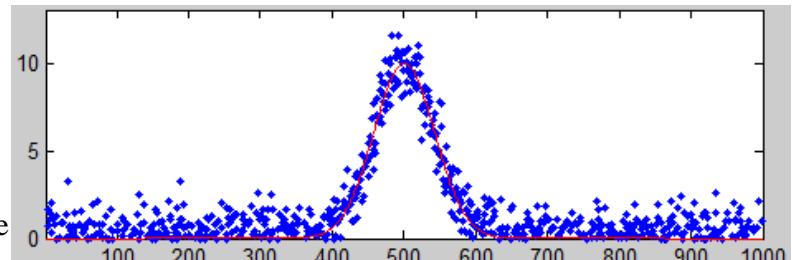
```
>> signal=mean(NoisyVoltage) ;
>> noise=std(NoisyVoltage) ;
>> SignalToNoiseRatio=signal/noise
```

Ma di solito le cose sono più complicate.

Ad esempio, se il segnale è un impulso rettangolare (come nella figura a lato) con rumore casuale costante, la semplice formulazione sopra non darà risultati accurati. Se il segnale è sufficientemente stabile da poter ottenere due successive registrazioni del segnale m_1 e m_2 che sono *identiche tranne che per il rumore*, allora si può semplicemente sottrarre il segnale in uscita: la deviazione standard del rumore è quindi data da $\text{sqrt}((\text{std}(m_1-m_2)^2)/2)$, dove “ std ” è la funzione della [deviazione standard](#) (perché [il rumore casuale si aggiunge in modo quadratico](#)). Ma non tutte le sorgenti di segnale sono abbastanza stabili e ripetibili da funzionare perfettamente. In alternativa, si può provare a misurare il segnale medio appena sopra la parte superiore dell'impulso e il rumore solo nell'intervallo di riferimento prima e/o dopo l'impulso. Non è così difficile da fare

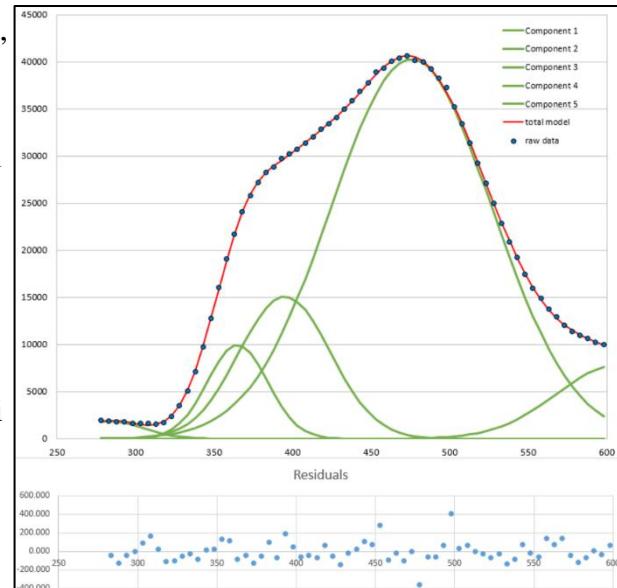


a mano, ma è più difficile da automatizzare con un computer, specialmente se la posizione o l'ampiezza dell'impulso cambia. È fondamentalmente lo stesso per le forme di picco con smoothing come il picco Gaussiano che si incontra solitamente (come nella figura a lato). È possibile stimare l'altezza del picco eseguendo uno [smoothing](#) e poi prendendo il massimo come segnale: `max(fastsmooth(y,10,3))`, ma la precisione si ridurrebbe se si sceglie una larghezza di smoothing troppo alta o troppo bassa. E chiaramente, tutto questo dipende dall'avere una linea di base ben definita nei dati in cui c'è solo rumore. Non funziona se il rumore varia con l'ampiezza del picco.



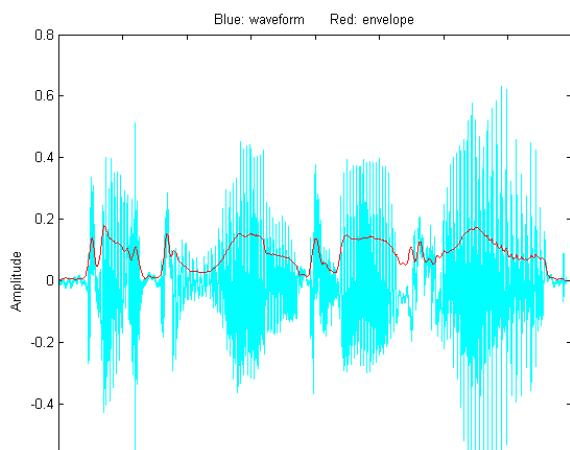
In molti casi, l'[approssimazione](#) può essere d'aiuto. Ad esempio, è possibile utilizzare l'[approssimazione](#) o un [rilevatore](#) per individuare più picchi e misurare le loro altezze e i loro rapporti S/N su una base picco-picco, calcolando il rumore come la deviazione standard della differenza tra i dati originali e l'approssimazione migliore sulla parte superiore del picco. È così che [iSignal](#) (pagina 366) misura i rapporti S/N dei picchi. Inoltre, iSignal ha funzionalità di correzione della linea di base che consentono di misurare il picco rispetto alla linea di base.

L'approssimazione [curve fitting] della curva funziona anche per segnali complessi di forma indeterminata che possono essere approssimati da un [polinomio di ordine elevato](#) o come [somma di un numero di funzioni di base come le Gaussiane](#), come nell'esempio di dati reali mostrato a lato. In questo caso, vengono utilizzate un numero crescente di Gaussiane per approssimare i dati al punto in cui i residui diventano casuali e non strutturati. I residui (mostrati in blu sotto il grafico) sono quindi solo il rumore rimanente nel segnale, la cui deviazione standard è facilmente calcolata utilizzando la funzione di deviazione standard nativa in un foglio di calcolo ("STDEV") o in Matlab/Octave ("std"). In questo esempio, la deviazione standard dei residui è 111 e il segnale massimo è 40748, quindi la deviazione standard relativa percentuale del rumore è 0,27% e il rapporto S/N è 367. (Le posizioni, le altezze e le larghezze delle componenti Gaussiane, di solito i risultati principali dell'approssimazione della curva, non vengono utilizzate in questo caso; l'approssimazione è usata solo per ottenere una misura del rumore attraverso i residui). Il vantaggio di questo approccio rispetto alla semplice sottrazione di due misure successive del segnale è che si aggiusta per lievi cambiamenti nel segnale dalla misura a misura. L'unica ipotesi è che il segnale sia una forma d'onda a bassa frequenza che può essere dotata di un polinomio o di una raccolta di forme di picco di base e che il rumore sia casuale e per lo più ad alta frequenza rispetto al segnale. Ma non si deve usare un ordine polinomiale troppo alto o troppi picchi nel modello, altrimenti si sta semplicemente "approssimando il rumore".

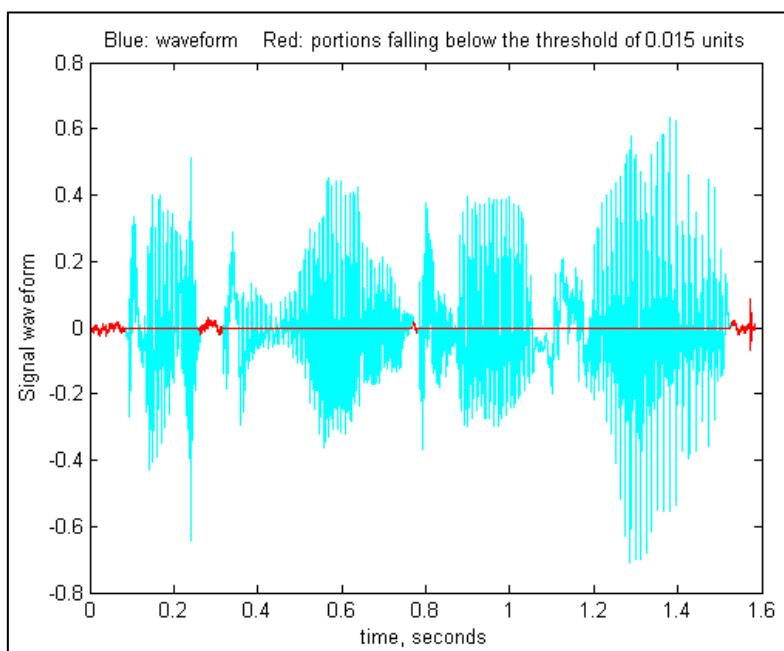


Con le forme d'onda periodiche del segnale, la situazione è un po' più complicata. Ad esempio, si consideri la registrazione audio della frase "Testing, one, two, three" (cliccare per scaricarla nel [formato .mat](#) o [WAV](#)). Lo script Matlab/Octave [PeriodicSignalSNR.m](#) carica il file audio nella variabile chiamata "waveform", quindi ne calcola l'ampiezza media ("inviluppo") con lo smoothing (pag. 39) del valore assoluto della forma d'onda:

```
envelope = fastsmooth(abs(waveform), SmoothWidth, SmoothType);
```

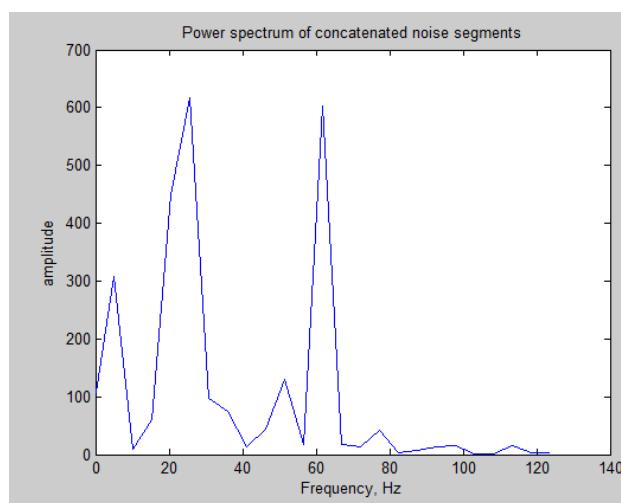


Il risultato è disegnato a sinistra, dove la forma d'onda è in blu e l'inviluppo è in rosso. Il segnale è facilmente misurabile come il massimo o forse la media della forma d'onda, ma il rumore non è così evidente. La voce umana non è abbastanza riproducibile per ottenere una seconda registrazione identica per sottrarre il segnale. Tuttavia, ci saranno spesso delle lacune nel suono, durante le quali il rumore di fondo sarà dominante. In una registrazione audio (vocale o musicale), ci saranno tipicamente tali spazi vuoti all'inizio, quando il processo di registrazione è già iniziato ma il suono non è ancora iniziato, e probabilmente in altri brevi periodi quando ci sono pause nel suono. L'idea è che, monitorando l'inviluppo del suono e notando quando scende al di sotto di un valore di soglia regolabile, si possa registrare automaticamente il rumore che si verifica in quegli spazi, ogni volta che avvengono in una registrazione.



In [PeriodicSignalSNR.m](#), questa operazione viene eseguita nelle righe 26-32 e la soglia viene impostata nella riga 12. Il valore di soglia deve essere ottimizzato per ogni registrazione. Quando la soglia è impostata su 0,015 nella registrazione di "Testing, one, two, three", i segmenti di rumore risultanti vengono individuati e contrassegnati in rosso nel grafico a lato. Il programma determina il livello di rumore medio in questa registrazione semplicemente calcolando la deviazione standard di quei segmenti (riga 46), quindi calcola e stampa il rapporto S/N picco-picco e il rapporto S/N del 'Valore efficace' RMS (root mean square).

```
PeakToPeak_SignalToNoiseRatio = 143.7296
RMS_SignalToNoiseRatio = 12.7966
```



Viene determinata anche la distribuzione di frequenza del rumore (righe 60-61) e mostrata nella figura a lato, utilizzando la funzione PlotFrequencySpectrum function, oppure si poteva usare iSignal (pagina 366) nella modalità spettro di frequenza (**Shift-S**). Lo spettro del rumore mostra una forte componente molto vicina ai 60 Hz, che è quasi certamente dovuta alla *linea elettrica* (la registrazione è stata fatta negli USA, dove l'alimentazione AC è di 60 Hz); questo suggerisce che una migliore schermatura e messa a terra dell'elettronica potrebbe aiutare a ripulire le registrazioni future. Il picco a 20 Hz

è più difficile da attribuire: forse è il debole ronzio di un ventilatore o di un condizionatore. La mancanza di componenti forti a 100 Hz e oltre (dove c'è la voce umana) suggerisce che i suoni vocali sono stati effettivamente soppressi a questa impostazione di soglia di ampiezza. Lo script può essere applicato ad altre registrazioni audio in formato WAV semplicemente cambiando il nome del file e l'asse del tempo nelle righe 8 e 9.

Gestione di segnali ad ampio raggio: Elaborazione segmentata

Per facilitare l'ispezione di segnali molto grandi e complessi, è utile essere in grado di "zoomare" le diverse parti della lungo l'asse x, cosa che può essere eseguita con gli strumenti interattivi Matlab *iSignal* (pagina 366), *iPeak* (pagina 246) e *ipf.m* (pagina 405) o con la funzione Matlab/Octave [segplot.m](#) (pagina 455). A volte un segnale sperimentale varierà così tanto lungo l'asse x che è impossibile trovare una singola impostazione per operazioni come lo smoothing o il rilevamento dei picchi che siano ottimizzate per tutte le regioni del segnale. È sempre possibile suddividere il segnale in pezzi e trattarli separatamente, ad esempio utilizzando *segplot*, ma in alcuni casi è più facile utilizzare una singola applicazione *segmentata* sull'intero segnale. Questa è l'idea alla base delle funzioni Matlab/Octave e dei fogli di calcolo Excel in questa sezione.

[SegmentedSmooth.m](#), illustrato a lato, è una variante segmentata di [fastsmooth.m](#), che può essere utile se le larghezze dei picchi o il livello di rumore varia sostanzialmente lungo il segnale. La sintassi è la stessa di *fastsmooth.m*, tranne per il fatto che il secondo argomento di input "smoothwidths" può essere un vettore: [SmoothedSignal, SmoothMatrix] =

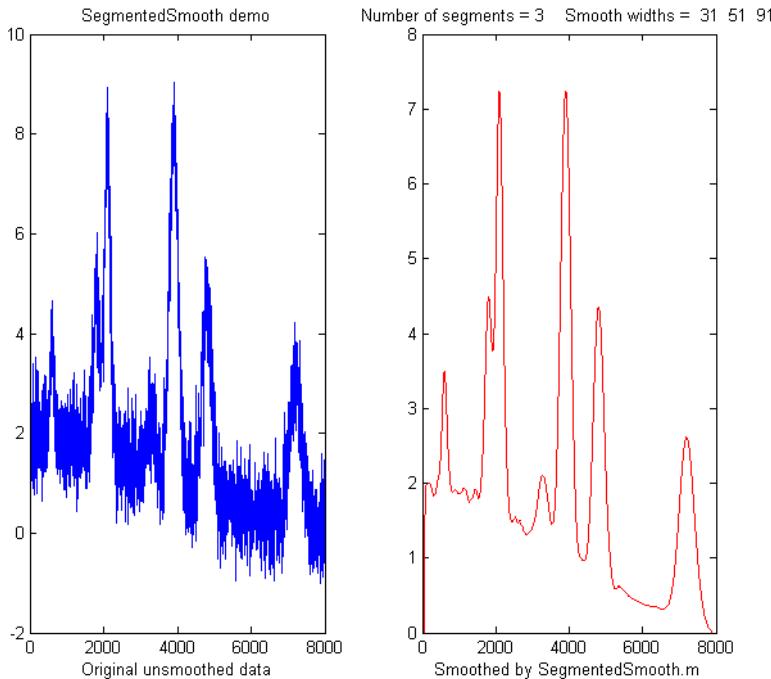
SegmentedSmooth (Y,

smoothwidths, type, ends).

La funzione divide Y in diverse regioni di uguale lunghezza definite dalla lunghezza del vettore 'smoothwidths', poi esegue il filtraggio di ciascuna regione col tipo di smoothing 'type' e con l'ampiezza definita dagli elementi del vettore 'smoothwidths'. Nel semplice esempio nella figura a lato, `smoothwidths = [31 52 91]`, che divide il segnale in tre regioni ed esegue lo smoothing della prima regione con smoothwidth 31, il secondo con 51 e l'ultimo con 91. Si può usare qualsiasi numero e sequenza di larghezze di smoothing.

Facoltativamente restituisce

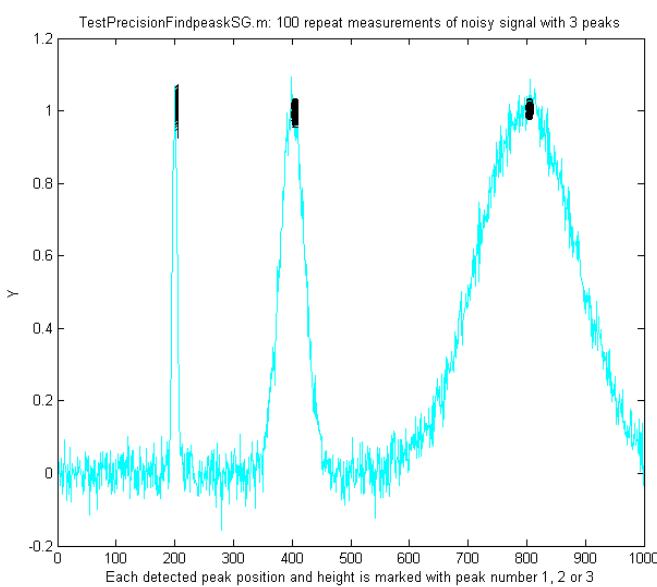
'SmoothMatrix' costituito da tutti i segmenti assemblati in una matrice. Digitare "help SegmentedSmooth" per altri esempi.



[DemoSegmentedSmooth.m](#) mostra il funzionamento con diversi segnali costituiti da picchi rumorosi di larghezza variabile che si allargano progressivamente, come la figura a lato. *FindpeaksSL.m* è la stessa cosa per i picchi Lorentziani.

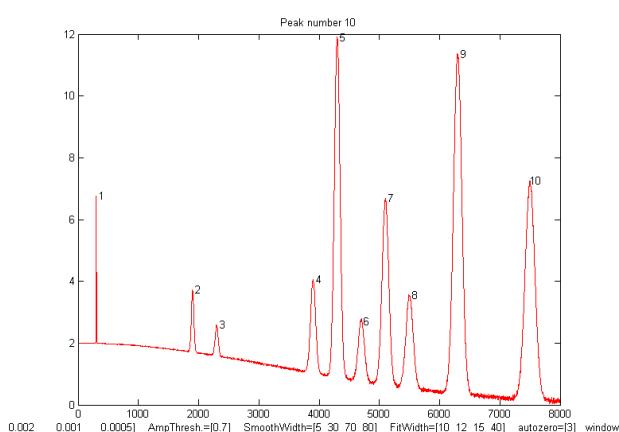
[SegmentedSmoothTemplate.xlsx](#) è uno *spreadsheet* per lo smoothing segmentato con multi-larghezze, che è funzionalmente simile a *SegmentedSmooth.m*. In questa versione, ci sono 20 segmenti. [SegmentedSmoothExample.xlsx](#) è un esempio di *spreadsheet* con i dati ([grafico](#)). Un foglio di calcolo correlato [GradientSmoothTemplate.xlsx](#) ([grafico](#)) esegue uno smoothing con una larghezza linearmente crescente (o decrescente) lungo tutto il segnale, dando solo il valore di inizio e quello di fine, genera automaticamente tutti i segmenti necessari. Ovviamente, come al solito con i fogli di calcolo, si dovrà modificare questi template per il numero di punti, di solito inserendo delle righe in mezzo a qualche parte e poi trascinando verso il basso quanto inserito, inoltre si potrebbe dover modificare l'intervallo dell'asse x del grafico. (Al contrario, le funzioni Matlab/Octave lo fanno automaticamente).

[SegmentedFouFilter.m](#) è una versione segmentata di [FouFilter.m](#) che applica diverse frequenze centrali e larghezze ai diversi segmenti del segnale. La sintassi, `[ffSignal, ffMatrix] = SegmentedFouFilter(y, samplingtime, centerFrequency, filterWidth, shape, mode)`, è come FouFilter.m in cui i due argomenti di input centerFrequency e filterWidth devono essere vettori con i valori di centerFrequency di filterWidth per ogni segmento. Il segnale viene diviso in diversi segmenti uguali determinato dalla lunghezza di centerFrequency e filterWidth, che devono essere uguali in lunghezza. Facoltativamente restituisce ffMatrix di tutti i segmenti assemblati in una matrice. Digitare help SegmentedFouFilter per l'help e per altri esempi; la figura a lato mostra l'Esempio 2. Potrebbe essere utile visualizzare il segnale utilizzando una funzione correlata, [PlotSegFreqSpect.m](#), che crea e visualizza uno spettro di potenza di Fourier *segmentato nel tempo* (vedere pagina 97 e seguenti).



[findpeaksSG.m](#) è una variante della funzione [findpeaksG function](#), con la stessa sintassi, tranne che i quattro parametri di rilevamento del picco possono essere *vettori*, dividendo il segnale in regioni ottimizzate per picchi di larghezze diverse. È possibile dichiarare qualsiasi numero di segmenti in base alla lunghezza dell'argomento di input SlopeThreshold. (Nota: i vettori si devono inserire solo per quei parametri che si desiderano variare tra i segmenti; per consentire a qualsiasi altro parametro di restare invariato per tutti i segmenti, è sufficiente immettere un singolo valore scalare per quel parametro; solo SlopeThreshold deve essere un vettore). Nell'esempio mostrato a lato, lo script

[TestPrecisionFindpeaksSG.m](#) crea un segnale rumoroso con tre picchi di larghezze molto diverse, ne misura le posizioni, le altezze e le larghezze di ciascuno utilizzando findpeaksSG e stampa le deviazioni standard relative percentuali dei parametri dei tre picchi in 100 misure con rumore casuale indipendente. Con i parametri per la ricerca di 3 segmenti, findpeaksSG rileva e misura in modo affidabile tutti e tre i picchi. Al contrario, findpeaksG, sintonizzato sul picco medio (usando la riga 26 invece della 25), misura male il primo e l'ultimo picco. Si può anche vedere che la precisione dei parametri migliora progressivamente (deviazione standard relativa minore) quando maggiore è la larghezza dei picchi, semplicemente perché ci sono più punti in quei picchi. (È possibile modificare qualsiasi variabile nelle righe 10-18).



Una funzione correlata è [findpeaksSGw.m](#) che è come la precedente tranne per il fatto che utilizza il *denoising wavelet* (pagina 130) anziché lo smoothing. Come argomento di input accetta il livello della wavelet anziché che l'ampiezza uniforme. Lo script [TestPrecisionFindpeaksSGvsW.m](#) confronta la precisione e l'accuratezza per la posizione del picco e la misurazione dell'altezza sia per entrambe le funzioni [findpeaksSG.m](#) e [findpeaksSGw.m](#).

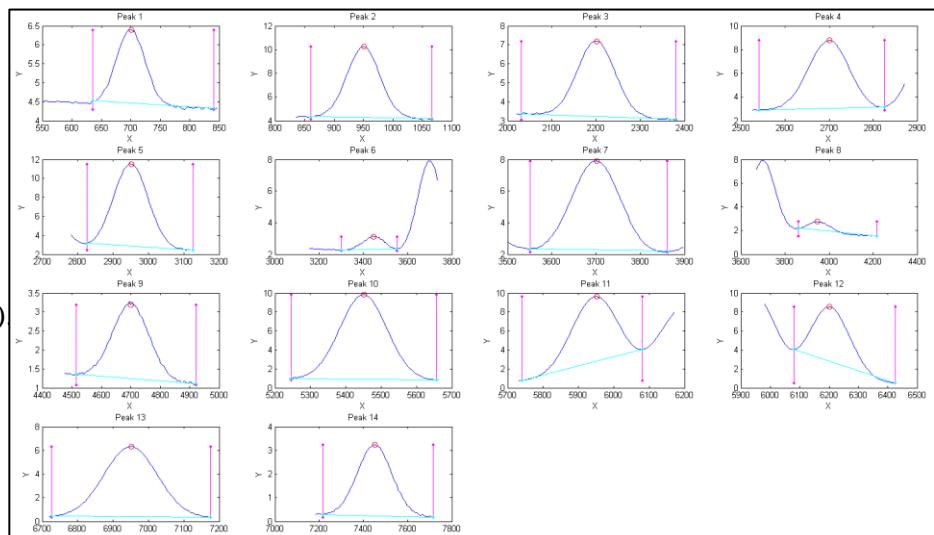
Pagina | 327

[findpeaksSb.m](#) è una variante segmentata della funzione [findpeaksb.m](#). Ha la stessa sintassi di findpeaksSb, tranne per gli argomenti "SlopeThreshold", "AmpThreshold", "smoothwidth", "peak-group", "window", "PeakShape", "extra", "NumTrials", "BaselineMode" e "fixedparameters" possono essere tutti optionalmente scalari o vettori con una voce per ogni segmento. Ciò consente alla funzione di gestire segnali molto variabili con picchi di forme, larghezze e background molto diversi. Nell'esempio a lato, lo script Matlab/Octave DemoFindPeaksSb.m crea una serie di picchi Gaussiani le cui larghezze aumentano di un fattore di 25 sovrapposte ad una linea di base curva con rumore bianco casuale che aumenta gradualmente lungo il segnale. In questo esempio, vengono utilizzati quattro segmenti, modificando i valori di rilevamento e di approssimazione della curva in modo che tutti i picchi vengano accuratamente misurati.

```
SlopeThreshold = [.01 .005 .002 .001];
AmpThreshold = 0.7;
SmoothWidth = [5 15 30 35];
FitWidth = [10 12 15 20];
windowspan = [100 125 150 200];
peakshape = 1;
BaselineMode = 3;
```

Lo script calcola anche l'errore percentuale relativo della misura di posizione, altezza, larghezza e area per ogni picco.

[measurepeaks.m](#) è un rilevatore automatico di picchi di forma arbitraria. Si basa su findpeaksSG, con cui condivide i primi 6 argomenti di input; i quattro argomenti di rilevamento possono essere vettori per accogliere segnali con picchi di larghezze molto variabili. Restituisce una [tabella](#) contenente il numero del picco, la posizione, l'altezza assoluta, la differenza picco-valle, l'area di caduta perpendicolare e l'area di scorrimento tangente di ciascun picco rilevato. Se l'ultimo argomento di input ('plots') è impostato su 1, [disegnerà l'intero segnale](#) con i picchi numerati e [anche i singoli picchi](#) contrassegnando il massimo, i punti di avvallamento e le linee tangenti (come mostrato a lato). Digitare "help measurepeaks" e provare o eseguire [testmeasurepeaks.m](#) ([animazione grafica](#)). Le funzioni correlate [autopeaks.m](#)



e [autopeakssplot.m](#) sono simili, tranne per il fatto che i quattro parametri di rilevamento del picco possono essere omessi e la funzione calcolerà i valori iniziali stimati.

Lo script [HeightAndArea.m](#) verifica l'accuratezza della misura dell'altezza e dell'area del picco con i segnali che hanno più picchi con larghezza, rumore, sfondo e sovrapposizione variabili. In generale, i valori per l'altezza assoluta e l'area di taglio verticale (pagina 140) sono migliori per i picchi che non hanno background, anche se sono leggermente sovrapposti, mentre i valori per la differenza picco-avvallamento e per l'area col [tangential skim] è migliore per i picchi isolati su un background diritto o leggermente curvo. Nota: questa funzione utilizza lo smoothing (specificato dall'argomento di input SmoothWidth) solo per il rilevamento dei picchi; esegue le sue misurazioni sui dati y originali. Se i dati originali sono rumorosi, potrebbe essere meglio eseguire lo smoothing dei dati y prima di chiamare measurepeaks.m, usando qualsiasi funzione per lo smoothing.

Altre funzioni segmentate. Lo stesso codice di segmentazione utilizzato in [SegmentedSmooth.m](#) (righe53-65) può essere applicato ad altre funzioni semplicemente modificando la prima riga del primo ciclo for/end (riga 59) per fare riferimento alla funzione che si desidera applicare in modo segmentato. Ad esempio, lo [sharpening](#) segmentato può essere utile quando un segnale ha più picchi che variano in larghezza e la [deconvoluzione](#) segmentata può essere utile quando un segnale ha più picchi che variano in larghezza o in coda variano sostanzialmente lungo il segnale:

[SegExpDeconv\(x,y,tc\)](#) esegue la deconvoluzione di y con un vettore di funzioni esponenziali le cui costanti di tempo sono specificate dal vettore tc. [SegExpDeconvPlot.m](#) è lo stesso tranne per il fatto che disegna i segnali originale e deconvoluto e mostra le divisioni tra i segmenti mediante linee magenta verticali.

Calibrazione della Misura

La maggior parte delle misure scientifiche implica l'uso di uno strumento che misura qualcosa' altro e lo converte nella misura desiderata. Esempi sono semplici bilance (che misurano la compressione di una molla), i comuni termometri (che misurano l'espansione termica), i pH-metri (che misurano una tensione) e la maggior parte dei dispositivi per misurare la frequenza cardiaca, l'emoglobina nel sangue, il CO₂ nell'aria, o lo zucchero nel vino (che misura un raggio di luce). Questi strumenti hanno *un solo scopo*, progettati per misurare una quantità e convertono automaticamente ciò che hanno misurato nella quantità desiderata e la visualizzano direttamente. Ma per garantire l'accuratezza, tali strumenti possono essere [calibrati](#), ovvero utilizzati per misurare uno o più standard di calibrazione di precisione nota, come un peso standard o un campione preparato con cura a una temperatura, pH o col contenuto di zucchero noto. La maggior parte vengono pre-calibrati in fabbrica per la misura di una sostanza specifica in un tipo specifico di campione.

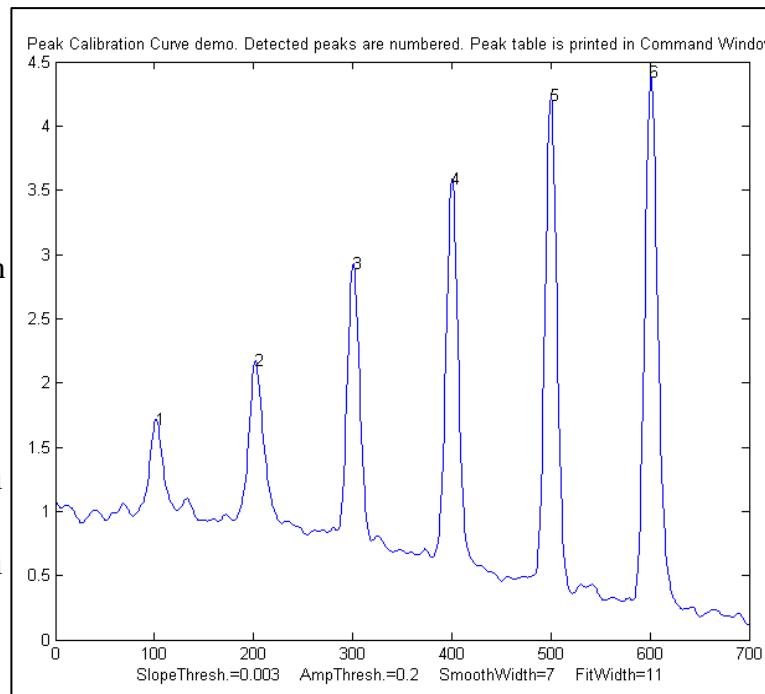
Calibrazione analitica. A differenza delle misure specializzate, i metodi strumentali *generici* vengono utilizzati per misurare la quantità di molti componenti chimici in vari tipi di campioni. Tra questi metodi ci sono vari tipi di spettroscopia, cromatografia ed elettro-chimica o combinazioni di tecniche come la "[GC-MS](#)". Anche questi devono essere calibrati, ma poiché questi strumenti possono essere utilizzati per misurare un'ampia gamma di composti o elementi, devono essere calibrati *dall'utente* per ciascuna sostanza e per ogni tipo di campione. Di solito, ciò viene ottenuto preparando con cura (o acquistando) uno o più "campioni standard" di concentrazione nota, come campioni di soluzione in un solvente adatto. Ogni standard viene inserito o iniettato nello strumento e le letture risultanti vengono poste in un grafico rispetto alle concentrazioni note degli standard, utilizzando i [calcoli dei quadrati minimi](#) per calcolare la *pendenza* e l'*intercetta*, nonché la deviazione standard della pendenza (*sds*) e dell'intercetta (*sdi*). In seguito le "incognite" (cioè i campioni le cui concentrazioni devono essere determinate) vengono misurate dallo strumento e i loro segnali vengono convertiti in concentrazioni con l'aiuto della curva di calibrazione. Se la calibrazione è lineare, la concentrazione del campione C di qualsiasi incognita è data da $(A - intercetta) / pendenza$, dove A è il segnale misurato (altezza o area) di quell'incognita. La deviazione standard prevista nella concentrazione del campione è $C * \text{SQRT}((sdi/(A-intercetta))^2 + (sds/pendenza)^2)$ per le [regole della propagazione dell'errore](#). Tutti questi calcoli vengono eseguiti nel template [CalibrationLinear.xls](#). In alcuni casi la cosa misurata non può essere rilevata direttamente ma deve subire una reazione chimica che la renda misurabile; in tal caso, la stessa reazione deve essere eseguita su tutte le soluzioni standard e le soluzioni campione ignote.

Vari metodi di calibrazione vengono utilizzati per compensare problemi come gli errori casuali nella preparazione dello standard o nelle letture dello strumento, [interferenze](#), [derive](#) e [non-linearietà](#) nella relazione tra la concentrazione e la lettura dello strumento. Per esempio, la [tecnica della](#)

calibrazione delle addizioni standard si può usare per compensare le interferenze moltiplicative. Ho preparato una serie di template di tipo “riempire gli spazi vuoti” per vari metodi di calibrazione, con le istruzioni, così come una serie di spreadsheet per le simulazioni della propagazione dell'errore in metodi di calibrazione analitica ampiamente utilizzati, compreso un esercizio passo-passo.

Calibrazione ed elaborazione dei segnali. L'elaborazione del segnale spesso si interseca con la calibrazione. Ad esempio, se si usa lo smoothing o il filtraggio per ridurre il rumore, o la differenziazione per ridurre l'effetto del background, o si misura l'area per ridurre l'effetto dell'ampliamento del picco, o si usa la modulazione per ridurre l'effetto della deriva a bassa frequenza, allora si *deve* usare la stessa esatta elaborazione del segnale sia per i campioni standard che per quelli ignoti, perché la scelta della tecnica di signal processing può avere un grande impatto sulla grandezza e anche sulle *unità* del risultato processato (come per esempio nella tecnica derivativa e nella scelta tra altezza e area del picco).

PeakCalibrationCurve.m ne è un esempio in Matlab/Octave. Questo script simula la calibrazione di un sistema di iniezione di flusso che produce picchi correlati a una concentrazione e un'ampiezza in esame ('amp'). In questo esempio, sei standard noti vengono misurati in sequenza, ne risultano sei picchi separati nel segnale osservato. (Si assume che il segnale del rivelatore sia linearmente proporzionale alla concentrazione in qualsiasi istante). Per simulare una misura più realistica, lo script aggiunge quattro sorgenti di "disturbo" al segnale osservato:

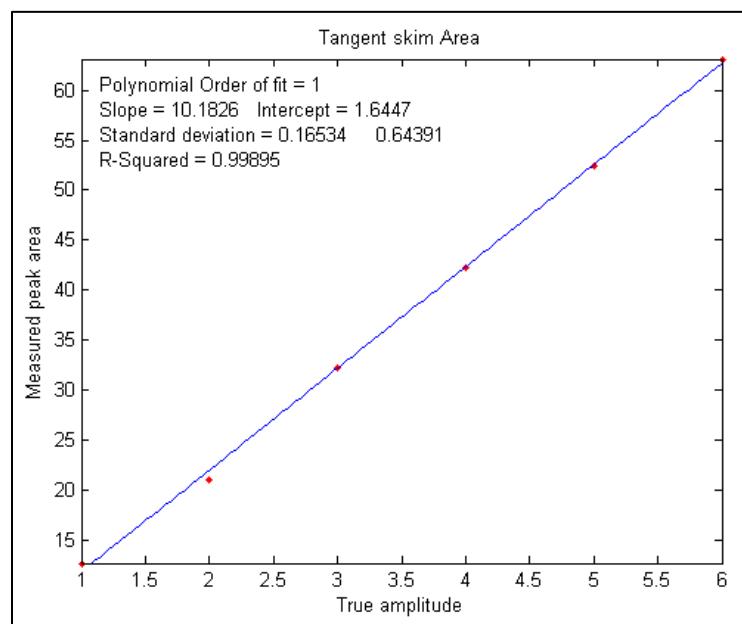


- a. un *casuale rumore bianco* aggiunto a tutti i punti del segnale, controllato dalla variabile "Noise";
- b. *background* - ampio background curvo di *ampiezza casuale*, inclinazione e curvatura, controllato da "background";
- c. *ampliamento* - ampliamento esponenziale del picco che *varia casualmente* da picco a picco, controllato da "broadening";
- d. uno *smoothing* finale prima di misurare i picchi, controllato da "FinalSmooth".

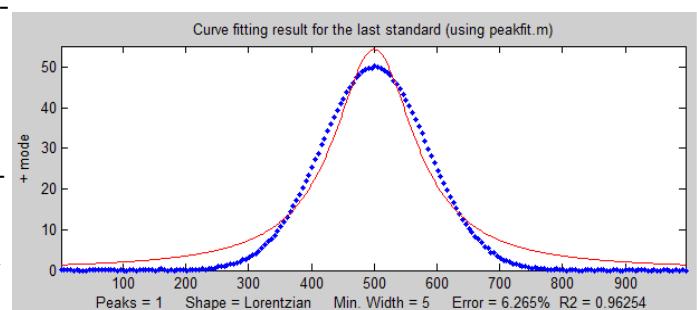
Lo script usa measurepeaks.m come funzione interna per determinare l'altezza assoluta del picco, la differenza picco-avallamento, l'area col taglio verticale e quella col taglio tangente (pagina 135). Disegna le curve di calibrazione separate per ciascuna di queste misure nelle Windows 2-5 di Matlab rispetto alle vere ampiezze in esame (nel vettore "amp"), approssimando i dati a una linea retta e calcolando la pendenza, l'intercetta e R2. (Se la risposta del rivelatore non fosse lineare, funzionerebbero meglio i quadrati minimi con una quadrica o una cubica). La pendenza e l'intercetta della retta 'best-fit' sono diverse per i diversi metodi, ma se R^2 è prossimo a 1.000, è possibile eseguire una misura corretta. (Se tutti i disturbi casuali sono impostati a zero nelle righe 33-36, i valori degli R^2 saranno tutti 1.000. In caso contrario, le misure non saranno perfette e alcuni metodi porteranno a misure migliori - R^2 più vicino a 1.000 - di altri). Ecco un risultato tipico:

Peak	Position	PeakMax	Peak-val.	Perp drop	Tan skim
1	101.56	1.7151	0.72679	55.827	11.336
2	202.08	2.1775	1.2555	66.521	21.425
3	300.7	2.9248	2.0999	58.455	29.792
4	400.2	3.5912	2.949		
	66.291	41.264			
5	499.98	4.2366	3.7884		
	68.925	52.459			
6	601.07	4.415	4.0797		
	75.255	61.762			
R2 values:		0.9809	0.98615		
	0.7156	0.99824			

In questo caso, il metodo di taglio tangente funziona meglio, fornendo una curva di calibrazione lineare (mostrata a lato) con l'R2 più alto.



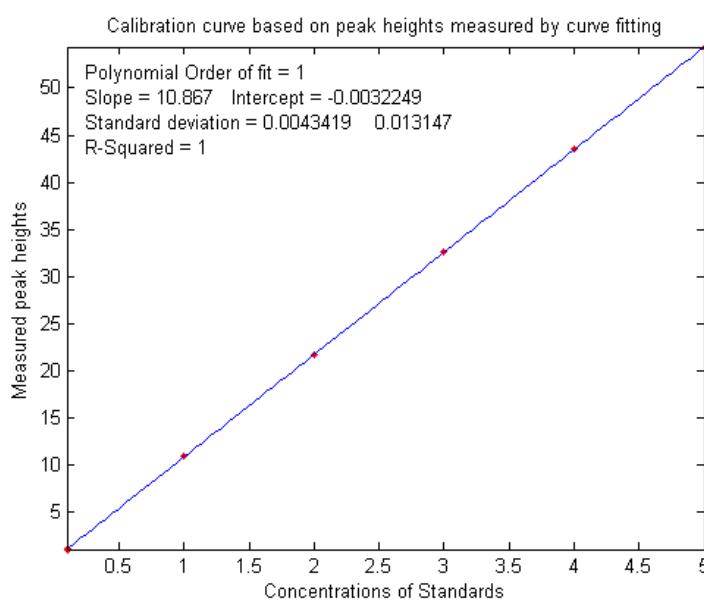
In questo tipo di applicazione, le altezze e/o le misure dell'area dei picchi non devono essere effettivamente *accurate*, ma devono essere *precise*. Questo perché l'obiettivo di un metodo analitico come l'iniezione del flusso o la cromatografia *non* è la misura delle *altezze* delle *aree* dei picchi, ma quella delle *concentrazioni*, motivo per cui vengono utilizzate le curve di calibrazione. La [Window 6](#) mostra il grafico della correlazione tra le aree misurate col taglio tangente e le aree reali effettive dei picchi nel segnale mostrato sopra, a destra; la pendenza di questo grafico mostra che le aree col taglio tangente sono inferiori di circa il 6% rispetto alle aree reali, ma ciò non fa differenza in questo caso perché gli standard e i campioni ignoti vengono misurati allo stesso modo. In alcune *altre* applicazioni, potrebbe essere necessario misurare accuratamente le altezze e/o le aree dei picchi, nel qual caso l'approssimazione della curva è generalmente il modo migliore per procedere.



Se i picchi si sovrappongono parzialmente, le altezze e le aree dei picchi misurate potrebbero essere influenzate. Per ridurre il problema, potrebbe essere possibile ridurre la sovrapposizione utilizzando i [metodi di sharpening](#), per esempio il [metodo derivativo](#), la [deconvoluzione](#) o il [metodo di trasformazione con la potenza](#), come mostrato dalla funzione autonoma Matlab/Octave [PowerTransformCalibrationCurve.m](#).

Approssimazione del segnale. Normalmente nell'approssimazione della curva, come nel metodo [dei minimi quadrati classico](#) (CLS) e in quello [iterativo non-lineare dei minimi quadrati](#), la selezione

ne di un modello per il profilo è molto importante. Tuttavia nelle applicazioni di *analisi quantitativa* dell'approssimazione [curve fitting], dove l'altezza o l'area del picco misurata dall'approssimazione della curva viene utilizzata solo per determinare la concentrazione della sostanza che ha creato il picco costruendo una [curva di calibrazione](#), con la forma esatta del modello è *sorprendentemente influente*. Lo script Matlab/Octave [PeakShapeAnalyticalCurve.m](#) mostra che, per un singolo picco isolato la cui forma è costante e indipendente dalla concentrazione, se viene utilizzata la forma del modello errata, le *altezze* misurate mediante approssimazione della curva saranno imprecise, ma tale errore sarà *esattamente lo stesso* per i campioni ignoti e la calibrazione nota standard, quindi l'errore verrà "cancellato" e le concentrazioni misure saranno ancora accurate, a condizione che si utilizzi lo *stesso* modello impreciso sia per gli standard noti che per i campioni ignoti. Nell'esempio mostrato sopra, la forma effettiva del picco è Gaussiana (punti blu), ma il modello utilizzato per approssimare i dati è Lorentziano (linea rossa). Questa è un'approssimazione intenzionalmente sbagliata; il valore R^2 per l'approssimazione del segnale è solo di 0.962 (a un'approssimazione scadente per gli standard della scienza delle misurazioni). Il risultato è che la *pendenza* della curva di calibrazione



(mostrata a lato) è *maggior del previsto*; avrebbe dovuto essere 10 (perché è il valore di "sensitivity" nella riga 18), ma in realtà è 10,867 nella figura a lato, ma ciò nonostante la *curva di calibrazione* è ancora lineare e il suo valore di R^2 è 1.000, il che significa che l'analisi dovrebbe essere accurata. (Si noti che l'approssimazione della curva viene effettivamente applicata *due volte* in questo tipo di applicazione, una volta utilizzando l'approssimazione iterativa dei *dati del segnale*, e poi di nuovo utilizzando l'approssimazione polinomiale per i *dati di calibrazione*).

Nonostante tutto ciò, è comunque meglio utilizzare un profilo quanto più accurato possibile per il segnale, poiché è possibile utilizzare l'errore di approssimazione percentuale o l' R^2 del segnale approssimato possono essere usati per evidenziare che c'è qualcosa che non va, come un aumento del rumore o la comparsa di un picco di interferenza da una sostanza estranea.

Precisione numerica del software

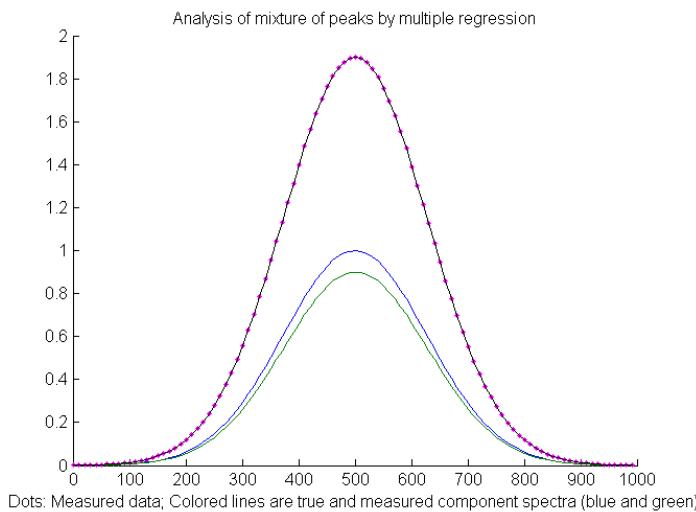
I calcoli effettuati dal software con numeri non interi hanno un limite naturale alla precisione con cui possono essere rappresentati; per esempio, il numero $1/3$ è rappresentato come $0,3333333 \dots$, utilizzando un numero grande ma finito di "3", mentre in teoria c'è una stringa *infinita* di "3" nella rappresentazione decimale di $1/3$. È lo stesso con i numeri irrazionali come "pi" e la radice quadrata di 2; non possono mai avere una rappresentazione decimale *esatta*. In linea di principio, questi piccoli errori potrebbero accumularsi in un calcolo con più passaggi molto complesso e potrebbero plausibilmente diventare una fonte significativa di errori. Nella maggior parte delle applicazioni al calcolo scientifico, tuttavia, questi limiti saranno minuscoli rispetto agli errori e al rumore casuale che è già presente nella maggior parte delle misure del mondo reale. Ma è meglio sapere quali sono questi limiti numerici, in quali circostanze potrebbero verificarsi e come minimizzarli.

Spettroscopia multi-componente. Probabilmente il calcolo più comune in cui la precisione numerica è un problema è nei metodi matriciali utilizzati nella [Spettroscopia multi-componente](#).

Nella derivazione del metodo [Classico dei Quadrati Minimi \(CLS\)](#), la *matrice inversa* viene utilizzata per risolvere grandi sistemi di equazioni lineari. La matrice inversa è una funzione standard nei linguaggi di programmazione come *Matlab*, *Octave*, Python, *Mathematica* di Wolfram e negli spreadsheet. Ma se si usa la funzione in Matlab, il nome ("inv") viene *automaticamente contrassegnato dall'editor* con il seguente avviso:

"Per risolvere un sistema di equazioni lineari, l'inversa di una matrice è principalmente un valore teorico. Non usare mai l'inverso di una matrice per risolvere un sistema lineare $Ax=b$ con $x=inv(A)*b$, perché è lento e impreciso.... Invece di moltiplicare per l'inverso, usare la divisione destra della matrice (/) o la divisione sinistra della matrice (\)."

Ovvero: Sostituire $inv(A)*b$ con $A\b$... [e]... sostituire $b*inv(A)$ con b/A "



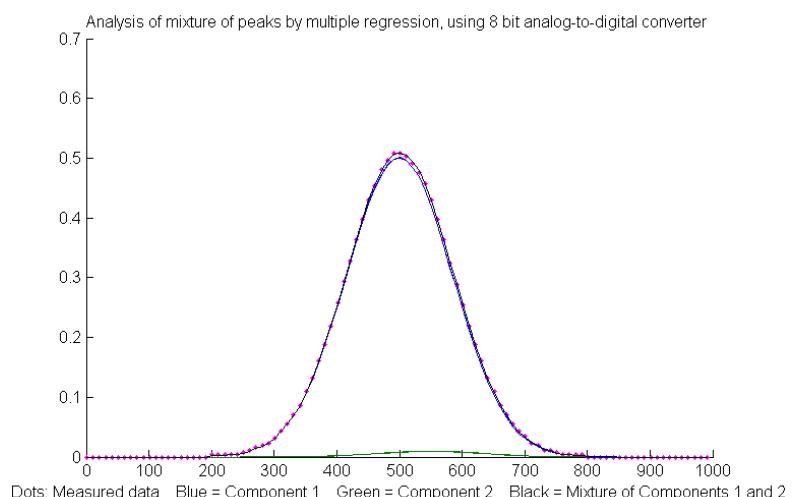
significano picchi molto sovrapposti che sono difficili da misurare con precisione. In questo esempio il rapporto di separazione/larghezza è 0,0033, che è molto piccolo (cioè difficile); questo equivale a provare a misurare una miscela di due picchi di spettroscopia di assorbimento larghi 300 nm e *separati solo da 1 nm*, una piccola differenza che non si noterebbe nemmeno a occhio nudo. I risultati di questo script mostrano che il metodo della matrice inversa ("inv") ha effettivamente un *errore migliaia di volte maggiore* del metodo che utilizza la divisione della matrice, ma anche che l'errore della divisione matriciale resta molto piccolo. In pratica, è improbabile che la differenza tra questi metodi sia significativa se applicata a dati sperimentali reali, perché anche il minimo bit di instabilità del segnale (come quello causato da piccoli cambiamenti nella temperatura del campione o rumore casuale nel segnale, che si può simulare nella riga 15) produce un errore molto maggiore. Quindi fondamentalmente quel messaggio di avvertimento viene da un matematico o da un programmatore di computer, *non* da uno scienziato sperimentale.

Risoluzione Analogico-digitale.

Potenzialmente più significativa della risoluzione numerica del computer è la risoluzione del *convertitore analogico-digitale* (ADC) utilizzato per convertire i segnali analogici (come la tensione) in un numero. Lo script Matlab/Octave [RegressionADCbitsTest.m](#) lo mostra, con due bande Gaussiane leggermente

"Lento e impreciso"? Termini spaventosi!

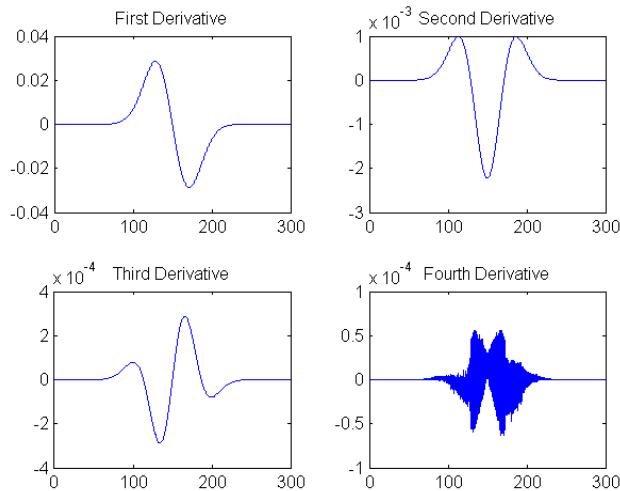
Ma quanto è serio questo problema nelle applicazioni reali? Per rispondere a questa domanda, lo script Matlab/Octave [RegressionNumericalPrecisionTest.m](#) applica il metodo CLS a una miscela di due picchi Gaussiani sovrapposti *molto ravvicinati senza rumore* (linee blu e verdi nella figura a lato) utilizzando *tre diverse formule matematiche* per il calcolo dei minimi quadrati che danno risultati diversi. La difficoltà di una tale misura dipende dal rapporto tra la separazione e la semi-larghezza dei picchi; rapporti piccoli



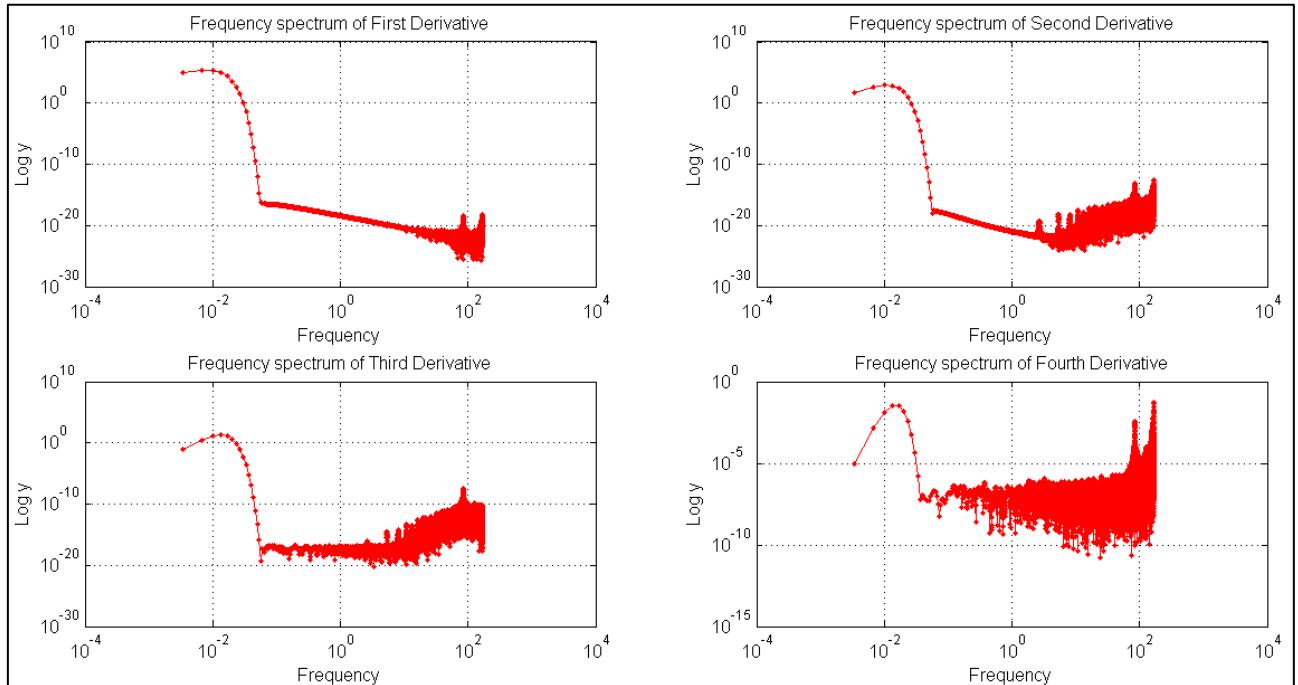
sovraposte con una *grande* differenza (50 volte) di altezza (le linee blu e verdi nella figura a lato, con picco rispettivamente a 500 e 550 nm); in questo caso il rapporto tra separazione e larghezza è 0.25, molto più grande (cioè più facile) rispetto all'esempio precedente. Per questo esempio, la simulazione mostra che gli errori percentuali relativi della misura dell'altezza sono dello 0,19% per il picco più grande e del 6,6% per quello più piccolo. È possibile modificare la risoluzione del convertitore analogico-digitale simulato nel *numero di bit* (riga 9). La risoluzione in ampiezza di un convertitore analogico-digitale è 2 elevata alla potenza del numero di bit. Le risoluzioni degli ADC sono solitamente 10, 12 e 14 bit, corrispondenti alle risoluzioni di una parte in 1024, 4096 e 16384, rispettivamente. Naturalmente, la risoluzione effettiva per il picco più piccolo, in questo caso, è 50 volte inferiore, e non si può semplicemente aumentare l'amplificazione sul picco più piccolo senza sovraccaricare l'ADC per quello più grande. Sorprendentemente, se la maggior parte del rumore nel segnale è questo tipo di rumore di digitalizzazione, può effettivamente aiutare ad aggiungere un po' di rumore casuale (specificato nella riga 10 in questo script), come si è visto a pagina 300.

Differenziazione. Un'altra applicazione in cui è possibile vedere il rumore della precisione numerica è nella [differenziazione](#), che implica la sottrazione di numeri adiacenti molto simili in una serie di dati. La funzione autonoma Matlab/Octave [DerivativeNumericalPrecisionDemo.m](#) mostra come i limiti della precisione numerica del computer influenzano le derivate dalla prima alla quarta di una banda Gaussiana che viene campionata molto finemente (oltre 16.000 punti nella semi-larghezza in questo caso) e che non ha rumore aggiunto. Il grafico a sinistra mostra le quattro forme d'onda a destra e i loro spettri di frequenza sono mostrati nella figura appena sotto. Il limite della

precisione numerica del computer crea rumore casuale a frequenze molto alte, che viene enfatizzato dalla differenziazione. Negli spettri di frequenza sottostanti, la grande macchia a bassa frequenza vicino a una frequenza di 10^{-2} è il *segnaletico* e tutto ciò che è sopra che è *rumore* numerico. Le derivate di ordine inferiore costituiscono raramente un problema, ma quando si raggiunge la derivata quarta, le frequenze di quel rumore si avvicinano all'intensità delle frequenze del segnale, come si può vedere nello spettro di frequenza della derivata quarta in basso a destra. Ma questo rumore è solo un rumore ad altissima frequenza, quindi lo smoothing con un minimo di 3 punti di di scorimento della media ne rimuove la maggior parte ([click per visualizzare](#)).



Un metodo alternativo di derivazione basato sulla trasformata di Fourier (pag. 84) ha errori numerici leggermente inferiori ma è usato raramente nella pratica (riferimento 88 La differenza fondamentale tra i due è che il metodo delle differenze finite funziona *localmente*, su un piccolo segmento di dati per volta, mentre il metodo FT funziona *globalmente* perché ogni frequenza nella rappresentazione di Fourier si estende per tutto il dominio del tempo. Lo script Matlab/Octave [FDvsFTderivative.m](#) confronta gli errori numerici dei metodi di differenziazione della differenza finita (FD) e della trasformata di Fourier (FT). Crea un picco Gaussiano molto ampio e finemente



campionato e poi ne calcola la derivata quarta in entrambi i modi. (Il rumore è causato solo dalle limitazioni della risoluzione numerica del software). Il risultato ([grafico](#)) è che gli errori numerici sono inferiori per il metodo FT in prossimità del picco ma maggiori lontano dal centro del picco.

Smoothing. Infine, potrebbe esserci un potenziale problema numerico con l'algoritmo [fastsmooth](#), trattato nella sezione sullo [smoothing](#), perché è un algoritmo *ricorsivo* che utilizza i risultati di un passaggio precedente per calcolare il passo successivo. La precisione numerica di [fastsmooth.m](#) è mostrata dallo script Matlab/Octave [FastsSmoothNumericalPrecisionTest.m](#). Anche per uno smoothing P-spline da 4000 punti applicato a un segnale di 100,000 punti, la deviazione standard relativa del rumore numerico è solo dello 0.00027%, e la maggior parte si verifica alle estremità del segnale (il primo 4% e l'ultimo 4% dei punti); l'errore oltre il 90% del segnale è *ordini di grandezza inferiore*, un problema trascurabile nei casi pratici.

Elaborazione miniaturizzata del signal processing: La Famiglia del Raspberry Pi



L'elaborazione dei segnali non richiede necessariamente costosi sistemi informatici. [Raspberry Pi](#) è una famiglia di computer a scheda singola piccoli ma straordinariamente potenti il cui costo varia tra \$ 10 e \$ 80. La maggior parte dei modelli ha all'incirca le dimensioni *di un mazzo di carte*. La maggior parte dispone di diverse porte USB, pin di ingresso-uscita

generici, porta HDMI, porta Ethernet, jack audio e video composito, interfacce per una fotocamera, slot per scheda micro-SD per archiviazione di massa, core grafico, Wireless LAN e Bluetooth. Si possono avere con una serie di software installati, inclusa una versione del sistema operativo Linux, un browser Web, la suite [LibreOffice](#) completa, [Mathematica](#) di Wolfram ([Screenshot](#)), diversi linguaggi di programmazione e varie utilità. [Tutti questi sono installati di default](#) sul programma di installazione del sistema operativo [del Raspberry Pi](#). (I modelli più piccoli ed economici sono ideali in situazioni in cui potrebbero essere danneggiati o persi, come negli esperimenti su razzi o palloncini). Sono disponibili molti [starter packs](#) che raggruppano tutti i componenti necessari.

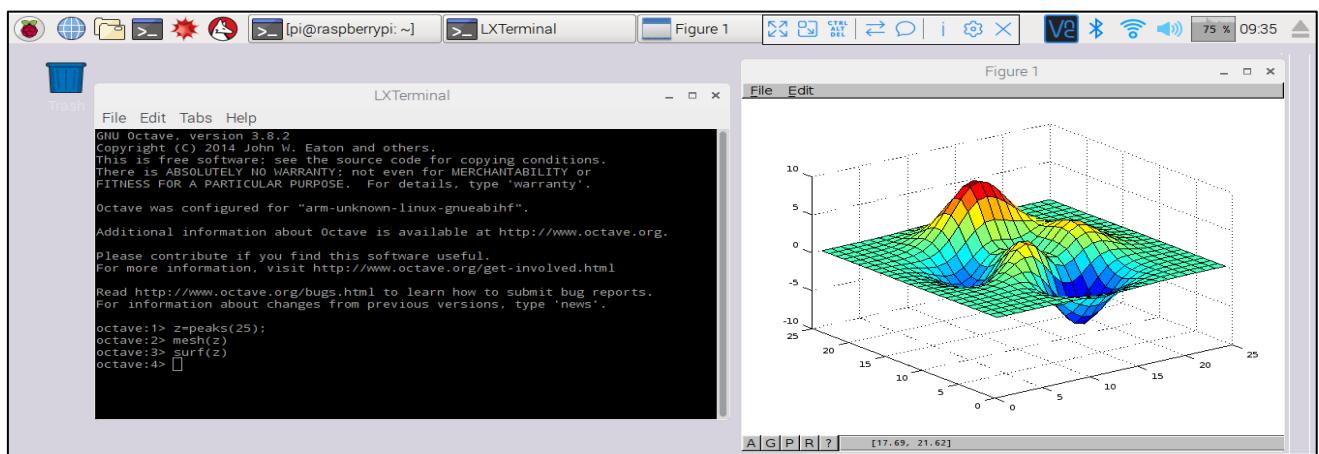
Si può facilmente costruire un computer Windows 10 completo da un Raspberry Pi versione 4 (\$ 70) integrato con una tastiera completa; serve solo un alimentatore USB-C da 5 volt, una TV/monitor con ingresso HDMI e un mouse (acquistabile di seconda mano) e una scheda mini SD (da 8 a 16 Gbyte) per l'archiviazione di massa (acquistabile con tutto il software già installato, oppure utilizzarne uno vuoto su cui scaricare autonomamente il software gratuito). Per scopi didattici, il Pi è stato utilizzato come alternativa a basso costo per i laboratori informatici scolastici, utilizzando il software incluso per entrambe le applicazioni di tipo Office (elaboratore di testi *Writer*, *Calc* foglio di calcolo, ecc.) e per istruzioni di programmazione (Python (pag. 427), C, C++, Java, Scratch, Ruby, ecc.).

Le versioni a scheda singola sono ideali per applicazioni “[headless](#)” (ovvero senza monitor, tastiera o mouse) dove, dopo la configurazione, è possibile accedervi solo da remoto tramite WiFi o Bluetooth, utilizzando [Putty](#) (per l'accesso in stile UNIX da riga di comando) o utilizzando un sistema grafico di condivisione del desktop come [RealVNC](#) (gratuito per Windows, Mac, IOS e Android), che riproduce l'intero desktop grafico sul dispositivo locale, completo di tastiera virtuale pop-up. Applicazioni tipiche sono come [server di file di rete](#), [stazione meteorologica](#), [media center](#) o come [telecamera di sicurezza](#) in rete. Può anche [condividere file con Windows](#).

Per le applicazioni di acquisizione di dati scientifici e [applicazioni di elaborazione dei segnali](#), la versione Pi di Linux ha tutti i ["soliti" comandi da terminale UNIX](#) per la raccolta, la ricerca, il filtraggio e il riepilogo dei dati. Inoltre, ci sono molte librerie aggiuntive per [Python](#), compreso [SciPi](#), [NumPy](#) e [Matplotlib](#), tutti scaricabili gratuitamente (pag. 427). Le 153 pagine del libro PDF di Allen B. Downey's ["Think DSP"](#) contengono molti esempi di codice Python per le tradizionali applicazioni di ingegneria. Sono disponibili dispositivi hardware aggiuntivi a basso costo, tra cui [video camere](#) e una [scheda sensore](#) [piggyback] che [legge e visualizza i dati provenienti da](#) diversi sensori integrati: giroscopio, accelerometro, magnetometro, barometro, termometro e per l'umidità relativa. (Si basa sullo [stesso hardware che al momento della stesura di questo articolo era in orbita sulla Stazione Spaziale Internazionale](#)).

Gli [spreadsheet per il signal processing](#) (pag. 480) funzionano perfettamente sulla versione di *Calc* del Pi, così come quelli di [calibrazione](#) (pag. 439) e i [modelli di strumenti analitici](#) (pag. 347).

Per le applicazioni scolastiche, [Element14](#) commercializza numerosi kit e attività didattici basati sul Raspberry Pi.



[Octave 3.6 può girare direttamente su alcune versioni](#) di Raspberry Pi; la schermata sopra mostra Octave 3.6 in esecuzione all'interno dell'interfaccia utente grafica integrata del Pi (mostrando le funzioni grafiche 3D "mesh" e "surf").

Ci sono molte [applicazioni per il laboratorio e sul campo](#), specialmente in combinazione con un [micro controllore Arduino](#). Ma se nessuno dei modelli Raspberry Pi disponibili è sufficientemente veloce per le proprie esigenze di elaborazione del segnale, allora si utilizzerà il Pi solo per l'acquisizione dei dati e si trasferiranno su un computer più veloce. (Per gli utenti MATLAB, esiste un [MATLAB Support Package for Raspberry Pi Hardware](#) che lo supporta. In alternativa, si potrebbe semplicemente fare in modo che il Raspberry Pi salvi i dati o i risultati in una [cartella condivisa](#) a cui si accede tramite Wi-Fi da un altro computer).

[Python](#) è il linguaggio di programmazione principale fornito con Raspberry Pi. Questo linguaggio è diverso da quelli più vecchi usati tradizionalmente dagli scienziati, come il Fortran o il Pascal. I programmatore Matlab possono utilizzare ChatGPT per convertire il proprio codice in Python (pagina 434). A titolo di confronto, ecco un [esempio in tempo reale di acquisizione e grafico dei dati su un Raspberry Pi](#), che misura la propria temperatura in funzione del tempo mentre si riscalda, utilizzando il componente aggiuntivo disponibile in commercio la scheda [Sense Hat](#) con un [programma scritto in Python](#) (cliccare per l'[animazione in tempo reale](#)). Se non si possiede una Sense Hat, ecco una [modifica dello stesso programma Python](#) che disegna la media corrente di un numero casuale, utilizzando la stessa tecnica grafica di auto-scala, mostrando graficamente un risultato che gradualmente si assesta sempre di più alla media man mano che avanza. Questo [script Matlab/Octave](#) fa la stessa cosa alla stessa velocità, ma nel caso dello script Matlab/Octave è sostanzialmente più corto. (Per un confronto più esteso di Python con Matlab per diverse attività di elaborazione del segnale, vedere pag. 427).

Tra i sistemi concorrenti c'è [BeagleBoard](#) e [LattePanda](#), un minuscolo PC con Windows-10 da \$130 con 2 Gbyte di RAM e 32 Gbyte di memoria flash. [Esistono molti prodotti simili](#).

Elaborazione batch

In situazioni in cui si dispone di una grande quantità di dati simili da elaborare, è utile automatizzare il processo. Si supponga di aver già acquisito dati sotto forma di più file di testo o file di dati numerici di un formato standardizzato memorizzati in una directory nota (cartella) da qualche parte sul computer. Ad esempio, potrebbero essere file ASCII .txt (testo normale) o .csv ("valori separati da virgola") con la variabile indipendente ('x') nella prima colonna e una o più variabili dipendenti ('y') nella altre colonne. Potrebbe esserci un numero variabile di file i cui nomi e

le lunghezze potrebbero essere variabili, ma, soprattutto, il *formato* dei dati è coerente da file a file. Si potrebbe scrivere uno script o una funzione Matlab che elaborerà quei file *uno per uno*, ma sarebbe bello se i computer potesse esaminare *tutti* i file nella directory *automaticamente*, determinare i nomi dei file, caricare ciascuno nello spazio di lavoro delle variabili, applicare le operazioni di elaborazione desiderate (rilevamento del picco, deconvoluzione, approssimazione della curva, wavelet, qualunque cosa), mostrare tutto l'output nella finestra del terminale, ognuno etichettato con il nome del file, aggiungere tali risultati a un file di "log" sempre crescente, quindi passare al file successivo. Idealmente, il programma *non dovrebbe fermarsi* incontrando un qualsiasi tipo di errore fatale; dovrebbe, invece, semplicemente *saltarlo e passare al successivo*. Sembra complicato, ma è più facile di quanto sembri.

[BatchProcess.m](#) è un esempio Matlab/Octave di un processo automatizzato di questo tipo utilizzabile come framework per le proprie applicazioni. Per adattare questo script ai propri scopi, si deve cambiare solo:

- (a) il nome della directory in cui sono archiviati i dati sul computer - ("DataDirectory") nella riga 11;
- (b) il nome della directory in cui sono memorizzate le funzioni di signal processing di Matlab sul computer - ("FunctionsDirectory") nella riga 12; e
- (c) le effettive funzioni di elaborazione da applicare a ciascun file (che in questo esempio eseguono l'approssimazione dei picchi utilizzando la funzione "[peakfit.m](#)" nelle righe 34–41, ma potrebbero essere *qualsiasi cosa*).

All'avvio, la routine crea e apre un file di "log" nella riga 21, che sarà posto nella FunctionsDirectory, con il nome del file "BatchProcess<date>.txt" (dove <date> è la data corrente e, p.es. 12-Jun-2022). Questo file cattura tutto l'output del terminale durante l'elaborazione: in questo esempio, si sta usando la funzione peakfit.m che genera una matrice FitResults (con Peak#, Position, Height, Width e Area del modello trovato), e una matrice Goodness of Fit (GOF) [approssimazione migliore] contenente l'errore di approssimazione percentuale e il valore R², per ciascun file di dati in quella directory. Le successive esecuzioni del programma nella stessa data vengono aggiunte a questo file. Ogni giorno successivo, viene avviato un nuovo file per quel giorno. È anche possibile salvare facoltativamente alcune delle variabili dello spazio di lavoro nel file dei dati; aggiungere una funzione "save" dopo l'elaborazione e prima dell'istruzione "catch me" (digitare "help save" al prompt dei comandi per le opzioni).

Questo programma utilizza delle tecniche di codifica particolarmente utili nell'elaborazione automatica dei file. Utilizza le "function forms" di diversi comandi "ls" (riga 13), "diary" (riga 21) e "load" (riga 29) per consentire poi di accettare le variabili calcolate all'interno del programma. Utilizza anche la struttura "[try/catch/end](#)" (righe 28, 47, 49), che impedisce al programma di arrestarsi se incontra un errore su uno dei file di dati. Se si verifica un errore, aggiunge una riga al log che riporta l'errore per quel file e passa a quello successivo. (Nota: non sorprende che Python abbia funzioni simili chiamate "[try..except..finally](#)" e [pydiary](#)).

Dopo aver eseguito questo script, il file di log "BatchProcess ..." conterrà tutto l'output del terminale. Ecco un estratto da un tipico file di log. In questo esempio, *i primi due file nella directory hanno restituito errori*, ma il terzo ("2016-08-05-RSCT-2144.txt") e tutti i successivi hanno funzionato normalmente e hanno riportato i risultati delle operazioni di approssimazione del picco:

Error with file number 1.

Error with file number 2.

3: 2016-08-05-RSCT-2144.txt

Peak#	Position	Height	Width	Area
1	6594.2	0.1711	0.74403	0.13551
2	6595.1	0.16178	0.60463	0.1041

% fitting error R2
2.5735 0.99483

4: 2016-09-05-RSCT-2146.txt

Peak#	Position	Height	Width	Area
1	6594.7	0.11078	1.4432	0.17017
2	6595.6	0.04243	0.38252	0.01727

% fitting error R2
4.5342 0.98182

5: 2016-09-09-RSCT-2146.txt

Peak#	Position	Height	Width	Area
2	6594	0.05366	0.5515	0.0315
1	6594.9	0.1068	1.2622	0.1435

% fitting error R2
3.709 0.98743

6: Ecc....

Nota: facoltativamente, è possibile importare il file di log in Excel aprendo un foglio di lavoro Excel, cliccare su una cella, cliccare su **Data > From Text**, selezionare il file di log, cliccare per *specificare che gli spazi devono essere utilizzati come separatori di colonna*, e cliccare su **Import**. Questo metterà tutto l'output del terminale raccolto in quel foglio di calcolo. Inoltre, si possono salvare le variabili dell'area di lavoro (ad esempio, come file .mat).

Il signal processing in tempo reale

Tutte le tecniche di signal processing trattate finora presuppongono che i dati siano stati acquisiti e archiviati nella memoria del computer prima di iniziare l'elaborazione. In alcuni casi, tuttavia, è necessario eseguire l'elaborazione del segnale in "tempo reale", cioè punto per punto mentre i dati

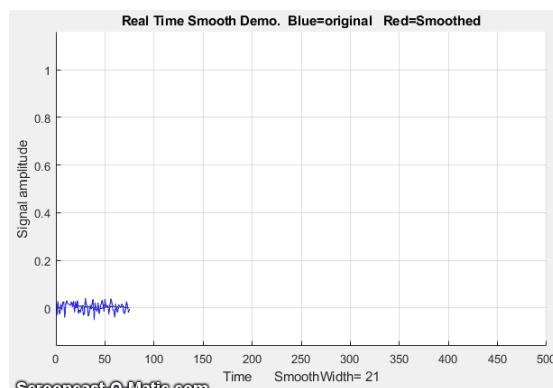
vengono acquisiti dal sensore o dallo strumento. Ciò richiede alcune modifiche al software, ma le i conci principali sono ancora validi. In questa sezione esamineremo i modi per eseguire i grafici dei dati, lo smoothing, la differenziazione, il rilevamento dei picchi, l'analisi delle armoniche (spettri di frequenza) e il filtraggio di Fourier in tempo reale. Poiché i dettagli per l'acquisizione dei dati variano con ogni singolo sperimentatore e configurazione strumentale, questi script dimostrativi *simuleranno* i dati in tempo reale in modo che si possano eseguire immediatamente sul proprio computer per vedere come funzionano, senza alcun hardware aggiuntivo. Lo si farà in due modi:

- (a) utilizzando i clic del mouse per generare i dati punto per punto, utilizzando la funzione "ginput" di Matlab, o
- (b) pre-calcolando alcuni dati simulati e poi accedendovi punto per punto in un ciclo.

Il primo metodo è illustrato dal semplice script [realtime.m](#). Quando si esegue questo script, viene visualizzato un sistema di coordinate grafico. Posizionare il puntatore del mouse lungo l'asse y (verticale) e cliccare per inserire i punti mentre ci si sposta col puntatore del mouse su e giù. La funzione "ginput" aspetta ogni clic del pulsante del mouse, quindi il programma registra la posizione delle coordinate y e conta il numero dei click. I punti vengono assegnati al vettore y (riga 17), disegnati come punti neri sul grafico (riga 18), e stampati nella finestra di comando (riga 19). Lo script [realtimeplotautoscale.m](#) è una versione espansa che cambia la scala del grafico man mano che arrivano i dati. Se il numero di punti supera 20 ('maxdisplay'), il massimo dell'asse x viene ridimensionato al doppio di del numero di punti (riga 32). Se l'ampiezza dei dati è uguale o superiore a ('maxy'), l'asse y viene ridimensionato a 1,1 volte l'ampiezza dei dati (riga 36).

Il secondo metodo è illustrato dallo script [realtimeplotautoscale2.m](#), che simula i dati in tempo reale utilizzando [dati pre-calcolati](#) (caricati dal disco rigido nella riga 13) a cui si accede punto per punto nelle righe 25 e 26 (Se l'animazione non è visibile, cliccare sulla figura per aprirla in un browser web). Un altro script, [realtimeplotdatedtime.m](#), mostra un modo per utilizzare la funzione 'clock' di Matlab per registrare la data e l'ora di ogni punto acquisito col clic. (Si potrebbe anche fare in modo che il computer controlli l'ora di acquisizione dei dati leggendo l'orologio in un ciclo fino all'arrivo dell'ora e della data desiderate, poi prendere un punto). Ovviamente, una macchina Windows non è proprio l'ideale per l'acquisizione di dati ad alta velocità e con tempi precisi, perché in genere ci sono tantissime interruzioni e altri processi in esecuzione in background, ma è adeguata per le applicazioni a bassa velocità. Per velocità più elevate, sono disponibili [hardware specializzati](#) e [software](#).

Smoothing. Lo script [RealTimeSmoothTest.m](#) mostra lo [smoothing \(pag. 38\)](#) in tempo reale,

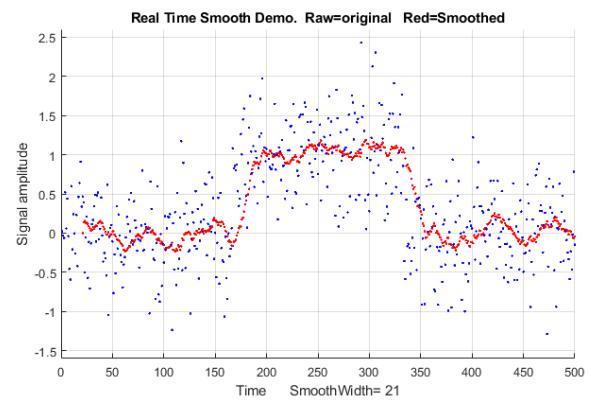


disegna i dati originali con linee blu e quelli filtrati con smoothing in rosso. In questo caso, lo script pre-calcola i dati simulati nella riga 28 e poi accede ai dati singolarmente nel ciclo 'for' di elaborazione (righe 30-51). Il numero totale di punti è controllato da 'maxx' nella riga 17 (inizialmente impostato a 1000) e la larghezza dello smoothing(in punti) è controllata da 'SmoothWidth' nella riga 20. (Per farlo con i dati in

tempo reale provenienti dal sensore, commentare la riga 29 e sostituire la riga 32 con il codice che acquisisce un punto dal sensore).

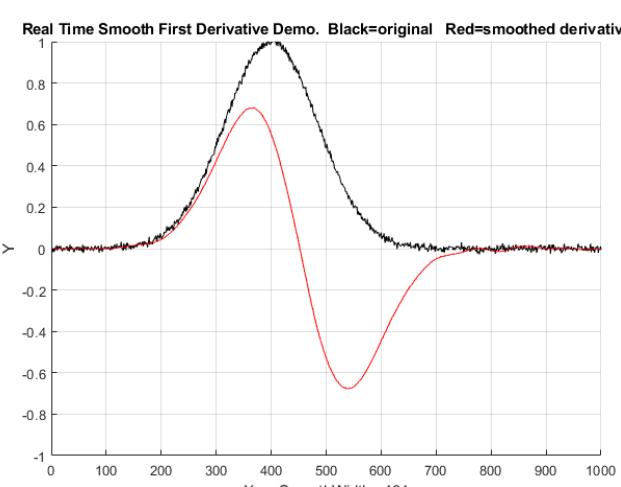
Come si può vedere nell'immagine sullo schermo sopra a sinistra ([link all'animazione](#)), i dati filtrati con smoothing (in rosso) sono *in ritardo* rispetto ai dati originali, perché lo smoothing non può essere calcolato fino a quando non è stato acquisito un numero di punti pari alla larghezza dello smoothing - 21 punti in questo esempio. (Tuttavia, conoscendo l'ampiezza dello smoothing, è possibile correggere le posizioni registrate sull'asse y delle caratteristiche del segnale, come massimi, minimi, picchi e punti di flesso). Questo particolare esempio implementa uno **smoothing a slittamento della media**, ma se ne possono implementare di altri tipi semplicemente de-commentando la riga 24 (rettangolare), la 25 (triangolare), o la 26 (Gaussiano), che richiedono che '[triangle](#)' e '[gaussian](#)' stiano nel path di ricerca di Matlab/Octave.

Un'applicazione pratica di smoothing a 'slittamento della media' come questo è in un sistema di controllo in cui un segnale rumoroso accende una valvola, un interruttore o un segnale di allarme ogni volta che il segnale supera un certo valore. Nell'esempio mostrato nella figura a lato, il valore della soglia è 0,5 e il segnale (punti blu) è così rumoroso che è necessario uno smoothing per evitare che il segnale attivi prematuramente il controllo. Un smoothing eccessivo, però, causerà un ritardo inaccettabile nel funzionamento. In questo caso, l'allarme si attiva a circa $t=160$ e si disabilita a 350.



Su un PC standard (Intel Core i5 3 Ghz) con Windows 10 home, l'operazione di smoothing aggiunge circa 2 microsecondi per ogni punto al tempo di acquisizione dei dati (senza disegnare, PlottingOn=0 nella riga 20) e 20 millisecondi (50 Hz max) col disegno punto per punto (PlottingOn=1). Col disegno disattivato, lo script acquisisce, esegue lo smoothing e memorizza i dati filtrati nella variabile "sy" in tempo reale, poi disegna i dati solo dopo che l'acquisizione sia stata completata, il che è molto più veloce della stampa in tempo reale.

Differenziazione. Lo script [RealTimeSmoothFirstDerivative.m](#) mostra la **differenziazione con smoothing in tempo reale** (pagina 58), utilizzando un semplice algoritmo basato sulle differenze adiacenti (riga 47) e disegna i dati originali come una linea nera e la derivata prima in rosso. Lo



script dimostrativo

[RealTimeSmoothSecondDerivative.m](#) calcola la derivata *seconda* con smoothing utilizzando un algoritmo basato sulla differenza centrale (riga 47). Entrambi gli script pre-calcolano i dati simulati nella riga 28 per poi accedervi punto per punto nel ciclo di elaborazione (righe 31-52). In entrambi i casi, il numero massimo di punti è impostato nella riga 17 e la larghezza dello smoothing è impostata nella riga 20. Anche in questo caso, le derivate sono in ritardo rispetto al segnale originale.

Qualsiasi ordine di derivazione è calcolabile in questo modo utilizzando i coefficienti di

derivazione nelle funzioni derivative Matlab/Octave elencate nella [pagina 71](#).

Rilevamento dei picchi. Il piccolo script [realtimepeak.m](#) mostra un semplice rilevamento del picco

in tempo reale basato sul passaggio per lo zero della derivata (pagina 227), utilizzando i clic del mouse per simulare i dati. Ogni volta che il clic del mouse forma un picco (cioè, va su e poi di nuovo giù), il programma registrerà ed etichetterà il picco sul grafico e stamperà i suoi valori x e y .

Peak detected at $x=13$ and $y=7.836$

Peak detected at $x=26$ and $y=1.707$

In questo caso, un picco è definito come qualsiasi punto che ha punti adiacenti di ampiezza inferiore su entrambi i lati, determinato dai cicli "for" annidati nelle righe 31-36. Naturalmente, un picco non può essere registrato fino a quando non viene registrato il punto successivo, perché non c'è modo di prevedere in anticipo se quel punto sarà inferiore o superiore al punto precedente. Se i dati sono rumorosi, è meglio eseguire lo *smoothing* del flusso di dati prima di rilevare i picchi, che è esattamente ciò che fa lo script Matlab/Octave [RealTimeSmoothedPeakDetection.m](#), il che riduce la possibilità di falsi picchi dovuti al rumore casuale, ma ha lo svantaggio di ritardare ulteriormente il rilevamento dei picchi. Ancora meglio, lo script Matlab/Octave

[RealTimeSmoothedPeakDetectionGauss.m](#)

utilizza la tecnica descritta nella [pagina](#) 230; individua i picchi positivi in un insieme di dati rumoroso che salgono al di sopra di una soglia di ampiezza impostata ("AmpThreshold" nella riga 55), esegue un'[approssimazione della curva dei quadrati minimi a una funzione Gaussiana](#) alla parte superiore del picco dei dati originali (riga 58), identifica ogni picco (riga 59), ne calcola la posizione, l'altezza e la larghezza (FWHM) di ciascuno da quell'approssimazione e stampa ogni picco trovato nella finestra di comando. I

parametri dei picchi vengono misurati sui dati originali, quindi non vengono distorti dallo smoothing. L'etichetta "peak" viene visualizzata accanto a ciascun picco rilevato una frazione di secondo dopo la parte superiore del picco, ma *i tempi effettivi elencati nella tabella stampata si basano sui dati originali e non vengono ritardati*. In questo esempio, i tempi effettivi dei picchi sono $x=500, 1000, 1100, 1200, 1400$. (Notare anche che il primo picco visibile, in $x = 300$, non viene elencato perché scende al di sotto della soglia dell'ampiezza, che in questo caso è 0,1). Se però quel picco è importante, si può semplicemente impostare la soglia su un valore più basso, ad es. 0.02). Link all'[animazione](#).

Peak detected at $x=500.1705$, $y=0.42004$, width= 61.7559

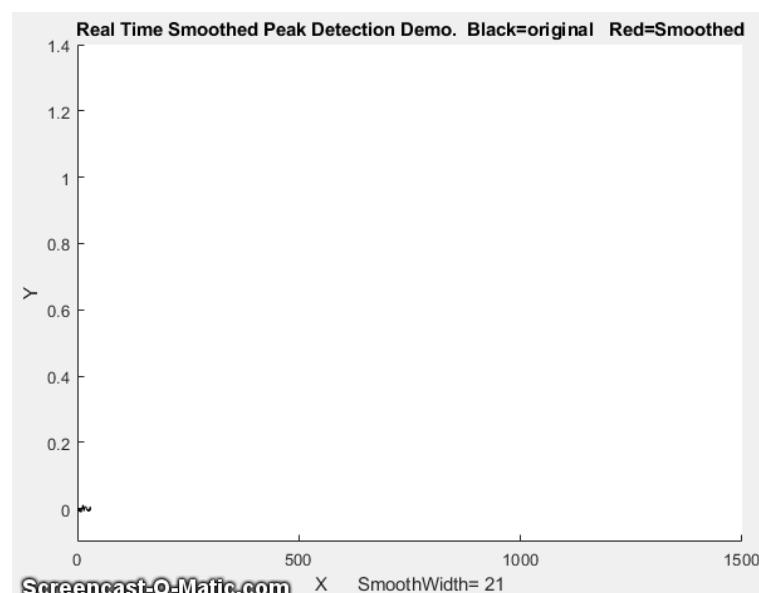
Peak detected at $x=1000.0749$, $y=0.18477$, width= 61.8195

Peak detected at $x=1100.033$, $y=1.2817$, width= 60.1692

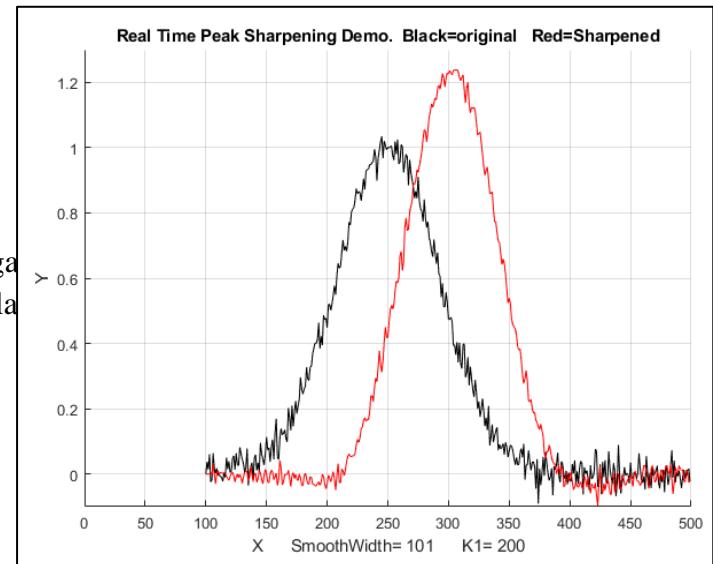
Peak detected at $x=1199.8493$, $y=0.36407$, width= 63.8316

Peak detected at $x=1400.1473$, $y=0.26134$, width= 58.9345

Lo script inoltre [numera i picchi](#) e salva i parametri di tutti i picchi in una matrice chiamata PeakTable, che è possibile interrogare ogni volta che si incontra ogni picco se si cercano particolari profili. Vedere pagina 242 per alcune idee sull'uso della notazione Matlab/Octave e sulle funzioni per farlo.



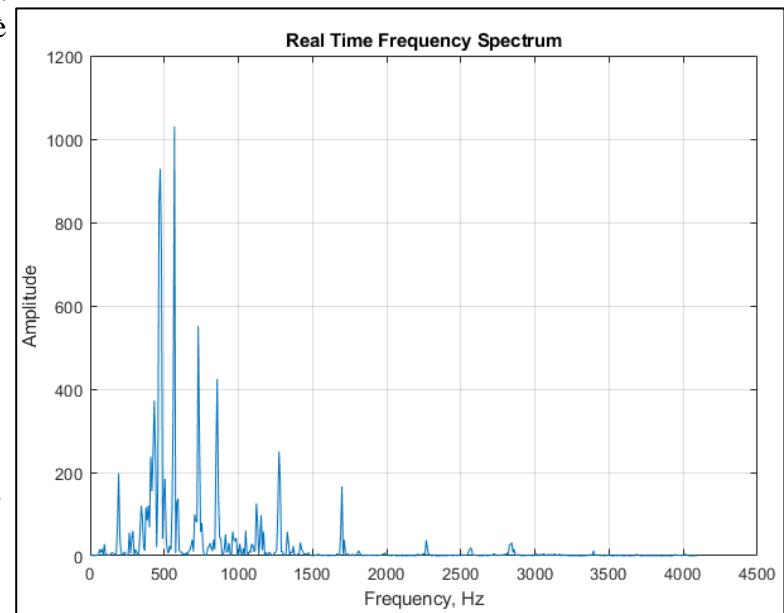
Sharpening del picco. Lo script Matlab/Octave [RealTimePeakSharpening.m](#) mostra lo **sharpening del picco** in tempo reale (pag. 74) utilizzando la tecnica della derivata seconda. Utilizza dati simulati pre-calcolati nella riga 30 e poi accede ai dati punto per punto nel ciclo di elaborazione (righe 33-55). In entrambi i casi il numero massimo di punti è impostato nella riga 17, l'ampiezza dello smoothing è impostata nella riga 20 e il fattore di ponderazione (K_1) è settato nella riga 21. Nell'esempio a lato, l'ampiezza dello smoothing è di 101 punti, che tiene conto del ritardo nel picco con sharpening rispetto all'originale.



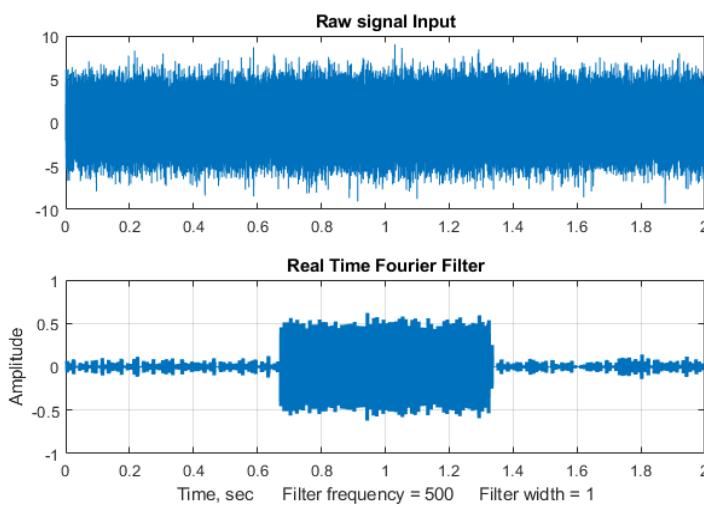
Spettro di Frequenza in tempo reale. Lo script

[RealTimeFrequencySpectrumWindow.m](#) calcola e disegna lo spettro di frequenza di Fourier di un segnale (pag. 87). Come gli script precedenti, simula il real-time dei dati caricandoli da un "file .mat" e poi accede a tali dati punto per punto nel ciclo di elaborazione. Una variabile critica in questo caso è "WindowWidth", il numero di punti presi per calcolare ogni spettro di frequenza. Maggiore è questo numero, minore sarà il numero di spettri che verranno generati, ma maggiore sarà la risoluzione in frequenza. Su un PC medio (Intel Core i5 3 Ghz con Windows 10 home), questo script genera circa 50 spettri al secondo con una velocità media (punti al secondo) di circa 50.000 Hz. Spettri più piccoli ((cioè valori inferiori di WindowWidth) generano velocità medie dei dati proporzionalmente inferiori (perché il flusso del segnale viene interrotto più spesso per calcolare e rappresentare graficamente uno spettro).

Se il flusso di dati è un segnale audio, è anche possibile riprodurre il suono attraverso gli altoparlanti del computer sincronizzato con la visualizzazione degli spettri di frequenza; per fare ciò, impostare la variabile `PlaySound` ad un valore di 1. Ciascun segmento del segnale viene riprodotto appena prima che venga visualizzato lo spettro di quel segmento, come mostrato a lato. (La riproduzione del suono non sarà perfetta, a causa del leggero ritardo durante il quale il computer calcola e visualizza lo spettro prima di passare al segmento successivo. In questo script dimostrativo, il file di dati è musica - infatti è un estratto di 8 secondi di una registrazione audio del 'Coro dell'Alleluia' dal *Messiah* di Handel, con una frequenza di campionamento di 8192 Hz. Questo file è incluso come dati dimostrativi nella distribuzione Matlab (`handel.mat`). La figura a lato mostra uno dei 70 spettri generati con una WindowWidth di 1024. È possibile modificare l'argomento della funzione 'pause' per il computer per ridurre al minimo questo problema e per riprodurre il suono regolarmente con la tonalità corretta.



Filtro di Fourier in tempo reale. Lo script [RealTimeFourierFilter.m](#) è una dimostrazione di un



larghezza della finestra di 1000 punti, questo script genera circa 35 segmenti filtrati al secondo con una velocità media (punti al secondo) di circa 34.000 Hz. Segmenti più piccoli (cioè, valori inferiori di WindowWidth) generano una velocità media proporzionalmente inferiore (perché il flusso del segnale viene interrotto più spesso per calcolare e rappresentare graficamente lo spettro filtrato). Il risultato dell'applicazione del filtro a ciascun segmento viene visualizzato in tempo reale durante l'acquisizione dei dati, poi alla fine lo script confronta l'intero input del segnale originale con l'uscita filtrata riassemblata, mostrata nella figura a lato.

In questa dimostrazione, il filtro di Fourier è in modalità passa-banda e viene utilizzato per rilevare un'onda sinusoidale di 500 Hz (la frequenza 'f' è nella riga 28) che appare a metà di un segnale molto rumoroso (riga 32), da circa 0,7 sec a 1,3 sec; l'onda sinusoidale a 500 Hz è *così debole che non è per nulla visibile* nel segnale originale (pannello superiore della figura a lato), ma spicca nell'uscita filtrata (pannello inferiore). La frequenza centrale (CenterFrequency) e la larghezza (FilterWidth) del filtro sono impostate nelle righe 46 e 47.

I quadrati minimi classici in tempo reale. La classica tecnica dei quadrati minimi (pagina 179) è applicabile in tempo reale alla cromatografia 2D con rilevatori a matrice che acquisiscono uno spettro completo più volte in un secondo per l'intero cromatogramma. Questo viene indagato in "Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo" a pagina 354.

Per applicare uno qualsiasi di questi esempi a dei dati in tempo reale provenienti da un sensore o da uno strumento, c'è bisogno solo del ciclo 'for' principale di elaborazione, sostituendo le prime righe dopo l'istruzione 'for' con una chiamata a una funzione che acquisisca un singolo punto di dati originali e lo assegna a y(n). Se non c'è bisogno di disegnare i dati punto per punto in tempo reale, si possono velocizzare notevolmente le cose rimuovendo l'istruzione "drawnow" alla fine del ciclo 'for' o rimuovendo tutto il codice per il disegno.

Negli esempi qui, l'*output* dell'elaborazione viene utilizzato per disegnare o stampare i dati elaborati punto per punto, ma ovviamente potrebbe anche essere utilizzato come input per un'altra funzione o verso un convertitore digitale-analogico o semplicemente per attivare un allarme se si ottengono determinati risultati specifici (ad esempio, se il segnale supera un certo valore per un periodo di tempo specificato, o se un picco viene rilevato in una posizione o altezza specificata, eccetera.).

filtro di Fourier in tempo reale. Lo script pre-calcola un segnale simulato a partire dalla riga 38, poi accede ai dati punto per punto (righe 56, 57) e divide il flusso di dati in segmenti per calcolare ciascuna sezione filtrata. "WindowWidth" (riga 55) è il numero di punti in ogni segmento.

Maggiore è questo numero, minore sarà il numero di segmenti che verranno generati, ma maggiore sarà la risoluzione della frequenza all'interno di ciascun segmento. Su un PC desktop medio (Intel Core i5 3 Ghz con Windows 10 home), con una

Gestione degli array di dati variabili negli spreadsheet

Quando si utilizzano spreadsheet del tipo descritto in questo libro, spesso si devono modificarli per adattarli a un numero diverso di punti o di componenti. Ciò può risultare noioso, soprattutto perché è necessario ricordare la sintassi di ciascuna delle funzioni del foglio di calcolo che si desidera modificare. Questa sezione descrive i modi per costruire fogli di calcolo che si adattano *automaticamente* a diversi set di dati, senza che si debba dedicare tempo e fatica a modificare le formule per ogni caso. Ciò implica l'utilizzo di alcune funzioni native poco utilizzate in Excel o in OpenOffice Calc, come MATCH, INDIRECT, COUNT, IF e AND.

La funzione MATCH. Nel signal processing con gli spreadsheet, è normale avere degli array x-y array di lunghezza variabile, come spettri (x =lunghezza d'onda, y =assorbanza o intensità) o chromatogrammi (x =tempo, y =risposta del rilevatore). Ad esempio, si consideri questo piccolo array di valori x e y raffigurati nel frammento del foglio di calcolo a lato. Le formule del foglio di calcolo normalmente si riferiscono alle celle in base all'indirizzo di riga e colonna, ma per un set di dati x-y come questo, è più naturale fare riferimento a un punto tramite la sua variabile indipendente x , piuttosto che al suo indirizzo di riga e colonna. Ad esempio, supponiamo di voler selezionare il punto corrispondente a $x=2$, indipendentemente dalle celle in cui si trovano. Lo si può fare con la funzione MATCH. Ad esempio, se imposta la cella B2 sul valore x desiderato (p.es. 2), allora $\text{MATCH}(\text{B2},\text{A5}: \text{A11})+\text{ROW}(\text{A5})$ restituirà il numero di riga di quel punto, che in questo caso è 6. Successivamente, se si dovesse spostare o espandere questa tabella, trascinandola o inserendo o eliminando righe o colonne, il foglio di calcolo regolerà automaticamente la funzione MATCH per compensare, restituendo il nuovo numero di riga del punto richiesto.

La funzione INDIRECT. Il modo usuale per fare riferimento al valore in una cella è specificarne l'indirizzo di riga e colonna. Ad esempio, nella matrice dei valori x e y nella foto sopra, per fare riferimento al contenuto della colonna B, riga 6, è possibile scrivere “=B6”, che in questo caso restituirà 5.9. Questo è indicato come indirizzamento “diretto”. Al contrario, per utilizzare l'indirizzamento “indiretto” si può scrivere “=INDIRECT(“B”&A1)”, poi inserire il numero “6” nella cella A1. Il carattere “&” è semplicemente “colla” che unisce “B” al contenuto di A1, quindi in quel caso “B”&A1 viene valutato come “B6” e il risultato è lo stesso di prima: il contenuto della cella B6, che è 5.9. Tuttavia, se si modifica la cella A1 in 9, allora “B”&A1 viene valutato come “B9” e il risultato sarà il contenuto della cella B9, che è 9.1. In altre parole, la funzione INDIRECT consente di calcolare delle celle *all'interno del foglio di calcolo* anziché digitare dei numeri fissi. Ciò consente ai fogli di calcolo di regolare i propri indirizzi in base a un risultato calcolato, ad esempio per regolare i propri calcoli in modo che si adattino al numero di punti in quel set di dati.

Questi esempi sono stati realizzati in quello che viene chiamato lo stile di riferimento “A1”, in cui le colonne sono indicate con lettere; è anche possibile utilizzare lo stile di riferimento “R1C1”, dove sia alle righe che alle colonne ci si riferisce con numeri. Per esempio, “=INDIRECT(“R”&A2&“C”&A1, FALSE)”, con il numero di riga in A2 e il numero della colonna in A1. (“FALSE” significa semplicemente che viene utilizzato lo stile di riferimento “R1C1”).

È possibile utilizzare la stessa tecnica per calcolare *intervalli* di indirizzi di celle. Ad esempio, supponiamo di voler calcolare la somma di tutti i numeri nella colonna B tra una prima riga specificata e l'ultima riga specificata. Se si inserisce il numero della prima riga in A1 e il numero dell'ultima riga in A2, l'indirizzo della *prima* cella sarà “B”&A1 e quello dell'*ultima* cella sarà “B”&A2. Quindi, l'intervallo di indirizzi di cella si formerebbe utilizzando il carattere “&” per incollare assieme questi due indirizzi, con un carattere “:” nel mezzo (“B”&A1&”:B”&A2). La

A	B
1	
2	
3	
4	x y
5	1 5
6	2 5.9
7	3 7
8	4 8
9	5 9.1
10	6 10
11	7 11

somma sarebbe `SUM(INDIRECT("B"&A1&":B"&A2))`, che è 56. Sì, è più lungo, ma il vantaggio rispetto all'indirizzamento diretto è che si può regolare l'intervallo cambiando solo due celle anziché ri-digitare la formula. È lo stesso per altre funzioni che richiedono un intervallo di celle, come `AVERAGE`, `MAX`, `MIN`, `STDEV`, ecc. Per esempi del suo utilizzo, vedere pagina 51.

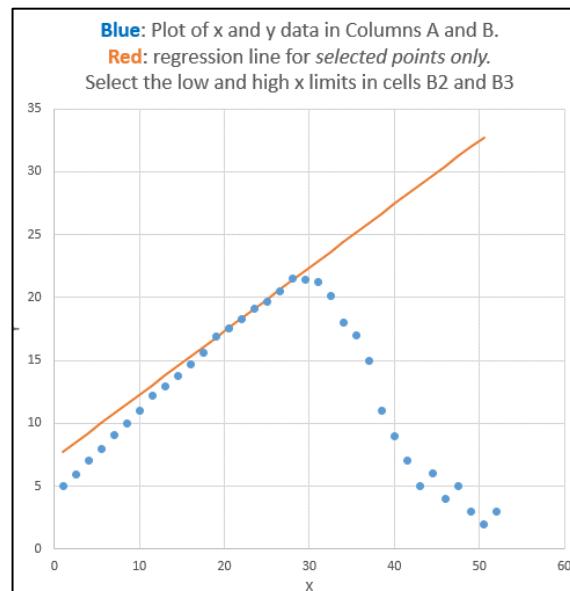
Per le funzioni che richiedono *due* intervalli, separati da una virgola, si può utilizzare la stessa tecnica. Supponiamo di voler calcolare la pendenza della linea di regressione lineare tra i valori x nella colonna A e i valori y nella colonna B nello stralcio di foglio di calcolo nella pagina precedente, utilizzando la funzione nativa `SLOPE`. `SLOPE` richiede due intervalli, prima i valori dipendenti (y) e poi i valori x indipendenti. Con l'indirizzamento *diretto*, la pendenza è `SLOPE(B5:B11,A5:A11)`. Con l'indirizzamento *indiretto*, sono necessarie due funzioni "indirette", una per ogni intervallo, separate da una virgola. Ecco come appare tutto insieme: `SLOPE(INDIRECT("B"&A1&":B"&A2),INDIRECT("A"&A1&":A"&A2))`, dove i valori x stanno nella colonna A, le y nella colonna B e i numeri della prima e dell'ultima riga stanno nelle celle A1 e A2 rispettivamente. Funziona allo stesso modo per le due funzioni correlate che calcolano `INTERCEPT` e `RSQ` (il valore di R^2) della retta della regressione. Sono d'accordo sul fatto che inizialmente sia fonte di confusione, ma funziona.

Un esempio funzionante. Un esempio dell'uso delle funzioni `MATCH` e `INDIRECT` che lavorano in coppia è mostrato nello spreadsheet "[SpecialFunctions.xlsx](#)" ([Grafico](#)), che ha una tabella più grande di dati x-y memorizzati nelle colonne A e B, a partire dalla riga 7. L'idea qui è che si può

	A	B	C	D	E	F
1	Select data range			Use of MATCH function		
2	First x value	20.0	first selected row number =	19		
3	Last x value	29.0	last selected row number =	25		

selezionare un intervallo limitato di valori x con cui lavorare digitando il valore *più basso* di x e quello *più alto* di x nelle celle B2 e B3, le due celle con uno sfondo giallo. Il foglio

di calcolo utilizza le funzioni `MATCH` nelle celle F2 e F3 per calcolare i numeri di riga corrispondenti, che vengono quindi utilizzati nelle funzioni `INDIRECT` nella sezione "Properties of selected data range" per calcolare il massimo, la media e la media di x e y nonché i valori della pendenza, dell'intercetta e di R^2 della retta di regressione lineare delle y *rispetto a x* (pagina 154) sull'intervallo delle x selezionato. La retta della regressione, che approssima *solo* i dati da x=20 a 29, è mostrata in rosso nel grafico a lato, sovrapposta all'intero set di dati (i punti blu). Modificando semplicemente i limiti dell'asse x nelle celle B2 e B3, *il foglio di calcolo e il grafico vengono ricalcolati, senza che si debba modificare nessuna delle formule delle celle*. Da provare. (A proposito, si può spostare il puntatore del mouse su qualsiasi cella con un segno rosso nell'angolo in alto a destra per rivelare la formula della cella o una spiegazione).



Le colonne J e K di questo foglio mostrano anche come utilizzare le funzioni "IF" e "AND" per copiare i dati dalle colonne A e B nelle colonne J e K solo quei punti che rientrano tra i due limiti di x specifici.

Volendo, si possono aggiungere più dati alla fine delle colonne A e B, limitati solo dall'intervallo delle funzioni `MATCH` nelle celle F2 e F3 (che sono inizialmente impostate su 1000, ma potrebbero essere grandi quanto serve). Il numero totale di valori numerici nel set di dati viene calcolato nella cella I15, utilizzando la

funzione “COUNT” (che, come suggerisce il nome, conta il numero di celle, che contengono un numero, in un intervallo).

Misura della posizione di un picco. Una comune operazione di elaborazione del segnale è trovare il valore dell'asse x dove il valore dell'asse y è massimo. Questo può essere suddiviso in tre fasi: (1) determinare il valore y massimo nell'intervallo selezionato con la funzione MAX; (2) determinare il numero di riga in cui appare quel numero con la funzione MATCH, e (3) determinare il valore di x in quella riga con la funzione INDIRECT. Questi passaggi sono illustrati nello stesso foglio di calcolo “[SpecialFunctions.xlsx](#)” nella colonna H, righe 20-23. Il risultato è che il massimo y (21,5) si verifica in x=28. I tre passaggi possono anche essere combinati in un'unica lunga formula (cella H23), sebbene sia più difficile da leggere rispetto alle formule con i passaggi separati. Lo [spreadsheet per la ricerca dei picchi](#) discusso a pagina 265 utilizza questa tecnica.

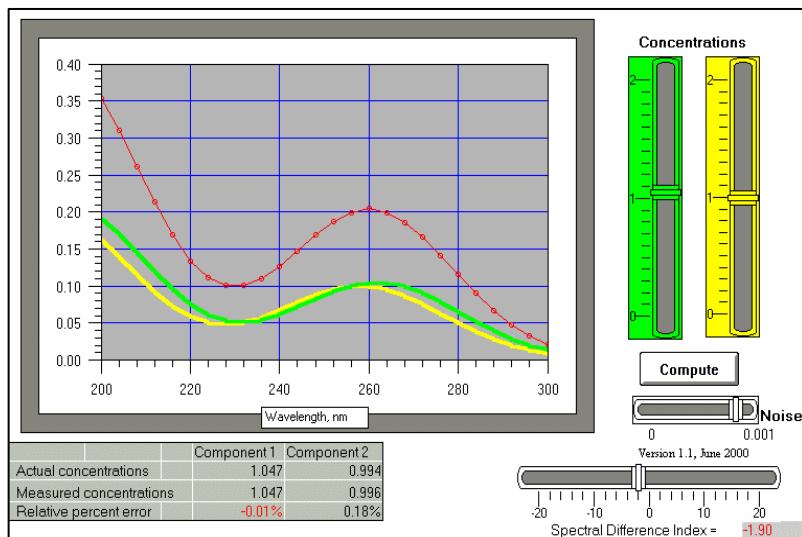
La funzione LINEST. L'indirizzamento indiretto è particolarmente utile quando si utilizzano *funzioni matriciali* come LINEST (pagina 170) o le funzioni di algebra matriciale (pagina 183). Il foglio di calcolo dimostrativo “[IndirectLINEST.xls](#)” ([Link al Grafico](#)) mostra come funziona per l'analisi spettroscopica a lunghezze d'onda multiple di una miscela di tre componenti sovrapposti col Metodo CLS (pagina 179). Lo spettro misurato della miscela è nella colonna C, righe 29-99 e gli spettri dei tre componenti puri sono nelle colonne D, E, e F. La cella C12 “=COUNT(C29:C1032)” conta il numero di righe di dati (cioè il numero di *lunghezze d'onda*) nella colonna C a partire dalla riga 29, e la cella G3 conta il numero di *componenti* (in questo caso 3). Questi vengono utilizzati per determinare la prima e l'ultima riga e la colonna per gli indirizzi indiretti in LINEST nella cella C17. Le altezze dei picchi calcolate da LINEST per i tre picchi sono indicate nella riga 17, colonne C, D ed E, e le deviazioni standard previste sono nella riga sottostante. In questo foglio di calcolo i dati sono simulati (nelle colonne O – U), quindi le vere altezze dei picchi sono note e pertanto *si può calcolare la precisione assoluta* (riga 26, C, D ed E) e confrontata con le deviazioni standard previste. Premere il tasto **F9** per ricalcolare con un campione di rumore indipendente, che equivale a prendere un'altra misura dello stesso campione. Poiché si utilizza l'indirizzamento con INDIRECT, si possono aggiungere o sottrarre punti alla fine delle colonne C – E e i calcoli funzionano senza altre modifiche. Per esempi del suo utilizzo nell'elaborazione del segnale, vedere pagina 184.

Illuminare l'invisibile: Simulazione di strumenti col computer

In questo libro, sono state spesso utilizzate simulazioni al computer per testare, dimostrare e determinare la gamma di applicabilità e l'accuratezza di varie tecniche di elaborazione del segnale. Lo scopo è quello di generare segnali realistici simulati al computer sommando assieme

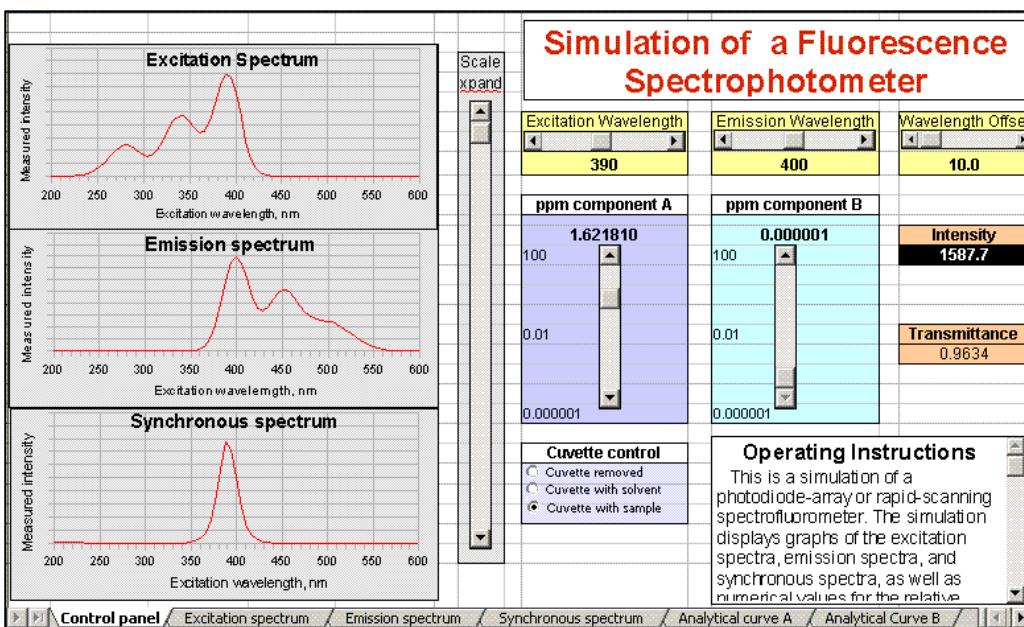
- (a) componente nota del *segnale*, come uno o più picchi, impulsi o fasi sigmoidali,
- (b) una *linea di base*, che può essere piatta, inclinata, curva o a gradini, e
- (c) *rumore casuale*, (pag. 22), che può essere di vari colori (pag. 28) e può dipendere dall'ampiezza (pag. 29).

Questo può essere fatto sia in Matlab/Octave, utilizzando le funzioni native o scaricabili (pagina 451) per vari profili di picco e tipi di rumore casuale, o in fogli di calcolo, che possono anche essere utilizzati per creare interfacce utente attraenti e intuitive. Le “app” di Matlab e i normali spreadsheet hanno un sistema nativo per creare una “GUI” (Graphic User Interface) con pulsanti, cursori, menu a discesa, ecc. Alcuni esempi di fogli di calcolo sono [SimulatedSignal6Gaussian.xlsx](#), [PeakSharpeningDemo.xlsx](#), [PeakDetectionDemo2.xls](#), [TransmissionFittingDemoGaussian.xls](#), [BeersLawCurveFit2.xls](#) e [RegressionDemo.xls](#) (di seguito, a lato).



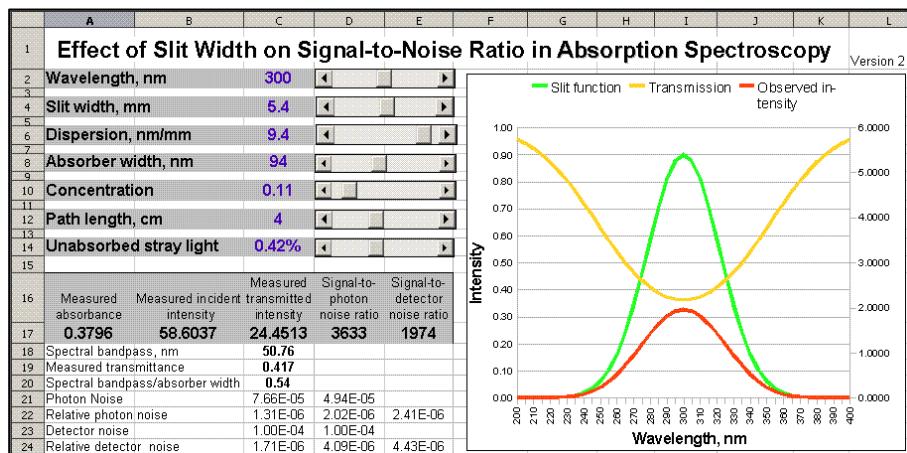
È possibile creare qualsiasi aspetto di un segnale generato dal computer casualmente variabile dalla misura a misura, con l'obiettivo di rendere la simulazione il più vicino possibile al comportamento reale del segnale che potrebbe essere necessario misurare. Ad esempio, nella sezione “Il Torneo: un confronto dei metodi” a pagina 291, il segnale da misurare è un picco Gaussiano situato vicino al centro

del segnale registrato, con una forma e larghezza fisse. La linea di base, però, è molto variabile, sia in ampiezza che come profilo, e si aggiunge anche del rumore bianco. In un'altra simulazione,

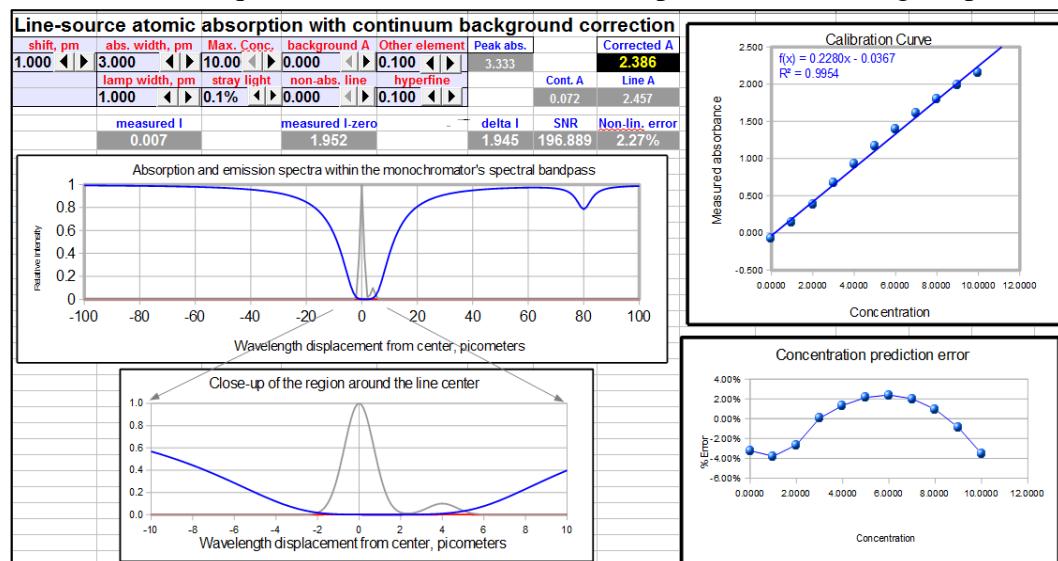


“Perché misurare l’area del picco piuttosto che l’altezza del picco?”, pagina 306, il picco del segnale stesso è soggetto a un processo di ampliamento variabile che fa sì che il picco misurato sia più basso e più largo, ma che non si ha alcun effetto sull’area totale. Nella sezione “La misura di un picco sepolto”, pagina 315, il segnale è un picco “bambino” sepolto sotto la coda di un picco “genitore” molto più forte. In tutti questi casi, il software conosce il *vero segnale in esame*, così che, dopo che il software misura il *segnale osservato* simulato con tutta la sua variabilità della linea di base e del rumore, può calcolare l’errore della misura, consentendo di confrontare i diversi metodi o di ottimizzare le variabili del metodo per ottenere la massima precisione.

In alcuni casi, può essere possibile simulare aspetti importanti di un *intero sistema di strumenti di misura*. Diversi esempi sono mostrati in <https://terpconnect.umd.edu/~toh/models/>. Ciò è particolarmente utile se sia l'ampiezza del segnale che il rumore possono essere previsti dai primi



principi. Ad esempio, nella spettroscopia ottica, i principi della fisica e dell'ottica geometrica possono essere utilizzati per prevedere l'intensità di una [sorgente di luce a incandescenza](#), la [trasmissione di un monocromatore](#) e il segnale generato da un [fotomoltiplicatore](#), compreso il [rumore fotonico](#). Quando questi sono combinati assieme, è possibile simulare gli aspetti



fondamentali di strumenti quali uno [spettrometro a fluorescenza a scansione](#) (sopra) o uno [strumento di assorbimento atomico](#) (sotto), per prevedere le [curve analitiche di calibrazione della spettroscopia di assorbimento](#), per confrontare i rapporti teorici segnale-rumore della misura dell'[assorbimento](#) e della [fluorescenza](#) e per prevedere i limiti di rilevamento della [misura dell'emissione atomica di vari elementi](#), e l'effetto della [larghezza della fenditura sul rapporto segnale/rumore](#) nella spettroscopia di assorbimento (sopra). È inoltre possibile simulare il funzionamento di un [amplificatore lock-in](#) (pag. 312), un sistema di spettroscopia a [modulazione di lunghezza d'onda](#) e persino [circuiti elettronici analogici elementari e amplificatori operazionali](#). Si noti che queste non sono simulazioni di strumenti commerciali che potrebbero essere utilizzati per addestrare gli operatori degli strumenti. Piuttosto, sono modelli matematici manipolati in modo interattivo che descrivono varie parti o aspetti di ciascun sistema, allo scopo di evidenziare aspetti nascosti delle loro operazioni interne.

Chi usa questo libro, il relativo sito web, i documenti e il software?

Negli ultimi anni, questo libro e il sito web associato (<http://terpconnect.umd.edu/~toh/spectrum/>) sono stati consultati dai provider di servizi Internet in oltre **162 paesi** e 6 categorie non specifiche di una regione (ad es. Fornitori di servizi satellitari), inclusi molti paesi in via di sviluppo, alcuni paesi molto piccoli (ad es. Liechtenstein, Isole Faroe), paesi relativamente isolati (Cuba, Corea del Nord, Myanmar/Birmania) e persino alcune regioni devastate dalla guerra (Afghanistan, Siria, Iraq e Ucraina). Il controllo dell'accesso a Internet da parte del governo è spesso un problema. Ad esempio, sono state ricevute meno visualizzazioni da Cuba che molti altri paesi di lingua spagnola con *popolazioni più piccole*, come Bolivia, Repubblica Dominicana, Costa Rica, Porto Rico, Panama e Uruguay, anche se Cuba ha molti scienziati attivi, specialmente in campo medico e farmaceutico.

La prima versione Web è stata pubblicata nel 1996, ma non si è iniziato a tenere traccia delle visualizzazioni delle pagine fino al 2008; da allora ce ne sono state oltre *2 milioni*. La distribuzione del conteggio delle visualizzazioni tra i paesi è molto lunga, con un terzo proveniente dagli Stati Uniti (tranne durante le principali festività degli Stati Uniti), metà delle visualizzazioni proviene da soli 5 paesi (USA, India, Germania, Regno Unito e Cina) e il 99% delle visualizzazioni proviene da soli 39 paesi. Tra i paesi che hanno un numero relativamente elevato di visualizzazioni *rispetto alla popolazione* ci sono Stati Uniti, Germania, Regno Unito, Canada, Australia, Paesi Bassi, Svizzera, Singapore, Israele, Belgio, Taiwan, Corea del Sud e Scandinavia. Un altro mio sito web su un argomento correlato, Interactive Computer Models for Analytical Chemistry Instruction, ha ottenuto altre 820.000 visualizzazioni.

I provider di servizi Internet con il maggior numero di visualizzazioni sono Comcast, Verizon FIOS, Time Warner, Cloudflare, At & t U-verse, Deutsche Telekom (Germania), BSNL (India) e Cox Communication. La maggior parte delle visualizzazioni in tutto il mondo proviene da macchine Windows, circa il 20% da Linux e Macintosh e il 10% da dispositivi mobili. Si è lavorato per rendere le pagine più utilizzabili da dispositivi mobili come gli smartphone.

Circa un quarto delle visualizzazioni proviene *direttamente dagli ISP degli istituti di istruzione* che hanno "School", "Ecole", "College", "Hochschule", "Univ...", "Academic" o "Institute of Technology" nei loro nomi. (Il numero di utenti accademici è certamente maggiore perché alcuni utenti accedono senza dubbio da altri ISP a casa o in azienda). Un'analisi di 200.000 visualizzazioni nel 2015 ha mostrato che i maggiori utenti accademici sono stati il sistema dell'Università della California (UCLA, Berkley, ecc.), Il sistema dell'Indian Institute of Technology, il sistema dell'Università del Texas, il Massachusetts Institute of Technology, l'Università del Michigan, l'Università del Maryland (il mio istituto di origine), la Delft University of Technology (Paesi Bassi), la Stanford University, il China Education and Research Network Center, l'Università del Wisconsin System e l'Università dell'Illinois.

Molti dei grandi laboratori nazionali sono utenti, tra cui Bell Canada, Oak Ridge, Pacific Northwest, Lawrence Livermore, Sandia, Brookhaven, National Renewable Energy Laboratory, SLAC, Fermilab, Lawrence Berkeley, NRC Canada, CERN, NIST, NASA, JPL e NIH.

Le pagine più popolari del sito di recente sono state Ricerca e Misura dei Picchi, Smoothing, Integrazione, Deconvoluzione, InteractivePeakFitter e Signal Processing Tools. Circa il 50% delle visualizzazioni proviene dai motori di ricerca (l'80% di coloro che utilizzano Google). Le parole

chiave di ricerca più comuni utilizzate sono: "peak area", "convolution", "deconvolution", "peak detection", "signal processing pdf", "findpeaks matlab", "Fourier filter" e "smoothing". Circa il 40% del traffico proviene da link diretti (segnalibri o URL digitati) e circa il 10% proviene da siti Web di riferimento, di solito da [Wikipedia](#) o da [MathWorks](#). Sfortunatamente, i caricamenti delle pagine e i termini di ricerca sono diventati quasi completamente crittografati negli ultimi anni, quindi non si possono più dire quali pagine vengono visualizzate e quali scaricate. (È interessante notare che questo non è il caso di [Interactive Computer Models for Analytical Chemistry Instruction](#), che ha solo il 75% di crittografia).

Sono stati effettuati oltre 100.000 download del mio software e dei file di documentazione, attualmente in media diverse centinaia di download di file al mese, sia dal [mio sito web](#) che dai miei file su [Matlab File Exchange](#). I file più comunemente scaricati sono [IntroToSignalProcessing.pdf](#), [PeakFinder.zip](#), [ipf12](#), [CurveFitter...xlsx](#), [iSignal](#), [ipeak](#), [PeakDetection.xlsx](#) e l'archivio completo del sito [SPECTRUM.zip](#).

Quali fattori influenzano il numero di visualizzazioni dai diversi paesi? Gli strumenti di analisi dei dati, in particolare la regressione (ad esempio, l'utilizzo di LINEST), possono aiutare a rispondere a questa domanda. Ovviamente, ci si aspetterebbe che la popolazione di un paese sia un fattore, ma risulta che *la correlazione tra log(visualizzazioni) e log(popolazione) è molto scarsa*, con un coefficiente di determinazione (coefficiente di correlazione log-log o valore R^2) di soli 0.36 ($n=163$ paesi; oltre 160,000 caricamenti di pagine totali nel periodo dal 2008 al 2017; [grafico su pagina successiva](#)). Si noti che, a causa della gamma molto ampia di dimensioni della popolazione, è stata eseguita una correlazione *log-log* (pagina 441) per evitare che i risultati fossero totalmente dominati da i primi paesi.

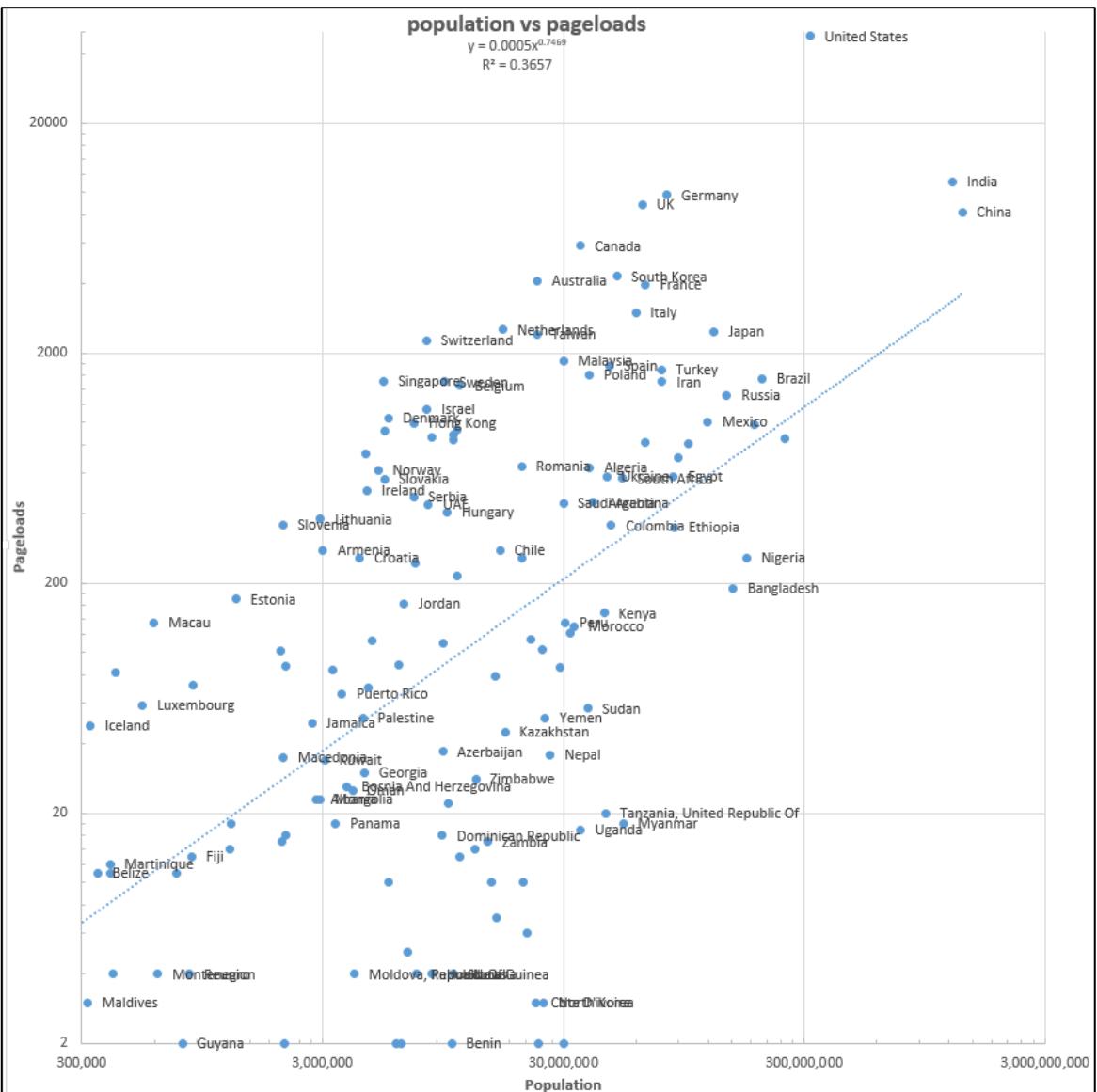
Ho anche studiato l'effetto di altri fattori che potrebbero essere più specifici per la lingua e l'argomento del mio sito particolare, incluso

- il numero di *persone che parlano inglese* in ciascun paese,
- il numero di *utenti Internet* in ciascun paese,
- il numero di *università* in ciascun paese, e
- il *budget totale di ricerca e sviluppo* di ciascun paese.

Tutte queste informazioni sono disponibili gratuitamente su Internet per *la maggior parte* (ma non tutti) dei paesi ([link al grafico](#)). Con un buon margine, *il fattore più influente è stato il budget di ricerca e sviluppo*, per il quale il valore di R^2 era 0.76. Questo forse non sorprende visto che il sito riguarda un argomento molto ristretto e specializzato: gli aspetti tecnici del trattamento dei dati scientifici computerizzati.

Un log-log di *regressione multilineare* su tutti e 5 questi fattori ha prodotto un valore R^2 di 0.84 ($n=53$ paesi per i quali sono stati riportati *tutti e 5* fattori), che rappresenta un modesto miglioramento rispetto al solo budget di ricerca e sviluppo. (Da quando questi calcoli sono stati effettuati nel 2017, le visualizzazioni di pagina dalla Cina sono aumentate notevolmente e ora sono generalmente seconde a quelle degli Stati Uniti).

Per un foglio di calcolo Excel con tutti questi dati e calcoli (tra il 2008 e il 2015), vedere [FinalCountriesSummary.xlsx](#)

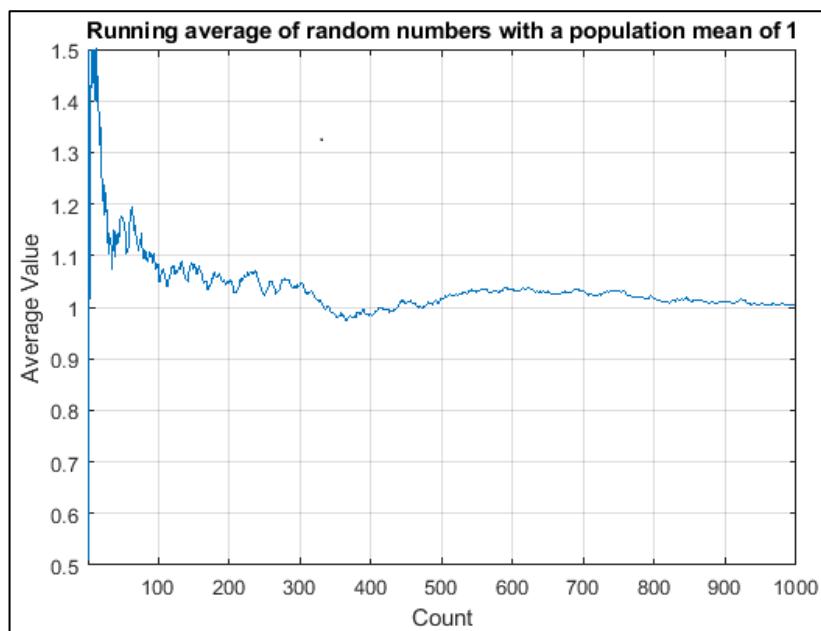


Satellite Provider	n	Log-log correlations to hits	n	R2
Asia/pacific Region	105	population	163	0.4
Anonymous Proxy	64	English speakers	92	0.579
Europe	955	Number of universities	146	0.6814
Non-country hits	1130	Internet users	145	0.6812
Country hits	159297	R&D spending	72	0.75
	1%	Patent applications, 2008-2012	133	0.591

Quali campi di studio sono rappresentati? Gli utenti del sito includono studenti, istruttori, lavoratori e ricercatori nei settori dell'industria, dell'ambiente, della medicina, dell'ingegneria, delle scienze della terra, dello spazio, militare, finanziario, agricolo, delle comunicazioni e persino della musica e linguistica. Questa conclusione si basa sulle e-mail che ho ricevuto, sui [titoli di articoli di riviste che hanno citato il mio lavoro](#), e sugli ISP dei principali visitatori del web. A giudicare dal rapporto tra download ed e-mail, la maggior parte delle persone che hanno scaricato il software non scrivono riguardo a ciò che stanno facendo, il che ovviamente è del tutto comprensibile. Inoltre, delle persone che scrivono, la maggior parte non dice nello specifico quali siano le loro applicazioni, che è una loro prerogativa. Di conseguenza, si hanno informazioni incomplete sulle aree applicative in cui vengono usati i programmi. Un'indicazione molto migliore sul numero di applicazioni può essere ottenuta esaminando i titoli dei quasi 700 [documenti e brevetti pubblicati](#) che hanno citato queste pagine web e il libro (pagina 492).

La Legge dei Grandi Numeri

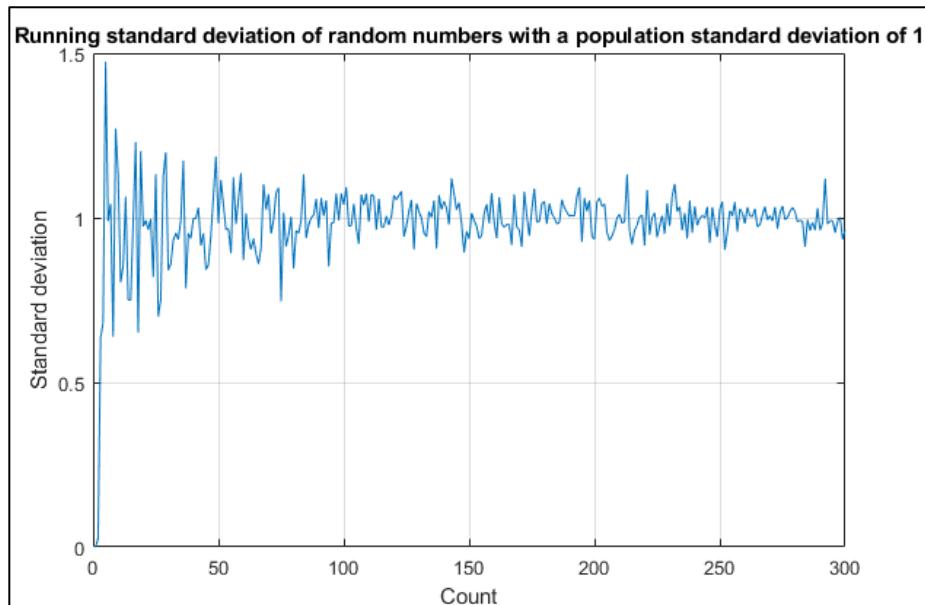
[La Legge dei Grandi Numeri](#) è un teorema che descrive grandi raccolte di numeri e osservazioni che sono soggette a variazioni casuali indipendenti e identicamente distribuite in modo casuale, come il risultato dell'esecuzione dello stesso esperimento o delle misure un gran numero di volte. La media dei risultati ottenuti da un gran numero di prove dovrebbe essere vicina al valore a lungo termine (chiamato "media della popolazione") e tenderà ad avvicinarsi man mano che vengono eseguite più prove. È un'idea importante perché garantisce risultati stabili a lungo termine per le medie di alcuni eventi casuali. Questo è il motivo per cui i casinò sono in grado di fare tanti soldi; i loro giochi sono progettati per dare al casinò un significativo vantaggio a lungo termine ma risultati altamente variabili a breve termine, garantendo molti vincitori (rumorosi) che tendono ad incoraggiare gli altri giocatori, ma anche abbastanza altri perdenti (silenziosi) da poter fare soldi. Ed è per questo che gli investitori nel mercato azionario guadagnano a lungo termine, nonostante l'imprevedibile variazione giornaliera, un giorno in rialzo e in ribasso il giorno successivo. Questo è anche il motivo per cui è così difficile vedere il cambiamento del *clima* nelle variazioni giornaliere o annuali a breve termine tra caldo e freddo del *meteo*.



viene calcolata la media di sempre più numeri di quella popolazione, fino a 1.000. (Questo è generato dal semplice script Matlab [RunningAverage.m](#)). Si noti che la media si aggira, raggiungendo e superando la media reale della popolazione (1.000) due volte in questo caso prima di finire vicino a 1.0 dopo che sono stati accumulati 1000 punti. Ma eseguendo di nuovo questo script, la media finale potrebbe *non* essere così vicina a 1.0. Infatti, la deviazione standard prevista

della media di *tutti* i 1000 numeri casuali è ridotta di un fattore di $1/\sqrt{1000}$, che è circa 0.031, ovvero il 3% relativo, il che significa che la maggior parte dei risultati cadrà solo [entro più o meno 6% della media reale](#), cioè da 0,94 a 1,06.

L'incertezza dell'incertezza. La



situazione è ancora più variabile se si desidera stimare la *deviazione standard* di una popolazione da piccoli campioni. Il grafico a lato mostra lo script Matlab [RunningStandardDeviation.m](#), che esegue tale simulazione per la stessa popolazione dell'esempio precedente. La deviazione standard del campione varia in modo allarmante per piccoli campioni e si assesta solo lentamente. Ancora peggio, la deviazione standard per campioni molto piccoli è [distorta verso il basso](#), spesso restituendo valori molto più bassi del solito. (Ad esempio, la deviazione standard calcolata di due punti dati ricavati da una distribuzione normale è quasi sempre inferiore alla deviazione standard effettiva della popolazione).

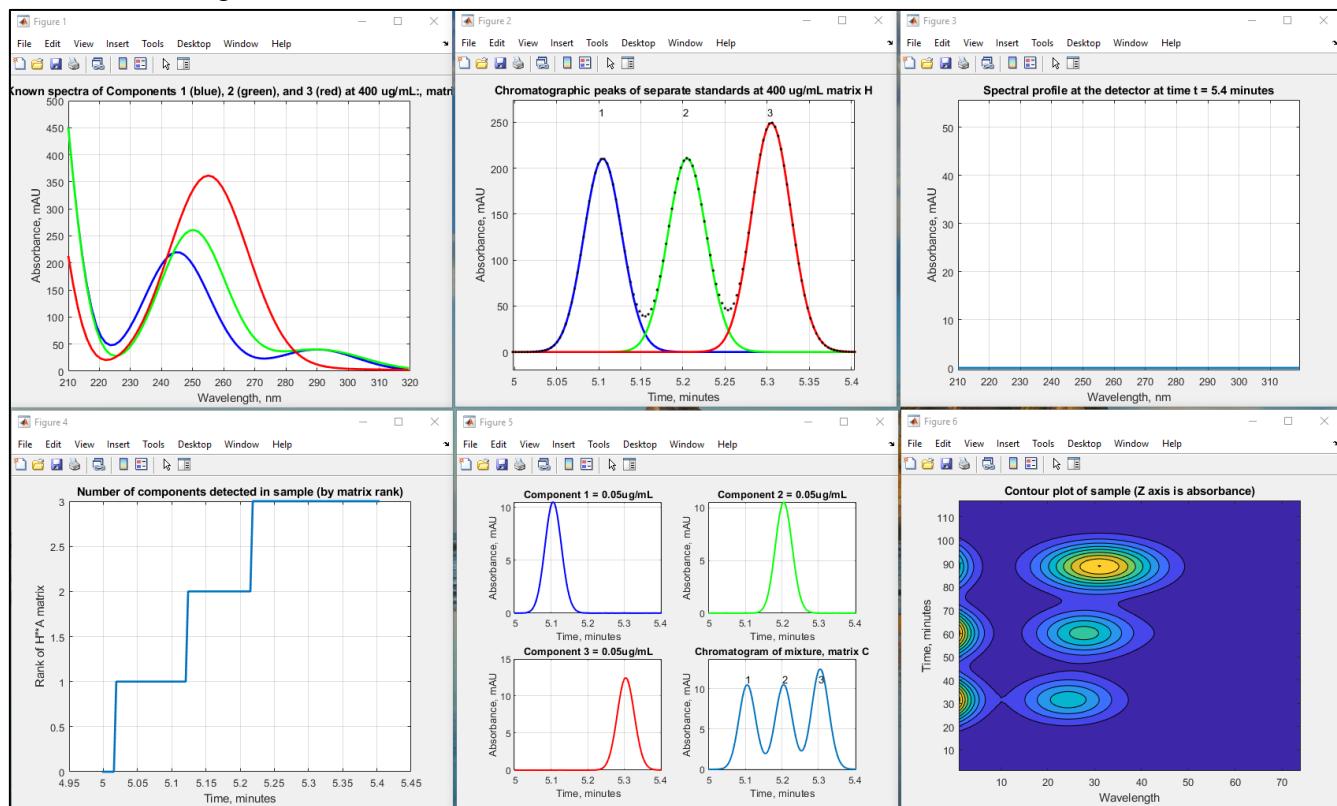
Esiste una tendenza ben documentata da parte delle persone a *sovrastimare* la qualità di un piccolo numero di misure, a volte indicata come [generalizzazione frettolosa](#), o [insensibilità alla taglia del campione](#), o l'[errore del giocatore d'azzardo](#). Questo è legato al campo di studio di una famosa coppia di psicologi di nome Amos Tversky e Daniel Kahneman, che hanno collaborato a uno studio di lunga durata sui pregiudizi cognitivi umani negli anni '70. Hanno formulato un'ipotesi che le persone tendano erroneamente a credere in una falsa "[Legge dei Piccoli Numeri](#)", il nome che hanno coniato per l'errata convinzione che un piccolo campione tratto da una vasta popolazione sia rappresentativo di quella grande popolazione. Ci piacerebbe credere che gli scienziati siano immuni a queste debolezze e che pensino sempre in modo logico e corretto. Ma gli scienziati sono solo umani, quindi vale la pena essere consapevoli di questa tendenza, *in particolare quando un piccolo campione di dati supporta l'ipotesi preferita*. Si è tentati di fermarsi qui, "finché si è avanti". Questo si chiama "[Bias di conferma](#)". Da evitare come la peste.

Naturalmente, in molte misure sperimentali pratiche, si potrebbe essere davvero vincolati a un piccolo numero di misure ripetute. Potrebbe esserci un numero fisso di punti e nessuna possibilità di raccoglierne altri. Oppure il costo, in denaro o in tempo, per raccogliere più dati può essere eccessivo. Ad esempio, il processo di calibrazione di uno strumento analitico per la misura quantitativa (pagina 329) può comportare la preparazione e la misura di diversi campioni standard o soluzioni di composizione nota. Se la curva di calibrazione (la relazione tra la lettura dello strumento e la composizione del campione) non è lineare, sono necessari diversi standard per definire la curva, più sono e meglio è. Ma è necessario considerare non solo il costo della preparazione di molti standard, ma anche il costo della pulizia e dello stoccaggio o dello smaltimento in sicurezza delle sostanze chimiche (potenzialmente pericolose) in seguito. In altre parole, potrebbe essere necessario accettare un numero inferiore all'ideale degli standard. La conclusione è, se si è limitati a un piccolo numero di dati, non sopravvalutare la qualità dei risultati. Per determinare gli intervalli di incertezza per un insieme di dati utilizzando la [regola 3-sigma](#) (pagina 31), la distribuzione deve essere normale (Gaussiana) e dev'essere nota la deviazione standard. Per piccoli insiemi di dati, sono *entrambi* incerti.

Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo

L'introduzione dei veloci [array di rilevatori UV-Visibile](#) negli strumenti di [cromatografia liquida ad alte prestazioni](#) (HPLC) ha notevolmente aumentato la potenza di questo metodo. La velocità di tali rivelatori consente loro di acquisire uno spettro completo UV-Visibile più volte al secondo sull'intero cromatogramma. In questa sezione si useranno i dati in un rapporto tecnico di Shimadzu Scientific Instruments (<https://solutions.shimadzu.co.jp/an/n/en/hplc/jpl217011.pdf>) che descrivono la separazione di tre isomeri posizionali del metil acetofenone: o-metil (o-MAP), m-metil (m-MAP) e p-metil (p-MAP) mediante cromatografia liquida con un rilevatore UV a serie di diodi. Gli spettri di assorbimento ultravioletti di ciascuno di questi tre isomeri a una concentrazione di 400 µg/mL sono mostrati sotto a sinistra; sono distinti ma fortemente sovrapposti. Al centro è mostrata la separazio-

ne cromatografica, utilizzando la colonna e le condizioni specificate nel loro rapporto; i picchi sono solo parzialmente risolti. Il rapporto Shimadazu descrive il loro software commerciale proprietario, che utilizza un approccio iterativo complesso per estrarre gli spettri e le caratteristiche cromatografiche dai dati originali.



Qui viene presentata una tecnica non iterativa più semplice basata sullo stesso sistema chimico, in cui consideriamo ogni spettro acquisito dal rivelatore come una miscela di diversi campioni e applichiamo il metodo dei minimi quadrati classici [presentato in precedenza \(pag. 179\)](#), in cui gli *spettri delle componenti sono note in anticipo* e dove è prevista l'aderenza alla legge di Beer-Lambert. Gli spettri e i picchi cromatografici vengono simulati digitalmente nello script Matlab/Octave [TimeResolvedCLS.m](#), mostrato nella figura seguente, modellando lo spettro di ciascun componente come somma di tre picchi Gaussiani e picchi cromatografici come Gaussiane modificate esponenzialmente. Questa è una “simulazione *basata sui dati*”: i parametri sono stati accuratamente adattati per corrispondere il più fedelmente possibile ai grafici della relazione tecnica, per rendere questa simulazione la più realistica possibile. Anche gli altri parametri, come la risoluzione spettrale, la frequenza di campionamento e il rumore del rivelatore (2 unità di milli-assorbanza, mAU), erano basati su quel rapporto. Si noti che i picchi cromatografici (figura centrale) non sono risolti sulla linea di base. Pertanto è prevedibile che la calibrazione quantitativa basata sulla misura delle aree dei picchi in questo cromatogramma (ad esempio con il metodo del taglio verticale, pagina 140, possa essere imprecisa, specialmente se le altezze dei picchi sono molto diverse. In effetti, in questo caso, anche se le concentrazioni delle tre componenti sono molto inferiori ($0.05 \mu\text{g/mL}$ ciascuna), le aree misurate per taglio perpendicolare sono abbastanza precise, differendo solo del 2% dai valori reali, principalmente a causa della leggera asimmetria e altezze quasi uguali delle tre cime. Gli spettri (figura a sinistra) sono ancora più sovrapposti rispetto ai picchi cromatografici, ma sono *distinti nella forma*, e questa è la chiave.

Fondamentalmente, lo trattiamo come una serie di calcoli CLS a 3 componenti, uno per ogni intervallo di tempo del rivelatore. I calcoli effettivi possono essere eseguiti in due modi, a seconda che gli spettri vengano elaborati uno per uno ("Alternative calculation #1", righe 113-146) o raccolti per l'intero cromatogramma e quindi elaborati tutti in una volta ("Alternative calculation #2", righe

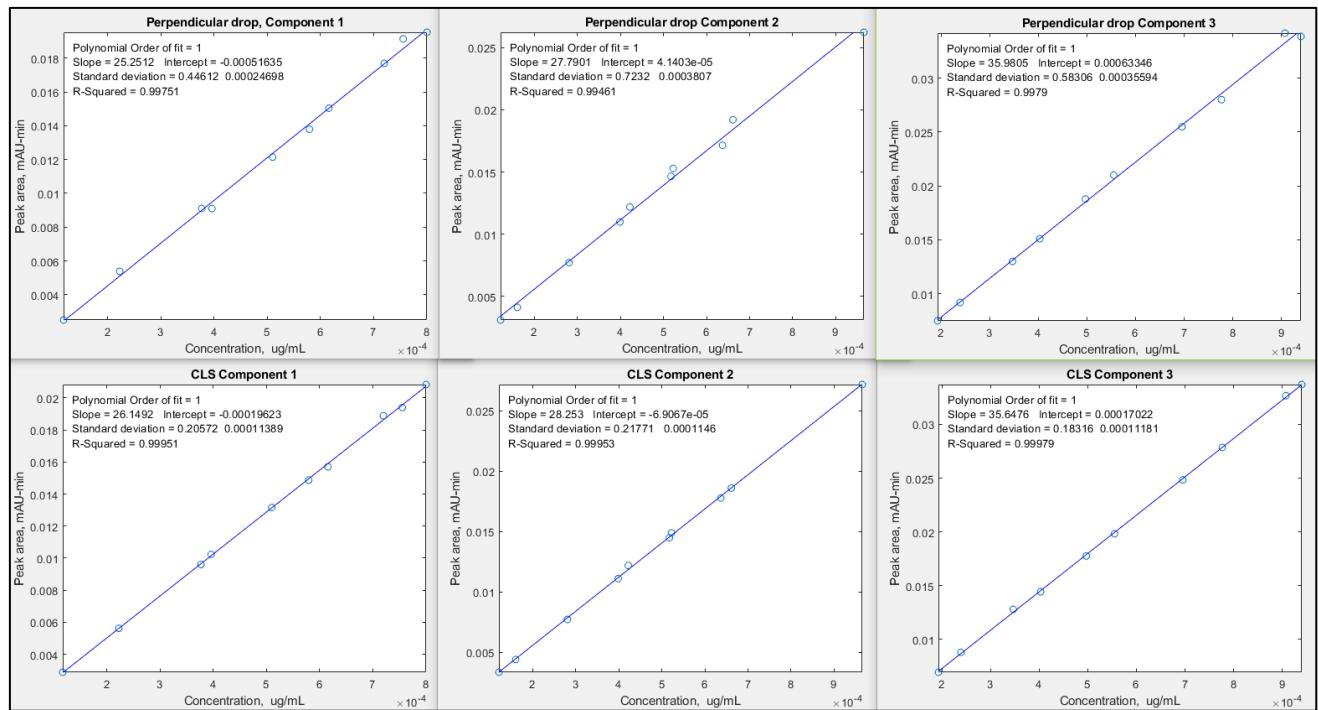
150-170). Il primo metodo assomiglia alla cromatografia durante l'esecuzione dello script; calcola punto per punto i picchi cromatografici delle tre componenti man mano che si evolvono nel tempo e li traccia nei primi tre quadranti della finestra di figura 3 (a destra). Il secondo metodo calcola l'intero chromatogramma in un unico passaggio alla fine e crea gli stessi grafici finali. (Il secondo metodo è più veloce dal punto di vista computazionale, ma non è significativo perché è la *cromatografia* che determina la velocità, non i *calcoli*). In ogni caso, il risultato è lo stesso; i picchi cromatografici delle tre componenti sono *completamente separati matematicamente*, quindi le loro aree sono facilmente misurabili, *indipendentemente da quanto si sovrappongono chromatograficamente*. Si noti che, sebbene i tre spettri debbano essere noti, non è richiesta alcuna conoscenza dei picchi cromatografici; emergono separati e intatti dai dati, computazionalmente.

Stress test. Il vantaggio di una simulazione "data-based" come questa è che si può testare come il metodo potrebbe funzionare in applicazioni più impegnative modificando i parametri di simulazione del segnale. Per testare le capacità e i limiti di questo metodo, è stata preparata una serie di scenari sempre più impegnativi, partendo dal sistema sperimentale originale illustrato sopra e poi rendendolo *progressivamente più difficile* modificando i parametri della simulazione. Nella tabella seguente sono elencati sei scenari, oltre ai tipici errori percentuali nella misura dell'area col metodo CLS e con collegamenti al grafico corrispondente e agli m-file Matlab/Octave. Ciascuna è una variante più impegnativa dell'analisi originale dell'analisi del metil-acetofenone; in #2 ha una sovrapposizione di picchi cromatografici molto maggiore; il #3 ha picchi più asimmetrici (un *tau* molto più alto); il #4 ha spettri molto più simili - infatti, le lunghezze d'onda dei picchi differiscono di soli 0,1 nm, facendole apparire identiche; nel #5, la componente 2 (il picco medio) ha una concentrazione 100 volte inferiore; e nel #6 è uguale al #5 tranne per il fatto che i picchi sono molto asimmetrici. In tutti questi casi, la normale tecnica di misura dell'area col taglio perpendicolare è impossibile (perché non ci sono picchi distinti per ogni componente) o risulta molto errata, ma la tecnica CLS funziona bene, dando errori molto bassi, tranne quando il picco medio della concentrazione è di 0.0001 ug/mL, che si avvicina al limite del rumore casuale del rivelatore, 2 mA.U. (Un'altra variazione nello script [TimeResolvedCLSbaseline.m](#), comprende la [correzione continua della deriva della linea di base](#) aggiungendo una 4^a componente spettrale piatta per tener conto delle possibili bolle o impurità lungo il percorso della luce verso il rilevatore ottico).

Risoluzione del picco	Similarità spettrale	Asimmetria del picco	Concentrazione ug/mL	Errori percentuali tipici nella misura dell'area	Link
1. Normale	Normale	Leggero: <i>tau</i> =10	.05 .05 .05	.002% .002% .0016%	Grafico mfile
2. Non risolto	Normale	Leggero: <i>tau</i> =10	.01 .01 .01	-.06% -.053% -.041%	Grafico mfile
3. Miscelato	Normale	Grande: <i>tau</i> =40	.05 .05 .05	-.0004% -.013% -.066%	Grafico mfile
4. Non risolto	Molto prossimo	Leggero: <i>tau</i> =10	.01 .01 .01	.054% .049% .04%	Grafico mfile
5. Non risolto	Molto prossimo	Leggero: <i>tau</i> =10	.01 .0001 .01	0.026% 2.4% 0.019%	Grafico mfile
6. Non risolto	Molto prossimo	Grande: <i>tau</i> =40	.01 .0001 .01	-0.04% -3.8% -0.03%	Grafico mfile

Anche quando i picchi sono abbastanza distinti da far funzionare il metodo del taglio verticale, può soffrire di un'interazione tra le altezze dei picchi adiacenti; cioè una variazione dell'altezza di un

picco può influire sulla misura dell'area dei picchi sovrapposti adiacenti, a causa delle variazioni nel punto di avvallamento tra di loro. Ciò è illustrato da [TimeResolvedCLScalibration.m](#), che simula le curve di calibrazione (concentrazione rispetto all'area del picco) per una miscela a tre componenti simile alla precedente ma modificata in modo che ci sia sempre un avvallamento tra i picchi, e quindi consente alle tre componenti di variare in modo indipendente e casuale in un intervallo da 2×10^{-4} a $9 \times 10^{-4} \mu\text{g/mL}$. (Ogni volta che lo si esegue, si otterrà un diverso mix di concentrazioni). Di seguito è mostrato un tipico insieme di curve di calibrazione. In questo caso, l'errore percentuale medio assoluto nella misura dell'area è di circa il 5% per il metodo del taglio verticale, con un R^2 di 0.995, ma è *inferiore all' 1% per la misura CLS*, con un R^2 di 0.9995, un grande miglioramento.



Il metodo CLS è chiaramente molto efficace, ma tutto ciò dimostra solo che la *matematica* funziona bene; il metodo richiede ancora che gli spettri di tutti i componenti siano accuratamente noti. Questo requisito può essere soddisfatto in alcune applicazioni, ma nella cromatografia liquida esiste una potenziale insidia. Se si utilizza l'eluizione in gradiente e/o la programmazione della temperatura e se gli spettri di quei composti chimici sono sensibili al solvente e/o alla temperatura, eventualmente spostando leggermente i loro picchi, allora ci saranno probabilmente ulteriori errori nella procedura CLS. Ovviamente questo dipende dal sistema chimico e dovrà essere valutato caso per caso.

In altre applicazioni, alcuni o tutti i componenti potrebbero essere semplicemente sconosciuti e si potrebbe voler ottenere i loro spettri. Questo può essere fatto *in situ* se la separazione dei picchi è buona almeno quanto quella illustrata a pagina 354, perché in ogni massimo di picco non c'è virtualmente alcun contributo dai picchi adiacenti.

Ma questo suggerisce un altro uso interessante di questo metodo. Se si dispone di un metodo cromatografico che ottiene la separazione della linea di base, è possibile utilizzarlo per ottenere spettri accurati di ciascuno in situ. Poi è possibile modificare le condizioni per ottenere un cromatogramma *più veloce*, ad esempio utilizzando un'altra lunghezza della colonna e/o una velocità di flusso. Anche se ciò comporta cromatogrammi non completamente risolti, è possibile applicare questo metodo CLS per ottenere risultati accurati in un tempo molto più breve per più campioni.

Ma cosa succede se i picchi sono più sovrapposti ancora, in modo tale che gli spettri dei componenti

puri non vengano mai ottenuti? In tal caso, devono essere utilizzati metodi più sofisticati, come quello descritto nel rapporto tecnico Shimadzu. Ciò comporta la realizzazione di stime iniziali dei picchi spettrali e cromatografici, seguita da una [ricerca iterativa](#) per l'approssimazione migliore ai dati sperimentali, soggetta all'imposizione di alcuni importanti vincoli già noti, come la non negatività degli spettri e dei picchi cromatografici (tali picchi sono sempre positivi, tranne che per il rumore casuale sulla linea di base), e l'[unimodalità](#) dei picchi cromatografici (ovvero, ogni componente fornisce uno e un solo picco cromatografico). Metodi di questo tipo verranno lasciati a una futura espansione di questo libro.

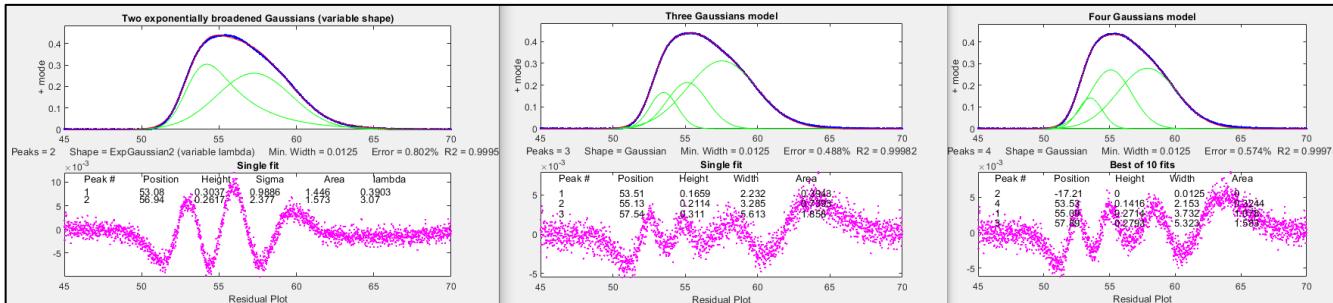
La sfida del picco misterioso

L'obiettivo di questo esercizio è apprendere il più possibile sulle proprietà di un segnale digitalizzato utilizzando gli strumenti di elaborazione del segnale in questo libro e, se possibile, ottenere una descrizione matematica del segnale ([script](#)). A prima vista, il segnale ([MysteryPeak.mat](#)) sembra essere un singolo picco asimmetrico con un massimo a $x=55.5$. Il rapporto segnale/rumore sembra essere molto buono - c'è poco rumore visibile a meno che non si guardi molto da vicino - e il segnale inizia e finisce vicino allo zero, quindi la correzione della linea di base probabilmente non è un problema. La cattiva notizia è che non sappiamo nient'altro. L'asimmetria potrebbe essere dovuta a un processo asimmetrico applicato a un profilo originariamente simmetrico, ma potrebbe essere un gruppo di picchi sovrapposti ravvicinati, il che è suggerito dalle deboli protuberanze nella forma. Alcune rapidi approssimazioni preliminari della curva possono essere eseguite utilizzando la riga di comando [peakfit.m](#) (pagina 384):

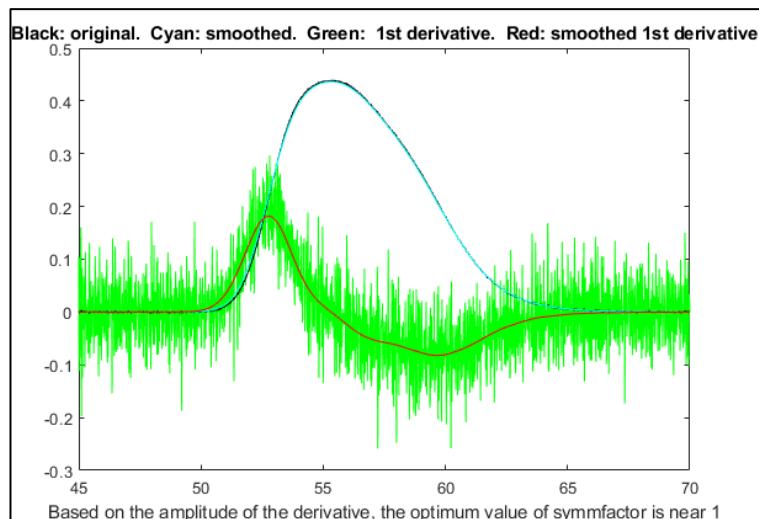
`[FitResults, GOF] = peakfit([x y], 0, 0, 2, 1)` per un modello con due Gaussiane (profilo 1)

`[FitResults, GOF] = peakfit([x y], 0, 0, 1, 3)` per un modello Logistico singolo (profilo 3)

`[FitResults, GOF] = peakfit([x y], 0, 0, 4, 39, 1)` per un modello Gaussiano ampliato in modo esponenziale 4 (profilo 39)

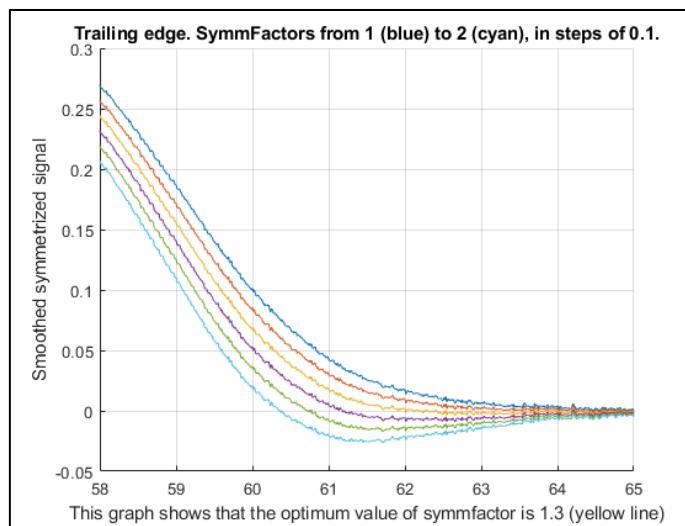


Oppure si potrebbe usare a questo scopo il peak fitter interattivo [ipf.m](#) (pagina 405); è possibile modificare rapidamente il profilo del modello, il numero di picchi, le ipotesi iniziali, l'area dati da approssimare, ecc., con singole sequenze di tasti e clic del mouse. In ogni caso, le tre approssimazioni iniziali nelle figure mostrano che il segnale contiene una piccola quantità di rumore casuale, che sembra essere bianco (quindi il segnale probabilmente non è stato smussato, il che è una fortuna) e che ha una deviazione standard relativa di circa 0.2%, in base a $1/5^{\circ}$ del valore visivo picco-picco (pag. 22). Ma sfortunatamente, questi accoppiamenti non hanno successo perché i loro errori di approssimazione (da 0,5 a 0,8%) sono tutti significativamente maggiori del rumore casuale dello 0,2%! Anche provare forme diverse e un numero maggiore di picchi non aiuta, poiché si traduce in errori di approssimazione più



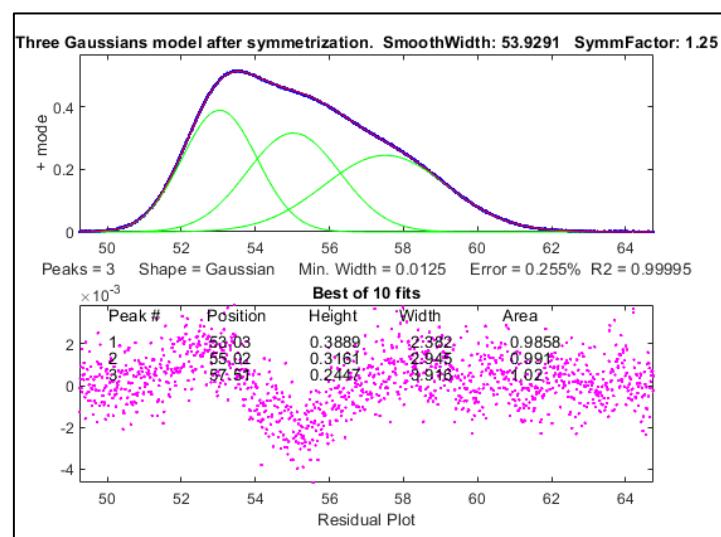
elevati, accoppiamenti instabili o altezze pari a zero; c'è troppa sovrapposizione per l'approssimazione della curva (pagina 218).

Un altro approccio al problema dei picchi asimmetrici consiste nell'utilizzare la tecnica di *simmetrizzazione della derivata prima* descritta a pagina 77. Ciò si applica specificamente all'ampliamento *esponenziale*, un comune meccanismo di allargamento dei picchi. L'idea è che se si calcola la derivata prima di un picco ampliato esponenzialmente, la si moltiplica per un fattore di ponderazione uguale alla costante di tempo *tau* dell'esponenziale, e la si somma al segnale ampliato originale, il risultato sarà il picco originale prima dell'ampliamento, il che *rende la sovrapposizione del picco meno grave*. Questo vale per [qualsiasi profilo di picco originale](#). Anche se non si conosce il *tau* in anticipo, si possono provare diversi valori fino a quando la linea di base dopo il picco diventa la più bassa possibile ma non negativa, come mostrato in [questa animazione GIF](#). Questo può essere fatto facilmente usando la funzione [symmetrize.m](#), o interattivamente in [iSignal](#), che ha lo smoothing (tasto **S**), le derivate (**D**), la simmetrizzazione (**Shift-Y**) e il 'curve fitting' (**Shift-F**). La derivata di *y* rispetto a *x*, dalla funzione "[derivxy.m](#)", mostrata dalla linea verde nella figura sopra, è piuttosto rumorosa. Come al solito, dobbiamo smussare [smoothing] le derivate dei segnali rumorosi per renderli utili, ma non eccessivamente per non distorcere i segnali. Come regola generale, una larghezza di smoothing uguale a $1/10^6$ del numero di punti nella semi-larghezza non distorce visibilmente il segnale, come mostrato a pagina 56. Il nostro picco ha circa 530 punti, misurati da [hal-](#)



[fwidth\(1:length\(y\),y\)](#), quindi una larghezza di smoothing prossima a 53 non distorce il picco del segnale, ma elimina la maggior parte il rumore dalla derivata (sopra a destra). Inoltre, possiamo vedere che la derivata, in unità *y/x*, è paragonabile come grandezza numerica al segnale originale, quindi la costante di tempo *tau* (in unità *x*) è probabilmente vicino a 1.0. Successivamente, si aggiunge il prodotto della derivata prima e *tau* al segnale originale, tenendo d'occhio il bordo di uscita mentre si provano sei diversi valori di *tau* vicino a 1.0. Il grafico in alto a sinistra mostra che il valore ottimale è circa 1.25.

Quando viene applicato all'intero segnale, il risultato, mostrato a destra, ha *protuberanze più distinte*. Quando quel segnale modificato viene utilizzato per l'approssimazione della curva, si trova che un modello di 3-Gaussiane funziona abbastanza bene, con un errore di approssimazione di solo lo 0,25%, prossimo al rumore. Questa è la prova che il segnale è composto da tre Gaussiane ravvicinate modificate esponenzialmente (EMG). Normalmente non esiste un modo indipendente per verificare l'accuratezza dei parametri del picco misurati così, ma - full disclosure - *il segnale nel caso non era effettivamente ignoto* ma piuttosto è stato generato dallo script [MysteryPeaks.m](#); in ef-

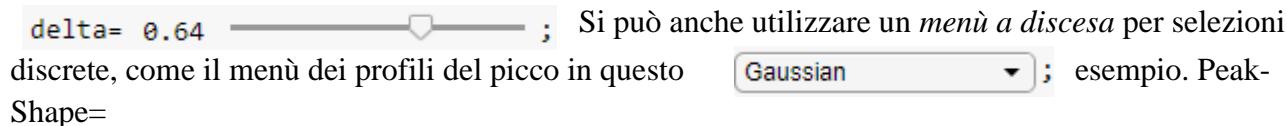


fetti è costituito da tre EMG, con massimi di picco a $x=53$, 55 e $57,5$, ciascuno con una costante di tempo di $1,3$ e tutti con aree di picco di $1,0$. I risultati dell'approssimazione *dopo* la simmetrizzazione sono entro lo 0.1% per le posizioni e entro il 2% delle aree. Al contrario, l'approssimazione diretta delle Gaussiane esponenziali ai dati *originali*, utilizzando il *tau* appena determinato, sembra buona ([grafico](#)) ma fornisce *parametri meno accurati* e impiega tre volte più tempo per il calcolo.

Sviluppo di Script Live Matlab e App

Il solito modo di sviluppare programmi in Matlab consiste nello scrivere script o funzioni (vedere pag. 33), sviluppati e utilizzati in un ambiente visuale simile a quanto [mostrato in precedenza](#). In questi ambienti si possono modificare i valori dei parametri digitandoli nella finestra dell'editor o sulla riga di comando. Se ci sono molti parametri e selezioni da esplorare, il ciclo di modifica e ri-esecuzione può essere lento e macchinoso. Ad esempio, "[DenomAdditionDemo.m](#)" ha nove variabili interattive regolabili. Il suo scopo è quello di mostrare la deconvoluzione di Fourier (pagina 107) di una coppia di picchi sovrapposti, con l'obiettivo di aumentare la risoluzione dei picchi. La sequenza di calcoli è (1) creare il segnale simulato con due picchi e rumore casuale, (2) creare una funzione di convoluzione centrata sullo zero della stessa forma e con larghezza variabile, (3) dividere la trasformata di Fourier del segnale simulato per la trasformata di Fourier modificata di una funzione di convoluzione calcolata della stessa forma, (4) aggiungere una costante al numeratore di quella divisione per ridurre gli spike di rumore (pagina 116), (5) calcolare la trasformata inversa risultante e (6) applicare il filtro passa-basso per ridurre il rumore. Per esplorare l'abilità di questo metodo, è possibile regolare diversi fattori: la forma, l'ampiezza, l'intervallo di campionamento e la separazione dei due picchi, l'ampiezza del rumore, l'ampiezza della funzione di deconvoluzione, l'aggiunta del denominatore e la frequenza di taglio del filtro. Per farlo, si dovrebbe modificare lo script, salvarlo ed eseguirlo nuovamente, potenzialmente molte volte. In questa appendice svilupperemo due diversi modi alternativi per codificare questa demo di deconvoluzione con un'interfaccia utente grafica (GUI) più moderna che utilizza cursori, menù a discesa, caselle di controllo, ecc., per controllare le variabili regolabili, anziché modificare lo script stesso.

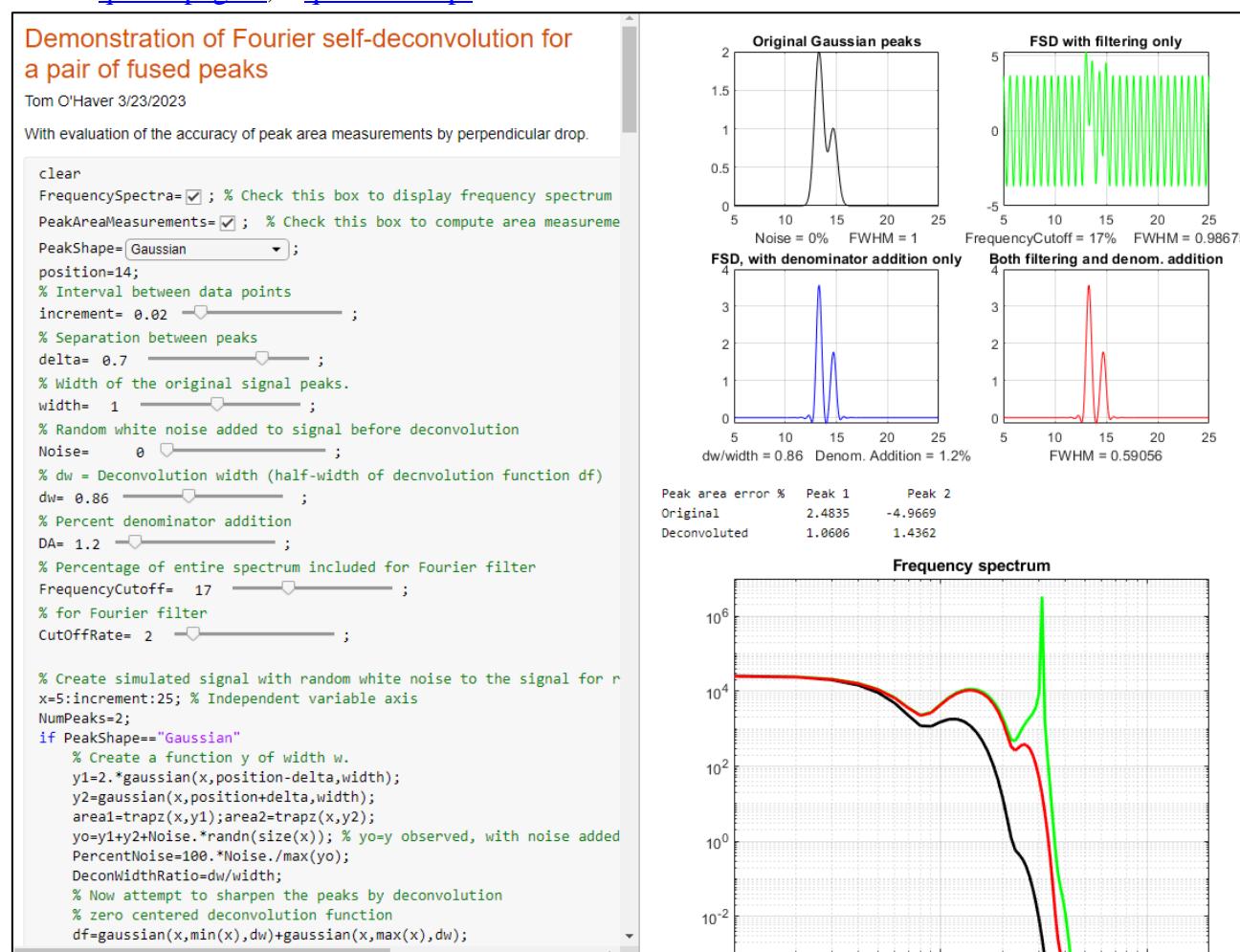
I Live Script di Matlab. Un'alternativa molto semplice è quella di utilizzare [Live Script](#) (disponibili a partire da MATLAB R2016b). I Live Script sono documenti che combinano codice, output, controlli interattivi e testo formattato in un unico ambiente. Per convertire il semplice script [DenomAdditionDemo.m](#) in un Live Script, cliccare su **Editor**, **Save as**, **Save as type > Live Code files (.mlx)**. Per aggiungere i controlli per le istruzioni di assegnazione, (ad esempio $\text{delta}=0.64$), selezionare il valore numerico (0.64), fare clic su **Control** e selezionare un controllo appropriato, ad esempio un *cursoro numerico [slider]* o uno *spinner* per le variabili continue. Poi, nella casella popup, si digittano i valori minimo, massimo e passo per quel controllo. L'editor aggiunge uno slider numerico come questo:



`delta= 0.64` ; Si può anche utilizzare un *menù a discesa* per selezioni discrete, come il menù dei profili del picco in questo `Gaussian` ; esempio. Peak-Shape=

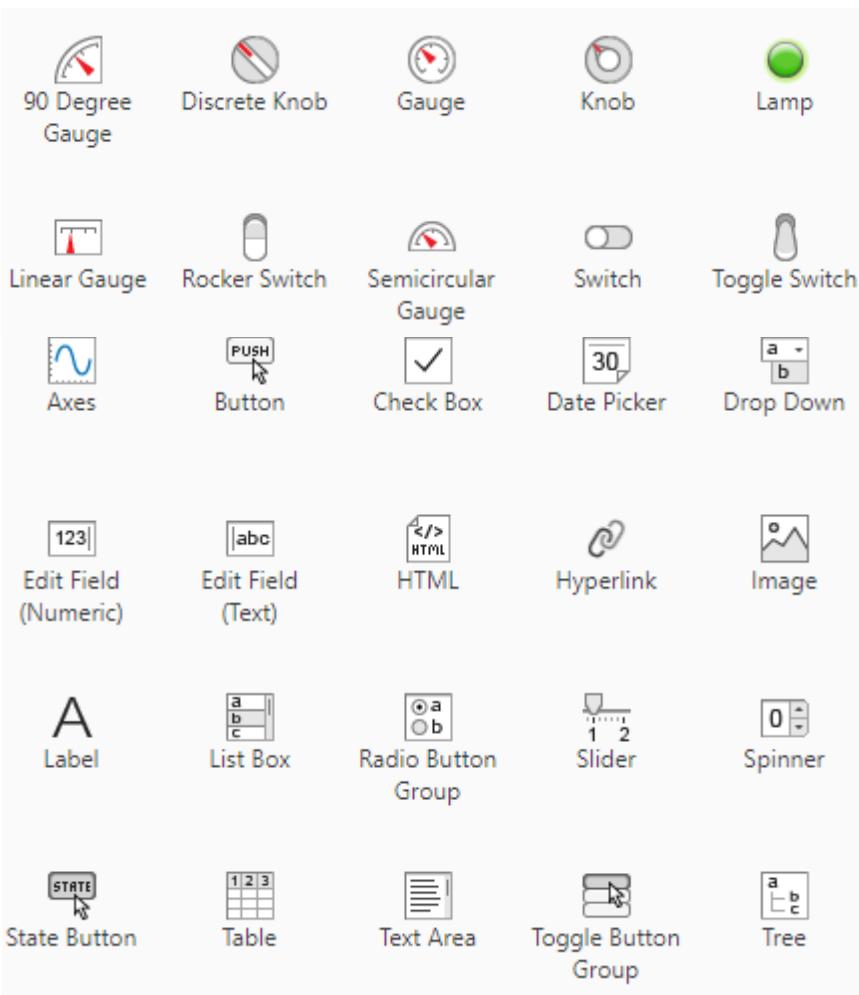
Ecco una [versione Live Script dell'app di auto-deconvoluzione](#), computazionalmente equivalente alla funzione ".m". Dispone di cursori per ciascuna delle variabili regolabili, che consentono di modificare tali variabili semplicemente facendo scorrere i puntatori. Ogni slider ha un intervallo numerico impostabile in un "intervallo ragionevole" per la variabile. Inoltre, l'app dispone di un menù a discesa per scegliere la forma del picco (Gaussiano, Lorentziano, ecc.) e di caselle di controllo per il calcolo opzionale dello spettro di frequenza e della precisione di misura dell'area del picco. Quando uno qualsiasi di questi controlli viene modificato, l'app ricalcola e aggiorna continuamente i grafici mentre il puntatore dello slider viene spostato. (Fare clic con il pulsante destro del mouse sul pannello di destra e selezionare "Disable synchronous scrolling"). Se lo spazio sullo schermo è limitato,

si può nascondere il codice cliccando sulla scheda **View** e poi su “**Hide Code**”. Per altri dettagli, vedere [questa pagina](#), o [questi esempi](#).



Oltre a questo script dimostrativo, esiste anche uno script correlato per applicare il metodo ai dati sperimentali archiviati su disco in formato .xlsx o .csv, chiamato [DeconvoluteData mlx \(schermata\)](#). Basta digitare il nome del file di dati in modo che inizi con "mydata=" nella parte superiore dello script. Lo script presuppone che i dati x,y siano nelle prime due colonne.

App di Matlab. Esiste un altro percorso di sviluppo che porta a programmi con un'interfaccia utente grafica *ancora più contemporanea e raffinata* (GUI). Questo è ideale per la distribuzione agli utenti che non hanno bisogno di avere accesso al codice, per evitare modifiche o cancellazioni accidentali di parti dello script. Queste sono chiamate “app” Matlab. Ne sono presenti esempi nei toolbox che potrebbero essere inclusi nella propria versione di Matlab (o possono essere facoltativamente acquistati da Matlab); digitare “ver” sulla riga di comando per vedere quali sono inclusi. Il processo di *sviluppo* di tali app è più complesso della codifica del Live Script o della funzione matematicamente equivalenti. Ma fortunatamente, Matlab ha un ambiente di sviluppo drag-and-drop integrato per creare interfacce utente; basta cliccare sul pulsante APPS in alto a sinistra. Questo fa apparire diversi pulsanti relativi alle app e un elenco di app già installate.



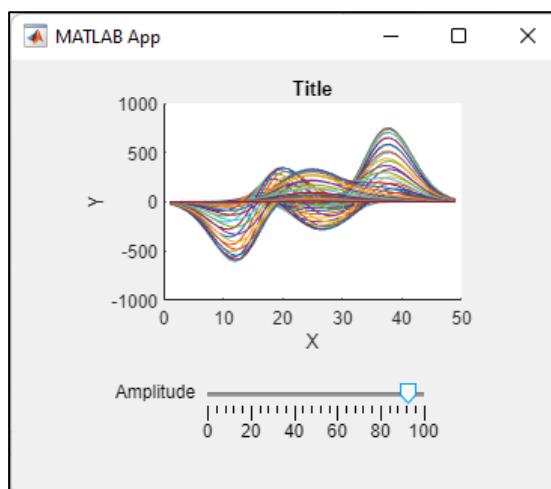
Facendo clic sul pulsante **De-sign App**, o digitando "appdesigner" al prompt dei comandi, viene visualizzata la schermata di App Designer, che ha due modalità principali, selezionate dai pulsanti sulla destra: la **De-**

sign View e la **Code View**. In Design View, si crea la propria interfaccia utente - il "look" dell'app - che potrebbe includere menù, pulsanti, slider, caselle di selezione, tabelle, grafici, ecc., selezionati da un ampio elenco di componenti (mostrati a lato), trascinando e rilasciando quelli che servono sul layout vuoto a destra e si dispongono come desiderato. Ciò che accade quando si aziona il controllo dipende dallo scopo dell'app, poi naturalmente si deve fornire quel codice, che viene chiamato

"funzione di callback". Tutto questo codice viene mostrato quando si clicca sul pulsante "Code View", sia il codice che viene generato automaticamente quando si aggiungono componenti al progetto (mostrato con uno sfondo *grigio*), che non si può modificare direttamente, e il codice che si può digitare per eseguire i calcoli desiderati (mostrato con uno sfondo *bianco*).

L'esempio mostrato a lato. "[Create and Run a Simple App Using App Designer](#)" mostra una forma d'onda di cui è possibile regolarne l'ampiezza in modo interattivo con uno slider. Si può imparare molto studiando esempi semplici come questo. Esistono molti esempi simili integrati nell'*App Designer*. Cliccando su "New" nella modalità Designer, verranno visualizzati (mostrati di seguito) diversi esempi già costruiti.

Anziché dettagliare tutti i passaggi richiesti qui, si farà riferimento ai molti eccellenti tutorial e video su YouTube già disponibili. Per esempio, c'è un tutorial video intitolato "Getting Started and Hello World app" all'indirizzo <https://www.youtube.com/watch?v=iga-YS6VbyE>.



Interactive Tutorial

Respond to Numerical Input

Respond to User Selections

Embed HTML Content

Lay Out Controls in a Grid

Link Data to a Tree

Analyze an Image

Configure a Timer

Display Specialized Axes

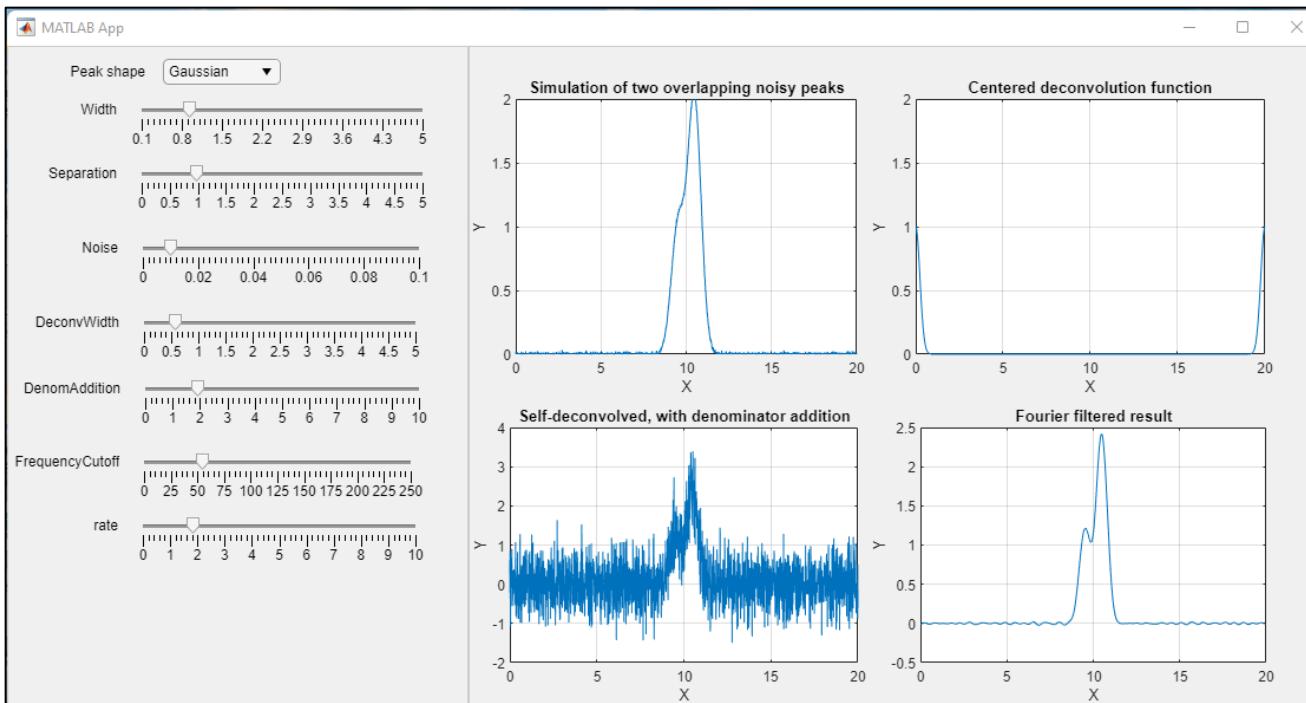
Create a Table

Query Website Data

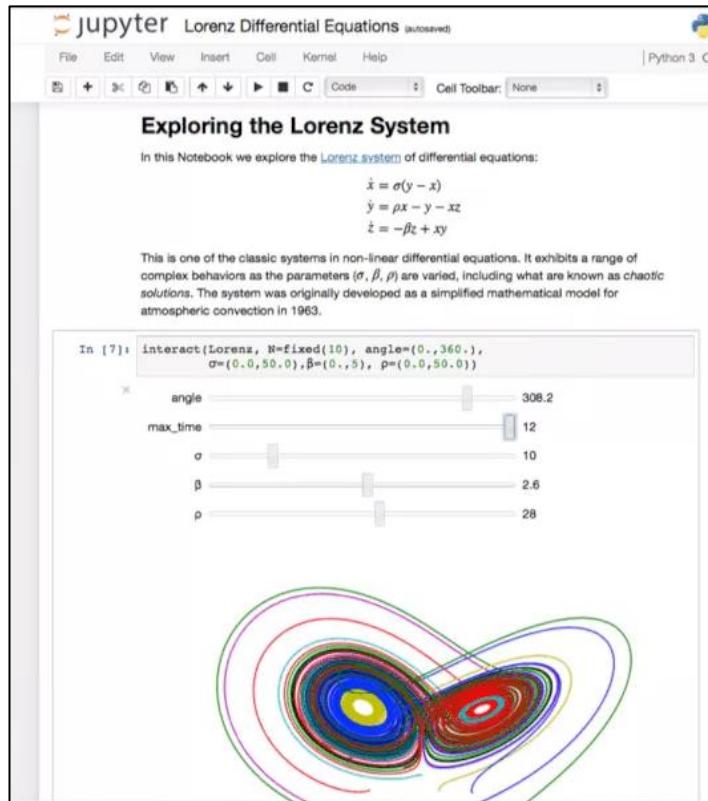
Lo *svantaggio* delle app è che sono più complesse per il programmatore. In effetti, la quantità di codice, il tempo e lo sforzo di codifica necessari alla progettazione e all'interattività dell'interfaccia utente di solito supera di gran lunga il codice richiesto per gli effettivi calcoli matematici.

Si deve aggiungere la programmazione specifica per i calcoli matematici, come in un normale script, che si digita nello spazio bianco del Code View, e *si può chiamare qualsiasi funzione scritta in precedenza* e salvata nel path di Matlab. Si può anche impacchettare qualsiasi app Matlab creata in un unico file, incluse tutte le funzioni che richiamate, in modo che possa essere facilmente condivisa con altri.

Ecco una versione dell'app Matlab della demo di auto-deconvoluzione. Si può scaricare il programma di installazione da [Self-deconvolution demo.mlappinstall](#).



La codifica richiesta per far funzionare tutto questo è considerevolmente più complessa rispetto alla creazione dell'equivalente Live Script mostrato sopra, ma l'app finisce per sembrare un'applicazione finita per l'utente finale ed è più semplice e infallibile per una persona inesperta da utilizzare.



Al momento Octave non dispone di funzionalità di sviluppo mobile integrate.

Python dispone dei [Jupyter Notebooks](#) utilizzati per creare una narrazione interattiva attorno al codice, come l'esempio a sinistra che dispone di cursori numerici per controllare le variabili in un sistema di equazioni non lineare.

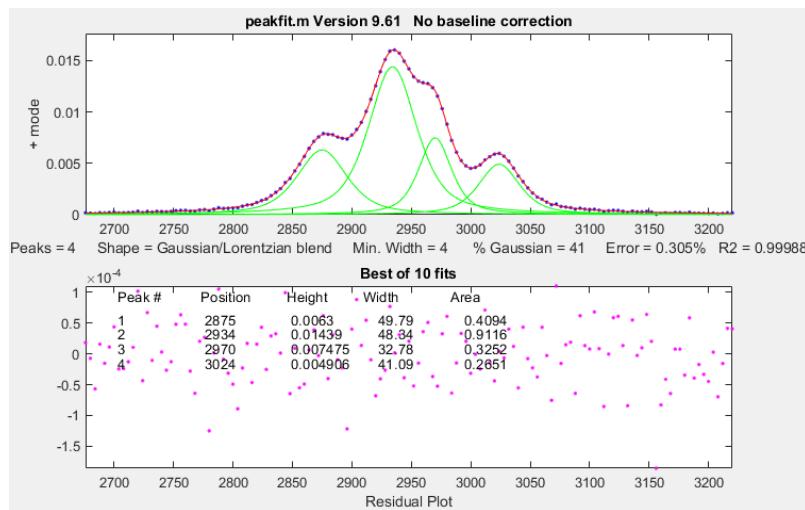
Esistono altri pacchetti che utilizzabili per creare applicazioni mobili in Python, come *Kivy*, *PyQt* o la libreria *Toga* di BeeWare; tali librerie sono tutti i principali attori nello spazio mobile di Python.

Uso della modellazione del segnale reale per determinare l'accuratezza della misura.

È comune utilizzare segnali generati dal computer le cui caratteristiche sono esattamente note per stabilire l'accuratezza di un metodo di elaborazione del segnale proposto, analogamente all'uso di materiali di riferimento standard nella chimica analitica (pagina 329). Ma il problema con i segnali generati al computer è che spesso sono troppo semplici o troppo ideali per essere realistici, come una serie di picchi tutti uguali in altezza e larghezza, di una forma idealizzata come una Gaussiana pura e con in più del rumore bianco casuale idealizzato. Per la misura delle aree dei picchi

parzialmente sovrapposti (pagina 135), tali picchi produrranno stime eccessivamente ottimistiche dell'accuratezza della misura dell'area. Un modo per creare segnali noti più realistici per una particolare applicazione consiste nell'usare l'*approssimazione iterativa della curva* (pagina 190). Se si riesce a trovare un modello che si adatta ai dati sperimentali con un errore di approssimazione molto basso e con residui casuali, allora i parametri di picco di quell'approssimazione possono essere utilizzati per costruire un segnale sintetico realistico per stimare la precisione e l'accuratezza della misura. Il grande vantaggio è che i parametri dei segnali sintetici possono essere modificati a piacimento per esplorare come funzionerebbe un metodo proposto in altre condizioni sperimentali (ad esempio, se la forma del picco fosse diversa o se il rumore o la frequenza di campionamento fossero più alti o più bassi).

Per mostrare questa idea, è stato scaricato uno [spettro](#) dal [database NIST IR](#) che conteneva un

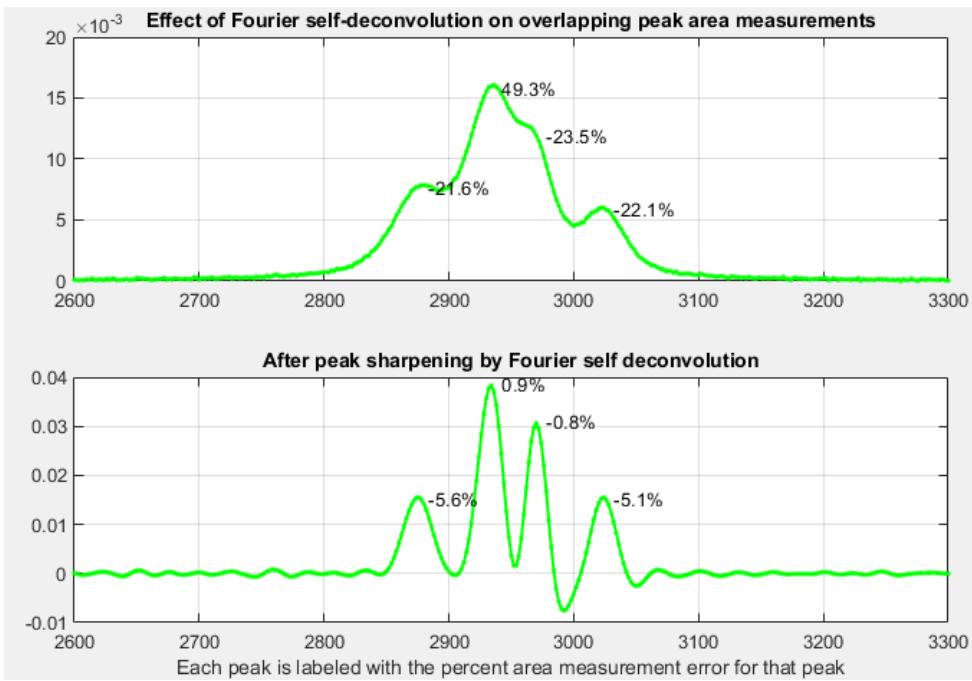


insieme di quattro picchi altamente fusi vicino a 3000 cm^{-1} . Per determinare le aree reali dei picchi nel modo più accurato possibile, sono stati approssimati iterativamente tali picchi con un modello costituito sommando quattro picchi Gaussiano-Lorentziano GLS (Gaussiano al 41%) di diverse larghezze, ottenendo un errore di approssimazione di solo 0.3% e un R^2 di 0,99988, con residui casuali

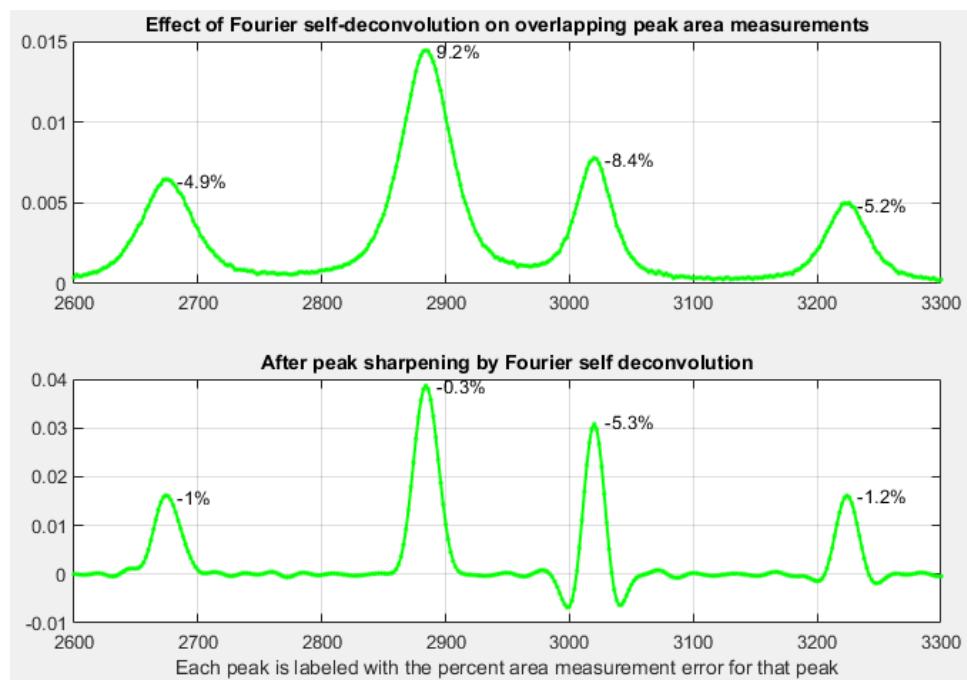
non strutturati, come mostrato a lato. I parametri più adatti e il rumore residuo sono stati poi utilizzati in uno [script 'self-contained'](#) per creare un modello sintetico di segnale che è *essenzialmente identico allo spettro sperimentale*, tranne per il fatto che ha aree di picco esattamente note. Quindi, lo script utilizza il [metodo del taglio verticale](#) più semplice e veloce per misurare tali aree, utilizzando la derivata seconda (pagina 58) per individuare le posizioni originali dei picchi, misura dell'area dei picchi mediante taglio verticale (pagina 133), che di per sé non si prevede che funzioni su picchi così sovrapposti, e infine ripetendo le misure dell'area dopo aver affinato i picchi mediante autodeconvoluzione di Fourier (pagina 111), utilizzando un filtro Fourier passa-basso per controllare il rumore (pagina 122)

Come mostrato dalla **prima figura** nella pagina seguente, lo sharpening dell'auto-deconvoluzione può infatti migliorare sostanzialmente la precisione dell'area del taglio verticale, da un errore medio del 29% a solo il 3,1% dopo la deconvoluzione. Ma poiché i picchi hanno larghezze diverse, non esiste un'unica larghezza di deconvoluzione ottimale. I test mostrano che i migliori risultati complessivi si ottengono quando la funzione di deconvoluzione *shape* [profilo] è la stessa del segnale originale e quando la funzione di deconvoluzione *width* [larghezza] è 1,1 volte l'*ampiezza media* del picco nel segnale. Nella **seconda figura**, i picchi nel segnale del modello sono sparsi artificialmente, senza altre modifiche, solo per mostrare più chiaramente che questa scelta dell'ampiezza della funzione di deconvoluzione fa sì che il terzo picco sia "eccessivamente nitido", risultando in lobi negativi per quello picco (ma ricorda che la deconvoluzione avviene in modo da *conservare l'area totale del picco*; vedere pagina 107). Un approccio più conservativo, utilizzando la massima larghezza di deconvoluzione possibile senza che il segnale diventi mai negativo (in questo caso circa 0,8 volte la larghezza media del picco) si traduce solo in un modesto miglioramento dell'accuratezza dell'area media (dal 27% al 12%; [grafico](#)). Quindi "lo sharpening

“eccessivo” non è sempre negativo.



Sampling interval (cm⁻¹) = 2
 Change in peak separation (PeakSpread) = 0
 Noise = 5e-05
 GLS Shape (fraction Gaussian) = 0.41
 Deconvolution Width = 23.7 points (1.1 times the mean signal peak width)
 Frequency Cutoff = 20%

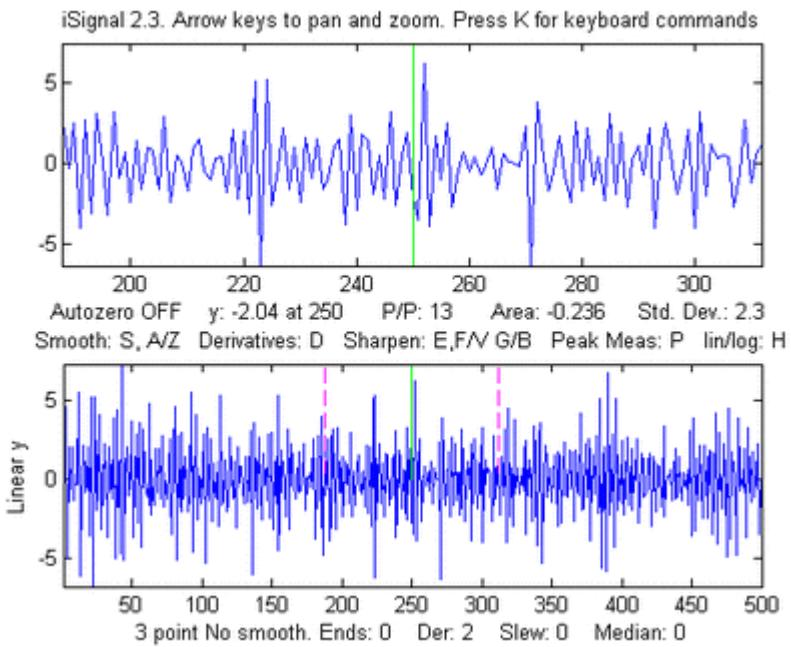


Change in peak separation (PeakSpread) = 100 (All other parameters unchanged)

Dettagli software del signal processing.

Smoothing, differenziazione e analisi del segnale interattivamente
(iSignal)

[iSignal](#) (o la versione Octave [isignal octave.m](#)) è un singolo file .m autonomo per l'esecuzione dello



smoothing, differenziazione, sharpening, interpolazione, sottrazione della linea di base, spettro di frequenza di Fourier, approssimazione dei minimi quadrati, deconvoluzione, e altre utili funzioni sui dati delle serie temporali. Utilizzando semplici sequenze di tasti, è possibile regolare continuamente i parametri di elaborazione del segnale osservandone dinamicamente l'effetto. [Cliccare qui per scaricare il file ZIP "iSignal8.zip"](#) che include anche alcuni dati di esempio per i test.

Si può anche scaricare iSignal dal [Matlab File Exchange](#). Si può anche eseguire iSignal [in un browser web](#) (basta fare clic sulla finestra della figura), anche su [Matlab Mobile](#). C'è una versione [separata per Octave](#). Lo script demo "[demoisignal.m](#)" è una dimostrazione a esecuzione automatica di diverse funzionalità del programma e ne verificherà la corretta installazione; il titolo di ogni figura descrive cosa sta succedendo. Il funzionamento di base di iSignal è simile a [iPeak](#) e [ipf.m](#).

La sintassi è: `pY=isignal(Data);` o, per specificare subito tutte le impostazioni :

```
[pY, Spectrum, maxy, miny, area, stdev] = isignal(Data, xcenter, xrange, SmoothMode, SmoothWidth, ends, DerivativeMode, Sharpen, Sharp1, Sharp2, Symize, Symfactor, SlewRate, MedianWidth, SpectrumMode);
```

"Data" può essere una matrice con 2 colonne con la variabile indipendente (valori x) nella prima colonna e quella dipendente (valori y) nella seconda colonna, o vettori x e y separati, o un singolo vettore y (in questo caso i punti vengono disegnati rispetto ai loro numeri di indice sull'asse x). È richiesto solo il primo argomento (Data); tutti gli altri sono opzionali. iSignal restituisce il vettore dell'asse dipendente processato ('pY') (e, nella [Modalità Spettro](#), la matrice dello spettro di frequenza, 'Spectrum') come argomenti di output. Disegna i dati nella finestra di 'Figure' di Matlab, la metà inferiore della finestra mostra l'intero segnale e quella superiore mostra una porzione selezionata controllata dai tasti pan e zoom (i quattro tasti freccia). Le impostazioni iniziali di pan e zoom sono facoltativamente controllate dagli argomenti di input 'xcenter' e 'xrange', rispettivamente. Doppio-click sulla barra del titolo della figura per vedere meglio a tutto schermo. Altre sequenze di tasti consentono di controllare il tipo di smoothing, la larghezza e il trattamento delle estremità, l'ordine della derivata (dalla 0^a fino alla 5^a) e lo sharpening. (I valori iniziali di tutti questi parametri possono essere passati alla funzione tramite gli opzionali argomenti di input **SmoothMode**, **SmoothWidth**, **ends**, **DerivativeMode**, **Sharpen**, **Sharp1**, **Sharp2**, **SlewRate** e **MedianWidth**. Si vedano gli esempi seguenti). Premere **K** per vedere tutti i comandi da tastiera.

Nota: Assicurarsi di non cliccare sul pulsante "Show Plot Tools" nella barra degli strumenti sopra la figura; che disabiliterà il normale funzionamento del programma. Se lo si fa, si deve chiudere la finestra "Figure" e ricominciare.

Smoothing

Il tasto **S** (o l'argomento di input "**SmoothMode**") cicla attraverso le 5 modalità di smoothing:
Se **SmoothMode**=0, Il segnale non viene filtrato.

Se **SmoothMode**=1, rettangolare (slittamento-della-media o boxcar)

Se **SmoothMode**=2, triangolare (2 passaggi dello slittamento-della-media)

Se **SmoothMode**=3, p-spline (3 passaggi dello slittamento-della-media)

Se **SmoothMode**=4, smoothing di [Savitzky-Golay](#) smooth (un grazie a [Diederick](#)).

I tasti **A** e **Z** (o gli argomenti opzionali di input **SmoothWidth**) controllano lo **SmoothWidth**, w . Il tasto **X** commuta "ends" tra 0 e 1. Questo determina come gestire le "estremità" del segnale (i primi $w/2$ punti e gli ultimi $w/2$ punti) per lo smoothing:

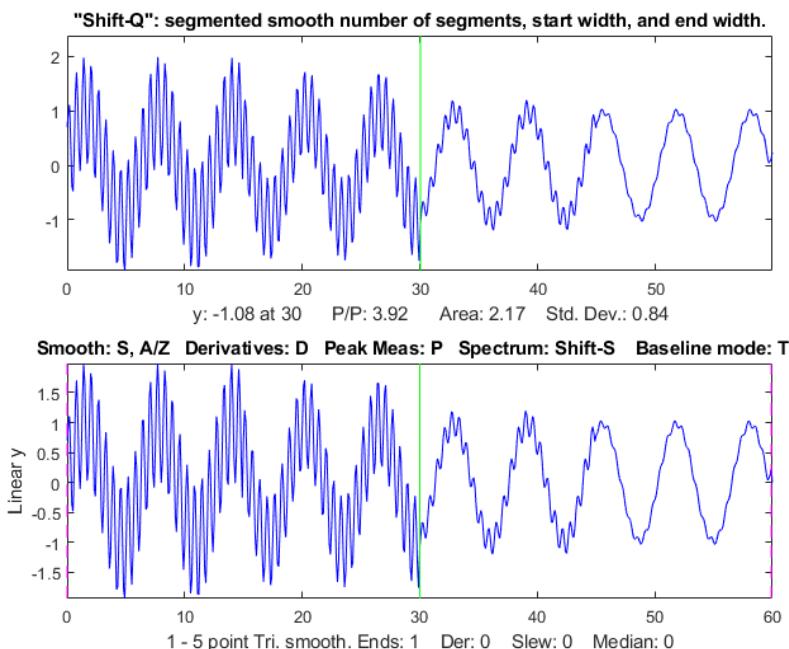
Se **ends**=0, le estremità vengono azzerate.

Se **ends**=1, le estremità subiscono lo smoothing con ampiezze progressivamente più piccole man mano che ci si avvicina agli estremi. Generalmente, **ends**=1 è meglio, tranne alcuni casi in cui con la derivata e quando **ends**=0 si ha una migliore centratura verticale del segnale.

Per specificare uno smoothing [segmentato](#) (approfondimenti a pagina 326) si preme **Shift-Q**. Si può indicare il vettore dell'ampiezza per lo smoothing in *due modi*: sulla riga di comando si può o (a) immettere il numero di segmenti (verrà quindi richiesto di inserire le ampiezze di smoothing del primo e nell'ultimo segmento e il computer calcolerà i valori interi per le ampiezze dello smoothing equamente divise tra il primo e l'ultimo valore specificato, oppure (b) si inserisce il vettore delle ampiezze per lo smoothing *direttamente* includendoli tra parentesi quadre, p.es [1 3 3 9]. In entrambi i casi, la successiva regolazione della larghezza dello smoothing con i tasti **A** e **Z** cambierà *tutti i segmenti con lo stesso fattore percentuale*. (Per tornare allo smoothing normale a segmento unico, si inserisce 1 come numero di segmenti). La figura a lato mostra uno smoothing a 4 segmenti con larghezze di smoothing di 1, 2, 4 e 5

Nota: quando si esegue lo smoothing dei picchi, se ne può facilmente misurare l'effetto sull'altezza e sulla larghezza attivando la modalità di misura del picco (premere **P**) e poi premere **S** per scorrere le modalità di smoothing.

Ci sono due funzioni speciali per rimuovere o ridurre gli spike stretti nei segnali: il tasto **M**, che implementa un filtro mediano (chiede di inserire l'ampiezza dello spike, ad es., 1,2, 3... punti) e il tasto **~**, che limita il tasso massimo di variazione.



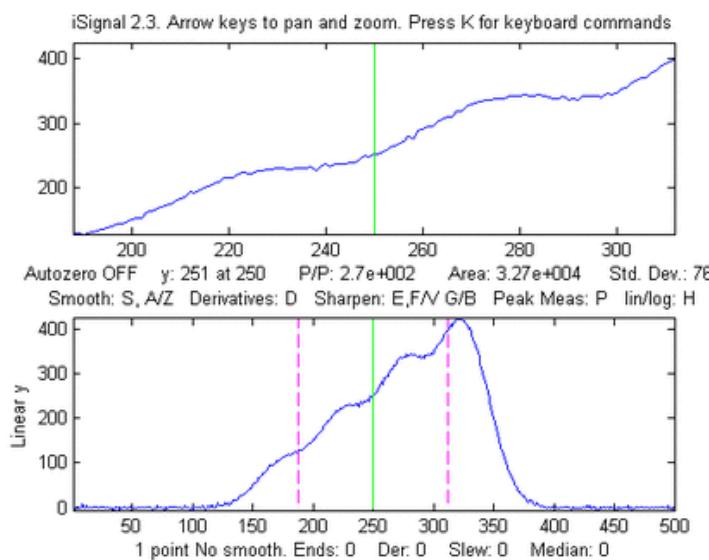
Differenziazione

I tasti **D** / **Shift-D** (o l'argomento opzionale di input "**DerivativeMode**") aumentano/diminuiscono l'ordine della derivata. Il default è 0. Un'accurata ottimizzazione dello smoothing delle derivate è essenziale per un accettabile rapporto segnale-rumore. Un esempio è mostrato nella figura a lato. Negli SmoothModes da 1 a 3, le derivate vengono calcolate rispetto alla variabile indipendente (valori di x), corrette per gli intervalli dell'asse x non-uniformi. Nello SmoothMode 4 (Savitzky-Golay) le derivate vengono calcolate co l'algoritmo di Savitzky-Golay. [Cliccare per un'animazione GIF](#).

Peak sharpening

Il tasto **E** key (o l'argomento opzionale di input "**Sharpen**") disattiva e attiva il peak sharpening.

L'intensità dello sharpening è controllato dai tasti **F** e **V** (o dall'argomento opzionale "Sharp1") e i tasti **B** e **G** ((o dall'argomento opzionale "Sharp2"). I valori ottimali dipendono dalla forma e dalla larghezza del picco. Per picchi di forma Gaussiana, un ragionevole valore per **Sharp1** è PeakWidth²/25 e per **Sharp2** è PeakWidth⁴/800 (o PeakWidth²/6 e PeakWidth⁴/700 per i picchi Lorentziani), dove PeakWidth è la larghezza a metà altezza del massimo dei picchi *espressa come numero di punti*.



Tuttavia, non ci sono calcoli da fare; *iSignal* può calcolare le impostazioni per lo sharpening e lo smoothing per il profilo Gaussiano e Lorentziano utilizzando i tasti **Y** e **U** rispettivamente. Basta isolare un solo picco tipico nella finestra superiore con i tasti pan e zoom, poi premere **Y** per picchi Gaussiani o **U** per i Lorentziani. L'aggiustamento finale dello sharpening avviene con i tasti **F/V** e **G/B** e quello dello smoothing con i tasti **A/Z**. Questa animazione si può vedere se si scarica la versione

[Microsoft Word 365](#), altrimenti cliccare su [questo link](#). (Le impostazioni ottimali dipendono dalla larghezza del picco, quindi se il segnale ha picchi con ampiezze molto diverse, un solo settaggio non sarà ottimale per tutti e si dovrà prendere in considerazione lo smoothing segmentato, come descritto precedentemente a pagina 326). Ci si può aspettare una diminuzione della larghezza del picco (ed un conseguente aumento della sua altezza) di circa il 20% - 50%, a seconda della forma del picco (l'area del picco, in gran parte, non è influenzata dallo sharpening). Uno sharpening eccessivo provoca artefatti alla linea di base e un aumento del rumore. *iSignal* consente di determinare sperimentalmente i valori di questi parametri che offrono il miglior compromesso tra sharpening, rumore e artefatti della linea di base, per i propri scopi. È possibile misurare facilmente l'effetto dello sharpening quantitativamente attivando la modalità di misurazione del picco (premere **P**) e premerendo **E** per attivare e disattivare la modalità di sharpening. Nota: per lo sharpening del picco viene usato solo lo smoothing di Savitzky-Golay.

Convoluzione e deconvoluzione interattiva

In [iSignal 8.3](#) si può premere **Shift-V** per visualizzare il menù delle operazioni di convoluzione e deconvoluzione di Fourier (pagina 106) che consentono di effettuare una convoluzione di una funzione Gaussiana, Lorentziana, o esponenziale col segnale, o una deconvoluzione di una funzione Gaussiana, Lorentziana, o esponenziale dal segnale.

Fourier convolution/deconvolution menu

1. Convolution
 2. Deconvolution
- Select mode 1 or 2: 2

Shape of convolution/deconvolution function:

1. Gaussian
2. Lorentzian
3. Exponential

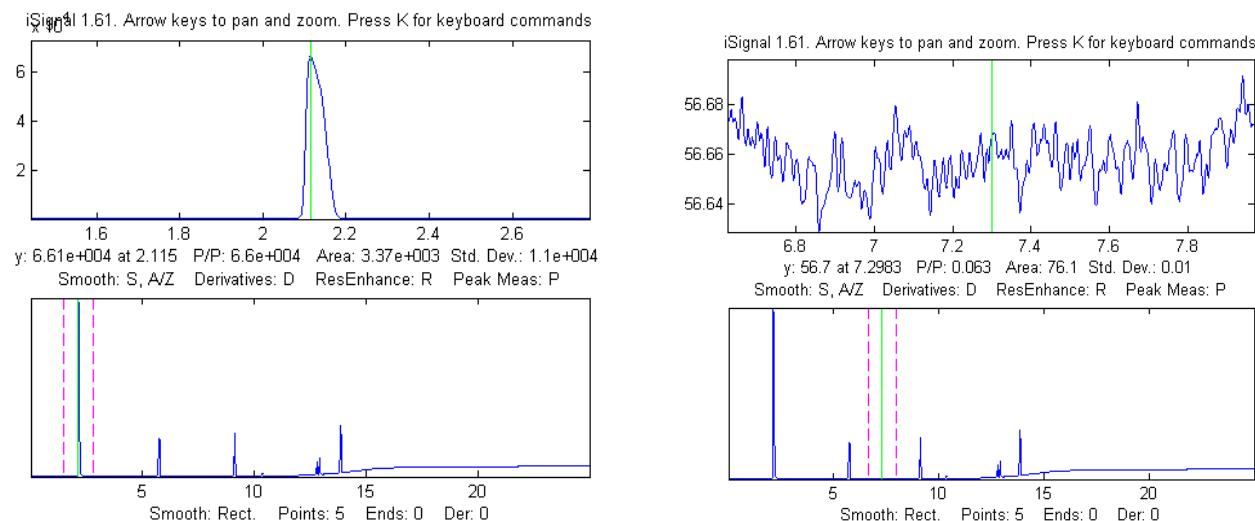
Select shape 1, 2, or 3: 2

Enter the width in x units:

Infine si inserisce la costante di tempo (in unità x) e si preme **Enter**. Poi si usano i tasti **3** e **4** per regolare la larghezza della funzione di deconvoluzione del 10% (o **Shift-3** e **Shift-4** per regolarla all'1%). Potrebbe essere necessario regolare anche lo smoothing, se il segnale è troppo rumoroso, ma uno smoothing eccessivo allargherà i picchi. Per un'applicazione con un segnale reale, vedere pagina 120. Nota: nei casi in cui le larghezze dei picchi all'interno di un gruppo di picchi sono sostanzialmente diverse, potrebbe essere preferibile utilizzare la [deconvoluzione segmentata](#).

La simmetrizzazione interattiva (o “de-tailing”) di segnali ampliati esponenzialmente viene eseguita mediante *addizione ponderata della derivata prima* (pagina 77). Si preme il tasto **Shift-Y**, si inserisce un fattore di ponderazione stimato (che è la costante di tempo o tau dell'esponenziale) e si preme **Enter**. Per regolare, si premono i tasti "1" e "2" per modificare il fattore di ponderazione del 10% e i tasti "**Shift-1**" e "**Shift-2**" per modificarlo dell'1%. Aumentare il fattore finché la linea di base dopo il picco non diventa negativa, poi si decrementa leggermente in modo che sia la più bassa possibile ma non negativa. Eseguire lo script [iSignalSymmTest.m](#) ([grafico](#)) per un segnale di esempio con due Gaussiane esponenzialmente allargate sovrapposte.

Misura del segnale

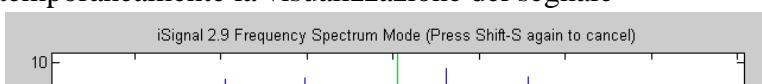


Misura del rapporto segnale-rumore (SNR) di un segnale con un SNR molto alto. Sinistra: L'altezza del picco più grande viene misurata ponendo il cursore verde al centro sul picco più grande; segnale picco-picco=66,000. Destra: Il rumore viene misurato su una porzione piatta della linea di base: deviazione standard del rumore=0.01, quindi SNR=66,000/0.01 = 6,600,000

I tasti cursore controllano la posizione del cursore verde (tasti freccia sinistra e destra) e la distanza tra i cursori i cursori rossi tratteggiati (tasti freccia su e giù) che contrassegnano l'intervallo selezionato visualizzato nella finestra del grafico superiore. L'etichetta sotto la finestra del grafico in alto mostra il valore del segnale (y) in corrispondenza del cursore verde, l'intervallo del segnale picco-picco (minimo e massimo), l'area sotto il segnale e la deviazione standard all'interno dell'intervallo selezionato (i cursori tratteggiati). Premendo il tasto **Q** viene stampata una tabella con le informazioni sul segnale nella finestra di comando. Se vengono specificati gli argomenti opzionali di output maxy, miny, area, stdev, iSignal restituisce il valore massimo di y, il valore minimo di y, l'area totale sotto la curva e la deviazione standard di y, nell'intervallo selezionato visualizzato nel pannello superiore.

Modalità Spettro delle Frequenze

La **modalità Spettro delle Frequenze**, attivata e disattivata dal tasto **Shift-S**, calcola lo spettro delle frequenze di Fourier (pagina 87) del segmento di segnale visualizzato nella finestra superiore e lo mostra in quella inferiore (rimpiazzando temporaneamente la visualizzazione del segnale)



completo). Si usano i tasti pan e zoom per regolare la regione del segnale da visualizzare. Si preme **Shift-A** per passare attraverso le quattro modalità del grafico (lineare, semilog X, semilog Y e log-log) e si preme **Shift-X** per alternare tra una *frequenza* sull'asse x e il *tempo* sull'asse x. È importante sottolineare che *tutte le funzioni di elaborazione del segnale rimangono attive nella modalità spettro di frequenza* (smoothing, derivata, ecc.) in modo da poter osservare immediatamente l'effetto di queste funzioni sullo spettro di frequenza del segnale. Questa animazione si può vedere scaricando la versione [Microsoft Word 365](#), altrimenti cliccare sulla figura per aprire una pagina Web). Si preme **Shift-T** per trasferire lo spettro delle frequenze del segnale nel pannello superiore in modo da poterne eseguire panoramiche e zoom oltre ad altre elaborazioni e misurazioni sullo spettro delle frequenze. Si preme ancora **Shift-S** per tornare alla modalità normale. La modalità spettro è una *modalità visibile*, indicata da un'etichetta nella parte superiore della figura. Per *avviare* con la modalità spettro, si imposta il 13° argomento di input, SpectrumMode, a 1. Per *salvare* lo spettro come nuova variabile, si chiama iSignal con gli argomenti di output [pY , Spectrum] :

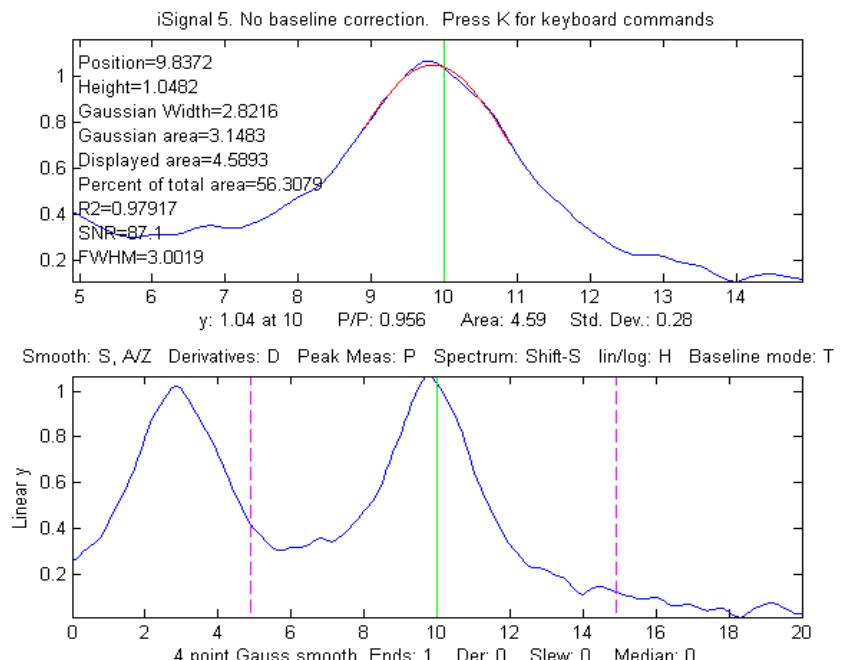
```
>> x=0:.1:60; y=sin(x)+sin(10.*x);
>> [pY,Spectrum]=isignal([x;y],30,30,4,3,1,0,0,1,0,0,0,1);
>> plot(Spectrum(:,1),Spectrum(:,2)) o plotit(Spectrum) o
isignal(Spectrum); o ipf(Spectrum); o ipeak(Spectrum)
```

Shift-Z attiva e disattiva il rilevamento dei picchi e l'etichettatura sullo spettro frequenza/tempo; i picchi vengono etichettati con le loro frequenze. È possibile regolare i parametri di rilevamento dei picchi nelle righe 2192-2195 nella versione 5. Il comando **Shift-W** visualizza lo [spettro 3D a cascata](#), dividendo il segnale in segmenti e calcolando la potenza spettrale di ogni segmento. Questa è principalmente una novità, ma può essere utile per segnali il cui spettro delle frequenze varia nel corso della durata del segnale. Viene richiesto di scegliere il numero dei segmenti in cui dividere il segnale (ovvero il numero di spettri) e il tipo di visualizzazione 3D (mesh, contorno, superficie, ecc.)

Sottrazione del background

Ci sono due modi per sottrarre il background dal segnale: automatico e manuale. Per selezionare la correzione automatica della linea di base, si preme il tasto **T** ripetutamente; si passa attraverso quattro modalità (pagina 211): *Nessuna* correzione della linea di base, sottrazione *lineare* della linea di base, sottrazione *quadratica* della linea di base, correzione linea di base *piatta*, poi di nuovo *nessuna* correzione della linea di base. Quando la modalità della linea di base è *lineare*, verrà sottratta automaticamente una linea retta di base che connette le due estremità della porzione di segnale nel pannello superiore. Quando la modalità della linea di base è *quadratica*, verrà sottratta automaticamente una linea di base

parabolica connessa tra le due estremità della porzione di segnale nel pannello superiore. La linea di base viene calcolata con un'approssimazione lineare (o quadratica) ai minimi quadrati del segnale del primo $1/10^\circ$ dei punti del segnale e dell'ultimo $1/10^\circ$ dei punti. Si provi a regolare il pan e lo zoom per includere una quantità sufficiente della linea di base all'inizio e alla fine del segmento



nella finestra superiore per accertarsi che con la sua sottrazione automatica si ottenga una buona lettura della linea di base. La modalità della linea di base *piatta* viene usata solo per l'[approssimazione del picco](#) (tasto **Shift-F**). I calcoli per l'ampiezza del segnale, del segnale picco-picco e dell'area del picco vengono ripetuti dopo la sottrazione della linea di base dal segnale nella finestra superiore. Se si stanno misurando picchi sovrapposti ad un background, l'uso della BaselineMode avrà un notevole effetto sulla misura dell'altezza, della larghezza e dell'area del picco, ma un effetto molto limitato sulla posizione sull'asse x del picco.

Oltre alle quattro modalità di BaselineMode per la sottrazione della linea di base per la misurazione del picco, una linea di base lineare a tratti *stimata manualmente* si può sottrarre dall'intero segnale con un'unica operazione. Col tasto **Backspace** si avvia l'operazione di correzione del background. Nella finestra comando, si digita il numero di punti del background da cliccare e si preme il tasto **Enter**. Il cursore si trasforma in mirino; si clicca lungo il presunto background nella finestra della figura, partendo da sinistra verso destra dell'asse x e piazzando l'ultimo click alla destra dell'asse x. Quando si clicca sull'ultimo punto, la linea di base interpolata linearmente tra questi punti viene sottratta dal segnale. Per ripristinare il background originale (cioè per ricominciare da capo), premere il tasto '\' (appena sotto il tasto backspace).

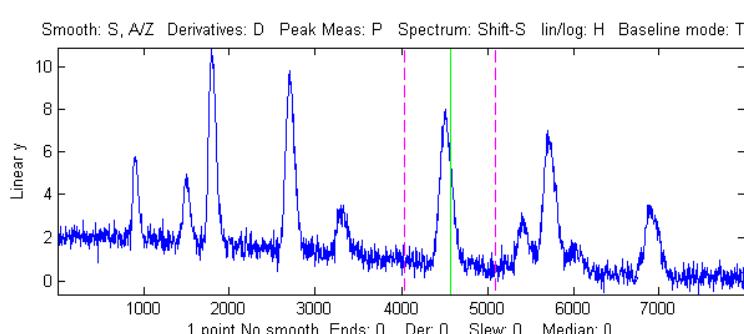
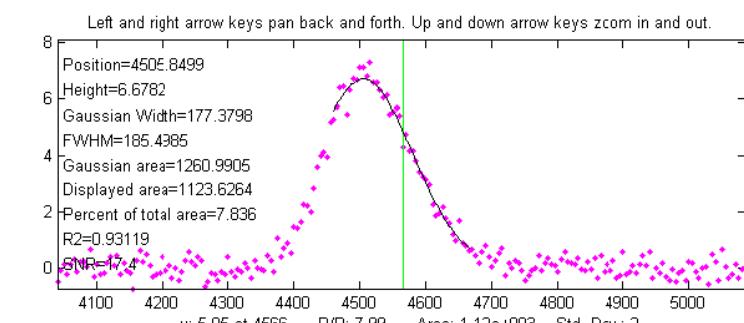
Misura di picchi e avvallamenti

Il tasto **P** attiva e disattiva la modalità "peak parabola", che tenta di misurare l'unico picco (o valle) centrato nella finestra superiore sotto il cursore verde sovrapponendo una [parabola approssimata dei minimi quadrati](#) (pagina 165) in rosso, sulla porzione centrale del segnale visualizzato nella finestra superiore. (Ingrandire in modo che il rosso si sovrapponga il più possibile solo alla cima del picco o al solo fondo della valle). La posizione, l'altezza e la larghezza del picco vengono misurati con la curva Gaussiana approssimata ai quadrati minimi (colorata in rosso nel pannello superiore) alla porzione centrale del segmento selezionato. (Modificare pan e zoom per cambiare la regione; i valori cambieranno modificando la porzione misurata). Il valore "RSquared" è il coefficiente di determinazione; più ci si avvicina a 1.000, meglio è. I parametri del picco saranno più accurati se i picchi sono Gaussiani. Altre forme di picchi, o picchi qualsiasi molto rumorosi, daranno solo risultati approssimativi. Tuttavia i valori della posizione, dell'altezza e dell'area sono abbastanza buoni per qualsiasi forma di picco se il valore di "RSquared" è almeno 0.99. "SNR" è il rapporto segnale/rumore del picco sotto il cursore verde; è il rapporto tra l'altezza del picco e la deviazione standard dei residui tra i dati e la linea dell'approssimazione in rosso.

Un esempio è mostrato nella figura a lato. Se i picchi sono sovrapposti su un background diverso da

zero, sottrarre il background prima di misurare i picchi, utilizzando BaselineMode (tasto **T**) o la sottrazione multi-punto del background (tasto backspace). Premere il tasto **R** per stampare i parametri del picco nella finestra di comando.

La *larghezza* del picco viene misurata *in due modi*: la "Gaussian Width" è la larghezza misurata dall'approssimazione della curva Gaussiana (sulla regione colorata in rosso nel pannello superiore) ed è molto accurata solo per i picchi Gaussiani. La versione **5.8** (mostrata sotto a sinistra) aggiunge la [misura diretta](#)



dell'intera larghezza a metà del massimo ('FWHM') del picco centrale nel pannello superiore (il picco contrassegnato dalla linea verticale verde); funziona per picchi con *qualsiasi profilo*, ma viene calcolata solo per il picco centrale e solo se i punti a metà del massimo rientrano nella regione di zoom visualizzata nel pannello superiore (altrimenti restituisce NaN). Non sarà molto preciso per picchi molto rumorosi. L'ampiezza Gaussiana sarà più precisa per i picchi rumorosi e per quelli scarsamente campionati, ma solo se i picchi sono almeno approssimativamente Gaussiani.

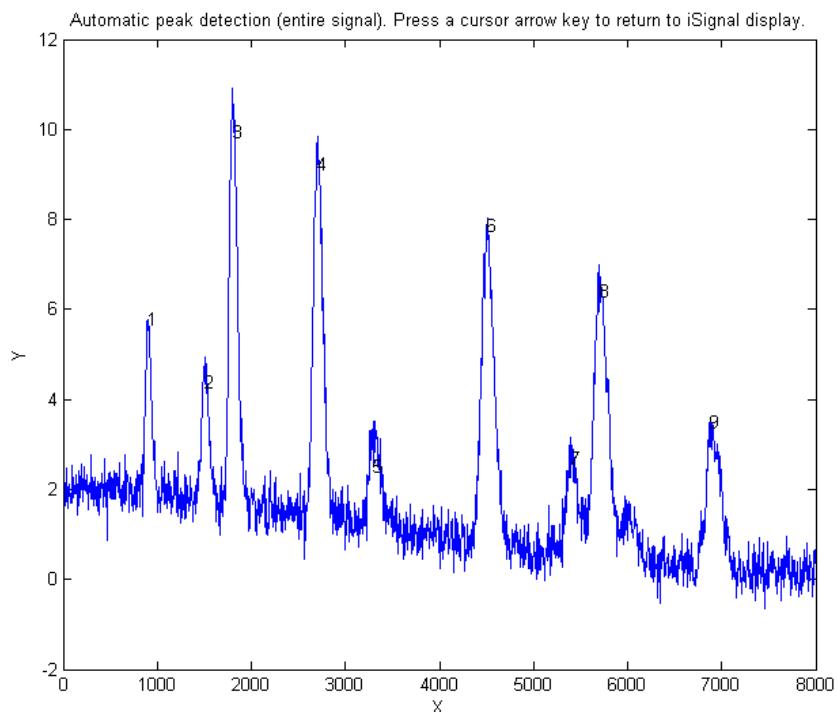
Nell'esempio a sinistra, i picchi sono Lorentziani, con una larghezza reale di 3.0, più il rumore aggiunto. In questo caso l'FWHM misurato (3.002) è più preciso della larghezza Gaussiana (2.82), soprattutto se si usa un po' di smoothing per ridurre il rumore.

Anche l'*area* viene misurata *in due modi*: la "Gaussian area" e la "Total area". La "Gaussian area" è l'area sotto la Gaussiana che approssimasi meglio la porzione centrale del segnale visualizzato nella finestra superiore, contrassegnata in rosso. "L'area totale" è l'area del metodo trapezoidale dell'intero segmento selezionato visualizzato nella finestra superiore. (Viene anche calcolata la percentuale rispetto all'area totale). Se la porzione del segnale visualizzato nella finestra superiore è una Gaussiana pura senza rumore e una linea di base a zero, allora le due misure dovrebbero quasi coincidere. Se il picco non ha una forma Gaussiana, allora è probabile che l'area totale sia più accurata, se il picco è ben separato

dagli altri. Se i picchi sono

sovraposti, ma hanno un profilo noto, l'approssimazione del picco (**Shift-F**) fornirà misure più accurate.

Nell'esempio sopra, il picco Lorentziano in $x=10$ ha un'area reale di 4.488, quindi in questo caso l'area totale (4.59) è più precisa dell'area Gaussiana (3.14), ma è troppo alta a causa della sovrapposizione con il picco in $x=3$. Il curve fitting di entrambi i picchi Lorentziani combinati produrrebbe le aree più accurate. Se il segnale viene spostato leggermente a sinistra e a destra, utilizzando i tasti cursore sinistro e destro o i tasti "[" e "]", i parametri del picco visualizzato cambiano leggermente a causa del rumore nei dati - più c'è rumore, più i valori cambiano, come nell'esempio a sinistra. Se il picco è asimmetrico, come in questo esempio, le larghezze visualizzate su un lato saranno maggiori di quelle sull'altro lato.



C'è un 'peak finder' automatico che è basato sulla funzione [autopeaks.m](#) (attivata dal tasto **J**); chiede la densità di picco (all'incirca il numero di picchi che si approssimano alla registrazione del segnale), poi rileva, misura e visualizza la posizione, l'altezza e l'area di tutti i picchi rilevati nel segnale elaborato attualmente visualizzato nella parte inferiore pannello, disegna e numera i picchi come mostrato a destra e [disegna anche ciascun picco separatamente nella Window 2](#) con i punti del picco, della tangente e dell'avallamento contrassegnati (clicare per il [grafico](#)). (La densità dei picchi richiesta controlla la sensibilità dei picchi - numeri grandi fanno sì che la routine rilevi un numero maggiore di picchi più stretti e numeri piccoli fanno sì che si ignorino le strutture più strette andando alla ricerca dei picchi più larghi). Stampa anche i parametri di rilevamento calcolati per

usarli con qualsiasi delle [funzioni findpeaks...](#) (pag. 227). Per ritornare alla solita visualizzazione iSignal, si preme un qualsiasi tasto freccia. (**Shift-J** a la stessa cosa per il segmento visualizzato nella finestra superiore).

Peak fitting

iSignal ha un metodo di [curve fitting interattivo](#) (pagina 190) eseguito da [peakfit.m](#). Questo è il metodo più accurato per la misura delle aree di picchi molto sovrapposti. Innanzitutto, si centra il segnale che si desidera approssimare utilizzando i tasti pan e zoom (tasti freccia), si seleziona la modalità della linea di base premendo il tasto 'T' per [scorrere le 4 modalità di linea di base](#): none, linear, quadratic e flat (vedere pagina 211). Premere il tasto **Shift-F**, poi digitare il profilo desiderato del picco in base al numero dal menù visualizzato nella finestra Command (pag. seguente), immettere il numero dei picchi, il numero di tentativi di approssimazioni (di solito 1-10), e infine *clicare sul grafico in alto dove si pensa che potrebbero stare i picchi*. (Per i picchi fuori schermo, cliccare *esternamente* ai limiti dell'asse ma all'interno della finestra del grafico). Viene mostrato un grafico del fitting nella finestra Figure e viene stampata una tabella dei risultati nella finestra comando. iSignal può approssimare molte diverse combinazioni di profili e vincoli dei picchi:

Gaussians: $y=\exp(-((x-pos) / (0.6005615.*width)).^2)$	
Gaussiane con posizioni e larghezze indipendenti (default)	1
Gaussiane espansa esponenzialmente (costanti di tempo uguali)	5
Gaussiane espansa esponenzialmente con larghezze uguali.....	8
Gaussiane espansa esponenzialmente con larghezza fissa.....	36
Gaussiane espansa esponenzialmente (costanti di tempo indipendenti)	31
Gaussiane con le stesse larghezze.....	6
Gaussiane con larghezze pre-impostate.....	11
Gaussiane in posizioni fisse.....	16
Gaussiane asimmetriche con diverse semi-larghezze su entrambi i lati.....	14
Lorentzians: $y=ones(size(x)) ./ (1+((x-pos) / (0.5.*width)).^2)$	
Lorentziane con posizioni e larghezze indipendenti.....	2
Lorentziana espansa esponenzialmente.....	18
Lorentziane con larghezze uguali.....	7
Lorentziana con larghezza fissa.....	12
Lorentziana con posizione fissa.....	17
Mix di Gaussiane/Lorentziana (mix uguali)	13
Mix di Gaussiane/Lorentziane a larghezza fissa.....	35
Mix di Gaussiane/Lorentziane con mix indipendenti)	33
Profilo Voigt con alfa uguali.....	20
Profilo Voigt con con larghezza fissa e alfa uguali.....	34
Profilo Voigt con con le alfa indipendenti.....	30
Logistica: $n=\exp(-((x-pos) / (.477.*wid)).^2); y=(2.*n) ./ (1+n)$	3
Pearson: $y=ones(size(x)) ./ (1+((x-pos) / ((0.5.^2/m).*wid)).^2).^m$	4
Pearson a larghezza fissa.....	37
Pearson con fattore di forma indipendente, m.....	32
Breit-Wigner-Fano.....	15

```

Impulso esponenziale: y=(x-tau2)./tau1.*exp(1-(x-
tau2)./tau1).....9
Funzione alfa: y=(x-spoint)./pos.*exp(1-(x-spoint)./pos);.....19
Sigmoide in salita (funzione logistica): y=.5+.5*erf((x-
tau1)/sqrt(2*tau2))...10
Funzione sigmoidea Down y=.5-.5*erf((x-
tau1)/sqrt(2*tau2)).....23
Triangolare.....21

```

Nota: se si ha un picco Gaussiano o Lorentziano espanso esponenzialmente, si può misurare sia l'altezza, la posizione e la larghezza "dopo l'ampliamento" utilizzando il tasto **P**, sia l'altezza, la posizione e la larghezza "pre-ampliamento", e approssimare la larghezza con l'approssimazione del picco ad un modello Gaussiano o Lorentziano espanso esponenzialmente (profili 5, 8, 36, 31 o 18) utilizzando il tasto **Shift-F**. Le aree saranno le stesse; l'ampliamento non influisce sull'area totale dei picchi.

Approssimazione polinomiale.

Shift-o approssima un polinomio semplice (lineare, quadratico, cubico, ecc.) al segmento nel pannello superiore e visualizza i coefficienti (con potenze discendenti) e il coefficiente di correlazione R^2 .

Salvataggio dei risultati

Per salvare il segnale elaborato su disco come una matrice x,y nel formato mat, si preme il tasto 'o', poi si digita il nome desiderato per il file nel campo "File name" e si preme **Enter** o si clicca su **Save**. Per caricare nell'area di lavoro, si digita "load" seguito dal nome del file digitato. Il segnale elaborato verrà salvato in una matrice chiamata "Output"; per creare i grafici dei dati, si digita "plot(Output(:,1),Output(:,2))".

Altri controlli da tastiera

Il tasto **Shift-G** attiva e disattiva una griglia temporanea sui pannelli inferiore e superiore.

Il tasto **L** attiva e disattiva la modalità Overlay, che mostra il segnale originale con una linea tratteggiata sovrapposta al segnale elaborato attuale, per confrontarli.

Shift-B apre la finestra Figure 2 e disegna il segnale originale nel pannello superiore e quello processato nel pannello inferiore (come mostrato a destra).

Il tasto **Tab** ripristina il segnale originale e le impostazioni del cursore.

Il tasto ";" imposta la regione selezionata a zero (si usa per eliminare artefatti e spike).

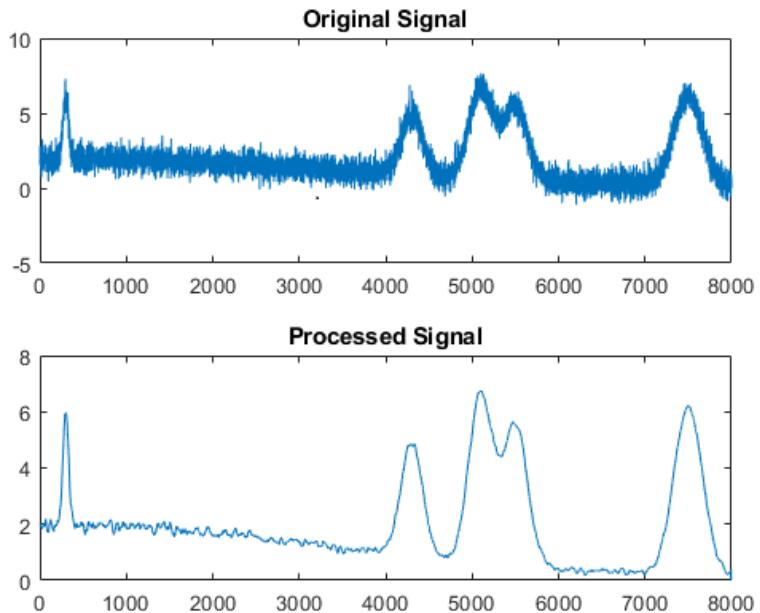
Il tasto "--" (segno meno) viene usato per negare il segnale (capovolge + per -).

Premere **H** per attivare o disattivare la visualizzazione semilog delle y nella finestra inferiore, utile per i segnali con una dinamica molto ampia, come nell'esempio della figura sotto (i punti zero e negativi vengono ignorati nel grafico log).

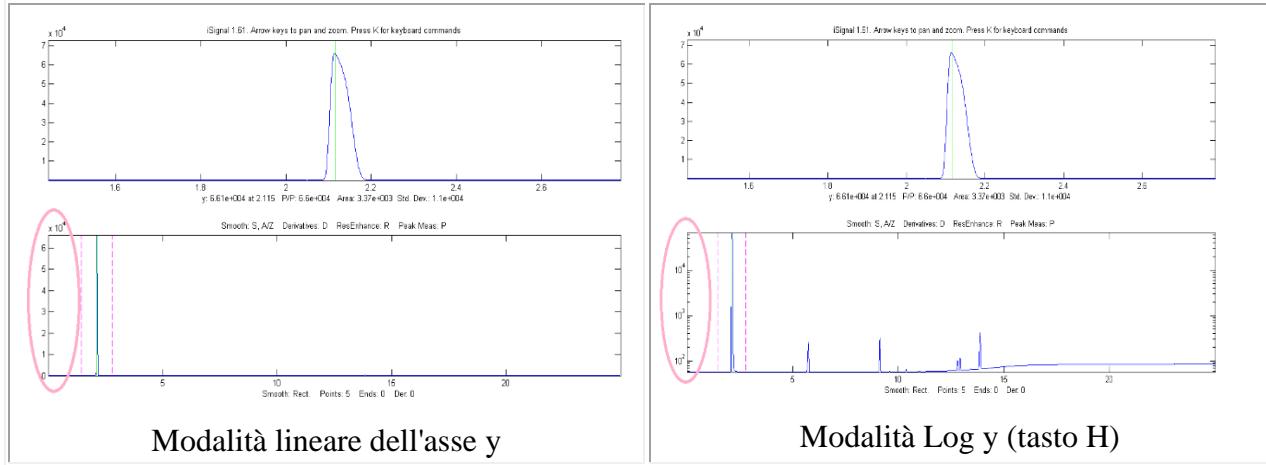
Il tasto '+' prende il valore assoluto di tutto il segnale.

Shift-L sostituisce il segnale con la versione processata di se stesso, per applicarvi ulteriori passaggi di diverse larghezze per lo smoothing o ordini maggiori di differenziazioni.

Il tasto ^ (**Shift-6**) eleva il segnale alla potenza specificata. Per invertirlo, semplicemente si eleva al



reciproco della potenza. Vedere il [Metodo di trasformazione con la potenza](#) dello sharpening a pagina 79.



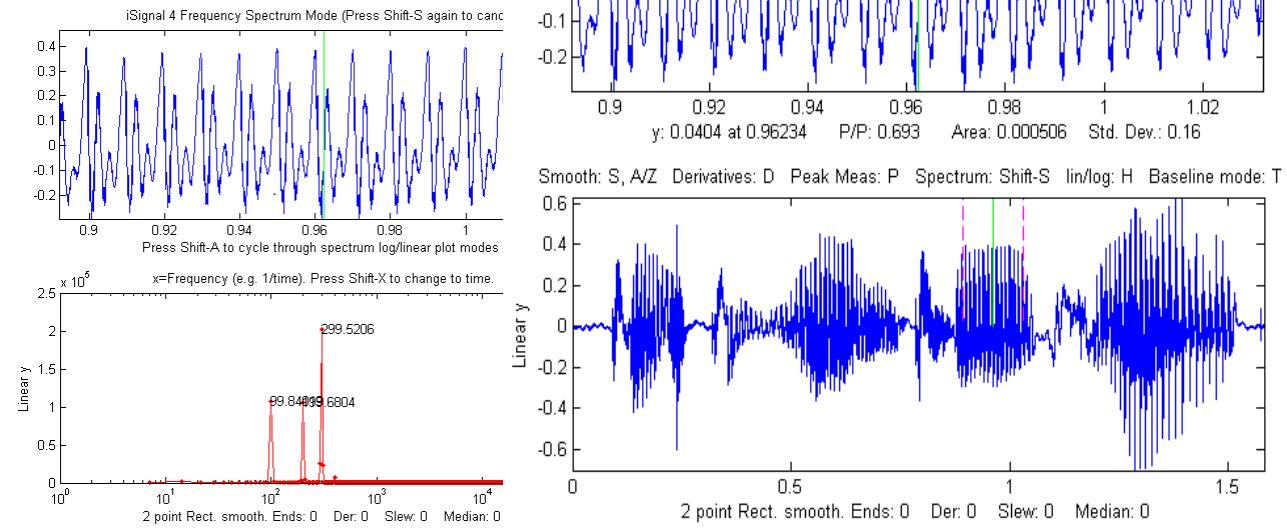
Il tasto **C** condensa il segnale di uno specifico fattore n , sostituendo ogni gruppo di n punti con la loro media (n dev'essere un intero, come 2,3, 4, ecc.). Il tasto **I** sostituisce il segnale con una versione interpolata linearmente contenente m data punti. Questo è utile per aumentare o diminuire l'intervallo sull'asse x del segnale o per uniformare la spaziatura dei valori. Dopo la pressione di **C** o **I**, si deve inserire il valore di n o m rispettivamente. Si può premere **Shift-C**, poi click sul grafico per stampare le coordinate x,y di quel punto. Funziona su entrambe le finestre, e sullo spettro delle frequenze.

Riprodurre i dati come suoni.

Si preme **Spacebar** o **Shift-P** per riprodurre il segmento del segnale visualizzato nella finestra superiore come audio tramite il computer. Si preme **Shift-R** per impostare la frequenza di campionamento - maggiore è il numero più breve ed acuto sarà il suono. Il valore di default è 44000Hz. Suoni o file musicali in formato WAV si possono caricare in Matlab con la funzione nativa "wavread". L'esempio a destra mostra una registrazione audio di 1.5825 secondi della frase "Testing, one, two, three" effettuata a 44000 Hz, salvata in formato WAV ([link](#)), caricata in iSignal e 'zoomata' sul suono "oo" della parola "two". Si preme **Spazio** per riprodurre il suono selezionato; si preme **Shift-S** per mostrare lo spettro delle frequenze (pag. 87) della regione selezionata.

```
>> v=wavread('TestingOneTwoThree.wav');
>> t=0:1/44001:1.5825;
>> isignal(t,v(:,2));
```

Si preme **Shift-Z** per etichettare i picchi nello spettro delle frequenze con le rispettive frequenze. ([a destra](#)). Si preme **Shift-R** e si digita



44000 per impostare la frequenza di campionamento.

Questo esempio di suono registrato consente di sperimentare l'effetto dello smoothing, della differenziazione e dell'interpolazione sul suono del parlato registrato. È interessante notare che diversi gradi di smoothing e differenziazione cambieranno il timbro della voce ma hanno *sorprendentemente scarso effetto sull'intelligibilità*. Ciò è dovuto al fatto che il parlato dipende dalla sequenza delle componenti di frequenza nel segnale, che non viene spostata nel tono o nel tempo, ma semplicemente modificata in ampiezza mediante attenuazione e differenziazione. Anche il calcolo del *valore assoluto* (tasto +), che effettivamente raddoppia la frequenza fondamentale, non rende il suono incomprensibile.

Shift-Ctrl-F trasferisce il segnale corrente a "Interactive Peak Fitter" (ipf.m, pagina 405) e **Shift-Ctrl-P** per trasferire il segnale corrente a "Interactive Peak Detector" (iPeak.m, pagina 246), se queste funzioni sono installate nel path di ricerca di Matlab.

Premere **K** per vedere *tutti* i comandi da tastiera.

ESEMPIO 1: Singolo argomento di input; i dati sono in due colonne di una matrice [x;y] o in un unico vettore y

```
>> isignal(y);  
>> isignal([x;y]);
```

ESEMPIO 2: Due argomenti di input. Dati in vettori x e y separati.

```
>> isignal(x,y);
```

ESEMPIO 3: Tre o quattro argomenti di input. Gli ultimi due argomenti specificano i valori iniziali di pan (xcenter) e zoom (xrangle) negli ultimi due argomenti di input. Utilizzo di dati in un file ZIP:

```
>> load data.mat  
>> isignal(DataMatrix,180,40); or  
>> isignal(x,y,180,40);
```

ESEMPIO 4: Come sopra, ma specifica inoltre i valori iniziali di SmoothMode, SmoothWidth, estremità e DerivativeMode negli ultimi quattro argomenti di input.

```
>> isignal(DataMatrix,180,40,2,9,0,1);
```

ESEMPIO 5: Come sopra, ma specifica inoltre i valori iniziali dei parametri di sharpening Sharpen, Sharp1 e Sharp2 negli ultimi tre argomenti di input. Premere il tasto **E** per attivare e disattivare lo sharpening per il confronto.

```
>>  
isignal(DataMatrix,180,40,4,19,0,0,1,51  
,6000);
```

ESEMPIO 6:

Uso della funzione nativa "humps":

```
>> x=[0:.005:2];y=humps(x);Data=[x;y];
```

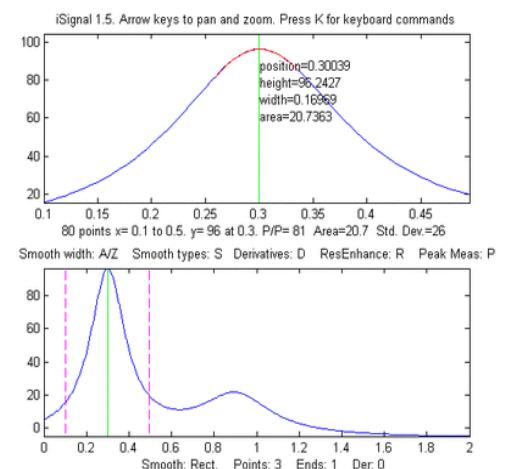
Derivata 4^a del picco a x=0.9:

```
>> isignal(Data,0.9,0.5,1,3,1,4);
```

Questa animazione si può vedere se si scarica la versione [Microsoft Word 365](#), altrimenti cliccare sulle figure.

Lo sharpening applicato al picco in x=0.3:

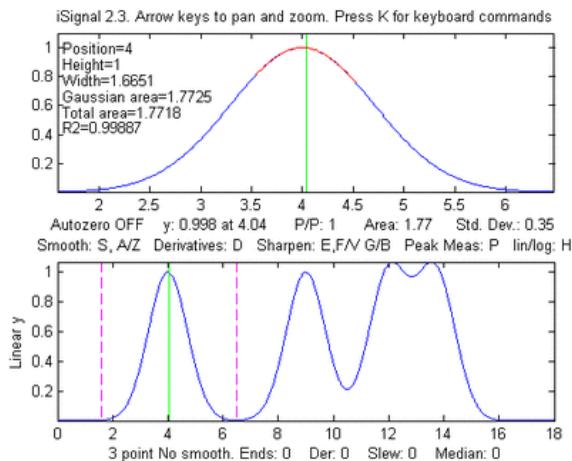
```
isignal(Data,0.3,0.5,1,3,1,0,1,220,5400)
```



) ;

(Premere il tasto 'E' per attivare/disattivare lo sharpening)

ESEMPIO 7: Misura dell'area del picco. Questo esempio genera quattro picchi Gaussiani, tutti con la stessa altezza (1.00) e la stessa area (1.77). Click sulla figura per la GIF animata.

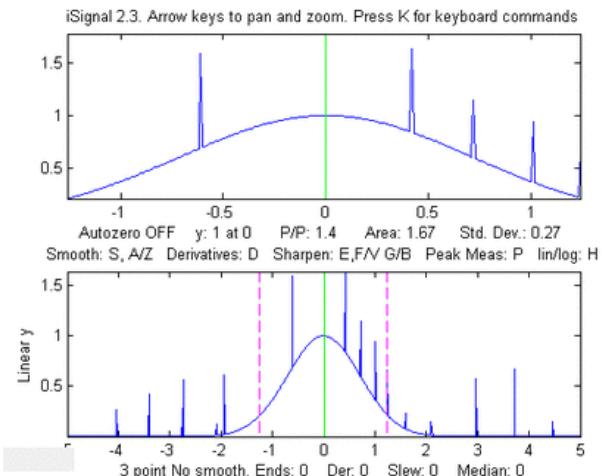


```
>> x=[0 : .01:20];
>> y=exp(-(x-4).^2)+exp(-(x-9).^2)+exp(-(x-13).^2)+exp(-(x-15).^2);
>> isignal(x,y);
```

Il primo picco (in $x=4$) è isolato, il secondo ($x=9$) è leggermente sovrapposto al terzo, e gli ultimi due (in $x=13$ e 15) sono molto accavallati. Per misurare l'area col metodo del taglio verticale (pagina 135), posizionare le linee tratteggiate rosse di demarcazione nel minimo tra i picchi sovrapposti. Una maggiore precisione nella misura dell'area utilizzando iSignal può es-

sere ottenuta con la [funzione di sharpening](#) per ridurre la sovrapposizione tra i picchi. Ciò riduce le larghezze dei picchi, ne aumenta le altezze, ma non ha alcun effetto sulle aree.

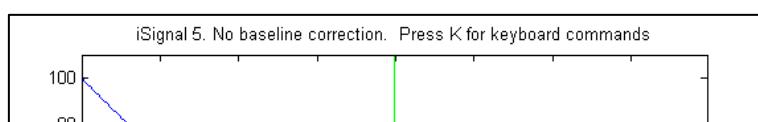
ESEMPIO 8: Picco singolo con picchi casuali (mostrato nella figura a destra). Confrontare lo smoothing rispetto al filtro spike (tasto **M**) e al [limite della velocità di variazione \[slew rate\]](#) (tasto **~**) per rimuovere gli spike.



```
x=-5:.01:5;
y=exp(-(x).^2);
for n=1:1000,
if randn()>2,y(n)=rand()+y(n),
end,
end;
isignal(x,y);
```

ESEMPIO 9: Picchi deboli su una forte linea di base.

Lo script demo [isignaldemo2](#) (mostrato a lato) crea un segnale di prova contenente quattro picchi con altezze 4, 3, 2, 1, con larghezze uguali, sovrapposti, su una linea di base curva molto forte, con in più del rumore bianco casuale. L'obiettivo è quello di estrarre una misura che sia proporzionale alle altezze dei picchi ma indipendente dalla preponderanza della linea di base. Approcci suggeriti: (a) Utilizzare la sottrazione della linea di base automatica o manuale per rimuoverla, misurare i picchi con la misura P-P nel pannello superiore; o (b) utilizzare la differenziazione (con smoothing)



per sopprimere la linea di base; o (c) utilizzare l'approssimazione della curva (**Shift-F**), con la correzione della linea di base (**T**), per misurare l'altezza dei picchi. Dopo l'esecuzione dello script, è possibile premere **Enter** per fare in modo che lo script esegua una calibrazione automatica con la derivata 3^a, eseguita dalla riga 56 alla 74. Come indicato nello script, è possibile modificare molte delle costanti; cercare la parola "change". (Per utilizzare il metodo derivativo, la *larghezza* dei picchi deve essere tutti uguale e stabile, ma le *posizioni* possono variare entro i limiti, impostati dall'Xrange per ciascuna picco nelle righe 61-67). Si devono avere **isignal.m** e **plotit.m** installate.

ESEMPIO 10: Accesso diretto alla modalità spettro di frequenza, disegnando lo spettro di frequenza restituito.

```
>> x=0:.1:60; y=sin(x)+sin(10.*x);
>> [pY, SpectrumOut]=isignal([x;y],30,30,4,3,1,0,0,1,0,0,0,1);
>> plot(SpectrumOut)
```

EXAMPLE 11: Lo script demo [demoisignal.m](#) gira da solo e richiede iSignal 4.2 o successivo e l'ultima versione di [plotit.m](#) installato.

ESEMPIO 12: Ecco un semplice esempio di un segnale molto rumoroso con molto rumore ad alta frequenza (blu) che oscura un picco perfettamente buono al centro a x=150, altezza=1e-4; SNR = 90. Scaricare prima il file dei dati [NoisySignal.mat](#) nel path di ricerca di Matlab, poi eseguire questi comandi:

```
>> load NoisySignal
>> isignal(x,y);
```

Usare i tasti **A** e **Z** per aumentare o diminuire la larghezza dello smoothing, e il tasto **S** per scorrere tra i vari tipi di smoothing disponibili. Suggerimento: usare lo smoothing P-spline e continuare ad aumentare l'ampiezza dello smoothing. Ingrandire il picco a centro, premere **P** per entrare nella modalità picco e verranno mostrate le caratteristiche del picco in alto a sinistra.

iSignal Controlli da tastiera (Versione 8.3):

```
Pan signal left and right...Coarse pan: < and >
Pan fine: tasti freccia sinistra e destra
Nudge: [ and ]
Zoom in and out.....Coarse zoom: / and "
Fine zoom: up and down cursor arrows
Resets pan and zoom.....ESC
Select entire signal.....Ctrl-A
Display grid.....Shift-G temporarily display grid on both panels
Adjust smooth width.....A,Z (A=>more, Z=>less)
Set smooth width vector.....Shift-Q for segmented smooth
Cycle smooth types.....S (No, Rect., Triangle, Gaussian, Savitzky-Golay)
Toggle smooth ends.....X (0=ends zeroed 1=ends smoothed (slower)
Symmetrize (de-tailing) Shift-Y allows entry of symmetrize weighting factor
Adjust Symmetrization.....1,2 keys: decrease, increase by 10%
Shift-1,Shift-2 decrease, increase by 1%
ConV/deconVolution mode.....Shift-V presents menu of conv/deconv choices
Adjust width.....3,4: decrease,increase by 10%
Shift-3,Shift-4: decrease,increase by 1%
Cycle derivative orders.....D/Shift-D Increase/Decrease derivative order
Toggle peak sharpening.....E (0=OFF 1=ON)
Sharpening for Gaussian.....Y Set sharpen settings for Gaussian
Sharpening for Lorentzian...U Set sharpen settings for Lorentzian
Adjust sharp1.....F,V F=>sharper, V=>less sharpening
Adjust sharp2 .....G,B G=>sharper, B=>less sharpening
Slew rate limit (0=OFF).....~ Largest allowed y change between points
Spike filter width (0=OFF) ..m spike filter eliminates sharp spikes
Toggle peak parabola.....P fits parabola to center, labels vertex
```

```

Fit polynomial to segment...Shift-o Asks for polynomial order
Fits peak in upper window...Shift-F (Asks for shape, number of peaks, etc.)
Find peaks in lower panel...J (Asks for Peak Density)
Find peaks in upper panel...Shift-J (Asks for Peak Density)
Spectrum mode on/off.....Shift-S (Shift-A and Shift-X to change axes)
Peak labels on spectrum.....Shift-Z in spectrum mode
Click graph to print x,y....Shift-C Click graph to print coordinates
Display Waterfall spectrum..Shift-W Allows choice of mesh, surf, contour, etc.
Transfer power spectrum.....Shift-T Replaces signal with power spectrum
Lock in current processing..Shift-L Replace signal with processed version
ConVolution/DeconVolution...Shift-V Convolution/Deconvolution menu
Power transform method..... ^ (Shift-6) Raises the signal to a specified power.
Print peak report.....R prints position, height, width, area
Toggle log y mode.....H semilog plot in lower window
Cycles baseline mode.....T none, linear, quadratic, or flat baseline mode
Restores original signal....Tab or Ctrl-Z key resets to original signal and
modes
Toggle overlay mode.....L Overlays original signal as dotted line
Display current signals.....Shift-B Original (top) vs Processed (bottom)
Baseline subtraction.....Backspace, then click baseline at multiple points
Restore background.....\ to cancel previous background subtraction
Invert signal.....Shift-N Invert (negate) the signal (flip + and -)
Remove offset.....0 (zero) set minimum signal to zero
Sets region to zero.....; sets selected region to zero
Absolute value.....+ Computes absolute value of entire signal
Condense signal.....C Condense oversampled signal by factor of N
Interpolate signal.....i Interpolate (resample) to N points
Print report.....Q prints signal info and current settings
Print keyboard commands.....K prints this list of keyboard commands
Print isignal arguments.....W prints isignal function with all current
arguments
Save output to disk.....O Save .mat file with processed signal matrix
Play signal as sound.....Spacebar or Shift-P Play selection through speaker
Play signal as sound.....Shift-R Change sampling rate for playing sound
Expand to full screen.....Double-click figure window title bar
Switch to ipf.m.....Shift-Ctrl-F transfers current signal to
Interactive Peak Fitter, ipf.m
Switch to iPeak.....Shift-Ctrl-P transfers current signal to
Interactive Peak Detector, ipeak.m

```

[ProcessSignal](#), una funzione Matlab/Octave a riga di comando che esegue lo smoothing e la differenziazione su dati temporali x,y (vettori colonna o riga). Digitare "help ProcessSignal". Restituisce il segnale processato come vettore che ha lo stesso profilo di x, indipendentemente da quello di y. La sintassi è

```
Processed=ProcessSignal (x,y,DerivativeMode,w,type,ends,Sharpen,factor1,
factor2,Symsize,Symfactor,SlewRate,MedianWidth)
```

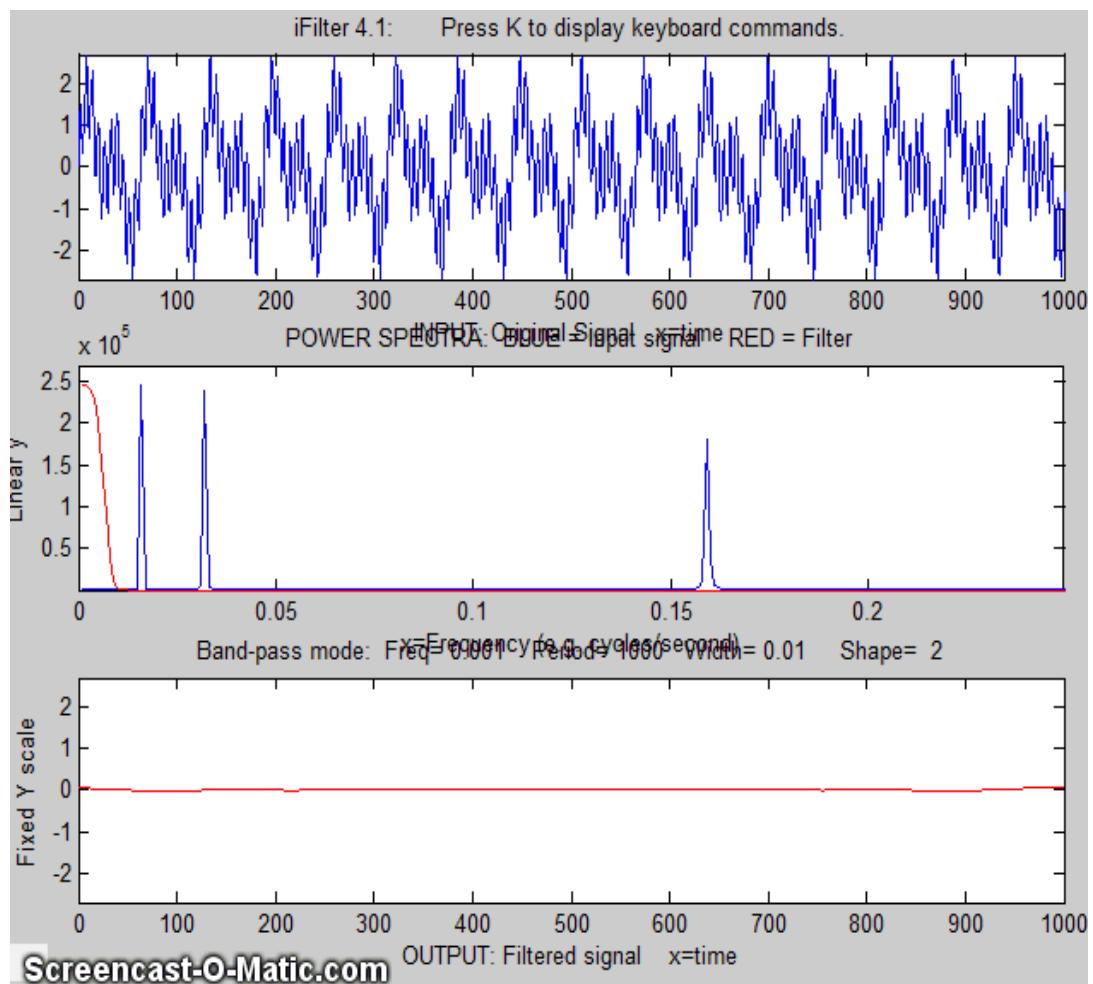
Filtro di Fourier gestito da tastiera

[iFilter.m](#) è una funzione Matlab per un filtro di Fourier interattiva da tastiera per un segnale temporale (x,y), con controlli da tastiera che consentono di regolare continuamente i parametri del filtro mentre, dinamicamente, se ne osserva l'effetto sul segnale. Gli argomenti opzionali di input impostano i valori per la frequenza centrale, la larghezza del filtro, il profilo, la modalità di disegno (1=lineare; 2=frequenza semilog; 3=ampiezza semilog; 4=log-log) e la modalità del filtro ('passa-banda', 'passa-basso', 'passa-alto', 'escludi-banda (notch)', 'comb pass' [pettine passante], e 'comb notch' [pettine notch]). Nelle modalità comb [pettine], il filtro ha più bande localizzate alle frequenze 1, 2, 3, 4... moltiplicato per la frequenza centrale, ciascuno con la stessa (controllabile) larghezza e profilo. L'interattività da tastiera funziona anche se si esegue [Matlab in un browser web](#), ma non con [Matlab Mobile](#). Gli utenti Octave devono utilizzare la versione alternativa [ifilteroctave](#),

che utilizza tasti diversi per il centro del filtro e la regolazione della larghezza e funziona nella versione più recente di Octave.

Il segnale filtrato si può restituire come valore della funzione, salvato come file ".mat" su disco o riprodotto tramite l'audio del computer. Premere **K** per l'elenco dei comandi da tastiera. Questa è una funzione Matlab autonoma che non richiede alcun toolbox o funzioni aggiuntive. Cliccare [qui](#) per visualizzarlo o scaricarlo e inserirlo nel path di ricerca di Matlab. Sulla riga di comando di Matlab, digitare:

```
FilteredSignal=ifilter(x,y) oppure ifilter(y) oppure ifilter(xymatrix) or
```



```
FilteredSignal=ifilter(x,y,center,width,shape,plotmode,filtermode)
```

Esempio 1 Si può vedere questa animazione scaricando la versione [Microsoft Word 365](#), altrimenti cliccare sulla figura. Forma d'onda periodica con 2 componenti di frequenza a 60 e 440 Hz.

```
x=[0:.001:2*pi];
y=sin(60.*x.*2.*pi)+2.*sin(440.*x.*2.*pi);
ifilter(x,y);
```

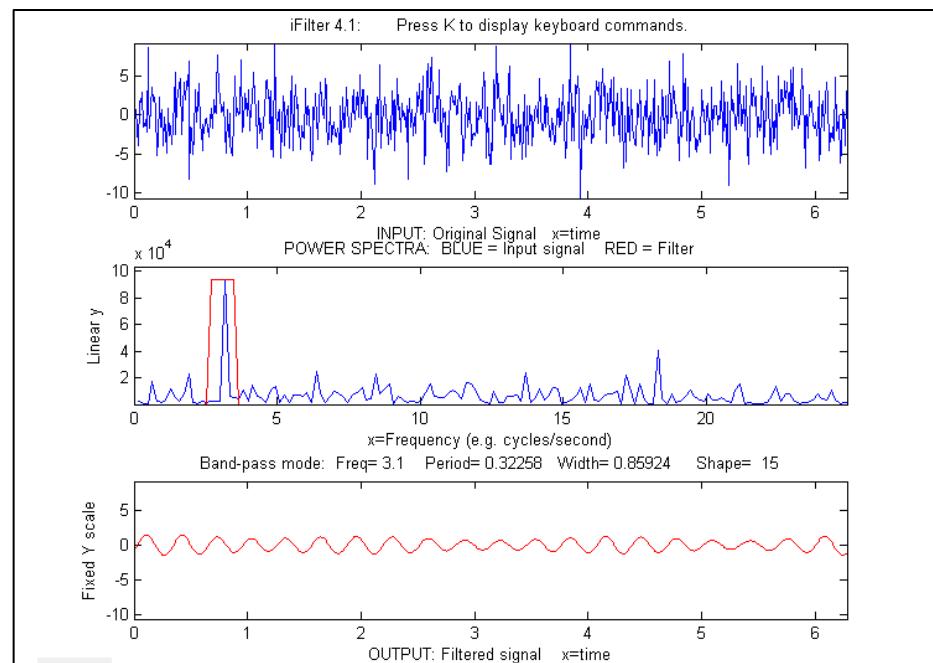
Esempio 2: utilizza argomenti di input opzionali per impostare i valori iniziali:

```
x=0:(1/8000):.3;
y=(1+12/100.*sin(2*47*pi.*x)).*sin(880*pi.*x)+(1+12/100.*sin(2*20*pi.*x))
.*sin(2000*pi.*x);
ry=ifilter(x,y,440,31,18,3,'Band-pass');
```

Esempio 3: Selezione di una frequenza da un'onda sinusoidale rumorosa (mostrato di seguito).

```
x=[0:.01:2*pi]';
y=sin(20*x)+3.*randn(size(x));
```

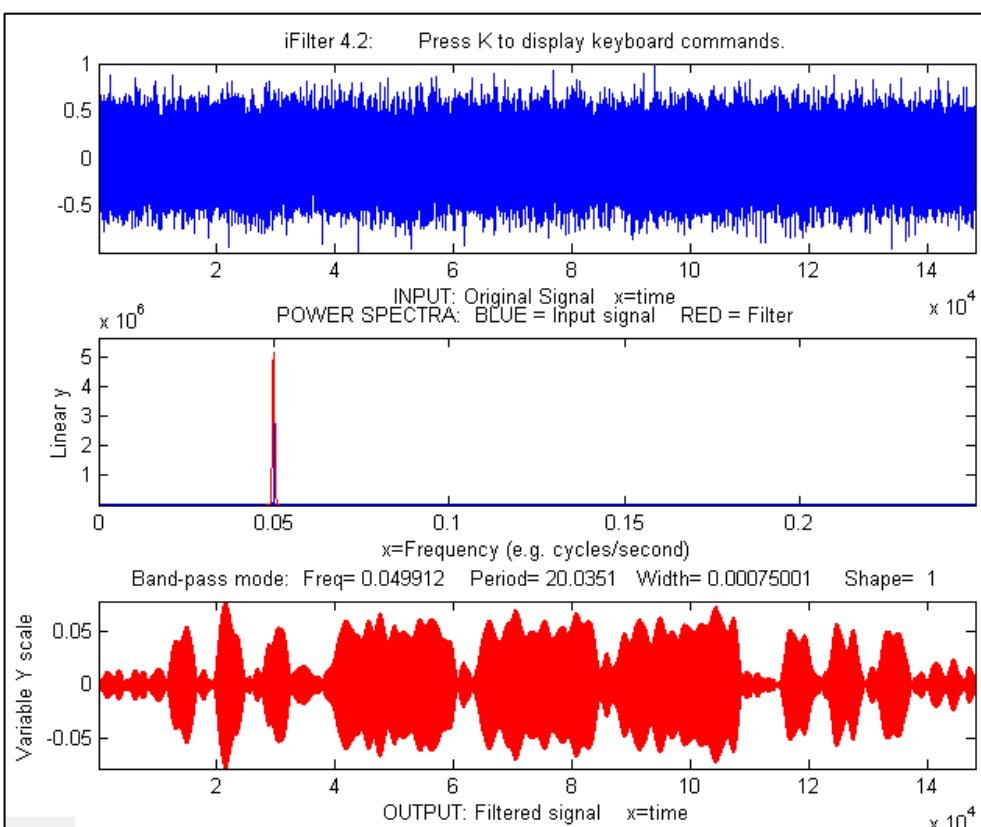
```
ifilter(x,y,3.1,0.85924,15,1,'Band-pass');
```



Esempio 4: Onda quadra con filtro passa banda rispetto al filtro a pettine passante

```
t = 0:.0001:.0625;
y=square(2*pi*64*t);
ifilter(t,y,64,32,12,1,'Band-pass');
ifilter(t,y,48,32,2,1,'Comb pass');
```

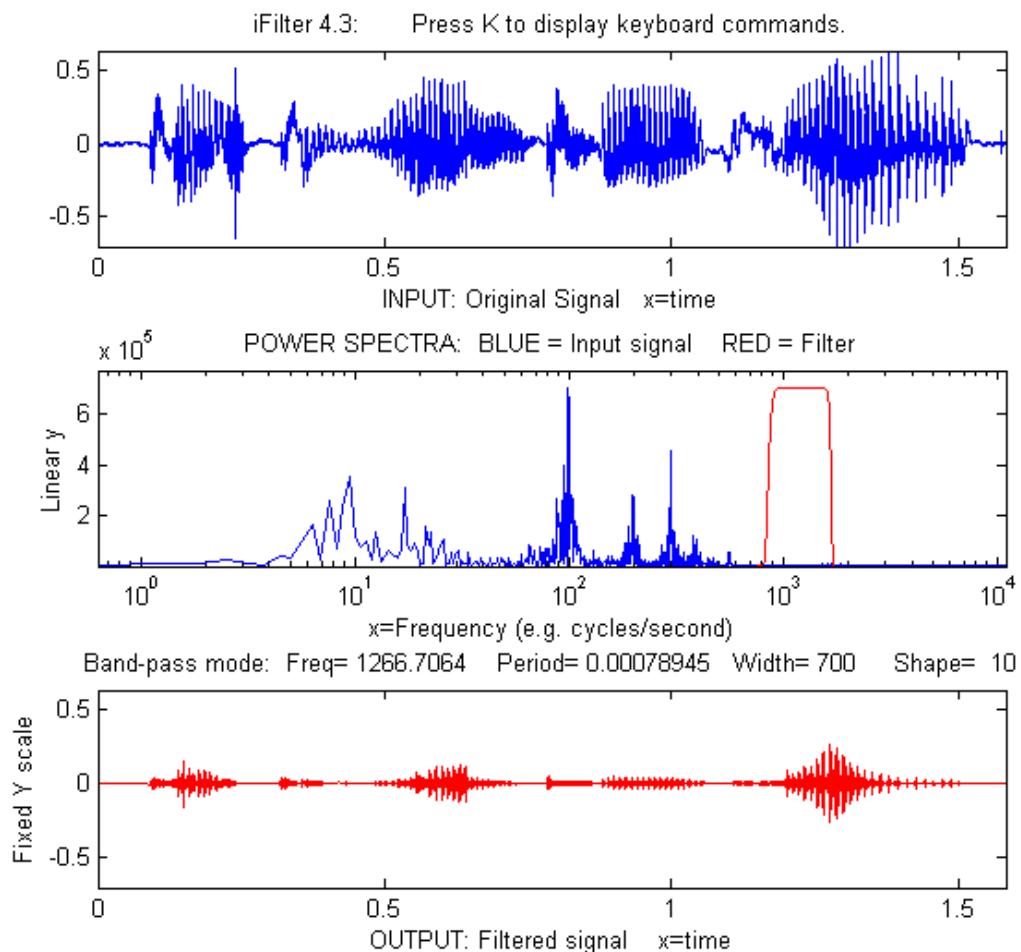
Esempio 5: [MorseCode.m](#) (sopra) uso di iFilter per mostrare le capacità e le limitazioni del filtro di Fourier. Crea, a [frequenza fissa](#), un'onda sinusoidale pulsante [indica “SOS” in codice Morse](#) (dit-dit-dit/dah-dah-dah/dit-dit-dit), poi aggiunge del rumore bianco casuale in modo che l'[SNR sia pessimo](#) (circa 0.1 in questo esempio). Il rumore bianco ha uno spettro di frequenza che è [distribuito su tutta la gamma di frequenze](#); il segnale stesso è concentrato principalmente a una frequenza fissa (0.05) ma [la modulazione dell'onda sinusoidale con gli impulsi del codice Morse](#) ne estende lo spettro su un [ristretto intervallo di frequenze di circa 0.0004](#). Ciò suggerisce che un filtro passa banda di Fourier sintonizzato sulla frequenza del segnale potrebbe essere in grado di isolare il segnale dal rumore. Man mano che la [larghezza di banda si riduce](#), il rapporto segnale-rumore migliora e il



segnale comincia ad emergere dal rumore fino a diventare chiaro, ma se la larghezza di banda è troppo stretta, il tempo di risposta del gradino è troppo lento per fornire dei "dits" e "dahs" distinti. Il tempo di risposta al gradino è inversamente proporzionale alla larghezza di banda. (Usare i tasti ? e " per regolare la larghezza di banda. Premere 'P' o lo Spazio per ascoltare il suono). Si può effettivamente sentire quella componente dell'onda sinusoidale meglio di quanto la si possa vedere nel grafico (pannello superiore), perché l'orecchio funziona come un analizzatore di spettro, con diverse terminazioni nervose assegnate a specifici intervalli di frequenza, mentre l'occhio analizza il grafico spazialmente, osservando l'ampiezza complessiva e non le singole frequenze. [Cliccare per il video mp4 di questo script in azione, col sonoro. Questo video è anche su YouTube:
https://youtu.be/agjs1-mNkmY.](#)

Esempio 6: Questo esempio (grafico nella pagina successiva) mostra una registrazione audio della durata di 1.5825 secondi della frase parlata "Testing, one, two, three", registrata precedentemente a 44001 Hz e salvata sia in formato WAV ([link per il download](#)) che in formato ".mat" ([link per il download](#)). Questo file viene caricato in [iFilter](#), impostato inizialmente in modalità passa-banda e sintonizzato su uno stretto segmento sopra i 1000 Hz che è ben al di sopra della gamma di frequenza della maggior parte del segnale. *Questa banda passante manca della maggior parte delle componenti di frequenza nel segnale*, ma anche in quel caso, il discorso è ancora intelligibile, dimostrando la notevole capacità del sistema orecchio-cervello di accontentarsi di un segnale molto compromesso. Premere P o spazio per ascoltare l'output del filtro sul proprio computer. Diverse impostazioni del filtro cambieranno il [timbro](#) del suono, ma sarà ancora comprensibile.

Nota: per modificare uno qualsiasi degli script demo precedenti di iFilter in Octave, si cambia semplicemente "ifilter" in "ifilteroctave" e ci si assicura che ifilteroctave sia nel path.



iFilter 4.3 Controlli da tastiera della versione Matlab (se la finestra non è in primo piano, cliccare su di essa):

(La [versione Octave](#) utilizza tasti diversi per il centro del filtro e la regolazione della larghezza.)
Adjust filter frequency.....Coarse (10% change) : < and >

```

Fine (1% change): left and right cursor
arrows
Adjust filter width.....Coarse (10% change): / and "
Fine (1% change): up and down cursor arrows
Filter shape.....A,Z (A more rectangular, Z more Gaussian)
Filter mode.....B=bandpass; N or R=notch (band reject);
H=High-pass;
L=Low-pass; C=Comb pass; V=Comb notch.
Select plot mode.....1=linear; 2=semilog frequency
3=semilog amplitude; 4=log-log
Print keyboard commands.....K Prints this list
Print filter parameters.....Q or W Prints ifilter with input
arguments: center, width, shape,
plotmode, filtermode
Print current settings.....T Prints list of current settings
Switch SPECTRUM X-axis scale...X switch between frequency and period
on the horizontal axis
Switch OUTPUT Y-axis scale.....Y switch output plot between fixed or
variable vertical axis.
Play output as sound.....P or Enter
Save output as .mat file.....S

```

Peak Fitter Matlab/Octave

Ho sviluppato un programma Matlab per l'approssimazione di picchi in segnali temporali, che utilizza un [algoritmo di ottimizzazione non lineare](#) non vincolato (pagina 190) per scomporre il segnale di un picco complesso e sovrapposto nelle sue parti componenti. L'obiettivo è quello di determinare se il segnale possa essere rappresentato come somma di profili fondamentali. Accetta segnali di qualsiasi lunghezza, inclusi quelli con valori x non interi e con le x non uniformemente spaziate. Ci sono **due diverse versioni**,

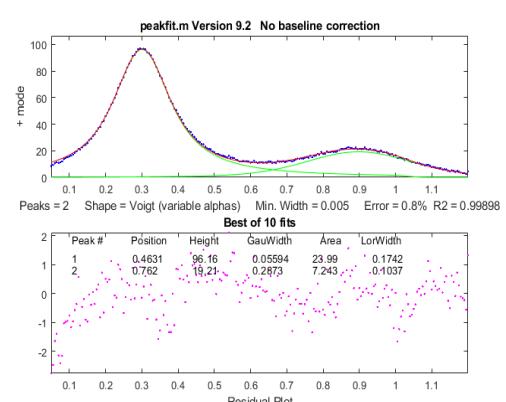
(1) una [versione a riga di comando \(peakfit.m\)](#) per Matlab o Octave, "[Scelta della Settimana](#)" in Matlab File Exchange. La versione corrente è la **9.61**.

(2) una [versione interattiva da tastiera \(ipf.m o ipfoctave.m\)](#), pagina 405. La versione corrente è la 13.4. Lo script demo corrispondente è [Demoipfoctave.m](#)

La differenza tra loro è che peakfit.m è completamente controllato dagli argomenti di input sulla riga di comando e restituisce le sue informazioni tramite gli argomenti di output; ipf.m consente il controllo interattivo tramite i comandi da tastiera. Per automatizzare l'approssimazione di un gran numero di segnali, **peakfit.m** è migliore (vedere pagina 337); ma **ipf.m** è la soluzione migliore per esplorare i segnali per determinare l'intervallo di approssimazione ottimale, i profili dei picchi, il numero, la modalità di correzione della linea di base, ecc. Per il resto hanno caratteristiche simili. I profili di base inclusi sono descritti a pagina 411; si possono aggiungere dei profili personalizzati (vedere pagina 424). Vedere [le pagine 420 e 425](#) per ulteriori informazioni e suggerimenti utili.

Funzione a riga di comando in Matlab/Octave: peakfit.m

[Peakfit.m](#) è una funzione utente per l'approssimazione del picco a riga di comando per Matlab o Octave, utilizzabile da un terminale remoto. È scritta come una funzione autonoma in un singolo m-file. (Per visualizzare o scaricare, cliccare su [peakfit.m](#)). Prende i dati sotto forma di una matrice 2 per n con le variabili indipendenti (valori X) nella riga 1 e quelle dipendenti (valori Y) nella riga 2, o come un singolo vettore della variabile dipendente. La sintassi è [**FitResults, GOF, baseli-**



`sults]=peakfit(signal, center, window, NumPeaks, peakshape, extra, NumTrials, start, BASELINEMODE, fixedparameters, plots, bipolar, minwidth, DELTA, clipheight)).` Solo il primo argomento di input, la matrice di dati, è assolutamente richiesto; ci sono valori predefiniti per tutti gli altri input. Tutti gli argomenti di input e output vengono spiegati di seguito.

La schermata è mostrata a lato; il pannello superiore mostra i dati come punti **blu**, il modello risultante come una linea **rossa** (idealmente sovrapposta ai **punti blu**), e le componenti del modello come linee **verdi**. La linea tratteggiata **magenta** è la prima ipotesi delle posizioni dei picchi per l'approssimazione finale. Il pannello inferiore mostra i residui (differenza tra i dati e il modello).

Si può scaricare un [file ZIP](#) contenente peakfit.m, DemoPeakFit.m, ipf.m, Demoipf.m, alcuni dati di esempio per le prove. Lo script di test [testpeakfit.m](#) o [autotestpeakfit.m](#) esegue tutti gli esempi di peakfit sequenzialmente per verificare il corretto funzionamento sul computer o per la versione di Matlab/Octave. Impiega 25 secondi.

Per una discussione sull'accuratezza e la precisione della misurazione dei parametri di picco utilizzando peakfit.m, cliccare [qui](#) o vedere pag. 159.

È possibile accedere alla funzionalità peakfit.m anche tramite le funzioni interattive da tastiera [**ipf**](#) (pagina 405), [**iPeak**](#) (pagina 405), e [**iSignal**](#) (pagina 366) per *Matlab* o *Matlab Online* o le versioni Octave i cui nomi terminano in “octave”.

Versione 9.62: luglio 2021. Risolto un bug nel profilo Voigt. Le versioni precedenti hanno aggiunto il profilo **50** che implementa il metodo della regressione multilineare ("quadrati minimi classici") per i casi in cui le forme, le posizioni e le larghezze siano *tutte note* e *solo* le altezze dei picchi sono da determinare. Vedere l'**Esempio 40** di seguito e gli script [peakfit9demo.m](#) e [peakfit9demoL.m](#) per una dimostrazione e un confronto tra questo e l'approssimazione iterativa non vincolata.

Peakfit.m può essere chiamato con diversi argomenti aggiuntivi opzionali sulla riga di comando. *Tutti gli argomenti di input (tranne il segnale stesso) possono essere sostituiti da zeri per utilizzare i valori predefiniti.*

```
peakfit(signal);
```

Esegue un'approssimazione dei minimi quadrati iterativa di un singolo picco Gaussiano non vincolato dell'intera matrice "signal", che ha valori x nella riga 1 e valori Y nella riga 2 (p.es. [x y]) o che può essere un singolo vettore di segnale (in tal caso i punti dati vengono disegnati rispetto ai indici sull'asse x).

```
peakfit(signal, center, window);
```

Approssima un singolo picco Gaussiano non vincolato a una porzione della matrice "signal". La porzione è centrata sul valore dell'asse x "center" ed ha la larghezza "window" (in unità x).

In questo e in tutti gli esempi seguenti, impostare "center" e "window" entrambi a 0 per approssimare l'intero segnale.

```
peakfit(signal, center, window, NumPeaks);
```

"NumPeaks" = number of peaks in the model (default is 1 if not specified).

```
peakfit(signal, center, window, NumPeaks, peakshape);
```

Numero o vettore che indica la/e forma/e del picco nel modello: **1**=Gaussiana non vincolata, **2**=Lorentziana non vincolata, **3**=*distribuzione* logistica, **4**=Pearson, **5**=Gaussiana espansa esponenzialmente; **6**=Gaussiane con le stesse larghezze, **7**=Lorentziane con le stesse larghezze, **8**=Gaussiane con le stesse larghezze, espanso esponenzialmente, **9**=impulso esponenziale, **10**=sigmoide in salita (*funzione* logistica), **11**=Gaussiane a larghezza fissa, **12**=Lorentziane a larghezza fissa, **13**=Mix di Gaussiana/Lorentziana; **14**=Gaussiana biforcata, **15**=Risonanza Breit-Wigner-Fano; **16**=Gaussiane in posizioni fisse; **17**=Lorentziane in posizioni fisse; **18**=Lorentziana espansa esponenzialmente; **19**=Funzione alpha; **20**=Profilo Voigt; **21**=Triangolare; **23**=Sigmoide in discesa; **25**=distribuzione lognormale; **26**=linea di base lineare (vedere Esempio 28); **28**=Polinomiale (extra=ordine del polinomio; Esempio 30); **29**=Articolata lineare segmentata (cfr. Esempio 29); **30**=Voigt con alfa indipendentemente variabile; **31**=ExpGaussiana con costante di tempo indipendentemente variabile; **32**=Pearson indipendentemente variabile; **33**=Mix Gaussiana/Lorentziana indipendentemente variabile; **34**=Voigt a larghezza fissa; **35**=Mix Gaussiana/Lorentziana a larghezza fissa; **36**=Gaussiana espansa esponenzialmente a larghezza fissa; **37**=Pearson a larghezza fissa; **38**=ExpLorentzian con costante di tempo indipendentemente variabile (cfr. Esempio 39); **40**=Onda sinusoidale; **41**=Rettangolo; **42**=Gaussiana appiattita; **43**=Funzione Gompertz (3 parametri logistici: $Bo \cdot \exp(-\exp((Kh \cdot \exp(1) / Bo) * (L-t) + 1))$); **44**= $1 - \exp(-k \cdot x)$; **45**: Quattro parametri logistici $y = maxy \cdot (1 + (miny - 1) / (1 + (x / ip)^{\text{slope}}))$; **46**=Linea di base quadratica (Cfr. Esempio 38); **47**=Emissione del corpo nero; **48**=Impulso esponenziale a larghezza uguale; **49**=Pearson IV; **50**=regressione multilineare (posizioni e larghezze dei picchi note). La funzione [ShapeDemo](#) mostra la maggior parte dei profili basilari dei picchi (grafico a pagina 411) mostrando i picchi con forma variabile su più righe.

Nota 1: "non vincolato" significa semplicemente che la posizione, l'altezza e la larghezza di ciascun picco nel modello possono variare indipendentemente dagli altri picchi, a differenza delle varianti di larghezza uguale, larghezza fissa e posizione fissa. I profili 4, 5, 13, 14, 15, 18, 20 e 34-37 sono vincolati alla stessa *costante di forma*; i profili 30-33 sono completamente svincolati dalla posizione, dalla larghezza e forma; le loro variabili di forma vengono determinate dall'iterazione.

Nota 2: Il valore del profilo costante "extra" è 1 se non viene specificato negli argomenti di input.

Nota 3: L'argomento peakshape può essere un *vettore con un profilo diverso per ciascun picco*, p.es. [1 2 1] per tre picchi in una sequenza Gaussiana, Lorentziana, Gaussiana. (L'argomento di input successivo, 'extra', deve essere un vettore della stessa lunghezza di 'peakshape'. Vedere gli **Esempi 24, 25, 28 e 38**, di seguito.)

```
peakfit(signal, center, window, NumPeaks, peakshape, extra)
```

Specifica il valore di 'extra', utilizzato nei profili Pearson, Gaussiana espansa esponenzialmente, Mix Gaussiana/Lorentziana, Gaussiana biforcata e Breit-Wigner-Fano per mettere a punto la forma del picco. Il valore di default di "extra" è 1 se non altrimenti specificato. Nella versione 5, 'extra' può essere un vettore con diversi valori extra per ciascun picco).

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,  
NumTrials);
```

Riavvia il processo di approssimazione "NumTrials" volte *con valori iniziali leggermente diversi* e seleziona quello migliore (quello con l'errore di approssimazione più basso). NumTrials può essere qualsiasi numero intero positivo (il valore di default è 1). In molti casi, NumTrials=1 sarà sufficiente, ma se ciò non fornisce risultati coerenti, aumentare NumTrials fino a quando il risultato non si stabilizza.

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,  
NumTrials, start)
```

Specifica il vettore di prima ipotesi "start" per le posizioni e le larghezze dei picchi, p.es., start=[position1 width1 position2 width2 ...]. Necessario solo per casi difficili, soprattutto quando ci sono molte variabili regolabili. Il vettore iniziale può essere costituito dai valori medi approssimativi basati sulla propria esperienza, oppure può essere calcolato da una precedente approssimazione più semplice, [come in questo esempio](#). Tralasciando "start", impostandolo a zero, il programma genererà i propri valori iniziali approssimativi (che spesso sono abbastanza buoni). Vedere gli esempi 14, 22, 28, 40 e 42 di seguito per situazioni in cui è utile o necessario specificare i valori iniziali.

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,  
NumTrials, start, BaselineMode)
```

Come sopra, ma "BaselineMode" imposta la modalità di correzione della linea di base nell'ultimo argomento: **BaselineMode**=0 (default) che *non* sottrae la linea di base dal segmento di dati. **BaselineMode**=1 interpola una linea di base *lineare* dalle estremità del segmento dei dati e la sottrae dal segnale (assumendo che il picco ritorni alla linea di base alle estremità del segnale); BaselineMode=2, come la modalità 1 tranne per il fatto che calcola una linea di base con una curva *quadratica*; BaselineMode=3 compensa una linea di base *piatta* senza riferimento al segnale stesso (non richiede che il segnale ritorni sulla linea di base alle estremità del segnale, così come le modalità 1 e 2). I coefficienti delle linee di base polinomiali vengono restituiti nel terzo argomento di output "baseline".

```
peakfit(signal,0,0,0,0,0,0,2)
```

Usare degli zeri come segnaposto, negli argomenti di input, per avere i valori di default. In questo caso, **BaselineMode** è impostato a 2, ma tutti gli altri sono i valori di default.

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,
```

```
NumTrials, start, BaselineMode, fixedparameters)
```

'fixedparameters' (10° argomento di input) specifica le larghezze o le posizioni fisse nei profili 11, 12, 16, 17, 34-37, una voce per ciascun picco. Quando si usa il profilo 50 (regressione multilineare), 'fixedparameters' deve essere una matrice che elenca il numero del profilo del picco (colonna 1), la posizione (colonna 2) e la larghezza (colonna 3) per ciascun picco, una riga per picco.

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,
```

```
NumTrials, start, BaselineMode, fixedparameters, plots)
```

'plots' (11° argomento di input regola il disegno grafico: 0=nessun disegno; 1=i grafici vengono disegnati come al solito (default)

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,
```

```
NumTrials, start, BaselineMode, fixedparameters, plots, bipolar)
```

(12° argomento di input) 'bipolar' = 0 vincola le altezze dei picchi ad essere positive; 'bipolar' = 1 consente altezze positive e negative.

```
peakfit(signal, center, window, NumPeaks, peakshape, extra,
```

```
NumTrials, start, BaselineMode, fixedparameters, plots,  
bipolar,minwidth)
```

'minwidth' (13° argomento di input) imposta la larghezza minima consentita del picco.. Il default, se non è specificato, è uguale all'intervallo dell'asse x. Se sono stati scelti i profili multipli, dev'essere un vettore con le larghezze minime, una per ogni picco.

```
peakfit(signal,center, window, NumPeaks, peakshape, extra,  
NumTrials, start, BaselineMode, fixedparameters, plots, bipolar,  
minwidth,DELTA)
```

'DELTA' (14° argomento di input) controlla la variabilità del riavvio quando NumTrials>1. Il default è 1.0. Con valori più alti si ottiene più variabilità. Solo dalla versione 5.8 in poi.

```
[FitResults,FitError]= peakfit(signal, center, window...);
```

Restituisce il vettore FitResults ordinato secondo il numero, la posizione, l'altezza, la larghezza e l'area del picco), e FitError (la differenza percentuale di RMS tra i dati e il modello nel segmento selezionato) dell'approssimazione migliore.

Etichettare la tabella FitResults: Utilizzando la funzione "table", si può visualizzare FitResults in una tabella ordinata con etichette di colonna, utilizzando solo una singola riga di codice:

```
disp(table(FitResults(:,2), FitResults(:,3), FitResults(:,4),
FitResults(:,5), 'VariableNames', {'Position' 'Height' 'FWHM' 'Area'}))
```

Position	Height	FWHM	Area
8.0763	3.8474	10.729	3.4038e-01
20	1	3	3.1934

È possibile aggiungere colonne aggiuntive di FitResults e di VariableNames per quei profili che visualizzano cinque o più risultati, come il profilo Voight:

```
disp(table(FitResults(:,2), FitResults(:,3), FitResults(:,4),
FitResults(:,5), FitResults(:,6), 'VariableNames', {'Position' 'Height'
'GauWidth' 'Area' 'LorWidth'}))
```

Position	Height	GauWidth	Area	LorWidth
0.80012	0.99987	0.30272	0.34744	0.39708
1.2003	0.79806	0.40279	0.27601	0.30012

Calcolo della precisione dei parametri del picco:

```
[FitResults, GOF, baseline, coeff, residuals, xi, yi, BootstrapErrors] = peakfit([x;y], 0, 0, 2, 6, 0, 1, 0, 0, 0);
```

Visualizza le stime degli errori dei parametri con il *metodo bootstrap*. Vedere pagina 162.

Parametri di output opzionali:

1. **FitResults**: una tabella dei parametri del modello, una riga per ogni picco, che elenca il numero, la posizione, l'altezza, la larghezza e l'area del picco (o, per il profilo 28, i coefficienti polinomiali e per il profilo 29, i punti di interruzione dell'asse x).
2. **GOF** ("Goodness of Fit" [Bontà dell'approssimazione]), un vettore a 2 elementi contenente l'errore di approssimazione RMS della migliore approssimazione di prova e l'R-quadro (coefficiente di determinazione).
3. **baseline**: restituisce i coefficienti polinomiali della linea di base interpolata in modalità lineare e quadratica (1 e 2) o il valore della linea di base costante in modalità 'piatta'.
4. **coeff**: Coefficienti per l'approssimazione polinomiale (solo per il profilo 28; per gli altri, coeff=0)
5. **residual**: vettore delle differenze tra i dati e l'approssimazione migliore. Può essere utilizzato per misurare le caratteristiche del *rumore* nel segnale.
6. **xi**: vettore contenente 600 valori x interpolati per i picchi del modello.
7. **yi**: matrice contenente i valori y dei picchi del modello a ogni xi. Digitare `plot(xi, yi(1, :))` per disegnare il picco 1 o `plot(xi, yi)` per disegnare tutti i picchi.
8. **BootstrapErrors**: la presenza di questo innesca le stime bootstrap delle deviazioni standard e degli intervalli interquartili per ciascun parametro di picco di ciascun picco nell'approssimazione (pagina 162).

Esempi

Nota: Lo script di test [testpeakfit.m](#) esegue automaticamente tutti gli esempi seguenti. (Si può effettuare il copia/incolla, o trascinare e rilasciare, uno qualsiasi di questi esempi di codice a riga

singola o multi-riga nell'editor di Matlab o di Octave o nella riga di comando e premere **Invio** per eseguirlo).

Esempio 1. Approssima i dati calcolati x rispetto a y con un singolo modello Gaussiano non vincolato.

```
> x=[0:.1:10];y=exp(-(x-5).^2); peakfit([x' y'])  
ans =  
Peak number Position Height Width Peak area  
1 5 1 1.665 1.7725
```

Esempio 2. Approssima un piccolo insieme di dati y inseriti manualmente a un singolo modello Gaussiano non vincolato.

```
> y=[0 1 2 4 6 7 6 4 2 1 0]; x=1:length(y);  
> peakfit([x;y],length(y)/2,length(y),0,0,0,0,0)  
Peak number Position Height Width Peak area  
1 6.0001 6.9164 4.5213 32.98
```

Esempio 3. Misura di picchi molto rumorosi con rapporto segnale/rumore = 1. (Provare diverse volte).

```
> x=[0:.01:10];y=exp(-(x-5).^2) + randn(size(x)); peakfit([x;y])  
Peak number Peak position Height Width Peak area  
1 5.0951 1.0699 1.6668 1.8984
```

Esempio 4. Approssima un segnale rumoroso con due picchi con un modello Gaussiano doppio non vincolato (NumPeaks=2).

```
> x=[0:.1:10]; y=exp(-(x-5).^2)+.5*exp(-(x-3).^2)  
+ .1*randn(1,length(x));  
> peakfit([x' y'],5,19,2,1,0,1)  
Peak number Position Height Width Peak area  
1 3.0001 0.49489 1.642 0.86504  
2 4.9927 1.0016 1.6597 1.7696
```

Esempio 5. Approssima una porzione della funzione humps, larga 0.7 unità e centrata su x=0,3, con una singola (NumPeaks=1) funzione di Pearson (peakshape=4) con extra=3 (controlla la forma della funzione di Pearson).

```
> x=[0:.005:1];y=humps(x);peakfit([x' y'],.3,.7,1,4,3);
```

Esempio 6. Crea una matrice di dati 'smatrix', approssima una porzione a un modello Gaussiano non vincolato a due picchi, prende la migliore tra 10 prove. Restituisce gli argomenti di output opzionali FitResults e FitError.

```
> x=[0:.005:1]; y=(humps(x)+humps(x-.13)).^3; smatrix=[x' y'];  
> [FitResults,FitError]=peakfit(smatrix,.4,.7,2,1,0,10)  
  
Peak number Position Height Width Peak area  
1 0.4128 3.1114e+008 0.10448 3.4605e+007  
2 0.3161 2.8671e+008 0.098862 3.0174e+007  
FitError = 0.68048
```

Esempio 7. Come sopra, ma specifica la prima ipotesi per la posizione e la larghezza dei due picchi, nell'ordine [posizione1 larghezza1 posizione2 larghezza2]

```
> peakfit([x' y'],.4,.7,2,1,0,10,[.3 .1 .5 .1]);
```

Fornire una prima ipotesi per posizione e larghezza è utile anche se si ha un picco sopra l'altro (come nell'esempio 4, con entrambi i picchi nella stessa posizione x = 5, ma con larghezze diverse, tra parentesi quadre):

```

>> x=[2:.01:8];
>> y=exp(-((x-5)/.2).^2)+.5.*exp(-(x-5).^2) + .1*randn(1,length(x));
>> peakfit([x' y'],0,0,2,1,0,1,[5 2 5 1])
Peak number Position Height Width Peak area
1 4.9977 0.51229 1.639 0.89377
2 4.9948 1.0017 0.32878 0.35059

```

Esempio 8. Come sopra, restituisce il vettore xi contenente 600 valori x interpolati per il modello dei picchi e la matrice yi contenente i valori y di ciascun modello per ogni xi. Digitare `plot(xi,yi(1,:))` per disegnare il picco 1 o `plot(xi,yi,xi,sum(yi))` per disegnare tutte le componenti del modello e il modello totale (somma delle componenti).

```

> [FitResults, GOF, baseline, coeff, residuals, xi, yi]= ...

peakfit(smatrix,.4,.7,2,1,0,10);

> figure(2); clf; plot(xi,yi,xi,sum(yi))

```

Esempio 9. Approssimazione di una singola Gaussiana non vincolata su un background lineare, utilizzando BaselineMode lineare (9° argomento di input = 1)

```
>> x=[0:.1:10]';y=10-x+exp(-(x-5).^2);peakfit([x y],5,8,0,0,0,0,0,1)
```

Esempio 10. Approssima un gruppo di tre picchi vicini x=2400 in DataMatrix3 con tre Gaussiane espansse esponenzialmente di larghezza uguale.

```

>> load DataMatrix3

>> [FitResults,FitError]= peakfit(DataMatrix3,2400,440,3,8,31,1)

Peak number Position Height Width Peak area
1 2300.5 0.82546 60.535 53.188
2 2400.4 0.48312 60.535 31.131
3 2500.6 0.84799 60.535 54.635

FitError = 0.19975

```

Nota: se i picchi si allungano a *sinistra*, anziché a destra come nell'esempio precedente, basta usare un valore *negativo* per la costante di tempo (in ipf.m, si preme **Shift-X** e si digita un valore negativo).

Esempio 11. Esempio di approssimazione instabile a un segnale costituito da due picchi Gaussiani non vincolati di uguale altezza (1.0). I picchi sono troppo sovrapposti per un'approssimazione stabile, anche se l'errore di approssimazione è piccolo e i residui non sono strutturati. Ogni volta che si rigenera questo segnale, si ottiene un'approssimazione diversa, con le altezze che variano di circa il 15% da segnale a segnale.

```

>> x=[0:.1:10];

>> y=exp(-(x-5.5).^2) + exp(-(x-4.5).^2) + .01*randn(size(x));

```

```

>> [FitResults,FitError]= peakfit([x y],5,19,2,1)

Peak number Position Height Width Peak area

1 4.4059 0.80119 1.6347 1.3941

2 5.3931 1.1606 1.7697 2.1864

FitError = 0.598

```

Risultati molto più stabili possono essere ottenuti utilizzando il modello Gaussiano con uguale larghezza (`peakfit([x y],5,19,2,6)`), ma ciò è giustificato solo se l'esperimento è legittimamente previsto per produrre picchi di uguale larghezza. Vedere pagina 202 - 211.

Esempio 12. Correzione della linea di base. Dimostrazione delle quattro “BaselineModes”, per una singola Gaussiana su un'ampia linea di base, con posizione=10, altezza=1 e larghezza=1.66. BaselineMode è specificato dal 9° argomento di input (che può essere 0,1,2 o 3).

BaselineMode=0 indica di ignorare la linea di base (modalità predefinita). In questo caso, si hanno grandi errori.

```

>> x=8:.05:12;y=1+exp(-(x-10).^2);

>> [FitResults,FitError,baseline]=peakfit([x;y],0,0,1,1,0,1,0,0)

Peak#      Position      Height      Width      Area

1 10 1.8561 3.612 5.7641

FitError =5.387

```

BaselineMode=1 sottrae la linea di base lineare da un estremo all'altro. Non funziona bene in questo caso perché il segnale non ritorna completamente alla linea di base alle estremità.

```

>> [FitResults,FitError,baseline]=peakfit([x;y],0,0,1,1,0,1,0,1)

Peak#      Position      Height      Width      Area

1 9.9984 0.96161 1.5586 1.5914

FitError = 1.9801

baseline = 0.0012608 1.0376

```

BaselineMode=2 sottrae la linea di base quadratico da un estremo all'altro. Non funziona bene in questo caso perché il segnale non ritorna completamente alla linea di base alle estremità.

```

>> [FitResults,FitError,baseline]=peakfit([x;y],0,0,1,1,0,1,0,2)

Peak#      Position      Height      Width      Area

1 9.9996 0.81762 1.4379 1.2501

FitError = 1.8205

```

```
baseline = -0.046619 0.9327 -3.469
```

BaselineMode=3 sottrae automaticamente una linea di base piatta, *senza* richiedere che il segnale ritorni alla linea agli estremi. Questa modalità funziona meglio per questo segnale.

```
>> [FitResults,FitError,baseline]=peakfit([x;y],0,0,1,1,0,1,0,3)
Peak#      Position      Height      Width      Area
1   10   1.0001   1.6653   1.7645
FitError = 0.0037056
baseline = 0.99985
```

In alcuni casi, è possibile considerare la linea di base come un "picco" aggiuntivo. Nell'esempio seguente, la linea di base pende fortemente, ma è diritta. In tal caso il risultato più accurato si ottiene utilizzando un'approssimazione con *due profili*, specificando la forma come un *vettore*, che approssima il picco con una *Gaussiana* (profilo 1) e una linea di base con una *retta inclinata* (**profilo 26**).

```
>> x=8:.05:12;y=x + exp(-(x-10).^2);

>> [FitResults,FitError]=peakfit([x;y],0,0,2,[1 26],[1 1],1,0)

Peak#      Position      Height      Width      Area
1   10   1   1.6651   1.7642
2   4.485   0.22297   0.05   40.045
FitError = 0.093
```

Nell'esempio seguente, la linea di base è *curva*, quindi si potrebbero ottenere buoni risultati con BaselineMode=2:

```
>> x=[0:.1:10]';y=1./(1+x.^2)+exp(-(x-5).^2);

>> [FitResults,FitError,baseline]=peakfit([x y],5,5.5,0,0,0,0,2)

Peak#      Position      Height      Width      Area
1   5.0091   0.97108   1.603   1.6569
FitError = 0.97661

baseline = 0.0014928   -0.038196   0.22735
```

Esempio 13. Come nell'esempio 4, ma con la Gaussiana a *larghezza fissa* (profilo 11), larghezza=1.666. Il 10° argomento di input è un vettore di larghezze fisse (tra parentesi quadre), una voce per ogni picco, che può essere uguale o diversa per ogni picco.

```
>> x=[0:.1:10];y=exp(-(x-5).^2)+.5*exp(-(x-3).^2)+.1*randn(size(x));

>> [FitResults,FitError]=peakfit([x' y'],0,0,2,11,0,0,0,0,[1.666 1.666])

Peak number      Position      Height      Width      Peak area
1   3.9943   0.49537   1.666   0.87849
2   5.9924   0.98612   1.666   1.7488
```

Esempio 14. Misura dell'area del picco. Quattro Gaussiane con un'altezza di 1 e una larghezza di 1.6651. Tutti e quattro i picchi hanno la stessa area teorica (1.772). I quattro picchi possono essere combinati in un'unica operazione di approssimazione utilizzando un modello Gaussiano a 4 picchi, solo con stime approssimative delle prime ipotesi per posizioni e larghezze (tra parentesi quadre). Le aree dei picchi così misurate sono molto più accurate rispetto al metodo del taglio verticale (pagina 135):

```
>> x=[0:.01:18];

>> y=exp(-(x-4).^2)+exp(-(x-9).^2)+exp(-(x-12).^2)+exp(-(x-13.7).^2);

>> peakfit([x;y],0,0,4,1,0,1,[4 2 9 2 12 2 14 2],0,0)

Peak number Position Height Width Peak area

1 4 1 1.6651 1.7725
2 9 1 1.6651 1.7725 ...
3 12 1 1.6651 1.7724
4 13.7 1 1.6651 1.7725
```

Funziona bene anche in presenza di notevoli quantità di rumore casuale:

```
>> x=[0:.01:18]; y=exp(-(x-4).^2)+exp(-(x-9).^2)+exp(-(x-12).^2)+exp(-(x-13.7).^2)+.1.*randn(size(x));

>> peakfit([x;y],0,0,4,1,0,1,[4 2 9 2 12 2 14 2],0,0)

Peak number Position Height Width Peak area

1 4.0086 0.98555 1.6693 1.7513
2 9.0223 1.0007 1.669 1.7779
3 11.997 1.0035 1.6556 1.7685
4 13.701 1.0002 1.6505 1.7573
```

A volte i picchi sperimentali sono influenzati dall'ampliamento esponenziale, che di per sé non modifica le aree, ma sposta le posizioni dei picchi e ne aumenta l'ampiezza, la sovrapposizione e l'assimmetria del picco, come mostrato quando si tenta di approssimare i picchi con Gaussiane. Usando lo stesso segnale del rumore precedente:

```
>> y1=ExpBroaden(y',-50);

>> peakfit([x;y1'],0,0,4,1,50,1,0,0,0)

Peak number Position Height Width Peak area

1 4.4538 0.83851 1.9744 1.7623
2 9.4291 0.8511 1.9084 1.7289
3 12.089 0.59632 1.542 0.97883
```

```
4 13.787 1.0181 2.4016 2.6026
```

Peakfit.m (and ipf.m) hanno un profilo Gaussiano ampliato esponenzialmente (#5) che funziona meglio in quei casi, recuperando le posizioni, le altezze le larghezze e le aree del picco *originale*. (L'aggiunta di un vettore di prima ipotesi come 8° argomento può migliorare l'affidabilità dell'approssimazione in qualche caso).

```
>> y1=ExpBroaden(y',-50);  
  
>> peakfit([x;y1'],0,0,4,5,50,1,[4 2 9 2 12 2 14 2],0,0)  
  
Peak#      Position  Height  Width  Area  
  
1  4   1  1.6651  1.7725  
  
2  9   1  1.6651  1.7725  
  
3  12  1  1.6651  1.7725  
  
4  13.7 0.99999  1.6651  1.7716
```

Un modo semplice per ottenere un buon vettore per la prima ipotesi è eseguire inizialmente un semplice approssimazione Gaussiana e fare in modo che lo script utilizzi i FitResults di tale approssimazione come elementi del vettore di prima ipotesi, [come in questo esempio](#).

Esempio 15. Visualizza una tabella degli errori stimati. Vedere [DemoPeakfitBootstrap](#) per una demo autonoma di questa funzione.

```
>> x=0:.05:9; y=exp(-(x-5).^2)+.5*exp(-(x-3).^2)+.01*randn(1,length(x));  
>> [FitResults, LowestError, baseline, residuals, xi, yi, BootstrapErrors]=  
peakfit([x;y],0,0,2,6,0,1,0,0,0);  
  
Peak #1  Position  Height  Width  Area  
  
Mean:      2.9987  0.49717  1.6657  0.88151  
  
STD:       0.0039508  0.0018756  0.0026267  0.0032657  
  
STD (IQR): 0.0054789  0.0027461  0.0032485  0.0044656  
  
% RSD:    0.13175  0.37726  0.15769  0.37047  
  
% RSD (IQR): 0.13271  0.35234  0.16502  0.35658  
  
  
Peak #2  Position  Height  Width  Area  
  
Mean:      4.9997  0.99466  1.6657  1.7636  
  
STD:       0.001561  0.0014858  0.00262  0.0025372  
  
STD (IQR): 0.002143  0.0023511  0.00324  0.0035296  
  
% RSD:    0.031241  0.14938  0.15769  0.14387
```

```
% STD (IQR) : 0.032875 0.13637 0.16502 0.15014
```

Esempio 16. Approssima entrambi i picchi della funzione Humps con un Mix Gaussiana/Lorentziana (profilo 13) che è al 15% Gaussiana (Extra=15). L'argomento 'Extra' imposta la percentuale della forma Gaussiana.

```
>> x=[0:.005:1];y=humps(x);[FitResults,FitError]=peakfit([x' y'],0.54,0.93,2,13,15,10,0,0,0)
```

Peak#	Position	Height	Width	Area
1	0.30078	190.41	0.19131	23.064
2	0.89788	39.552	0.33448	6.1999

```
FitError = 0.34502
```

Esempio 17. Approssima un picco leggermente asimmetrico con una Gaussiana biforcuta (profilo 14). L'argomento 'Extra' (=45) controlla l'asimmetria del picco (50 è simmetrico).

```
>> x=[0:.1:10];y=exp(-(x-4).^2)+.5*exp(-(x-5).^2)+.01*randn(size(x));[FitResults,FitError]=peakfit([x' y'],0,0,1,14,45,10,0,0,0)
```

Peak#	Position	Height	Width	Area
1	4.2028	1.2315	4.077	2.6723

```
FitError = 0.84461
```

Esempio 18. Restituisce solo gli argomenti di output, senza plottaggio né stampa dalla finestra dei comandi (11° argomento di input=0, il default è 1)

```
>> x=[0:.1:10]';y=exp(-(x-5).^2);FitResults=peakfit([x y],0,0,1,1,0,0,0,0,0)
```

Esempio 19. Come l'esempio 4, ma con una Gaussiana a *posizione fissa* (profilo 16), posizioni=[3 5].

```
>> x=[0:.1:10];y=exp(-(x-5).^2)+.5*exp(-(x-3).^2)+.1*randn(size(x));[FitResults,FitError]=peakfit([x' y'],0,0,2,16,0,0,0,[3 5])
```

Peak number	Position	Height	Width	Peak area
1	3 0.49153	1.6492	0.86285	
2	5 1.0114	1.6589	1.786	

```
FitError = 8.2693
```

Esempio 20. Lorentziana modificata esponenzialmente (profilo 18) con del rumore aggiunto. Come per il profilo 5, peakfit.m recupera la posizione (9), l'altezza (1) e la larghezza (1) del picco originale.

```
>> x=[0:.01:20];L=lorentzian(x,9,1)+0.02.*randn(size(x));L1=ExpBroaden(L',-100);peakfit([x;L1'],0,0,1,18,100,1,0,0)
```

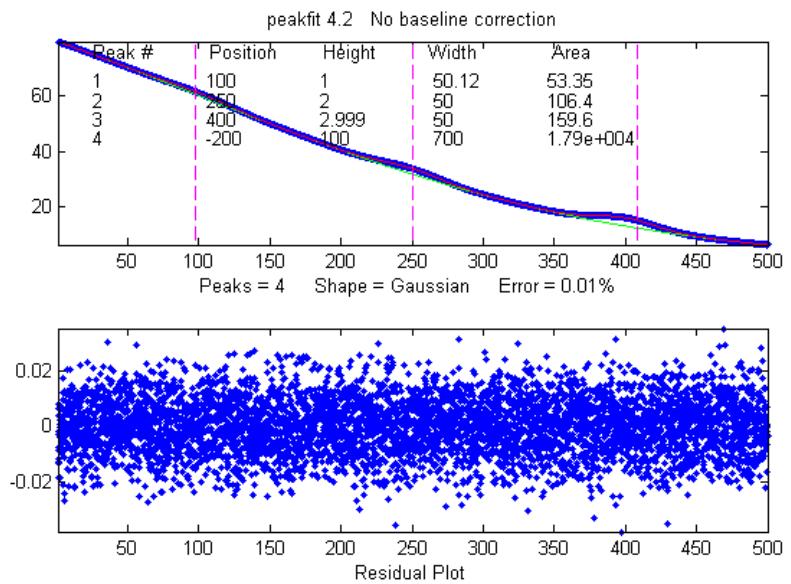
Esempio 21. Funzione di approssimazione 'humps' con due profili Voigt non vincolati (versione 9.5)

```
>>disp('Peak Position Height Width Area Alpha')>> [FitResults,FitError]=peakfit(humps(0:.01:2),60,120,2,30,1.7,5,0)
```

Peak	Position	Height	Width	Area	Alpha
1	31.629	95.175	19.469	2404.5	2.1355

GOF = 0.7618 0.9991

Esempio 22. Misura di tre deboli picchi Gaussiani a $x=100, 250, 400$, sovrapposti su una linea di base curva molto forte con in più del rumore. [peakfitdemob.m](#), illustrato di seguito. La funzione peakfit approssima quattro picchi, trattando la linea di base come un quarto picco la cui posizione è negativa. (Le vere altezze dei picchi sono rispettivamente 1, 2 e 3). Poiché questo si traduce in così tante variabili regolabili (4 picchi x 2 variabili/picco = 8 variabili), è necessario specificare un vettore "start", come nell'Esempio 7. È possibile testare l'affidabilità di questo metodo modificando i parametri nelle righe 11, 12 e 13 e vedere se la funzione peakfit seguirà con successo le modifiche e darà risultati accurati per i tre picchi senza dover cambiare il vettore start. Vedere l'[Esempio 9 su iSignal.html](#) per altri modi per gestire questo segnale.



Esempio 23. 12° argomento di input (modo +/-) impostato a 1 (bipolare) per consentire altezze di picco sia negative che positive. (Default is 0)

```
>> x=[0:.1:10];y=exp(-(x-5).^2)-.5*exp(-(x-3).^2)+.1*randn(size(x));
>> peakfit([x' y'],0,0,2,1,0,1,0,0,0,1,1)
Peak#      Position      Height      Width      Area
1  3.1636   -0.5433    1.62   -0.9369
2  4.9487    0.96859   1.8456   1.9029
FitError = 8.2757
```

Esempio 24. Versione 5 o successive. Approssima la funzione 'humps' a un modello costituito da un picco Lorentziano e uno Gaussiano.

```
>> x=[0:.005:1.2];y=humps(x);
[FitResults,FitError]=peakfit([x' y'],0,0,2,[2 1],[0 0])
Peak#      Position      Height      Width      Area
1  0.30178   97.402    0.18862   25.116
2  0.89615   18.787    0.33676   6.6213
FitError = 1.0744
```

Esempio 25. Cinque picchi, cinque diversi profilo, tutte le altezze = 1, tutte le larghezze = 3, incluso il vettore "extra" per i picchi 4 e 5.

```
x=0:.1:60;
y=modelpeaks2(x,[1 2 3 4 5],[1 1 1 1 1],[10 20 30 40 50],[3 3 3 3 3],[0 0
0 2 -20])+.01*randn(size(x));
peakfit([x' y'],0,0,5,[1 2 3 4 5],[0 0 0 2 -20])
```

Si può usare questa tecnica anche per creare dei modelli tutti della stessa forma ma con diversi valori di 'extra' usando un vettore di valori 'extra', o (nella versione 5.7) con vincoli diversi per le

ampiezze minime usando un vettore di valori 'minwidth' come 13° argomento di input.

Esempio 26. Vincolo di larghezza minima (13° argomento di input)

```
>> x=1:30;y=gaussian(x,15,8)+.05*randn(size(x));
```

Nessun vincolo (minwidth=0):

```
peakfit([x;y],0,0,5,1,0,10,0,0,0,1,0,0);
```

Larghezze vincolate a valori 7 o superiori:

```
peakfit([x;y],0,0,5,1,0,10,0,0,0,1,0,7);
```

Esempio 27. Test del rumore con segnale di picco molto rumoroso: altezza e rumore RMS entrambi uguali a 1.

```
>> x=[-10:.05:10];y=exp(-(x).^2)+randn(size(x));
```

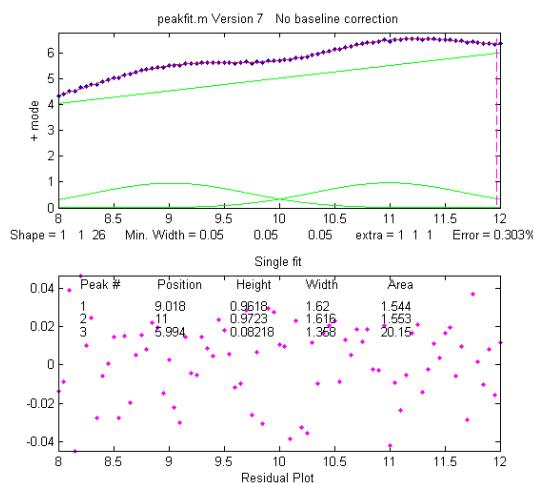
```
>> P=peakfit([x;y],0,0,1,1,0,10);
```

Esempio 28: Picco Gaussiano debole su linea di base retta inclinata, approssimazione con 2 picchi con una Gaussiana ed una linea retta a pendenza variabile ('slope', profilo 26, peakfit versione 6 e successive).

```
>> x=8:.05:12; y=x + exp(-(x-10).^2);
[FitResults,FitError]= peakfit([x;y],0,0,2,[1 26],[1 1],1,0)
Peak#    Position      Height      Width      Area
1    10    1    1.6651    1.7642
2    4.485    0.22297    0.05    40.045
FitError =0.093
```

Per fare un esempio più difficile, questo ha *due* deboli picchi Gaussiani su una linea di base retta inclinata. In questo caso si usa un modello con 3 picchi con peakshape=[1 1 26], che viene aiutato aggiungendo prime ipotesi approssimative ('start') utilizzando la funzione 'polyfit' per generare le prime ipotesi automatiche per la linea di base inclinata. La terza componente (picco 3) è la linea di base.

```
x=8:.05:12
y=x/2+exp(-(x-9).^2)+exp(-(x-
11).^2)+.02.*randn(size(x));
start=[8 1 10 1 polyfit(x,y,1)];
peakfit([x;y],0,0,3,[1 1 26],[1 1 1],1,start)
```



vedere l'esempio 38 (pagina 400) per un caso simile con una linea di base *curva*.

Esempio 29: Approssimazione lineare segmentata (Profilo 29). Si specifica il numero di segmenti nel 4° argomento di input ('NumPoints') e il programma tenta di trovare le *posizioni sull'asse x* ottimali dei punti di interruzione che si riducono al minimo l'errore di approssimazione. Le linee verticali magenta tratteggiate contrassegnano i punti di interruzione sull'asse x. [Un altro esempio con una singola banda Gaussiana](#).

```
>> x=[0.9:.005:1.7];y=humps(x);
>> peakfit([x' y'],0,0,9,29,0,10,0,0,0,1,1)
```

Esempio 30: Approssimazione polinomiale (Profilo 28). Specificare l'ordine del polinomio (qualsiasi numero intero positivo) nel 6° argomento di input ('extra'). (Il 12° argomento di input, 'bipolar', è impostato a 1 per rappresentare l'intero intervallo dell'asse y quando y diventa negativo).

```
>> x=[0.3:.005:1.7];y=humps(x);y=y + cumsum(y);
>> peakfit([x' y'],0,0,1,28,6,10,0,0,0,1,1)
```

Esempio 31: Lo script Matlab/Octave [NumPeaksTest.m](#) usa peakfit.m per mostrare un modo per

determinare il numero minimo di picchi del modello necessari per contenere un insieme di dati, disegnando l'errore di approssimazione rispetto al numero di picchi e cercando il punto in cui l'errore di approssimazione raggiunge un minimo. Questo script crea un segnale rumoroso generato al computer contenente 3, 4, 5 o 6 picchi selezionati dall'utente, approssima una serie di modelli contenenti da 1 a 10 picchi, disegna gli errori di approssimazione rispetto al numero di picchi nel modello e poi determina il vertice della parabola più adatta; il numero intero più vicino è solitamente il numero corretto di picchi. Richiede anche che sia installata la funzione [plotit.m](#).

Esempio 32: Esempi dei profili variabili *non vincolati* 30-33 e il profilo 39, tutte con *tre* variabili iterate (posizione, larghezza e profilo):

- a. **Voigt** (profilo 30). Restituisce le Alfa (rapporti tra le larghezze delle Lorentziane con quelle delle Gaussiane) come 6^a colonna.

```
x=1:.1:30; y=modelpeaks2(x,[13 13],[1 1],[10 20],[3 3],[20 80]);
disp('Peak# Position Height Width Area Alpha')
[FitResults,FitError] = peakfit([x;y],0,0,2,30,2,10).
```

- b. **Gaussiana esponenzialmente allargata** (profilo 31):

```
load DataMatrix3;
peakfit(DataMatrix3, 1860.5,364,2,31,3,5,[1810 60 30 1910 60 30])
```

La versione 8.4 include anche una Gaussiana esponenzialmente allargata alternativa, profilo 39, che è parametrizzata in modo diverso (vedere l'[Esempio 39](#) nella pagina successiva).

- c. **Pearson** (profilo 32)

```
x=1:.1:30;
y=modelpeaks2(x,[4 4],[1 1],[10 20],[5 5],[1 10]);
[FitResults,FitError] = peakfit([x;y],0,0,2,32,0,5)
```

- d. **Mix Gaussiana/Lorentziana** (profilo 33):

```
x=1:.1:30; 0
y=modelpeaks2(x,[13 13],[1 1],[10 20],[3 3],[20 80]);
[FitResults,FitError]=peakfit([x;y],0,0,2,33,0,5)
```

Esempio 34: Uso della funzione nativa "sortrows" per ordinare la tabella FitResults secondo la posizione (colonna 2) o l'altezza del picco (colonna 3).

```
>> x=[0:.005:1.2]; y=humps(x);
>> FitResults,FitError]=peakfit([x' y'],0,0,3,1)
>> sortrows(FitResults,2)
ans =
2 0.29898 56.463 0.14242 8.5601
1 0.30935 39.216 0.36407 14.853
3 0.88381 21.104 0.37227 8.1728
>> sortrows(FitResults,3)
ans =
3 0.88381 21.104 0.37227 8.1728
1 0.30935 39.216 0.36407 14.853
2 0.29898 56.463 0.14242 8.5601
```

Esempio 35: Versione 7.6 o successiva. Uso del Mix Gaussiana/Lorentziana con larghezza fissa (profilo 35).

```
>> x=0:.1:10; y=GL(x,4,3,50)+.5*GL(x,6,3,50) + .1*randn(size(x));
>> [FitResults,FitError]=peakfit([x;y],0,0,2,35,50,1,0,0,[3 3])
peak position height width area
```

```

1 3.9527 1.0048 3 3.5138
2 6.1007 0.5008 3 1.7502
GoodnessOfFit = 6.4783 0.95141

```

Rispetto all'approssimazione con la larghezza variabile (profilo 13), l'errore di approssimazione è maggiore ma i risultati sono più accurati (quando la vera larghezza del picco è nota, width = [3 3]).

```

>> [FitResults,GoodnessOfFit]= peakfit([x;y],0,0,2,13,50,1)
1 4.0632 1.0545 3.2182 3.9242
2 6.2736 0.41234 2.8114 1.3585
GoodnessOfFit = 6.4311 0.95211

```

Nota: per visualizzare la tabella FitResults con le etichette delle colonne, chiamare peakfit.m con gli argomenti di output [FitResults...] e digitare:

```
disp(' Peak number Position Height Width Peak area');disp(FitResults)
```

Esempio 36: Funzione Lorentziana esponenzialmente espansa variabile, profilo 38. (Solo dalla versione 7.7 in poi). FitResults ha in più una sesta colonna per la costante di tempo misurata.

```

>> x=[1:100]';
>> y=explorentzian(x',40,5,-10)+.01*randn(size(x));
>> peakfit([x y],0,0,1,38,0,10)

```

Esempio 37: Logistica a 3 parametri (Gompertz), profilo 43. (Solo dalla versione 7.9 in poi). I parametri etichettati come Bo, Kh, e L. FitResults estesa a 6 colonne.

```

>> t=0:.1:10;
>> Bo=6;Kh=3;L=4;
>> y=Bo*exp(-exp((Kh*exp(1)/Bo)*(L-t)+1))+.1.*randn(size(t));
>> [FitResults,GOF]=peakfit([t;y],0,0,1,43)

```

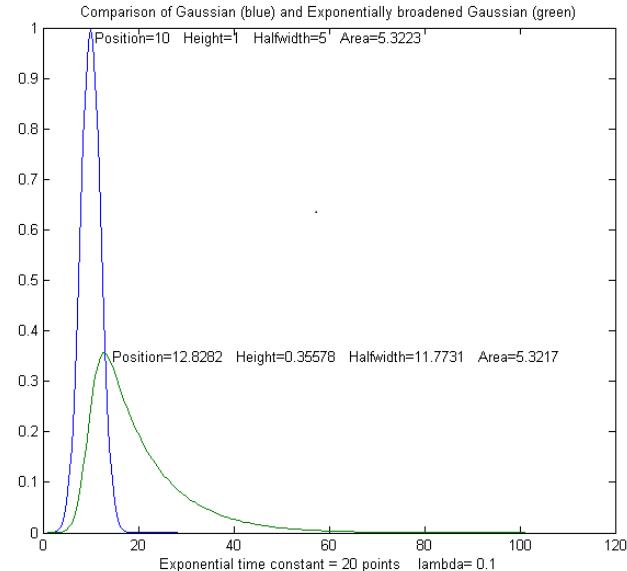
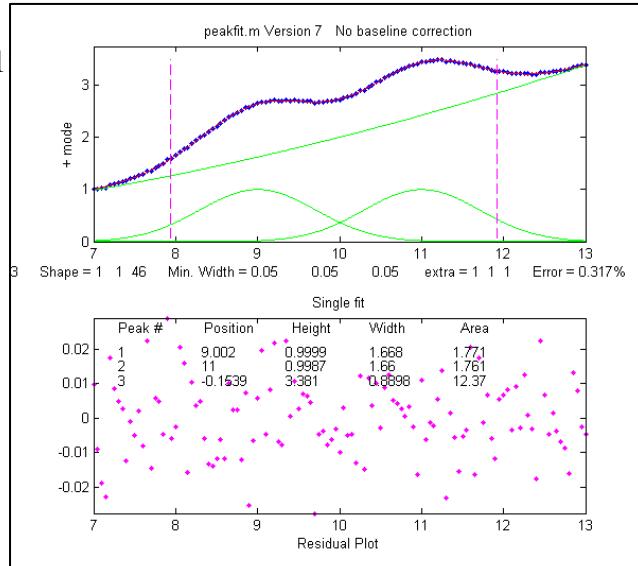
Esempio 38: Profilo 46, 'quadslope'. Due Gaussiane sovrapposte (posizioni=9,11; altezze=1 larghezze=1.666) su una linea di base curva, approssimate con 3 picchi con peakshape=[1 1 46], il NumTrials di default e start.

```

>> x=7:.05:13;
>> y=x.^2/50+exp(-(x-9).^2)+exp(-(x-11).^2)+.01.*randn(size(x));
>> [FitResults,FitError]=
peakfit([x;y],0,0,3,[1 1 46],[1 1 1])

```

La schermata a lato. Nota: se l'ampiezza della linea di base è molto più alta dell'ampiezza del picco, sarà utile fornire un valore iniziale 'start'



approssimativo e utilizzare NumTrials > 1.

Esempio 39: Confronto Gaussiane esponenzialmente espanso alternative senza vincoli, profili 31 e 39. Il profilo 31 ([exgaussian.m](#)) crea la forma eseguendo una convoluzione di Fourier di una Gaussiana specificata mediante un decadimento esponenziale della costante di tempo specificata, mentre la forma 39 ([exgaussian2.m](#)) usa un'espressione matematica per la forma finale così prodotta. Entrambi danno come risultato lo

stesso profilo del picco ma sono parametrizzati in modo diverso. Il profilo 31 riporta l'altezza e la posizione del picco come quella della gaussiana *originale* prima dell'espansione, mentre il profilo 39 riporta l'altezza del picco del *risultato espanso*. Il profilo 31 riporta la larghezza come FWHM (larghezza intera a metà

del massimo) e il 39 riporta la deviazione standard (sigma) della Gaussiana. Il profilo 31 riporta il fattore esponenziale e il *numero dei punti* e il profilo 39 riporta il *reciproco della costante di tempo* in unità di tempo. Vedere lo script [GaussVsExpGauss.m \(a lato\)](#). Vedere le Finestre di Matlab [2](#) e [3](#). Per approssimazioni a picchi multipli, entrambe le forme di solito richiedono un vettore ragionevole di prima ipotesi ('start') per ottenere i migliori risultati.

```
Method Position Height Halfwidth Area Exponential factor
Shape 31 10 1 5 5.3223 20.0001
Shape 39 12.8282 0.35578 11.7731 5.3217 0.1
```

Vedere lo script [DemoExpgaussian.m](#) per una spiegazione più dettagliata.

Esempio 40: Uso del vettore "start" in un'approssimazione con 4 Gaussiane alla funzione "humps"
`x=[-.1:.005:1.2];y=humps(x);`

Il primo tentativo con valori iniziali predefiniti fornisce un'approssimazione scadente che varia da prova a prova:

```
[FitResults,GOF]=peakfit([x;y],0,0,4,1,0,10)
```

Il secondo tentativo di specificare valori approssimativi di "start" nell'8° argomento di input fornisce un'approssimazione molto migliore:

```
start=[0.3 0.13 0.3 0.34 0.63 0.15 0.89 0.35];
[FitResults,GOF]=peakfit([x;y],0,0,4,1,0,10,start)
```

Esempio 41: Peakfit 9 e successive. Uso del profilo 50 ("[regressione multilineare](#)") quando le *posizioni e le larghezze* dei picchi sono note, solo le *altezze* sono sconosciute. Le forme, posizioni e larghezze dei picchi vengono specificate nel 10° argomento di input "fixedparameters", che in questo caso deve essere una *matrice* che elenca il numero del profilo (colonna 1), la posizione (colonna 2) e la larghezza (colonna 3) per ciascun picco, uno per ogni riga. Vedere gli script dimostrativi [peakfit9demo.m](#) e [peakfit9demoL.m](#).

Esempio 42: [RandPeaks.m](#) è uno script che mostra l'accuratezza dell'approssimazione iterativa del picco quando non vengono forniti valori di "start" personalizzati, ovvero conoscendo solo la forma e il numero di picchi. Genera un numero qualsiasi di picchi Gaussiani sovrapposti (NumPeaks nella riga 9) con posizione, altezza e larghezza casuali e chiama la funzione peakfit. Calcola la percentuale media degli errori di posizione, altezza e larghezza. Man mano che si aumenta il numero di picchi, la precisione diminuisce, anche se R² resta prossimo a 1.00.

Come trovare gli argomenti di input corretti per peakfit?

Se non si sa da dove iniziare, si può usare l'[Interattivo Peak Fitter \(ipf.m\)](#) per provare rapidamente diverse regioni di approssimazione, profili, numero di picchi, modalità di correzione della linea di base, numero di prove, ecc. Ottenuta una buona approssimazione, si preme il tasto "W" per stampare l'istruzione della riga di comando per peakfit.m che eseguirà quell'approssimazione in una singola riga di codice, con o senza grafica.

Lavorare con la matrice dei risultati dell'approssimazione "FitResults".

Si supponga di aver eseguito un'approssimazione multi-picco della curva a un insieme di dati, ma si è interessati solo a uno o a pochi picchi specifici. Non è sempre affidabile andare semplicemente per numero di indice del picco (la prima colonna nella tabella FitResults); i picchi a volte cambiano la

loro posizione nella tabella FitResults in modo arbitrario, perché *l'errore di approssimazione è indipendente dall'ordine del picco* (la somma dei picchi 1+2+3 è esattamente la stessa di 2+1+3 o 3+2+1, ecc.). Il problema si può risolvere utilizzando il comando Matlab/Octave "[sortrows](#)" per riordinare la tabella secondo la posizione del picco o della sua altezza. Utile in questi casi anche la funzione [val2ind](#)(v, val), che restituisce l'indice e il valore dell'elemento del vettore 'v' più vicino a 'val' (scaricare questa funzione e porla nel path di ricerca di Matlab). Ad esempio, supponiamo di voler estrarre l'altezza del picco (colonna 3 di FitResults) del picco la cui posizione (colonna 2 di FitResults) più vicina a un valore particolare, chiamiamolo "TargetPosition". Ci sono tre passaggi:

```
VectorOfPositions=FitResults(:,2);
IndexOfClosestPeak=val2ind(VectorOfPositions, TargetPosition);
HeightOfClosestPeak=Fitresults(IndexOfClosestPeak,3);
```

Per un esempio di utilizzo in un'applicazione pratica, vedere [RandomWalkBaseline.m](#).

Un altro modo per utilizzare la matrice FitResults è quello di [calcolare i valori iniziali per ulteriori approssimazioni](#).

Script dimostrativo per peakfit.m

[DemoPeakFit.m](#) è uno script dimostrativo per peakfit.m. Genera un segnale di picco Gaussiano sovrapposto, aggiunge del rumore bianco normalmente distribuito, lo approssima con la funzione [peakfit.m](#) (nella riga 78), lo ripete molte volte ("NumRepeats" nella riga 20), poi confronta i parametri (posizione, altezza, larghezza e area) delle misure con i loro valori effettivi e calcola l'accuratezza (errore percentuale) e la precisione (deviazione standard relativa percentuale). Si può modificare qualsiasi valore iniziale nelle righe 13-30. Ecco un risultato tipico per un segnale a due picchi Gaussiani:

Percentuale degli errori dei parametri misurati:

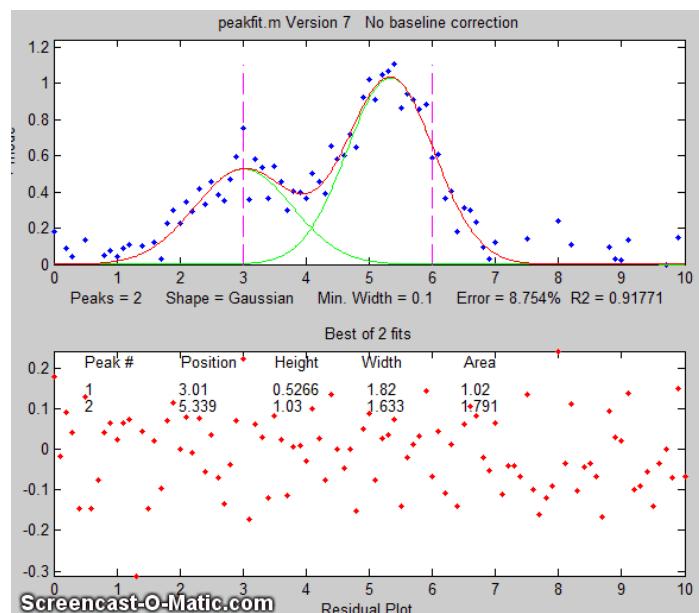
Position	Height	Width	Area
0.048404	0.07906	0.12684	0.19799
0.023986	0.38235	-0.36158	-0.067655

Average Percent Parameter Error for all peaks:

0.036195	0.2307	0.24421	0.13282
----------	--------	---------	---------

In questi risultati, è possibile vedere che l'accuratezza e la precisione (%RSD) della misura della *posizione* del picco sono sempre le migliori, seguite da quelle dell'*altezza*, della *larghezza* e dell'*area*, che di solito sono le peggiori.

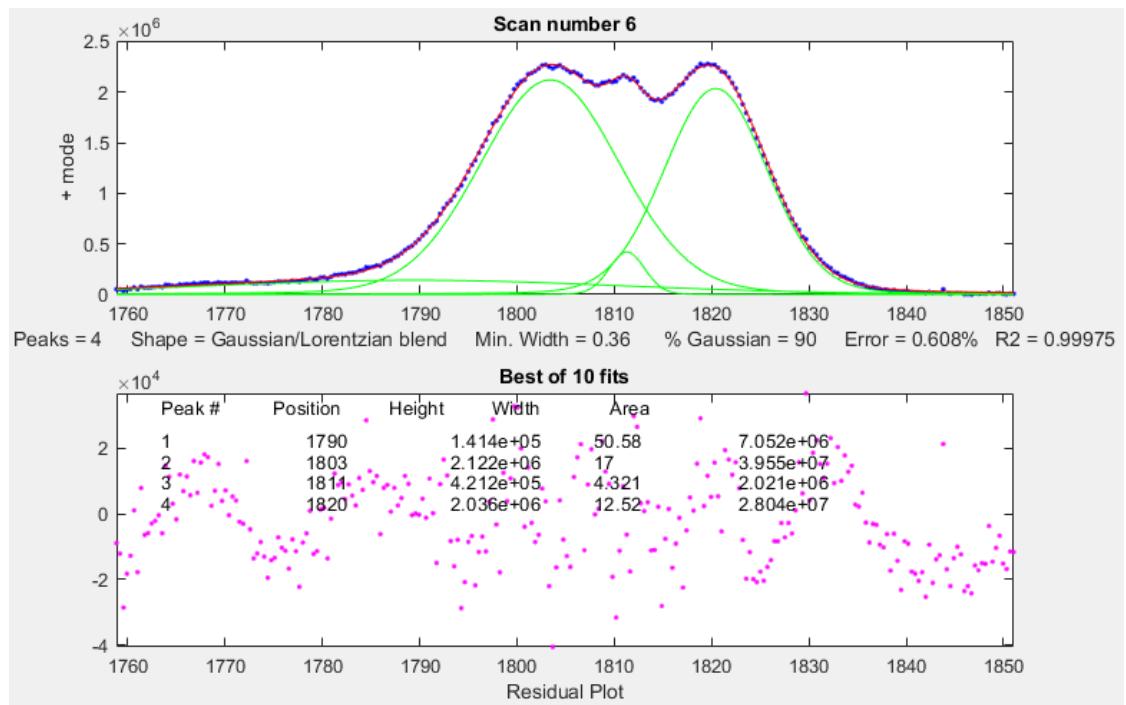
[DemoPeakFitTime.m](#) è una simulazione che mostra come applicare più approssimazione della curva a un segnale che cambia nel tempo. Il segnale contiene due picchi Gaussiani rumorosi (come l'illustrazione a lato) in cui la posizione del *secondo* picco aumenta col tempo e gli altri parametri restano costanti (eccetto il rumore). Lo script crea un set di 100 segnali rumorosi (nella riga 5) contenenti due picchi Gaussiani in cui la posizione del secondo picco cambia nel tempo (da x=6 a 8) e il primo picco rimane lo stesso. Poi approssima un modello con 2 Gaussiane a ciascuno di questi segnali



(nella riga 8), memorizza FitResults in una matrice $100 \times 2 \times 5$, visualizza i segnali e le approssimazioni graficamente con il tempo ([cliccare per riprodurre l'animazione](#)), quindi traccia la posizione misurata dei due picchi rispetto al tempo nella riga 12. Ecco un [esempio di dati reali con un impulso esponenziale](#) variabile nel tempo. Per un esempio di automazione dell'elaborazione di più file memorizzati, vedere pagina 337.

Approssimazione di picchi in dati multi-colonna

Lo script [FittingPeaksInMultiColumnData.m](#) mostra come leggere un set di dati a più colonne da un foglio di calcolo Excel e come utilizzare peakfit.m per approssimare una porzione selezionata del segnale in ciascuna colonna con un modello composto da 4 picchi sovrapposti. I dati, un sottoinsieme dal database NIST InfraRed estratti dallo spettro infrarosso del benzene. I risultati di ciascun sottoinsieme vengono stampati e memorizzati nella matrice PP 4x5x6.



Gestire i segnali complessi con molti picchi

Quando un segnale è costituito da molti picchi su un background altamente variabile, l'approccio migliore è spesso quello di utilizzare peakfit con gli argomenti "center" e "window" per suddividere il segnale in segmenti contenenti gruppi più piccoli di picchi sovrapposti con i loro segmenti di background, isolando i picchi che non si sovrappongono ad altri picchi. Le ragioni di ciò sono diverse:

- (a) peakfit.m funziona meglio se il numero di variabili per ogni approssimazione è ridotto;
- (b) è più facile compensare il background locale sui segmenti più piccoli;
- (c) con approssimazioni più piccole, potrebbe non essere necessario fornire ipotesi iniziali per le posizioni e le larghezze dei picchi;
- (d) si possono facilmente saltare i picchi o le regioni a cui non si è interessati;

(e) In realtà è più veloce per il computer eseguire una serie di comandi `peakfit()` più piccoli rispetto a uno solo che comprende l'intero intervallo di dati in una volta sola.

Un modo semplice per farlo è usare il peak fitter interattivo **ipf.m** (pag. 411) per esplorare vari segmenti del segnale eseguendo pan e zoom e per provare alcune approssimazioni di prova e impostazioni per la correzione della linea di base quindi premere il tasto "**w**" per stampare la sintassi `peakfit` per quel segmento, con tutti i suoi argomenti di input. Copiare, incollare e modificare la sintassi per ogni segmento come si desidera, quindi incollarla nel codice:

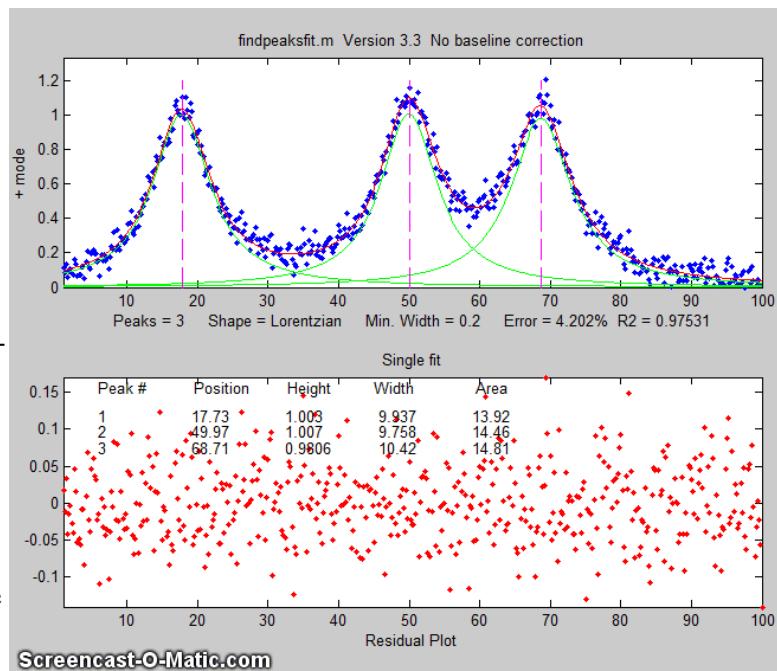
```
[FitResults1, GOF1] = peakfit(datamatrix, center1, window1...
```

```
[FitResults2, GOF2] = peakfit(datamatrix, center2, window2...
```

Assegna i dati della matrice a "datamatrix". Ogni riga utilizzerà gli stessi dati ma con diversi valori di "center" e "window". Anche gli altri argomenti di input (forma del picco, numero di picchi, "extra", numero di tentativi, valori di partenza, correzione della linea di base, ecc.) possono essere diversi se sono stati modificati in `ipf.m`.

Ricerca automatica e Approssimazione dei picchi

findpeaksfit.m è essenzialmente una combinazione di [findpeaks.m](#) e [peakfit.m](#). Utilizza il numero, le posizioni e le larghezze dei picchi trovati, determinate da `findpeaks`, come input per la funzione `peakfit.m`, che quindi approssima l'intero segnale col profilo specificato. Questa combinazione di funzioni è più comoda dell'uso separato di `findpeaks` e `peakfit`. Ottiene valori migliori di `findpeaks`, perché `peakfit` approssima l'intero picco, non solo alla parte superiore, e perché si occupa di picchi non Gaussiani e sovrapposti. Tuttavia, approssima solo quei picchi trovati da `findpeaks`, quindi ci si dovrà assicurare che ogni picco che contribuisce al segnale sia localizzato da `findpeaks`. La sintassi è:

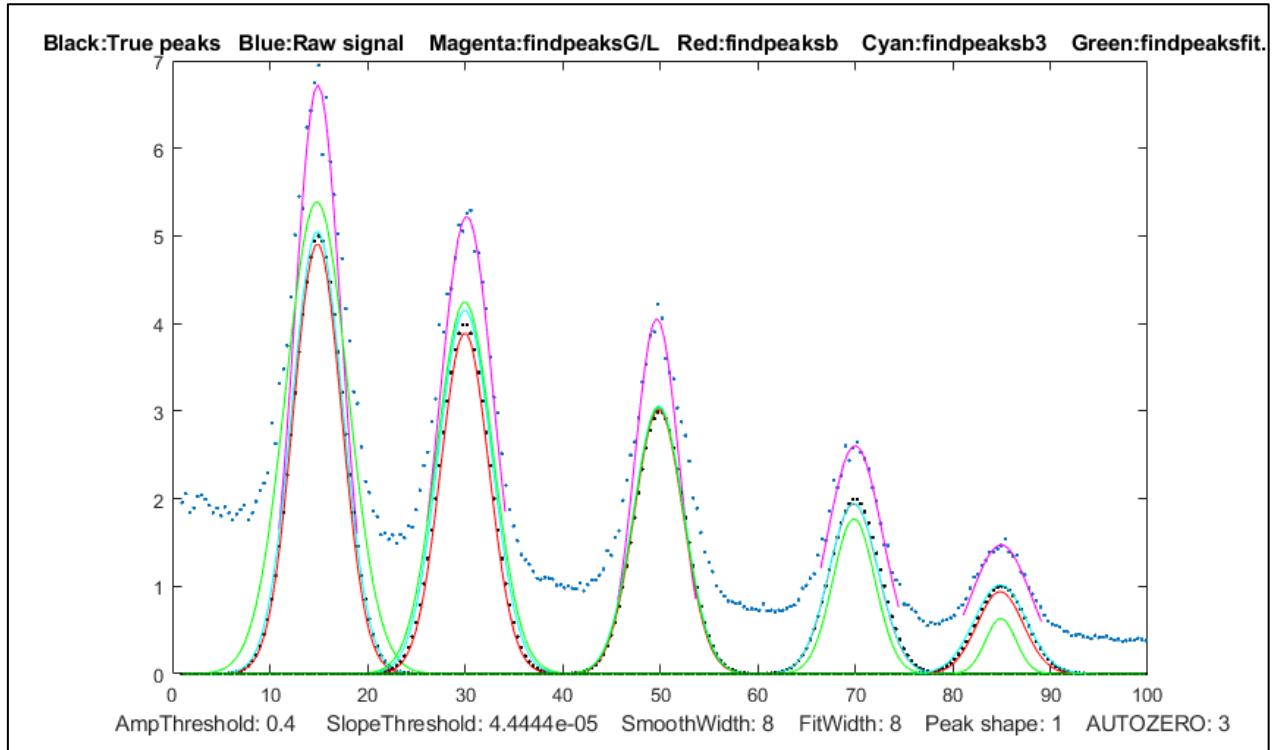


```
function [P,FitResults,LowestError,residuals,xi,yi] = findpeaksfit(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, peakshape, extra, NumTrials, BaselineMode, fixedparameters, plots)
```

I primi sette argomenti di input sono esattamente gli stessi delle funzioni [findpeaks....](#) (pagina 230); se è stata utilizzata `findpeaks` o `iPeak` per cercare e misurare i picchi, si possono utilizzare gli stessi valori per gli argomenti di input per `findpeaksfit.m`. I restanti sei argomenti di `findpeaksfit.m` sono per la funzione `peakfit` (pagina 384); se è stata utilizzata `peakfit.m` o [ipf.m](#) per approssimare i picchi, si possono usare gli stessi argomenti di input per `findpeaksfit.m`. Digitare "help `findpeaksfit`" per ulteriori informazioni. Questa funzione è inclusa nella distribuzione [ipf13.zip](#).

L'[animazione](#) a lato è stata creata con lo script demo [findpeaksfitdemo.m](#). Mostra findpeaksfit che trova e approssima i picchi in 150 segnali in tempo reale. Ciascun segnale ha da 1 a 3 picchi Lorentziani in posizioni variabili.

Lo script [FindpeaksComparison.m](#) confronta l'accuratezza dei parametri di quattro funzioni di rilevamento: [findpeaksG/L](#), [findpeaksb](#), [findpeaksb3](#) e [findpeaksfit](#) applicati a un segnale generato dal computer con più picchi di diversi tipi e quantità variabile di linea di base e rumore casuale. Le ultime tre funzioni comprendono l'approssimazione iterativa equivalente a peakfit.m, in cui il numero dei picchi e i valori iniziali della "prima ipotesi" vengono determinati da findpeaksG/L. Un tipico risultato è mostrato di seguito.

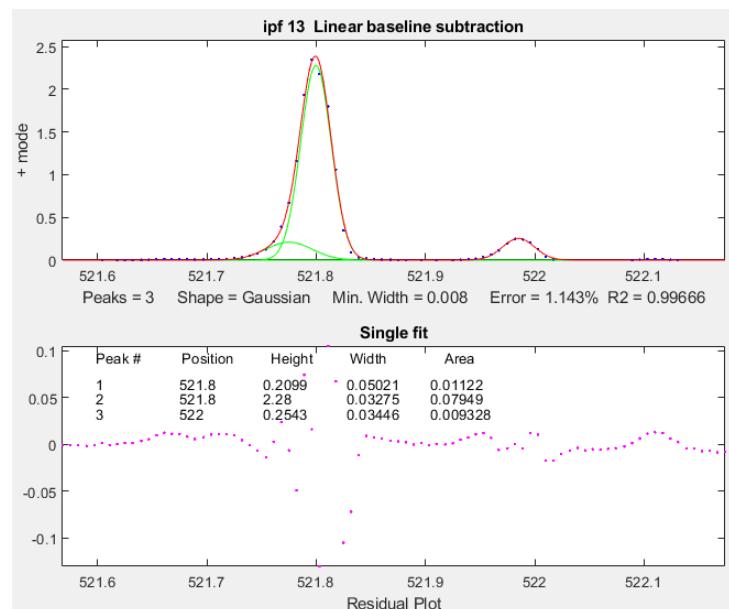


Average absolute percent errors of all peaks

Position error	Height error	Width error	Elapsed time, sec
findpeaksG	0.365331	35.5778	11.6649 0.005768
findpeaksb	0.28246	2.7755	3.4747 0.069061
findpeaksb3	0.28693	2.2531	2.9951 0.49538
findpeaksfit	0.341892	12.7095	18.3436 0.273

Peak Fitter Interattivo Gestito da Tastiera (ipf.m)

[ipf.m](#) per Matlab (Versione 13.3, Settembre 2019), o [ipfoctave.m](#) per Octave, è un "peak fitter" per dati x,y che utilizza comandi da tastiera e il mouse. È una funzione autonoma, in un unico m-file. L'operazione interattiva di pressione dei tasti funziona anche se si esegue [Matlab Online in un browser web](#), ma non funziona in [Matlab Mobile](#) né su iPad e né su iPhone. La flessibile sintassi degli input consente di specificare i dati come vettori x,y separati o come una matrice 2 x n e di



definire facoltativamente il focus iniziale aggiungendo valori “center” e “window” come argomenti di input aggiuntivi, dove 'center' è il valore x desiderato al centro della finestra superiore e “window” è la larghezza desiderata di quella finestra. Doppio-click sulla barra del titolo della figura per vedere meglio a tutto schermo. Esempi:

1 argomento di input:

```
ipf(y) or ipf([x;y]) or ipf([x;y]');
```

2 argomenti di input:

```
ipf(x,y) or ipf([x;y],center) or ipf([x;y]',center);
```

3 argomenti di input:

```
ipf(x,y,center) or ipf(y,center,window) or  
ipf([x;y],center,window) or ipf([x;y]',center,window);
```

4 argomenti di input:

```
ipf(x,y,center,window);
```

Come *iPeak* e *iSignal*, *ipf.m* inizia mostrando l'intero segnale nel pannello inferiore e la regione selezionata che verrà approssimata nel pannello superiore (regolata dagli stessi tasti cursore di *iPeak* e di *iSignal*). Dopo un'approssimazione, il pannello superiore mostra i dati come **punti blu**, il modello totale con una **linea rossa** (idealmente si sovrappone ai punti blu) e le componenti del modello come **linee verdi**. Le **linee magenta** tratteggiate sono le prime posizioni ipotizzate per l'ultima approssimazione. Il pannello inferiore mostra i residui (differenza tra i dati e il modello).

Nota importante: Non cliccare su “Show Plot Tools” nella barra degli strumenti sopra la figura; questo disabiliterà il normale funzionamento del programma. Se lo si fa, si deve chiudere la finestra "Figure" di Matlab e ricominciare. Le istruzioni animate sul suo utilizzo sono disponibili online su <https://terpconnect.umd.edu/~toh/spectrum/ifpinstructions.html>.

Cronologia delle versioni recenti. Versione 13.3: settembre 2019. Aggiunge le modalità di correzione della linea di base 4 e 5. Versione 13.2 mostra "Working..." mentre l'approssimazione è in corso; modificato il tasto "d" per salvare i dati del modello su disco come SavedModel.mat. Versione 13 aggiunge nuovi profili di picco, ora si possono selezionare 24 profili con un solo tasto e 49 dal menù "-". Versione 12.4: Cambiato IQR nel metodo bootstrap in IQR/1.34896, che stima l'RSD senza valori anomali per una distribuzione normale. Versione 11.1 aggiunto il vincolo di larghezza minima del picco (**Shift-W**); aggiunta la saturazione massima (**Shift-M**) per ignorare i punti al di sopra del massimo di saturazione (utile per approssimare i picchi diventati piatti perché hanno raggiunto la saturazione). Versione 11 aggiunta l'approssimazione polinomiale (tasto **Shift-o**). Versione 10.7 corretti dei bug nelle Lorentziane con larghezze uguali (profilo 7) e nella modalità bipolare (+ e -). Versione 10.5, agosto 2014 aggiunti i tasti **Shift-Ctrl-S** e **Shift-Ctrl-P** per trasferire il segnale corrente ad *iSignal* e *iPeak*, rispettivamente, se queste funzioni sono installate nel path di ricerca di Matlab; Versione 10.4, giugno 2014. Sposta il testo risultante dall'approssimazione nel pannello inferiore del grafico. La modalità log: (tasto **M**) attiva/disattiva la modalità di log, approssima il log(modello) a log(y). Sostituisce la Lorentziana biforcuta col picco della risonanza di Breit-Wigner-Fano (tasto **Shift-B**); vedere http://en.wikipedia.org/wiki/Fano_resonance. **Ctrl-A** seleziona tutto; **Shift-N** nega il segnale. Versione 10 aggiunti *modelli a forma multipla*; Versione 9.9 aggiunto il tasto '+=' per passare dalla modalità + (solo picchi positivi) alla bipolare (picchi +/-); Versione 9.8 aggiunto **Shift-C** per specificare la prima ipotesi personalizzata ('start'). Versione 9.7 aggiunto il profilo Voigt. Versione 9.6 inserito un ulteriore BaselineMode che sottrae una linea di base piatta senza richiedere che il segnale ritorni alla linea di base ad entrambe le estremità del segmento di segnale. Versione 9.5 aggiunta la Lorentziana espansa esponenzialmente (profilo 18) e la funzione alfa (profilo 19); Versione 9.4: bug fix per l'altezza del Mix Gaussiana/Lorentziana; Versione 9.3 aggiunto **Shift-S** per salvare la finestra Matlab "Figure" come grafica PNG nella directory corrente. Versione 9.2: correzione di bug; Versione 9.1 aggiunte Gaussiane a posizioni fisse (profilo 16) e Lorentziane a posizioni fisse (profilo 17) e la selezione del profilo da menu

(attivato dal tasto '-' key).

[Demoipf.m](#) è uno script demo per ipf.m, con un generatore di segnali simulati incluso. Per scaricare questi m-file, clic destro su questi link, selezionare **Save Link As...** e cliccare su **Save**. Si può anche scaricare un [file ZIP](#) contenente ipf.m, Demoipf.m e peakfit.m.

Esempio 1: Test con una funzione Gaussiana pura, settaggi di default per tutti gli argomenti di input.

```
>> x=[0:.1:10];y=exp(-(x-5).^2);ipf(x,y)
```

In questo esempio, l'approssimazione è essenzialmente perfetta, indipendentemente dalle impostazioni di pan e zoom o dai valori iniziali della prima ipotesi (start). (Tuttavia, l'area del picco, l'ultimo risultato dell'approssimazione riportata, include *solo l'area all'interno della finestra superiore*, quindi varia). Se ci fosse rumore nei dati o se il modello fosse imperfetto, allora *tutte* i risultati delle approssimazioni dipenderanno dalle esatte impostazioni di pan e zoom.

Esempio 2: Test con "center" e "window" specificati.

```
>> x=[0:.005:1];y=humps(x).^3;  
>> ipf(x,y,0.335,0.39) focus sul primo picco  
>> ipf(x,y,0.91,0.18) focus sul secondo picco
```

Esempio 3: Isola un segmento stretto verso la fine del segnale.

```
>> x=1:.1:1000;y=sin(x).^2;ipf(x,y,843.45,5)
```

Esempio 4: Picco molto rumoroso (SNR=1).

```
x=[0:.01:10];y=exp(-(x-5).^2)+randn(size(x));ipf(x,y,5,10)
```

Premere il tasto **F** per approssimare i dati con un modello Gaussiano.

Premere il tasto **N** più volte per vedere quanta incertezza nei parametri è causata dal rumore.

Esempio 5: 1-4 picchi Gaussiani, nessun rumore, linea di base zero, tutti i parametri sono numeri interi. Illustra l'uso del tasto **X** (la migliore tra 10 approssimazioni) all'aumentare del numero di picchi.

```
Height=[1 2 2 3 3 3 4 4 4 4];  
Position=[10 30 35 50 55 60 80 85 90 95]; Width=[2 3 3 4 4 4 5 5 5 5];  
x=[0:.01:100];y=modelpeaks(x,10,1,Height,Position,Width,0);  
ipf(x,y);
```

ipf controlli da tastiera (Versione 13.4): Ottenuti premendo il tasto **K**

```
Pan signal left and right...Coarse: < and >  
Fine: left and right cursor arrow  
Nudge: [ ]  
Zoom in and out.....Coarse zoom: ?/ and ''  
Fine zoom: up and down arrow keys  
Select entire signal.....Ctrl-A (Zoom all the way out)  
Resets pan and zoom.....ESC  
Select # of peaks.....Number keys 1-9, or press 0 key to  
enter number manually  
Peak shape from menu.....- (minus or hyphen), then type  
number or shape vector and Enter  
Select peak shape by key....g unconstrained Gaussian  
h equal-width Gaussians  
Shift-G fixed-width Gaussians  
Shift-P fixed-position Gaussians  
Shift-H bifurcated Gaussians  
(equal shape, a,z adjust)  
e Exponential-broadened Gaussian  
(equal shape, a,z adjust)
```

```
Shift-R ExpGaussian (var. tau)
j exponential-broadened equal-width Gaussians
(equal shape, a,z adjust)
l unconstrained Lorentzian
:: equal-width Lorentzians
Shift-[ fixed-position Lorentzians
Shift-E Exponential-broadened Lorentzians
(equal shape, a,z adjust)
Shift-L Fixed-width Lorentzians (a,z adjust)
o LOgistic distribution (Use
Sigmoid for logistic function)
p Pearson (a,z keys adjust shape)
Shift-L Pearson IV variable asymmetry
u exponential pUlse
y=exp(-tau1.*x).* (1-exp(-tau2.*x))
Shift-U Alpha: y=(x-tau2)./
tau1.*exp(1-(x-tau2)./tau1)
s Up Sigmoid (logistic function):
y=.5+.5*erf((x-tau1)/sqrt(2*tau2))
Shift-D Down Sigmoid
y=.5-.5*erf((x-tau1)/sqrt(2*tau2))
~` Gauss/Lorentz blend (equal shape,
Shift-V Voigt profile (a/z adjusts
a,z adjust shape)
Shift-B Breit-Wigner-Fano (equal
shape a,z adjust)
Fit.....f
Select BaselineMode .....t selects none, tilted, quadratic,
flat, tilted mode(y), flat mode(y)
+ or +/- peak mode.....+= Flips between + peaks only and
+/- peak mode
Invert (negate) signal.....Shift-N
Toggle log y mode OFF/ON....m Log mode plots and fits
log(model) to log(y).
2-point Baseline.....b, then click left and right baseline
Set manual baseline.....Backspace, then click baseline at
multiple points
Restore original baseline...|\ to cancel previous background subtraction
Cursor start positions.....c, then click on each peak position
Type in start vector.....Shift-C Type or Paste start vector
[p1 w1 p2 w2 ...]
Print current start vector..Shift-Q
Enter value of 'Extra'.....Shift-x, type value (or vector of values
in brackets), press Enter.
Adjust 'Extra' up/down.....a,z: 5% change; upper case A,Z:0.5% change
Print parameters & results..q
Print fit results only.....r
Compute bootstrap stats.....v Estimates standard deViations of
parameters
Fit single bootstrap.....n Extracts and Fits siNgle
bootstrap sub-sample.
Plot signal in figure 2.....y
save model to Disk.....d Save model to Disk as SavedModel.mat.
Refine fit.....x Takes best of 10 trial fits
(change number in line 227 of ipf.m)
Print peakfit function.....w Print peakfit function with all
input arguments
Save Figure as png file....Shift-S Saves Figure window as Figure1.png,
Figure2.png, etc.
Display current settings....Shift-? displays list of current settings
Fit polynomial to segment...Shift-o Asks for polynomial order
Enter minimum width.....Shift-W Constrains peak widths to
a specified minimum.
```

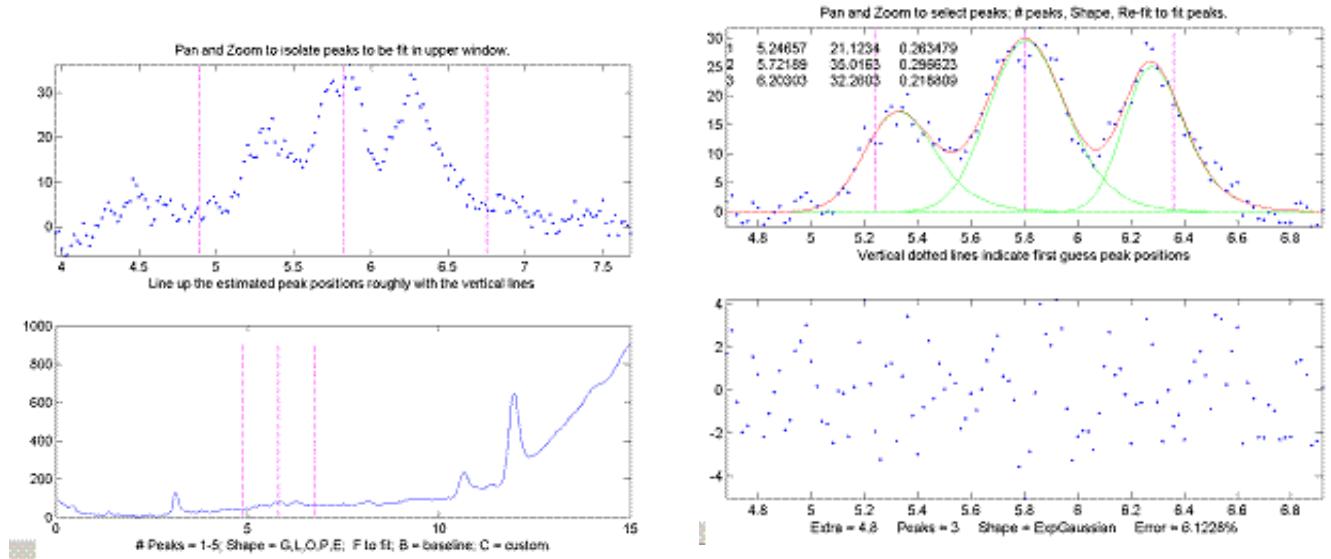
```

Enter saturation maximum....Shift-M Ignores points above a
specified saturation maximum.
Switch to iPeak.....Shift-Ctrl-P transfers current
signal to iPeak.m
Switch to iSignal.....Shift-Ctrl-S transfers current
signal to iSignal.m
(La funzione ShapeDemo illustra le forme dei picchi di base graficamente, mostrando i picchi a pro-
filo variabile come linee multiple; grafico a pagina 411)

```

Esempi pratici con dati sperimentali reali:

1. Approssimazione di picchi cromatografici deboli e rumorosi con Gaussiane modificate esponenzialmente.

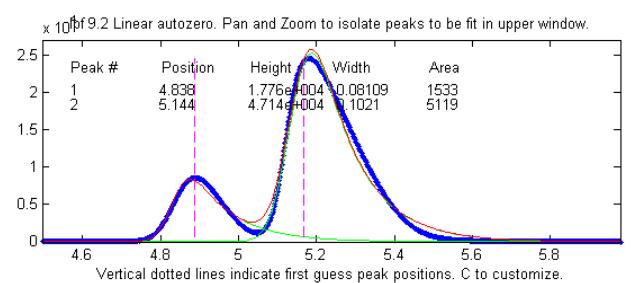


a. In questo esempio di dati reali, i controlli di pan e zoom vengono utilizzati per isolare un segmento di un cromatogramma da 4 a 7,5 minuti che contiene tre picchi molto deboli vicini a 5,8 minuti, su una linea di base sfalsata e leggermente inclinata. Il grafico inferiore mostra l'intero cromatogramma e quello superiore mostra il segmento selezionato. Solo i picchi di quel segmento sono soggetti all'operazione di approssimazione. Pan e zoom vengono regolati in modo che il segnale ritorni alla linea di base locale alle estremità del segmento.

b. Premendo **T** si seleziona Baseline Mode 1, facendo sì che il programma sottragga una linea di base lineare interpolata da questi punti dati. Premendo **3**, **E** si seleziona un modello Gaussiano esponenzialmente ampliato a 3 picchi (una forma di picco comune in cromatografia). Premendo **F** si avvia l'approssimazione. I tasti **A** e **Z** vengono poi utilizzati per regolare la costante di tempo ("Extra") per ottenere la migliore approssimazione. I residui (pannello inferiore) sono casuali e non mostrano una struttura evidente, indicando che l'approssimazione è la migliore possibile con questo livello di rumore. Un'analisi bootstrap degli errori (pag. 166) indica che la deviazione standard relativa delle altezze dei picchi misurate dovrebbe essere inferiore al 3%.

2. Misura delle aree.

Nell'esempio successivo, un campione di aria viene analizzato mediante gascromatografia ([sorgenti dei dati](#)). Il cromatogramma risultante mostra due picchi asimmetrici leggermente sovrapposti, il primo per l'[ossigeno](#) e il secondo per l'[azoto](#). Si presume che



l'area di ciascun picco sia proporzionale alla composizione del gas. Il [metodo del taglio verticale](#) (pagina 135) per misurare le aree fornisce aree di picco in un rapporto del 25% e 75%, rispetto agli effettivi 21% e 78% composizione, che non è molto precisa, forse perché i picchi sono molto asimmetrici. Tuttavia, una Gaussiana ampliata in modo esponenziale (che è una forma di picco comunemente riscontrata in cromatografia) fornisce una buona approssimazione ai dati, utilizzando ipf.m e regolando il termine esponenziale con i tasti A e Z per ottenere l'approssimazione migliore. I risultati per un'approssimazione a due picchi, nella schermata di ipf.m a lato e nel report tasto-R di seguito, mostrano che le aree dei picchi sono in un rapporto del 23% e 77%, il che è leggermente migliore. Con un'[approssimazione a 3-picchi](#), modellando il picco di azoto come somma di *due* picchi molto sovrapposti le cui aree vengono sommate, l'[approssimazione della curva è molto migliore](#) (un errore di approssimazione inferiore all'1%), e in effetti il risultato in questo caso è 21.1% e 78.9% - che è notevolmente più vicino alla composizione effettiva.

Percent Fitting Error =2.9318% Elapsed time = 11 sec.

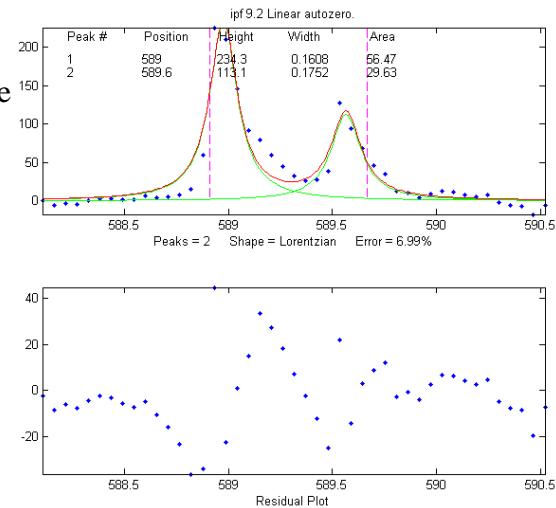
Peak#	Position	Height	Width	Area
1	4.8385	17762	0.081094	1533.2
2	5.1439	47142	0.10205	5119.2

3. L'accuratezza della misura della posizione dei picchi

può essere buona anche se l'errore di approssimazione è piuttosto scarso. In questo esempio, uno spettro di emissione atomica sperimentale ad alta risoluzione viene esaminato nella regione delle [ben note linee spettrali del sodio](#). Qui si trovano due linee (figura a lato), e quando un modello Lorentziano o Gaussiano non vincolato viene approssimato ai dati, le lunghezze d'onda sono determinate a 588.98 nm e a 589.57 nm:

Percent Fitting Error 6.9922%

Peak#	Position	Height	Width	Area
1	588.98	234.34	0.16079	56.47
2	589.57	113.18	0.17509	29.63



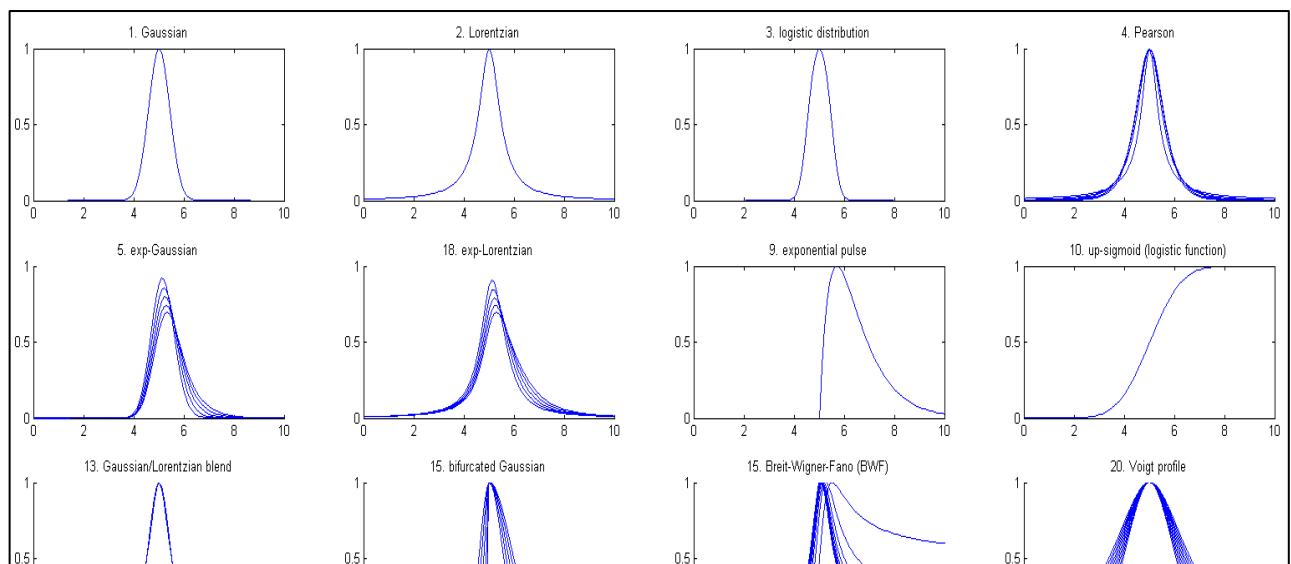
Confrontando con le [lunghezze d'onda raccomandate ASTM](#) per questo elemento (588.995 e 589.59 nm) e vedrà che l'errore non è maggiore di 0.02 nm, che è considerevolmente *inferiore all'intervallo tra i punti dei dati* (0.05 nm). E questo nonostante l'approssimazione non sia particolarmente buona, perché i profili dei picchi hanno una forma piuttosto strana (probabilmente a causa dell'[auto-assorbimento](#), dato che queste particolari righe atomiche sono fortemente *assorbenti* oltre che avere una forte emissione). Questo elevato grado di accuratezza assoluta rispetto a uno standard esterno affidabile è una testimonianza dell'eccellente calibrazione della lunghezza d'onda dello strumento su cui sono stati ottenuti questi dati sperimentali, ma mostra anche che la posizione del picco è di gran lunga il parametro più precisamente misurabile nell'approssimazione del picco, anche quando i dati sono rumorosi e l'approssimazione non è particolarmente buona. La [stima la deviazione standard bootstrap](#) (pagina 162) calcolata da ipf.m per entrambe le lunghezze d'onda è 0.015 nm (vedere #17 nella prossima sezione), e utilizzando la regola pratica della deviazione standard del 2 x avrebbe previsto un probabile errore entro 0.03 nm. (Un *errore di approssimazione* ancora più basso si può ottenere con un'[approssimazione di 4 picchi](#), ma l'*accuratezza della posizione* dei picchi più grandi rimane praticamente invariata).

4. Quanti picchi modellare? Nel secondo e nel terzo esempio sopra, il numero di picchi nel modello è stato suggerito dai dati e dalle aspettative di ciascuno di questi esperimenti (due gas principali nell'aria; il sodio ha una coppia ben nota a quella lunghezza d'onda). Nel primo esempio, *a priori* non c'era nessuna aspettativa a disposizione sul numero dei picchi, ma i dati proponevano tre picchi evidenti, e i residui erano più casuali e non strutturati con un modello a 3 picchi, suggerendo che nessun ulteriore picco era necessario. In molti casi, tuttavia, il numero di picchi del modello non è così chiaramente indicato. In un esempio descritto in precedenza a pagina 214, l'errore di approssimazione continua a diminuire man mano che vengono aggiunti più picchi al modello, tuttavia i residui rimangono "ondeggianti" senza mai diventare casuali. Senza un'ulteriore conoscenza dell'esperimento, è impossibile sapere quali sono i "picchi reali" e cosa sta semplicemente "approssimando il rumore". Ma in certi casi, i dati possono suggerire qualcosa di diverso dalle nozioni preconcette; a volte i dati sono il modo con cui la natura dà un colpetto sulla spalla.

Istruzioni sull'uso di ipf.m (versione 13.4).

Sono disponibili delle istruzioni animate su
<https://terpconnect.umd.edu/~toh/spectrum/ifpinstructions.html>

1. Nella riga di comando, digitare **ipf(x,y)**, (x = variabile indipendente, y = variabile dipendente) oppure **ipf(datamatrix)** dove "datamatrix" è una matrice che ha valori x nella riga o colonna 1 e valori y nella riga o colonna 2. O, se si ha un vettore di segnale y, digitare **ipf(y)**. Facoltativamente, si possono aggiungere altri argomenti numerici:
ipf(x,y,center,window) ; dove 'center' è il valore x desiderato al centro della finestra superiore e "window" è la larghezza di quella finestra.
- 2.
3. Utilizzare i quattro **tasti freccia** sulla tastiera per eseguire il pan e lo zoom del segnale per isolare il picco o il gruppo di picchi che si desidera inserire nella finestra superiore. (*L'operazione di approssimazione della curva si applica solo al segmento del segnale mostrato nel grafico superiore*. Il grafico in basso mostra l'intero segnale. Si cerchi di non avere picchi indesiderati nella finestra superiore altrimenti il programma approssimerà anche quelli. Per selezionare l'intero segnale, premere **Ctrl-A**.
- 4.
5. I tasti numerici (1–9) servono per scegliere il numero di picchi del modello, ovvero il numero



minimo di picchi che si ritiene sia sufficiente per approssimare questo segmento del segnale. Per più di 9 picchi, premere **0**, digitare il numero e premere **Enter**.

6. Selezionare il **profilo del picco** desiderato. In ipf.m versione 13, sono disponibili 24 diversi profili di picco premendo un tasto, ad esempio, **G**=Gaussiano, **L**=Lorentziano, **U**=impulso esponenziale, **S**=sigmoide (funzione logistica), ecc. Premere **K** per visualizzare l'elenco di tutti i comandi. Si può anche selezionare la forma in base al numero da un menù ancora più capiente con 49 profili premendo il tasto - (meno) e selezionando la forma in base al numero. Se si prevede che le larghezze dei picchi di ciascun gruppo siano uguali o quasi, selezionare le forme di "equal-width" [larghezza uguale]. Se le larghezze o le posizioni dei picchi sono note da esperimenti precedenti, selezionare i profili "fixed-width" [larghezza fissa] o "fixed position" [posizione fissa]. [Queste approssimazioni più vincolate](#) sono più veloci, più facili e molto più stabili delle normali approssimazioni tutte variabili, specialmente se il numero di picchi nel modello è maggiore di 3 (perché ci sono meno parametri variabili per il programma da regolare - piuttosto che un valore indipendente per ogni picco).

7.

- 8.** Sul grafico sono presenti delle linee tratteggiate verticali, una per ogni profilo. Provare a regolare finemente con i tasti **Pan** e **Zoom** in modo che il segnale vada sulla linea di base a entrambe le estremità del grafico superiore e in modo che i picchi (o i dossi) nel segnale *approssimativamente* siano allineati tra le linee tratteggiate verticali. Non deve essere precisissimo.

9.

- 10.** Se si desidera consentire picchi negativi e positivi, premere il tasto + per passare alla modalità +/- (indicata dal segno +/- nell'etichetta dell'asse y della parte superiore pannello. Premendolo ancora si torna alla modalità + (solo picchi positivi). In qualsiasi momento si può passare da una modalità all'altra. Per negare l'intero segnale, premere **Shift-N**.

11.

- 12.** **F** per avviare il calcolo dell'approssimazione della curva. Nella versione 13.2, il centro del grafico mostra "Working..." mentre l'approssimazione è in corso. Ogni volta che si preme **F**, viene eseguito un'altra approssimazione del modello selezionato ai dati con posizioni di partenza leggermente diverse, in modo da poter giudicarne la stabilità rispetto ai cambiamenti nelle prime ipotesi iniziali. Tenere d'occhio il grafico dei residui e la visualizzazione di "Error %". [Da fare più volte](#), cercando l'errore più basso e col grafico dei residui casuali il meno strutturato possibile. In qualsiasi momento, è possibile regolare la regione del segnale per approssimarla (passo 2), la posizione della linea di base (passi 9 e 10), modificare il numero dei picchi (passo 3) o il profilo (passo 4), quindi premere nuovamente il tasto **F** per calcolare un'altra approssimazione. Se l'approssimazione sembra instabile, prova a premere il tasto **X** un paio di volte (cfr. #14, di seguito).

13.

- 14.** I parametri del modello dell'ultima approssimazione vengono mostrati nella finestra superiore. Per esempio, per un'approssimazione di 3 picchi:

Peak#	Position	Height	Width	Area
1	5.33329	14.8274	0.262253	4.13361
2	5.80253	26.825	0.326065	9.31117
3	6.27707	22.1461	0.249248	5.87425

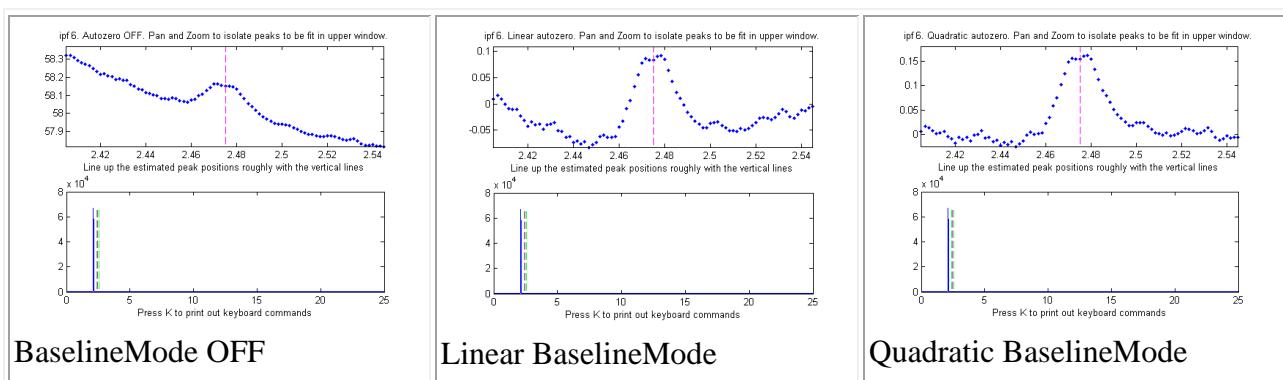
19.

- 20.** Le colonne sono, da sinistra a destra: il numero del picco, la posizione, l'altezza, l'ampiezza e l'area del picco. (Nota: per i profili dell'impulso esponenziale (**U**) e del sigmoide (**S**), Position e Width vengono sostituiti da Tau1 e Tau2). **R** per stampare questa tabella nella finestra di comando. I picchi vengono numerati da sinistra a destra. (L'area di ciascun picco componente all'interno della finestra superiore viene calcolata utilizzando il metodo trapezoidale e visualiz-

zata dopo la larghezza). Premendo **Q** viene stampato un report delle impostazioni e i risultati nella finestra di comando, in questo modo:

```
Peak Shape = Gaussian
Number of peaks = 3
Fitted range = 5 - 6.64
Percent Error = 7.4514 Elapsed time = 0.19 Sec.
Peak# Position Height Width Area
1 5.33329 14.8274 0.262253 4.13361
2 5.80253 26.825 0.326065 9.31117
3 6.27707 22.1461 0.249248 5.87425
```

- 21.** Per selezionare la modalità di correzione della linea di base, premere ripetutamente il tasto **T**; questo scorre 6 modalità di correzione: nessuna, inclinata lineare, quadratica, piatta, inclinata mode(*y*), piatta mode(*y*). Quando la sottrazione della linea di base è lineare, verrà automaticamente sottratta una linea di base retta che collega le due estremità del segmento di segnale nel pannello superiore. Quando la sottrazione della linea di base è quadratica, verrà sottratta automaticamente una linea di base parabolica che collega le due estremità del segmento di segnale nel pannello superiore. Il metodo mode(*y*) sottrae il valore di *y* più comune da tutti i punti nella regione selezionata. Per i segnali in cui i picchi di solito ritornano sulla linea di base tra i picchi, questa è solitamente la linea di base anche se il segnale non ritorna alla linea di base alle estremità come per le modalità 2 e 3 ([esempio grafico](#)). Utilizzare la correzione quadratica della linea di base se è curva, come in questi esempi:



- 22.** Se si preferisce impostare manualmente la linea di base, premere il tasto **B**, poi cliccare sulla linea di base alla SINISTRA del o dei picchi, quindi cliccare sulla linea di base a DESTRA. Verrà sottratta la nuova linea di base e verrà ricalcolata l'approssimazione. (La nuova linea di base rimane in vigore fino a quando non si utilizzano i controlli di pan o zoom). In alternativa, è possibile utilizzare la correzione multi-punto del background per l'intero segnale: premere il tasto **Backspace**, digitare il numero di punti desiderato del background e premere il tasto **Enter**, poi cliccare sulla linea di base iniziando a sinistra del valore x più basso e finendo a destra del valore x più alto. Premere il tasto **** per ripristinare il background precedente per ricominciare da capo.

- 23.**

24. In alcuni casi, sarà utile specificare manualmente la prima ipotesi delle posizioni: premere **C**, poi cliccare sulle posizioni stimate delle di picco nel grafico superiore, una volta per ogni picco. Dopo l'ultimo clic parte automaticamente un'approssimazione. I picchi vengono numerati secondo l'ordine dei click. Per le approssimazioni più difficili, si può digitare **Shift-C** e poi digitare o Incollare l'*intero vettore iniziale*, completo di parentesi quadre, ad esempio, “[pos1 wid1 pos2 wid2 ...]” dove “pos1” è la *posizione* del picco 1, “wid1” è la *larghezza* del picco 1 e così via per ciascun picco. I valori iniziali personalizzati rimangono in vigore finché non si modifica il numero di picchi o si utilizzano i controlli pan o zoom. Suggerimento: se si copia il vettore iniziale e lo si tiene nel buffer, si può usare il tasto Shift-C e incollarlo nuovamente dopo aver modificato i controlli di pan o zoom. Nota: È possibile cliccare oltre l'intervallo dell'asse x, per provare ad approssimare un picco il cui massimo è al di fuori dell'intervallo dell'asse x visualizzato. Questo è utile quando si desidera approssimare una linea di base curva trattandola come un picco aggiuntivo la cui posizione del picco è fuori scala (dati reali [esempio grafico](#)).

25.

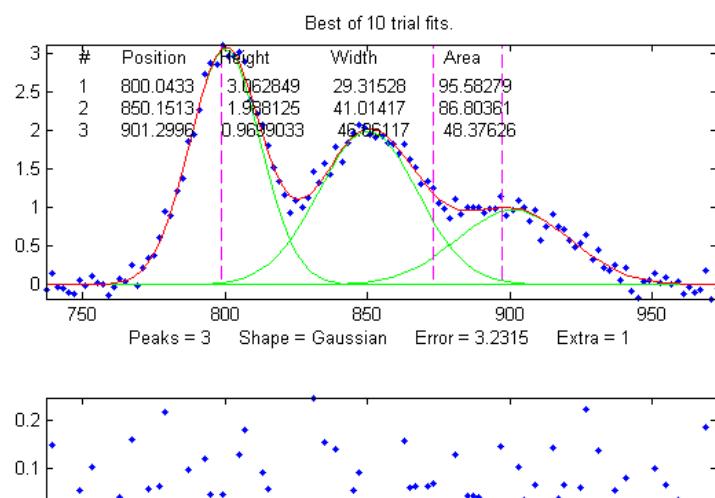
26. I tasti **A** e **Z** controllano il parametro "forma" ('extra') utilizzato solo si stanno usando i modelli a "forma uguale" come i profili Voigt, Pearson, Gaussiana esponenzialmente allargata (Exp-Gaussian), Lorentziana esponenzialmente allargata (ExpLorentzian), Gaussiana biforcata, Breit-Wigner-Fano o mix Gaussiana/Lorentziana. Per questi modelli, le forme sono variabili con i tasti **A** e **Z** ma sono le stesse per tutti i picchi nel modello. Per il profilo Voigt, il parametro "shape" controlla l'*alfa*, il rapporto della larghezza della Lorentziana con quella della Doppler. Per il profilo Pearson, un valore di 1.0 dà un profilo Lorentziano, un valore di 2.0 dà una forma all'incirca a metà strada tra una Lorentziana e una Gaussiana, e valori più grandi danno una forma quasi Gaussiana. Per le forme Gaussiane esponenzialmente ampliate, il parametro "shape" regola la "costante di tempo" esponenziale (espressa come numero di punti). Per il mix Gaussiana/Lorentziana e la Gaussiana biforcata, il parametro "shape" controlla l'asimmetria del picco (un valore di 50 fornisce un picco simmetrico). Per il Breit-Wigner-Fano, controlla il fattore di Fano. È possibile immettere un valore iniziale del parametro "shape" premendo **Shift-X**, digitando un valore e premendo il tasto **Enter**. Per i modelli multi-forme, inserire un vettore di valori "extra", uno per ogni picco, racchiusi tra parentesi quadre. Per i modelli a forma singola, è possibile regolare questo valore utilizzando i tasti **A** e **Z** (tenere premuto il tasto **Shift** per mettere a punto). Si cerchi di ridurre al minimo la % di errore o impostarlo su un valore precedentemente determinato. Nota: se si approssimano più picchi di forma variabile sovrapposti, è più facile approssimare prima un singolo picco, per ottenere un valore approssimativo per il parametro "shape", poi regolare finemente quel parametro per l'approssimazione multi-picco se necessario.

27.

28. Per le situazioni in cui le forme possono essere diverse per ogni picco e si desidera che il computer determini la forma più adatta per ciascun picco separatamente, utilizzare le forme con tre variabili iterate non vincolate: 30=Voigt con alfa variabile, 31=costante di tempo variabile Exp-Gaussian (**Shift-R**), 32=Pearson a forma variabile, 33=mix Gaussiana/Lorentziana a percentuale variabile. Questi modelli richiedono più tempo e sono difficili, soprattutto per più picchi sovrapposti.

29.

30. Per le approssimazioni difficili, può essere utile premere **X**, che riavvia l'approssimazione iterativa 10 volte *on*



prime ipotesi leggermente diverse e prende quello con l'errore di approssimazione più basso. Nella versione 13.2, il centro del grafico mostra "Working..." mentre l'approssimazione è in corso. Ovviamente ci vorrà un po' più di tempo. (Si può modificare il numero di tentativi, "Num-Trials", nella o vicino alla riga 227 - il valore di default è 10). *Le posizioni e le larghezze risultanti dei picchi dalla migliore approssimazione tra 10 diventano quindi i punti di partenza per le approssimazioni successive*, quindi l'errore di approssimazione dovrebbe gradualmente diminuire premendo **X** ripetutamente, finché non si assesta al minimo. Se nessuna delle 10 prove fornisce un errore di approssimazione inferiore a quello precedente, non viene modificato nulla. Questi valori iniziali rimangono in vigore fino a quando non si modifica il numero di picchi o si utilizzano i controlli di pan o zoom. (Ricordare: [approssimazioni a larghezze uguali, a larghezze fisse](#), e profili con posizioni fisse sono più veloci, più facili e molto più stabili delle normali approssimazioni, quindi usare le approssimazioni a larghezza uguale ogni volta che ci si aspetta che le larghezze dei picchi siano uguali quasi, oppure a larghezza fissa (o posizioni fisse) quando le larghezze o le posizioni sono note da esperimenti precedenti).

- 31.**
- 32.** Premere **Y** per visualizzare tutto il segnale a schermo intero senza i cursori, con l'ultima approssimazione visualizzata in verde. Il residuo viene visualizzato in rosso, sulla stessa scala dell'asse y dell'intero segnale.
- 33.**
- 34.** Premere **M** per passare avanti e indietro tra le modalità log e lineare. In modalità log, l'asse y del grafico superiore cambia in y semilog, e $\log(\text{modello})$ è approssimato a $\log(y)$, utile se i picchi variano notevolmente in altezza.
- 35.**
- 36.** Premere il tasto **D** per salvare i dati dell'approssimazione su disco come SavedModel.mat, contenente due matrici: DataSegment (la porzione dei dati originali da approssimare) e ModelMatrix (una matrice contenente ciascun componente del modello interpolato a 600 punti in quel segmento). Per posizionarli nell'area di lavoro, digitare load SavedModel. Per disegnare il DataSegment salvato, digitare plot(DataSegment(:,1), DataSegment(:,2)). Per disegnare SavedModel, digitare plot(ModelX,ModelMatrix); ciascuna componente nel modello verrà disegnata con un colore diverso.
- 37.**
- 38.** Premere **W** per stampare la funzione peakfit.m con tutti gli argomenti di input, inclusi gli ultimi valori migliori del vettore della prima ipotesi. Si può copiare e incollare la funzione peakfit.m nel proprio codice o nella finestra di comando, poi sostituire "datamatrix" con la propria matrice x-y.
- 39.**
- 40.** Sia [ipf.m](#) che [peakfit.m](#) sono in grado di stimare la variabilità attesa di posizione, altezza, larghezza ed area del picco dal segnale, utilizzando il [metodo di campionamento bootstrap](#) (pagina 162). Ciò comporta l'estrazione di 100 campioni bootstrap dal segnale, l'approssimazione di ciascuno di quei campioni col modello, quindi il calcolo dell'incertezza di ciascun picco: la deviazione standard (RSD) e la deviazione standard percentuale relativa (%RSD). Fondamentalmente, questo metodo calcola gli accoppiamenti ponderati di un singolo set di dati, utilizzando un diverso set di pesi per ogni campione. Questo processo ad alta intensità di calcolo può richiedere diversi minuti per essere completato, soprattutto se il numero di picchi nel modello e/o il numero di punti nel segnale sono elevati.
- 41.**
- 42.** Per attivare questo processo in ipf.m, premere il tasto **V**. Innanzitutto chiede di digitare il numero di approssimazioni di prova "best-of-x" per ogni campione bootstrap (il default è 1, ma si po-

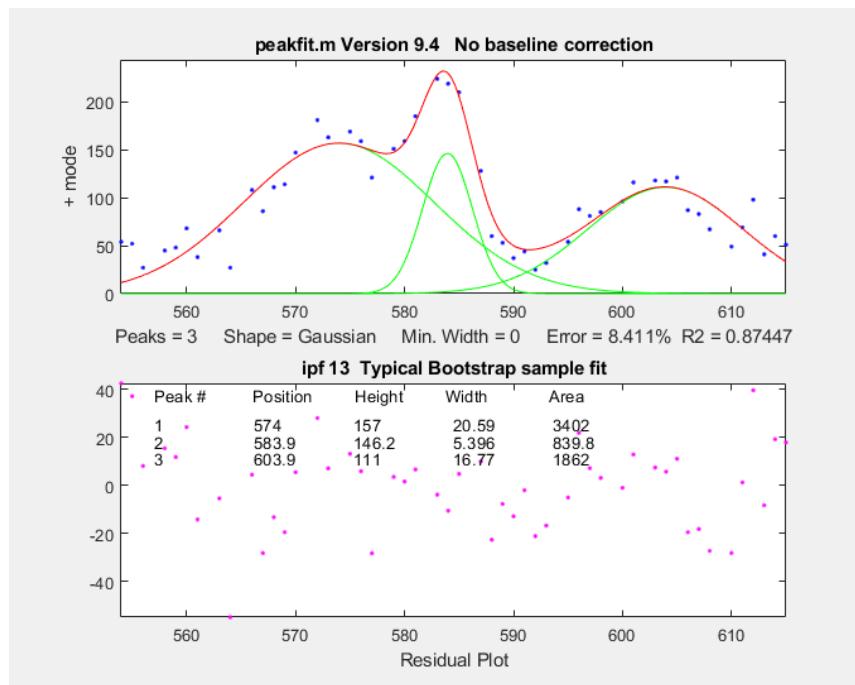
trebbe aver bisogno di un numero più alto se le approssimazioni sono occasionalmente instabili; provare 5 o 10 se i risultati iniziali danno NaN o numeri incredibilmente improbabili). (Per attivare questo processo in peakfit.m, è necessario utilizzare la versione 3.1 o successiva e includere tutti e sei gli argomenti di output, ad esempio [FitResults, LowestError, residuals, xi, yi, BootstrapErrors]=peakfit...). Nella versione 13.2, il centro del grafico mostra "Working..." mentre l'approssimazione è in corso. Il programma visualizza i risultati in una tabella nella finestra di comando. Ad esempio, ecco un'approssimazione a tre picchi Gaussiani per alcuni dati sperimentali rumorosi, seguita da un calcolo delle statistiche bootstrap:

43.

```

44. Shape= Gaussian % Fitting Error= 8.1876% R2= 0.89861
45. Peak# Position Height Width Area
46. 1 573.97 156.96 20.591 3401.6
47. 2 583.94 146.22 5.3956 839.78
48. 3 603.94 110.96 16.77 1861.7
49.
50. Number of fit trials per bootstrap sample (0 to cancel): 1
51. Computing bootstrap sampling statistics...May take several minutes.
52.

```



Peak #1

Parameter	Mean	STD	STDIQR	PercentRSD	PercentRSDIQR
{'Position'}	573.89	0.33752	0.28748	0.058813	0.050093
{'Height'}	158.53	3.6802	3.0271	2.3215	1.9095
{'width'}	19.038	1.408	1.2717	7.3954	6.6795
{'Area'}	3187.6	194.89	178.79	6.1141	5.6088

Peak #2

Parameter	Mean	STD	STDIQR	PercentRSD	PercentRSDIQR
{'Position'}	583.84	0.098986	0.1007	0.016954	0.017247
{'Height'}	156.03	10.837	11.916	6.9454	7.6372
{'width'}	5.5557	0.26153	0.24418	4.7074	4.3952
{'Area'}	924.28	95.719	76.147	10.356	8.2385

Peak #3

Parameter	Mean	STD	STDIQR	PercentRSD	PercentRSDIQR
{'Position'}	603.4	0.36056	0.33927	0.059754	0.056227
{'Height'}	113.24	3.9028	4.3535	3.4465	3.8445
{'width'}	16.128	1.2027	1.2781	7.457	7.9247
{'Area'}	1843.3	79.733	87.735	4.3256	4.7598

Elapsed time is 10.655257 seconds.

Si noti che, nonostante il rumore nei dati e l'errore di approssimazione dell'8.4%, le deviazioni standard relative del bootstrap non sono così negative, specialmente per le posizioni e le altezze dei picchi. Si noti che la RSD della *posizione del picco è il migliore* (il più basso), seguito da altezza, larghezza e area. Questo è uno schema tipico. Inoltre, si deve essere consapevoli del fatto che l'affidabilità della variabilità calcolata dipende dal presupposto che il rumore nel segnale sia rappresentativo del rumore medio nelle misure ripetute. Se il numero di punti nel segnale è piccolo, queste stime possono essere molto approssimative.

Se l'RSD e l'RSD IQR sono più o meno gli stessi (come nell'esempio sopra), la distribuzione dei risultati dell'approssimazione bootstrap è vicina alla normale e l'approssimazione è stabile. Se l'RSD è sostanzialmente maggiore dell'RSD IQR, allora l'RSD è sbilanciato verso l'alto da "valori anomali" (attacchi inconsapevolmente errati che cadono lontano dalla norma), e in tal caso si dovrebbe usare l'RSD IQR anziché l'RSD, perché l'IQR è molto meno influenzato dai valori anomali. (In alternativa, si può utilizzare un altro modello o un set di dati diverso per vedere se ciò fornisce approssimazioni più stabili).

Una probabile trappola con il metodo bootstrap, quando applicato alle approssimazioni iterative, è la possibilità che uno (o più) degli accoppiamenti bootstrap vada fuori strada, vale a dire, risulteranno parametri di picco largamente fuori dalla norma, e una variabilità stimata dei parametri troppo alta. Per questo motivo, in ipf 12.3, vengono calcolate *due misure* dell'incertezza: (a) la *deviazione standard* regolare (STD) e (b) la deviazione standard stimata dividendo il l'[intervallo interquartile](#) (IQR) per 1.34896. L'IQR è più resistente alle anomalie. Per una distribuzione *normale*, l'intervallo interquartile è in media pari a 1.34896 volte la deviazione standard. Se una o più approssimazioni del campione bootstrap fallisce, determinando una distribuzione dei parametri con grandi valori anomali, il normale STD sarà molto maggiore dell'IQR. In tal caso, una stima più realistica della variabilità è IRQ/1.34896. È meglio cercare di aumentare la stabilità dell'approssimazione scegliendo

un modello migliore (ad esempio utilizzando un [modello a larghezza uguale o fissa](#), o un profilo a posizione fissa, se appropriato), regolando intervallo (tasti pan e zoom), la sottrazione del background (tasti **T** o **B**), o le posizioni iniziali (tasto **C**), e/o selezionando un numero maggiore di prove di approssimazione per bootstrap (che aumenterà il tempo di calcolo). Come rapido test preliminare della stabilità dell'approssimazione bootstrap, premendo il tasto **N** verrà eseguito una singola approssimazione iterativa a un sotto-campione bootstrap casuale e verrà disegnato il risultato. Lo si faccia più volte per vedere se le approssimazioni bootstrap sono sufficientemente stabili da valere la pena calcolare un bootstrap da 100 campioni. **Nota:** è normale che la stabilità delle approssimazioni del campione bootstrap (tasto [N: cliccare qui per l'animazione](#)) sia inferiore alle approssimazioni del campione completo (tasto [F: cliccare qui per l'animazione](#)), perché quest'ultima include solo la variabilità causata dalla modifica delle posizioni di partenza per un insieme di dati e rumore, mentre i tasti **N** e **V** mirano a includere la variabilità causata dal rumore casuale nel campione approssimando sotto-campioni bootstrap. Inoltre, le migliori stime dei parametri misurati sono quelle ottenute dalle normali approssimazioni del segnale completo (tasti **F** e **X**), *non* dalle medie riportate per i campioni di bootstrap (tasti **V** e **N**), perché ci sono punti più indipendenti nelle approssimazioni completi e perché le medie del bootstrap sono influenzate dai valori anomali che si verificano più comunemente nelle approssimazioni bootstrap. I risultati del bootstrap sono utili solo per stimare la variabilità dei parametri di picco, non per stimarne i valori medi. Anche i tasti **N** e **V** sono metodi molto utili per determinare se si stanno utilizzando troppi picchi nel modello; *i picchi superflui saranno molto instabili quando N viene premuto ripetutamente* e avranno una deviazione standard molto più alta della sua altezza quando si usa il tasto **V**.

Peakfit e il metodo bootstrap possono funzionare bene per stimare la precisione delle misure dei parametri, anche quando il rapporto segnale/rumore è piuttosto scarso. Per un esempio basato sullo spettro IR del benzene, dal [NIST Quantitative Infrared Database](#), vedere [FittingWeakPeaks.pdf](#).

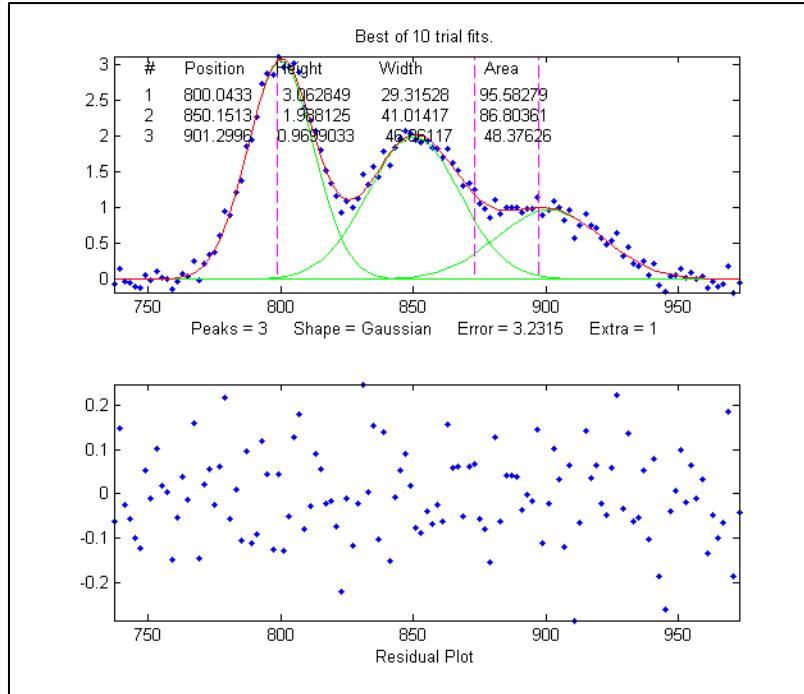
Shift-o approssima un semplice polinomio (lineare, quadratico, cubico, ecc.) al segmento del segnale visualizzato nel pannello superiore e mostra i coefficienti polinomiali (in potenze discendenti) e l'R².

20. Se dei picchi sono saturi e hanno una parte superiore piatta (tagliata all'altezza massima), è possibile fare in modo che il programma ignori i punti saturati premendo **Shift-M** e inserendo i valori Y massimi da mantenere. I valori Y superiori a questo limite verranno semplicemente ignorati; i picchi al di sotto di questo limite si approssimeranno come al solito.

Per vincolare il modello a picchi superiori a una certa larghezza, premere **Shift-W** e immettere la larghezza minima consentita.

Demoipf.m

Demoipf.m è uno script demo per ipf.m, con un generatore di segnali simulati incluso. I valori reali delle posizioni, delle altezze e delle larghezze dei picchi simulati vengono visualizzati nella finestra dei comandi di Matlab, per il confrontarli con i FitResults ottenuti mediante l'approssimazione del picco. Il segnale di default simulato contiene sei indipendenti gruppi di picchi utilizzabili per fare pratica: una tripletta prossima a $x = 150$, un picco isolato a 400, una coppia vicino a 600, una tripletta a circa 850, e due larghi picchi isolati a 1200 e 1700. Eseguire questa demo e vedere quanto ci si possa avvicinare ai parametri effettivi. L'utilità di una simulazione come questa è che si può avere un'idea dell'accuratezza delle misure dei parametri, ovvero la differenza tra i valori veri e quelli misurati. Per scaricare questi m-file, clic destro sui link, selezionare **Save Link As...** e cliccare su **Save**. Per l'esecuzione, piazzare sia [ipf.m](#) che [Demoipf](#) nel path di ricerca di Matlab, poi digitare **Demoipf** al prompt di Matlab.



Un esempio dell'utilizzo di questo script è mostrato nella figura. Qui ci si concentra sui tre picchi accavallati vicino a $x=850$. I veri parametri dei picchi (prima di aggiungervi il rumore casuale) sono:

Position	Height	Width	Area
800	30	95.808	
850	40	85.163	
900	50	53.227	

Quando questi picchi vengono isolati nella finestra superiore e approssimati con tre Gaussiane, i risultati sono:

Position	Height	Width	Area
800.04	3.0628	29.315	95.583
850.15	1.9881	41.014	86.804
901.3	0.9699	46.861	48.376

Come si può vedere l'accuratezza delle misure è eccellente per la posizione, buona per l'altezza e meno buona per la larghezza e l'area. Non sorprende che le misure meno accurate siano per il picco più piccolo con il rapporto segnale/rumore più scarso. Nota: la deviazione standard prevista di questi parametri può essere determinata mediante il metodo del campionamento bootstrap, come descritto nella sezione precedente. Ci aspetteremmo che i valori misurati dei parametri

(confrontando i valori veri con quelli misurati) rientrino in circa 2 deviazioni standard dai valori reali elencati sopra).

[Demoipf2.m](#) è identico, tranne per il fatto che i picchi sono sovrapposti su una linea di base fortemente curva, quindi si può testare l'accuratezza dei metodi di correzione della linea di base (# 9 e 10, sopra).

Tempo di esecuzione dell'approssimazione ed altri compiti del signal processing

Per "tempo di esecuzione" si intende il tempo necessario per eseguire un'approssimazione, escludendo il disegno e la stampa dei risultati, quando Matlab è in esecuzione su un PC standard. Per l'approssimazione iterativa dei picchi, i fattori principali che determinano il tempo di esecuzione sono (a) la velocità del computer, (b) il numero di picchi e (c) la forma del picco:

- a) Il tempo di esecuzione può variare di un fattore di 4 o 5 o più tra computer diversi, (ad esempio, confrontando un piccolo laptop, con una CPU Athlon dual-core da 1.6 GHz, con 4 Gbytes di RAM, con un desktop con una CPU i7 da 3.4 GHz con 16 Gbyte di RAM). Eseguire il test di benchmark Matlab "bench.m" per vedere come si comporta il proprio computer.
- b) Il tempo di esecuzione aumenta con il prodotto del numero di picchi nel modello per il numero di variabili iterate per ciascun picco. (Vedere [PeakfitTimeTest.m](#)).
- c) Il tempo di esecuzione varia notevolmente (a volte di un fattore 100 o più) con la forma del picco, dove le forme espanso esponenzialmente sono le più lente e le forme a larghezza fissa quelle più veloci. Vedere [PeakfitTimeTest2.m](#) e [PeakfitTimeTest2a.m](#). Le varianti a larghezza-uguale e a larghezza-fissa dei profili sono sempre più veloci dei corrispondenti modelli a larghezza-variabile.
- d) Il tempo di esecuzione aumenta direttamente con NumTrials in peakfit.m. La funzione "Best of 10 trials" (tasto X in ipf.m) richiede circa 10 volte più tempo di una singola approssimazione.

Altri fattori meno importanti sono il numero di punti nella regione approssimata (ma solo se il numero di punti è molto grande; vedere ad esempio [PeakfitTimeTest3.m](#)) e i valori iniziali (buoni i valori iniziali possono ridurre leggermente il tempo di esecuzione; [PeakfitTimeTest2.m](#) e [PeakfitTimeTest2a.m](#) hanno degli esempi a proposito). Nota: alcuni di questi script richiedono [DataMatrix2](#) e [DataMatrix3](#), scaricabili da <http://tinyurl.com/cey8rwh>.

[TimeTrial.txt](#) è un file di testo che confronta la velocità di 18 diverse attività di elaborazione del segnale in esecuzione su 5 diversi sistemi: (1) Windows 10, 64-bit, 3.6 GHz core i7, con 16 GBytes RAM, con Matlab 9.9 (R2020b) Update 3 academic, (2) Matlab 2017b Home; (3) Matlab Online R2018b nel browser Chrome, in esecuzione su PC desktop o laptop datati, (4) Matlab Mobile su un iPad, e (5) Octave 6.2.0 in esecuzione su un computer desktop. Il codice Matlab/Octave che ha generato questo è [TimeTrial.m](#), che esegue tutte le attività una dopo l'altra e stampa i tempi impiegati dalla particolare macchina, oltre ai tempi precedentemente registrati per ciascuna attività su ciascuno dei cinque sistemi software. [TimeTrial.xlsx](#) riassume i confronti tra Matlab e Octave. Vedere anche pagina 427 per un confronto di velocità tra Matlab e Python che eseguono alcune attività diverse.

Suggerimenti e consigli per il Curve Fitting Iterativo

1. Se l'approssimazione fallisce completamente, restituendo tutti zeri, i dati potrebbero essere formattati in modo errato. La variabile indipendente ("x") e quella dipendente ("y") devono essere

vettori o colonne separati di una matrice 2-x n, con le x nella prima riga o colonna.. Oppure potrebbe essere necessario fornire le prime ipotesi ("start") per tale approssimazione.

2. È preferibile *non* eseguire lo smoothing prima dell'approssimazione. Lo smoothing può distorcere la forma del segnale e la distribuzione del rumore, rendendo più difficile valutare l'approssimazione mediante l'ispezione visiva del grafico dei residui. Uno smoothing preventivo dei dati rende impossibile il raggiungimento dell'obiettivo di ottenere un diagramma casuale dei residui non strutturati e aumenta la possibilità che "si approssimi il rumore" piuttosto che il segnale effettivo. Le stime dell'errore di bootstrap non sono valide se i dati hanno subito uno smoothing.
3. Il fattore più importante nell'approssimazione iterativa della curva non lineare è la selezione della *funzione del modello del picco in esame*, ad esempio, Gaussiana, Gaussiane a larghezze uguali, Lorentziana, ecc. (cfr. pagina 202). Vale la pena dedicare un po' di tempo nel cercare e verificare una funzione adatta per i dati. Se si prevede che le larghezze in ciascun gruppo di picchi siano uguali o quasi, selezionare le forme a "larghezze uguali"; le approssimazioni a larghezze uguali (disponibili per le forme Gaussiana e Lorentziana) sono più veloci, più facili e molto più stabili delle normali approssimazioni a larghezza variabile. Ma è importante capire che una buona approssimazione *non è di per sé la prova* che la funzione del profilo scelto sia quella corretta; in alcuni casi, la funzione sbagliata può dare un'approssimazione che sembra perfetta. Ad esempio, si consideri un modello Gaussiano a 5 picchi che presenta un errore di approssimazione percentuale basso e per il quale i residui sembrano casuali - di solito è un indicatore di una buona approssimazione ([Click per un grafico](#) se si sta leggendo online). Ma in effetti, in questo caso, il modello è *sbagliato*; quei dati provenivano da un dominio sperimentale in cui la forma in esame è fondamentalmente non Gaussiana ma in alcuni casi può assomigliare molto a una Gaussiana. È importante ottenere il modello giusto per i dati e non dipendere esclusivamente dalla bontà dell'approssimazione.
4. Si dovrebbe sempre utilizzare il numero *minimo* di picchi che approssimano adeguatamente i dati. ([pagina](#) 202). L'uso di troppi picchi può fornire un'approssimazione instabile: le linee verdi nel grafico superiore, che rappresentano i picchi dei singoli componenti, rimbalzeranno all'impazzata per ogni approssimazione ripetuta, senza ridurre in modo significativo l'errore. Un modo molto utile per determinare se si stanno usando troppi picchi nel modello è usare il tasto **N** (cfr. #10, più sotto) per eseguire una singola approssimazione a un sotto-campione di punti bootstrap; *i picchi superflui saranno molto instabili quando N sarà premuto ripetutamente*. (È possibile ottenere statistiche migliori per questo test, a discapito del tempo, utilizzando il tasto **V** per calcolare la deviazione standard di 100 sotto-campioni bootstrap).
5. Se i picchi sono sovrapposti su un background o una linea di base, è necessario tenerne conto *prima* dell'approssimazione, altrimenti i parametri dei picchi (soprattutto altezza, larghezza e area) saranno imprecisi. O sottrarre la linea di base dall'*intero* segnale col tasto **Backspace** (#10 nelle [Istruzioni per l'uso](#), sopra) oppure usare il tasto **T** per selezionare una delle modalità di correzione automatica della linea di base (# 9 nelle [Istruzioni per l'uso](#), sopra).
6. Questo programma utilizza una funzione di ricerca non lineare iterativa ("*modified Simplex*") per determinare le posizioni e le larghezze dei picchi che meglio corrispondono ai dati. Ciò richiede delle ipotesi iniziali per le posizioni e le larghezze dei picchi. (Le *altezze* non richiedono ipotesi iniziali, perché sono parametri lineari; il programma le determina mediante regressione lineare). Le ipotesi iniziali di default per le posizioni dei picchi vengono create dal computer in base alle impostazioni di pan e zoom e sono indicate dalle linee tratteggiate verticali magenta. Le ipotesi iniziali per le larghezze dei picchi vengono calcolate dall'impostazione dello zoom, quindi i migliori risultati saranno ottenuti se si ingrandisce in modo che il gruppo di picchi sia isolato e distribuito come suggerito dai marker di posizione dei picchi (linee tratteggiate verticali).

7. Se le componenti del picco sono spaziate in modo molto irregolare, potrebbe essere meglio inserire manualmente le posizioni dei picchi come prime ipotesi premendo il tasto **C** e cliccando poi sul grafico in alto dove si pensa che potrebbero essere i picchi. Niente di tutto questo deve essere precisissimo: sono solo le prime ipotesi, ma se sono troppo lontane si può inficiare l'algoritmo di ricerca. È inoltre possibile digitare manualmente la prima ipotesi per le posizioni *e* le larghezze premendo **Shift-C**.
8. Ogni volta che si esegue un'altra approssimazione iterativa (ad esempio, premendo il tasto **F**), il programma aggiunge piccole deviazioni casuali alle prime ipotesi, per determinare se è possibile ottenere un'approssimazione migliore con prime ipotesi leggermente diverse. Ciò è utile per determinare la robustezza o la stabilità dell'approssimazione *rispetto ai valori iniziali*. Se l'errore e i valori dei parametri variano leggermente in una regione ristretta, significa che si ha un'approssimazione robusta (per quel numero di picchi). Se l'errore e i valori dei parametri rimbalzano notevolmente, significa che l'approssimazione non è robusta (provare a cambiare il numero di picchi, la forma e le impostazioni di pan e zoom), o potrebbe semplicemente essere che i dati non sono abbastanza buoni da approssimarsi a quel modello. Provare a premere il tasto **X**, che accetta la migliore tra 10 approssimazioni iterative e utilizza quei valori come *ipotesi iniziali* per le approssimazioni successive. Quindi, ogni volta che si preme **X**, se uno di questi accoppiamenti produce un errore di approssimazione inferiore al migliore precedente, questo viene considerato come l'inizio per l'approssimazione successiva. Di conseguenza, le approssimazioni tendono a migliorare gradualmente quando si preme ripetutamente il tasto **X**. Spesso, anche se la prima approssimazione è terribile, le successive col tasto **X** miglioreranno notevolmente.
9. La variabilità dei parametri dall'approssimazione utilizzando i tasti **X** o **F** è solo una stima dell'incertezza causata dalla procedura di approssimazione della curva (ma *non* dell'incertezza causata dal rumore nei dati, perché questo è solo per un campione e del rumore; per questo è necessaria l'approssimazione col tasto **N**).
10. Per esaminare la robustezza o la stabilità dell'approssimazione *rispetto al rumore casuale nei dati*, si preme il tasto **N**. **Ogni volta che si preme N**, eseguirà un'approssimazione iterativa su un diverso sottoinsieme di punti nella regione selezionata (chiamato "campione bootstrap"; vedere pagina 162). [Cliccare per l'animazione](#). Se questo fornisce approssimazioni dall'aspetto ragionevole, è possibile continuare a calcolare le statistiche dell'errore del picco premendo il tasto **V**. Se d'altra parte il tasto **N** fornisce approssimazioni molto diverse, con errori di approssimazione e dei parametri altamente variabili, allora l'approssimazione non è stabile e si potrebbe provare il tasto **X** per prendere la migliore tra 10 approssimazioni e azzerare le ipotesi iniziali, quindi premere di nuovo **N**. In casi difficili, potrebbe essere necessario aumentare il numero di prove quando richiesto (ma ciò aumenterà il tempo necessario per il completamento), o se ciò non aiuta, utilizzare un altro modello o un set di dati migliore. (I tasti **N** e **V** sono un buon modo per valutare un modello multi-picco per la possibilità di picchi superflui, vedere # 4 sopra).
11. Se non si trova la forma del picco di cui si ha bisogno in questo programma, guardare la sezione successiva per sapere come aggiungerne di nuovi, oppure [scrivetemi](mailto:scrivetemi@toh.umd.edu) a toh@umd.edu e vedrò cosa posso fare.
12. Se si tenta di approssimazione un segmento della variabile indipendente (asse x) molto piccolo di un segnale molto grande, ad esempio una regione che è solo 1/1000° o meno dell'intero intervallo dell'asse x, si potrebbe riscontrare un problema con approssimazioni instabili. In tal caso, provare a sottrarre una costante da x, eseguire poi l'approssimazione, e dopo aggiungere la quantità sottratta alle posizioni x misurate.
13. Se ci sono pochissimi punti dati sul picco, potrebbe essere necessario ridurre la larghezza minima (impostata da minwidth in peakfit.m o Shift-W in ipf.m) a zero o a qualcosa di più piccolo del minimo di default (che è il valore predefinito alla spaziatura dell'asse x tra punti adiacenti).
14. Differenza tra i tasti **F**, **X**, **N** e **V** in ipf.m:

- **F:** Varia leggermente i valori iniziali ed esegue una singola approssimazione iterativa utilizzando tutti i punti dati nella regione selezionata.
- **X:** Esegue 10 approssimazioni di prova iterative utilizzando tutti i punti dati nella regione selezionata, variando leggermente i valori iniziali prima di ogni prova, poi prende quello con l'errore più basso. Premerlo nuovamente per ripetere e affinare l'approssimazione. Richiede circa 10 volte più tempo del tasto **F**.
- **N:** Varia leggermente i valori iniziali ed esegue un'approssimazione singola iterativa utilizzando un sottoinsieme casuale dei punti nella regione selezionata. Utilizzarlo per visualizzare la stabilità dell'approssimazione rispetto al rumore casuale. Impiega lo stesso tempo del tasto **F**.
- **V:** Esegue diverse approssimazioni di prova, poi esegue 100 approssimazioni iterative ciascuna su un diverso sottoinsieme casuale dei punti nella regione selezionata, ciascuna utilizzando il numero di prove specificato e prendendo la migliore, poi calcola la media e la deviazione standard dei parametri di tutti i 100 migliori risultati. Utilizzarlo per quantificare la stabilità dei parametri rispetto al rumore casuale. Richiede circa 100 volte più tempo del tasto **X**.

Estrazione delle equazioni per i modelli migliori

Le equazioni per i profili dei picchi si trovano nel menù apposito in ipf.m, isignal.m e ipeak.m. Ecco le espressioni per i profili espressi matematicamente anziché algoritmicamente:

```
Gaussiana: y =exp(-((x-pos)/(0.60056120439323*width)) ^2)
Lorentziana: y =1/(1+((x-pos)/(0.5*width))^2)
Logistica: y =exp(-((x-pos)/(.477*wid))^2); y=(2*n)/(1+n)
Lognormale: y = exp(-(log(x/pos)/(0.01*wid)) ^2)
Pearson: y =1/(1+((x-pos)/((0.5^(2/m))*wid))^2)^m
Breit-Wigner-Fano: y =((m*wid/2+x-pos)^2)/(((wid/2)^2)+(x-pos)^2)
Funzione Alpha: y =(x-spoint)/pos*exp(1-(x-spoint)/pos)
Sigmoide in salita: y =.5+.5*erf((x-tau1)/sqrt(2*tau2))
Sigmoide discendente: y =.5-.5*erf((x-tau1)/sqrt(2*tau2))
Gompertz: y =Bo*exp(-exp((Kh*exp(1)/Bo)*(L-t) +1))
FourPL: y = 1+(miny-1)/(1+(x/ip)^slope)
OneMinusExp: y = 1-exp(-wid*(x-pos))
EMG (profilo 39) y = s*lambda*sqrt(pi/2)*exp(0.5*(s*lambda)^2-lambda*(t-mu))*erfc((1/sqrt(2))*(s*lambda-(t-mu)/s)))
```

I parametri del picco (altezza, posizione, larghezza, tau, lambda, ecc.) restituiti dai FitResult visualizzati sul grafico e nella finestra di comando. Per esempio, se si approssima un insieme di dati a una singola Gaussiana si ottiene...

Peak#	Position	Height	Width	Area
1	0.30284	87.67	0.23732	22.035

...quindi l'equazione sarebbe:

$$y = 87.67 * \exp(-((x-0.30284)/(0.6005612*0.23732)).^2)$$

Se si specifica un modello con più di un picco, l'equazione è la *somma* di ciascun picco nel modello. Ad esempio, approssimando la funzione nativa di Matlab "humps", utilizzando un modello con 2 Lorentziane

```
>> x=[0:.005:2];y=humps(x);[FitResults,GOF]=peakfit([x' y'],0,0,2,2)
```

```

Peak# Position Height Width Area
1 0.3012 96.9405 0.1843 24.9270
2 0.8931 21.1237 0.2488 7.5968

```

Utilizzando l'espressione per una Lorentziana nella pagina precedente, l'equazione per due picchi sarebbe:

```
y = 96.9405*(1/(1+((x-0.3012)/(0.5*0.1843))^2)) + 21.1237*(1/(1+((x-0.8931)/(0.5*0.2488)).^2))
```

È anche possibile utilizzare più forme in un'approssimazione, specificando il parametro della forma del picco come vettore. Ad esempio, si potrebbe approssimare il primo picco della funzione "humps" con una Lorentziana e il secondo con una Gaussiana utilizzando [2 1] come argomento della forma.

```
>> x=[0:.005:2];y=humps(x);peakfit([x' y'],0,0,2,[2 1])
```

```

Peak# Position Height Width Area
1 0.3018 97.5771 0.1876 25.4902
2 0.8953 18.8877 0.3341 6.7180

```

In tal caso l'espressione sarebbe $y = \text{height1} * \text{Lorentzian} + \text{height2} * \text{Gaussian}$:

```
y = 97.5771*(1/(1+((x-0.3018)/(0.5*0.1876))^2)) + 18.8877*exp(-( (x-0.8953)/(0.60056120439323*0.3341)).^2)
```

Nota: Per ottenere i dati del modello *campionati digitalmente*, utilizzare il 6° e il 7° parametro di *output* di peakfit, xi e yi, che restituiscono un modello interpolato di 600 punti come vettore di valori x e una matrice di valori y con una riga per ogni componente. Digitare `plot(xi, yi)` per disegnare ciascun picco del modello separatamente in un colore diverso o `plot(xi, yi(1,:))` per disegnare solo il picco 1.

Come aggiungere un nuovo profilo in peakfit.m, ipf.m, iPeak, o iSignal

È facile aggiungere la forma personalizzata di un picco a peakfit.m (o a quelle funzioni interattive che utilizzano peakfit.m internamente, come ipf.m, iSignal o iPeak), se si ha un'espressione matematica del profilo. Il modo più semplice è *modificare un profilo esistente* che non si intende utilizzare, sostituendolo con la nuova funzione. Attenzione, scegliere una forma da sacrificare che abbia lo stesso numero di variabili e vincoli e che abbia la nuova forma. Per esempio, se il proprio profilo ha *due* parametri iterati (p.es., posizione e larghezza variabili), si può modificare la forma Gaussiana, Lorentziana o la Triangolare (rispettivamente numero 1, 2 e 21). Se il proprio profilo ha *tre* variabili iterate, si possono usare forme come la 31, 32, 33, o la 34. Se il proprio profilo ha *quattro* variabili iterate, si usa la forma 49 ("Gaussiana doppia", Shift-K). Se il proprio profilo ha un parametro 'extra', come Voigt, Pearson, BWF, o Mix Gaussiana/Lorentziana, tutte a forme uguali, usare una di queste. Se c'è bisogno di una forma modificata esponenzialmente, si usa la Gaussiana modificata esponenzialmente (5 o 31) o la Lorentziana (18). Se c'è bisogno di larghezze uguali o fisse, ecc., si usa una di queste forme. **Questo è importante;** si deve avere lo stesso numero di variabili e vincoli, perché la struttura del codice è diversa per ciascuna classe di profili.

Ci sono solo due passaggi necessari per il processo:

- Si supponga che il nuovo profilo abbia *due* parametri iterati e che si intenda sacrificare la forma triangolare, numero 21. Aprire peakfit.m o ipf.m nell'editor di Matlab e ri-scrivere la funzione del *vecchio* profilo ("triangular", vicino alla riga 3672 in ipf.m - c'è una funzione simile per ciascuna forma - cambiando il *nome* della funzione e la *formula matematica* dell'istruzione di assegnazione (ad esempio, $g = 1 - (1 ./ wid) .* \text{abs}(x - pos);$). Si possono utilizzare le stesse variabili ("x" per la variabile indipendente, "pos" per la posizione del picco, "wid" per l'ampiezza, ecc.). Ridimensionare la funzione in modo da avere un'altezza del picco di 1.0 (ad esempio, dopo aver calcolato y come funzione di x, dividere per $\max(y)$).
- Usare la funzione di ricerca in Matlab per trovare tutte le istanze del nome della vecchia funzione e sostituirla con il nuovo nome, *selezionando "Wrap around" ma de-selezionando "Match case"* e *"Whole word"* nella casella Search. Se lo si fa bene, ad esempio, tutte le istanze di "triangular" e tutte le istanze di "fittriangular" verranno modificate con il nuovo nome che sostituirà "triangular". Effettuare il **Save** del risultato (o **Save as...** con un nome diverso).

Finito! La nuova forma sarà ora la 21 (o qualunque fosse il numero del vecchio profilo sacrificato). In ipf.m, sarà attivato dalla stessa sequenza di tasti usata dalla vecchia forma (p.es. Shift-T per il triangolare, tasto numero 84), e in iSignal e in iPeak, il menù delle forme risulterà modificato dalla sostituzione effettuata nel passaggio 2.

Volendo, si possono modificare le assegnazioni dei tasti in ipf.m. Dapprima, si trova un tasto o Shift-tasto non ancora assegnato (e che restituisca l'errore "UnassignedKey" quando lo si preme con ipf.m in esecuzione). Poi si cambia il vecchio numero del tasto 84 con quello non assegnato nella grande "switch double(key), " l'istruzione case in prossimità dell'inizio del codice.

Quale usare? *peakfit*, *ipf*, *findpeaks...*, *iPeak* o *iSignal*?

iPeak (pagina 405), *iSignal* (pagina 366), *peakfit* (pagina 384) e *ipf* (pagina 405), o le loro versioni Octave, sono state progettate ciascuna con un'enfasi diversa, sebbene vi siano alcune sovrapposizioni nelle loro funzioni. In breve, iSignal combina diverse funzioni di base, tra cui livellamento, differenziazione, analisi dello spettro, ecc.; iPeak e le varie funzioni findpeaks si concentrano sulla ricerca di picchi multipli in segnali di grandi dimensioni; peakfit e ipf si concentrano sull'approssimazione iterativo dei picchi. Ma c'è qualche sovrapposizione; iPeak e iSignal possono anche eseguire l'approssimazione dei minimi quadrati iterativa dei picchi e iSignal può eseguire la ricerca dei picchi. Inoltre, iSignal, iPeak e ipf sono *interattivi* e funzionano in *Matlab* o in *Matlab Online* in un browser web, mentre le loro versioni *a riga di comando* ProcessSignal.m, le numerose variazioni di findpeaks.m e peakfit.m sono funzioni. Il punto principale è che le funzioni interattive sono migliori per l'esplorazione e la prova diretta di diverse *impostazioni tecniche*, mentre le funzioni della riga di comando sono migliori per l'elaborazione automatica di grandi quantità di dati.

Caratteristiche comuni. I programmi interattivi iSignal, iPeak e ipf hanno tutti diverse caratteristiche in comune.

(a) Il tasto **K** visualizza i controlli della tastiera per ciascun programma.

(b) Le versioni Matlab usano come tasti pan e zoom i tasti freccia – sinistra e destra per pan, Su e Goù per lo zoom. *Le versioni Octave usano i tasti < e > (con e senza shift).*

(c) **Ctrl-Shift-A** seleziona l'intero segnale (cioè, lo rimpiccolisce completamente). Nelle versioni Octave, lo fa il tasto ";".

(d) Il tasto **W**. Per facilitare il trasferimento delle impostazioni da una di queste funzioni a un'altra o a una versione a riga di comando, tutte queste funzioni utilizzano il tasto **W** per stampare la sintassi delle altre funzioni correlate, con le impostazioni di pan e zoom e altri argomenti di input numerici specificati, pronti per essere copiati e incollati nei propri script o nuovamente nella finestra di comando. Ad esempio, è possibile convertire un'approssimazione della curva da *ipf* nella funzione a riga di comando *peakfit*; oppure si può convertire un'operazione di ricerca dei picchi da *ipeak* nella riga di comando delle funzioni *findpeaksG*, *findpeaksb* o *findpeaksb3*. Ciò fornisce un modo per gestire i segnali che richiedono una diversa elaborazione in diverse regioni dei loro intervalli dell'asse x, consentendo di creare una serie di funzioni a riga di comando per ciascuna regione locale che, se eseguite in sequenza, elaborano rapidamente ogni segmento del segnale in modo appropriato potendolo ripetere facilmente per un numero qualsiasi di altri esempi dello stesso tipo di segnale.

(e) Tutti questi programmi usano i tasti **Shift-Ctrl-S**, **Shift-Ctrl-F** e **Shift-Ctrl-P** tasti per trasferire il segnale corrente, come [variabili globali X e Y](#), a *iSignal.m*, *ipf.m* e a *iPeak.m*, rispettivamente.

Primo approccio? Dare uno sguardo a questi *demo Web animati* di [ipeak.m](#) e [ipf.m](#). Oppure scaricare queste funzioni demo di Matlab che confrontano *ipeak.m* con *peakfit.m* per segnali con [pochi picchi](#) e con [molti picchi](#) e che illustrano come regolare *ipeak* per rilevare [picchi larghi o stretti](#). Queste demo autonome comprendono tutte le funzioni Matlab richieste. Basta inserirle nel path e cliccare **Run** o digitare il nome al prompt dei comandi. Oppure si possono scaricare tutte assieme queste demo in [idemos.zip](#). [peakfitVSfindpeaks.m](#) esegue un confronto diretto dell'accuratezza dei parametri di *findpeaks* rispetto a *peakfit*.

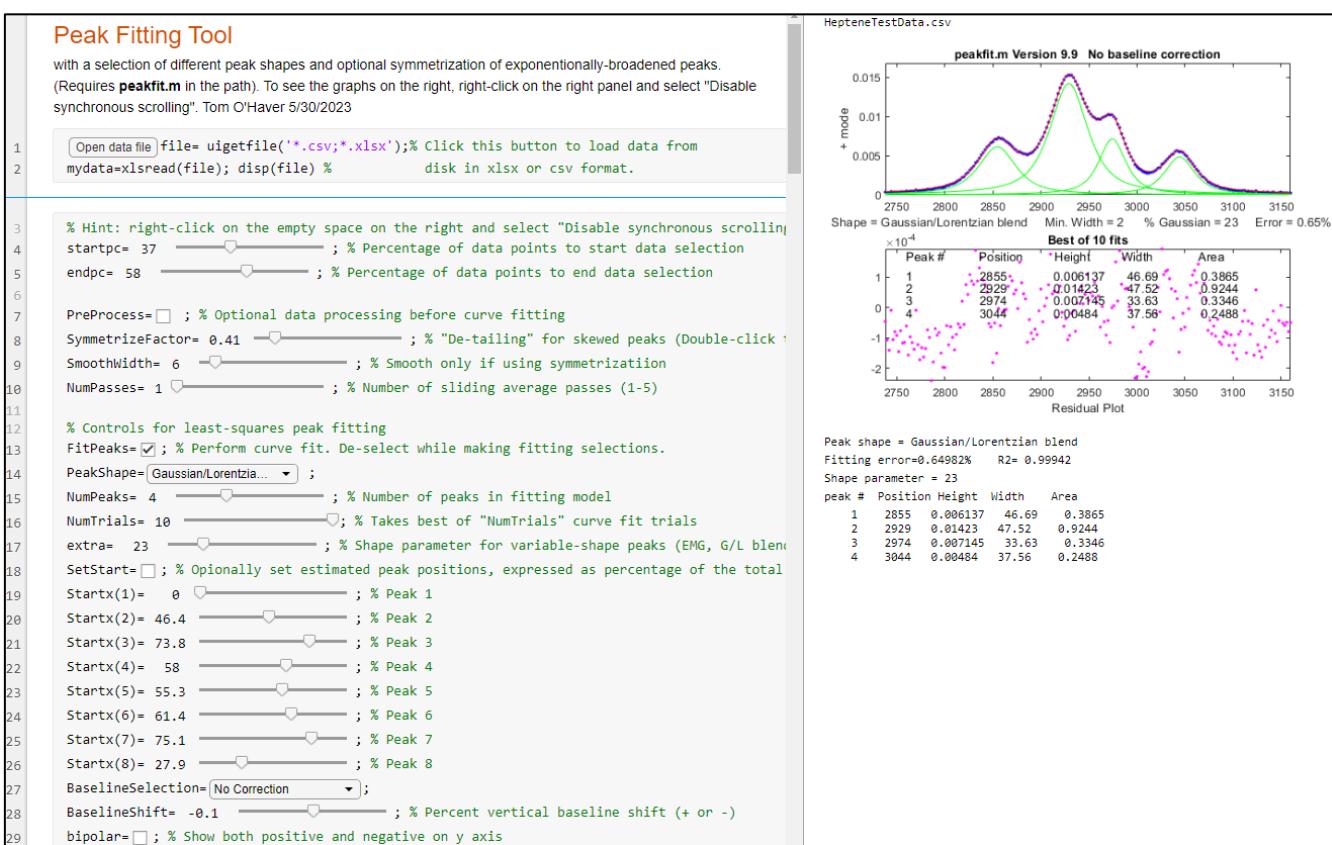
Nota 1: Cliccare sulla finestra Figure di Matlab per attivare gli strumenti della tastiera interattiva. Assicurarsi di non cliccare sul pulsante “Show Plot Tools” nella barra degli strumenti sopra la finestra; che disabiliterà il normale funzionamento del programma. Se lo si fa, chiudere semplicemente la finestra "Figure" e ricominciare

Nota 2: Le funzioni interattive *iPeak*, *iSignal*, *iFilter* e *ipf* operanti con la tastiera funzionano anche quando si esegue [Matlab in un browser web](#) basta cliccare prima sulla finestra "Figure"), ma attualmente non funzionano su [Matlab Mobile](#). Sono necessarie altre versioni se si utilizza Octave (con "octave" aggiunto al nome del file, ad esempio *ipfoctave.m* invece di *ipf.m*, ecc.)

Nota 3: Gli script e le funzioni devono trovarsi sul computer in una posizione specificata dal “path di ricerca di Matlab”, ovvero l’insieme di cartelle nel file system che Matlab usa per localizzare i file. Vedere [“What Is the MATLAB Search Path?”](#)

Tool Live Script Peak Fitter

[PeakFittingTool mlx](#) esegue l'approssimazione iterativa dei minimi quadrati applicata ai dati sperimentali archiviati su disco. Cliccando il pulsante "Open data file" nella riga 1 si apre un browser di file, che consente di navigare fino al file di dati (in formato .csv o .xlsx; lo script presuppone che i dati x,y siano nelle prime due colonne). Ma prima di aprire un file, è una buona idea deselezionare temporaneamente la casella di controllo "FitPeaks" nella riga 14, quindi attendere di aver impostato gli altri controlli. In questo modo si eviterà di attendere operazioni di approssimazione della curva non necessarie fino al completamento delle impostazioni appropriate. (A volte le operazioni di approssimazione della curva possono essere lente e nei casi difficili possono richiedere diversi secondi). Con FitPeaks disattivato, il programma visualizza semplicemente un grafico del file di dati selezionato. Nota: per visualizzare i grafici a destra del codice, come mostrato qui, clic destro sullo spazio vuoto a destra e selezionare "Disable synchronous scrolling".



Regolare i cursori **startpc** e **endpc** nelle righe 4 e 5 per isolare gruppi di picchi ravvicinati che possono essere approssimati insieme. Cercare di distribuirli nel modo più uniforme possibile, come mostrato nella figura sopra. (Se tutti i picchi sono ben separati e non si sovrappongono, e se il numero di picchi varia in modo imprevedibile da segnale a segnale, è possibile che siano stati disattivati utilizzando lo [Peak Detector Tool](#) ([Peakdetector mlx](#)), che ha anche una funzione di approssimazione del picco).

La casella di controllo "PreProcess" (riga 7) consente una pre-elaborazione preliminare opzionale. Lo slider **SymmetrizeFactor** esegue la ["simmetrizzazione"](#) o ["de-tailing"](#) per i picchi distorti dall'ampliamento esponenziale, mediante l'aggiunta della derivata prima. Aumentare il valore di SymmetrizeFactor finché il picco non è il più stretto possibile senza che il bordo finale scenda al di

sotto della linea di base. I cursori SmoothWidth e NumPasses (linee 9 e 10) consentono lo [slittamento-della-media](#) del segnale, che è utile nei casi in cui il rumore ad alta frequenza oscura i picchi. Il cursore "VerticalShift" (riga 11) consente lo spostamento positivo e negativo nella posizione della linea di base, per compensare l'offset della linea di base.

Il menù a discesa **PeakShape** consente di selezionare il profilo del picco del modello approssimato. **NumPeaks** imposta il numero di picchi nel modello. NumTrials, riavvia il processo di approssimazione "NumTrials" volte con valori iniziali leggermente diversi e seleziona quello migliore (con l'errore di approssimazione più basso). **NumTrials** può essere qualsiasi numero intero positivo. In molti casi, NumTrials=1 sarà sufficiente, ma se ciò non fornisce risultati coerenti, aumentarlo finché i risultati non saranno stabili. Lo slider "extra" viene utilizzato per ottimizzare determinati profili del picco, ad esempio Pearson, Gaussiano ampliato in modo esponenziale e mix Gaussiana/Lorentziana. Regolarlo per ridurre al minimo l'errore di approssimazione.

Dopo aver effettuato tutte queste impostazioni, si può cliccare sulla casella di controllo **FitPeaks** (riga 13); verrà eseguito un'approssimazione e la tabella dei picchi risultante verrà visualizzata nel pannello di destra, come nel grafico sopra. Successivamente, qualsiasi modifica all'impostazione causerà un immediato ricalcolo dell'approssimazione della curva.

In casi difficili, è possibile ottenere risultati migliori se si specificano le posizioni stimate dei picchi, soprattutto se i picchi sono distanziati in modo molto irregolare o se alcuni picchi appaiono solo come spalle o rigonfiamenti anziché come picchi distinti. Selezionare la casella di controllo **SetStart** e regolare gli slider sulle posizioni di picco relative previste, per ciascun picco nel modello nelle righe da 19 a 26. La lunghezza di questi cursori rappresenta l'intervallo dell'asse x visualizzato nella figura.

Se la linea di base per il gruppo di picchi è spostata dallo zero, la si può correggere utilizzando lo slider **BaselineShift** nella riga 28. Se la linea di base per il gruppo di picchi è inclinata o curva, è possibile utilizzare il menù **BaselineSelection** alla riga 27 per scegliere una correzione della linea di base che tenta di stimare la linea di base dai bordi dell'intervallo del segnale. La casella di controllo **Bipolar** (riga 29) controlla se visualizzare nel grafico sia i valori del segnale positivi che quelli negativi oppure solo i valori positivi.

È possibile aggiungere facilmente profili aggiuntivi al menù **PeakShape** selezionando altre forme dall'elenco dei profili predefiniti e il loro numero corrispondente su <https://terpconnect.umd.edu/~toh/specth/InteractivePeakFitter.htm>, aggiungendo quel nome e numero agli altri nell'istruzione switch/case nelle righe 52-73, aggiungendo poi quel nuovo profilo al menù a discesa nella riga 15. Basta seguire lo schema dei profili già presenti.

Un altro esempio di Peak Fitting Tool mostra l'approssimazione di un gruppo di picchi deboli nel mezzo di un segnale molto più grande ([chrom.csv](#)), in questo caso utilizzando il profilo Gaussiana ampliata in modo esponenziale e la "modalità Tilted" " di correzione della linea di base (riga 27).

Nell'approssimare picchi asimmetrici che hanno un disallineamento esponenziale, si può provare a rimuovere l'asimmetria utilizzando lo slider **SymmetrizeFactor** ([esempio](#)) seguito dall'approssimazione di un profilo di picco simmetrica oppure selezionando un profilo di picco ampliato in modo esponenziale ([esempio](#)); entrambi gli approcci possono produrre risultati simili a quelli di questi esempi, ma il primo metodo è spesso più veloce.

Nota: con un doppio clic su uno qualsiasi degli slider si possono modificare i relativi intervalli se quello iniziale è insufficiente.

Python: un linguaggio alternativo gratuito, open-source

Un'alternativa popolare a Matlab per la programmazione scientifica è [Python](#), che è un linguaggio gratuito e open source, mentre [Matlab](#) è chiuso, proprietario e può risultare costoso. I due linguaggi sono diversi in molti particolari e un programmatore Matlab esperto avrà probabilmente qualche difficoltà a [convertire in Python](#) e viceversa. Ma tutto ciò che è stato fatto in questo libro usando Matlab, lo si può fare in Python. Non si intende fornire una completa introduzione a Python, non più di quanto non sia stato fatto per Matlab o per i fogli di calcolo, perché ci sono molte buone [fonti online](#). Piuttosto, si esamineranno alcuni dei metodi computazionali cruciali, mostrando come eseguire i calcoli in Python e confrontare il codice fianco a fianco con il calcolo equivalente in Matlab. Inoltre, si confronteranno i tempi di esecuzione di entrambi, in esecuzione sullo stesso computer desktop, per vedere se c'è qualche vantaggio in termini di velocità di esecuzione o sulla dimensione del codice in qualche modo. Per questo test, stiamo eseguendo Python 3.8.8; Anaconda Individual Edition 2020.11, utilizzando il desktop Spyder 5.0.5 incluso ([schermata](#)) e Matlab 2021: 9.10.0.1602886 (R2021a), entrambi in esecuzione su un Dell XPS i7 3.5Ghz tower. Sia Matlab che Python/Spyder hanno un editor integrato, un analizzatore del codice, il debugging, la possibilità di modificare le variabili, ecc.

Smoothing a media mobile del segnale

In base all'articolo di Nita Ghosh: [Scientific data analysis with Python: Part 1](#)

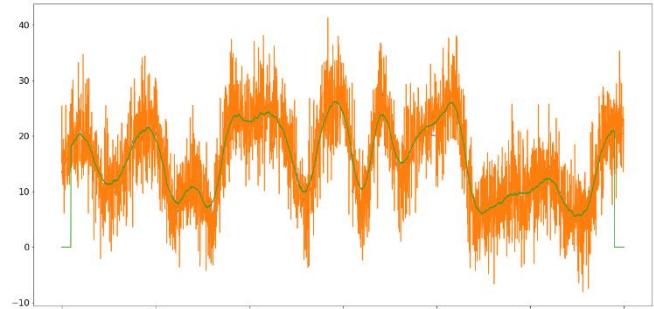
Uno smoothing a “media mobile” è semplice da implementare utilizzando la funzione “mean” nativa in entrambi i linguaggi.

Python

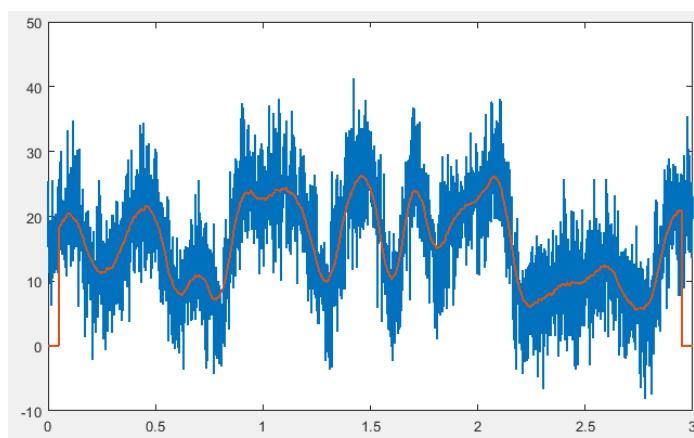
```
import numpy as np
import matplotlib.pyplot as plt
from pyptictoc import TicToc
t = TicToc()
plt.rcParams["font.size"] = 16
plt.rcParams['figure.figsize'] = (20, 10)
sigRate = 1000 #Hz
time = np.arange(0,3, 1/sigRate)
n = len(time)
p = 30 #poles for random interpolation
ampl = np.interp(np.linspace(0,p,n),np.arange(0,p),np.random.rand(p)*30)
noiseamp = 5
noise = noiseamp*np.random.randn(n)
signal = ampl + noise

t.tic() # start clock
plt.plot(time, ampl)
plt.plot(time, signal)
filtSig = np.zeros(n)
k = 50
for i in range(k,n-k-1):
    filtSig[i] = np.mean(signal[i-k:i+k])
plt.plot(time, filtSig)
t.toc() # stop clock print time

# Save signal to file
np.savetxt('SlidingAverageSignal.out',(time, signal), delimiter=',')
```



Nelle prime righe Python deve importare alcuni pacchetti necessari per il calcolo numerico; Matlab ha molti oggetti nativi, ma ci sono delle aggiunte opzionali “[toolbox](#)” (p.es. pag. 126). Il codice Python crea un “segnale” rumoroso simulato nel primo blocco di codice e poi utilizza la funzione “savetext” nativa per salvarlo su disco affinché Matlab possa leggerlo, assicurando che venga utilizzato *lo stesso identico segnale per entrambi*:



Matlab

```
clf
load SlidingAverageSignal.out
x=SlidingAverageSignal(1,:);
y=SlidingAverageSignal(2,:);

tic % start clock
lx=length(x);
filtsig=zeros(1,lx);
k=50; % smooth width
for i=k:lx-k-1
filtsig(i)=mean(y(i-k+1:i +k));
end
plot(x,y,'g',x,filtsig,'linewidth',1.5)
toc % stop clock and print elapsed time
```

La cosa fondamentale qui è l'uso della funzione “mean” in entrambi i linguaggi per calcolare la media di k punti adiacenti per il segnale con smoothing. In Python questo viene fatto dalle righe:

```
k = 50
for i in range(k,n-k-1):
filtSig[i] = np.mean(signal[i-k:i+k])
```

e in Matlab da:

```
k=50;
for i=k:lx-k-1
filtsig(i)=mean(y(i-k+1:i +k));
end
```

Si può vedere quanto sono simili. (Sono stati usati gli stessi nomi di variabili per facilitare il confronto). Una differenza fondamentale è il modo in cui viene specificato un blocco di codice, ad esempio un *ciclo* o una *definizione di funzione*. In Python, questo viene fatto usando l'*indentazione*, che è *applicata rigorosamente*. In Matlab, viene eseguito utilizzando le istruzioni “end”; l'indentazione è facoltativa in Matlab ma, per quanto riguarda uno stile di codifica buono, è facilmente attuabile con la voce “Smart Indent” nel menù contestuale con click destro. (Vedere un esempio di definizione di funzione nell'esempio iterativo dei minimi quadrati a pag. 432). Altre importanti differenze: Python usa le parentesi quadre per racchiudere l'indice negli array anziché le parentesi tonde come fa Matlab. Gli array Python sono indicizzati partendo da *zero*; quelli di Matlab partono da *1*; quindi, per esempio, i primi due elementi di un array A in Python sono A[0] e A[1] ma in Matlab sono A(1) e A(2). Le piccole cose significano molto.

Per confrontare i tempi di esecuzione, sono state usate le istruzioni “tic” e “toc” (native in Matlab ma aggiunte a Python dal pacchetto TicToc) per avviare e arrestare un timer. Sono state messe in **grassetto** quelle righe nel codice, per chiarire che si sta contando e cronometrando solo la parte “interna” del codice che dovrà essere ripetuta se ci sono più set di dati da elaborare. Non viene contato il tempo richiesto dalla configurazione iniziale, dal caricamento dei package, ecc. – cose che devono essere fatte una sola volta.

Python: 8 righe; 0.04 – 0.08 sec
Matlab: 7 righe: 0.007 – 0.008 sec

Entrambi i programmi hanno circa la stessa lunghezza ma Matlab è chiaramente più veloce in questo caso.

Trasformazione di Fourier e (de)convoluzione

La trasformata di Fourier (FT) è fondamentale per calcolare spettri di frequenza, convoluzioni e deconvoluzione. Il codici qui creano semplicemente un vettore "a" di numeri casuali, calcolano la FT, la moltiplicano per se stessa e poi la FT inversa del risultato, utilizzando entrambe le funzioni fft e ifft. Questa è fondamentalmente una convoluzione di Fourier. La deconvoluzione sarebbe la stessa, tranne per il fatto che le due trasformate di Fourier verrebbero *divise*. Come prima, **tic** e **toc** contrassegnano i blocchi misurati.

Python:

```
import numpy as np
from scipy import fft
from pyctictoc import TicToc
t = TicToc()
t.tic() # start clock
min_len = 93059 # la lunghezza principale è il caso peggiore per la velocità
a = np.random.randn(min_len)
b = fft.fft(a)
c=b*b
d=fft.ifft(c)
t.toc() # ferma l'orologio e stampa il tempo trascorso
```

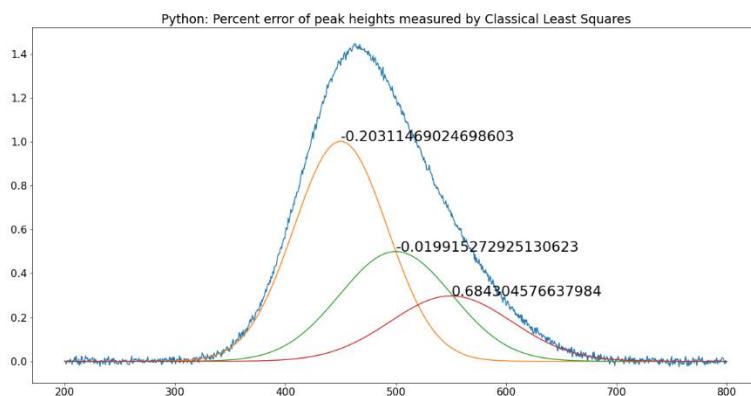
Matlab:

```
tic
MinLen = 93059; % la lunghezza principale è il caso peggiore per la velocità
a=randn(1,MinLen);
b = fft.fft(a);
c=b.*b;
d=ifft(c);
toc
```

Python: 5 righe; 0.01 – 0.04 sec (Il tempo di esecuzione apparentemente varia con le diverse sequenze numeriche casuali).

Matlab: 5 righe: 0.008 – 0.009 sec

I Quadrati Minimi Classici



La tecnica dei minimi quadrati classici (CLS) è stata a lungo utilizzata nell'analisi spettroscopica delle miscele, dove sono noti gli spettri dei singoli componenti ma che si sovrappongono, aggiungendosi, nelle miscele (pag. 179). Un confronto tra la codifica Python e Matlab per questo metodo è fornito a pagina 189. (Prima è stato

scritto il codice Matlab, poi è stato copiato e incollato nell'editor Spyder e infine convertito riga per riga). Dopo l'import necessario dei pacchetti Python, i codici ([NormalEquationDemo.py](#) e [NormalEquationDemo.m](#)) sono notevolmente simili, differendo principalmente nell'uso dell'indentazione, il modo in cui le funzioni sono definite, il modo in cui gli array sono indicizzati, il modo in cui i vettori sono concatenati in matrici e la codifica della trasposizione della matrice, dell'elevamento a potenza e dei prodotti scalari.

Python: 50 righe; 0.017 sec. Matlab: 41 righe. 0.018 sec.

Rilevamento dei picchi

Il rilevamento automatico dei picchi è una richiesta comune. La codifica richiesta è un po' più complessa. I codici sono basati sull'esempio Python nella documentazione di SciPy:

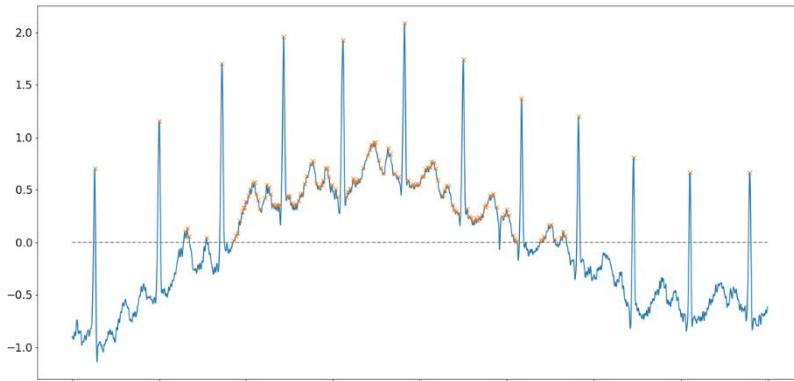
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. Entrambi i linguaggi hanno una funzione "find peaks", ma quella di Matlab è piuttosto lenta, quindi si è preferito codificare l'algoritmo in Matlab usando cicli annidati (ognuno con un'istruzione "end", ma anche indentato per chiarezza). Il segnale usato è una porzione di un elettrocardiogramma incluso come segnale di esempio in Python. Il compito è quello di rilevare i picchi che superano una soglia di ampiezza specificata e contrassegnarli con una "x" rossa sul grafico.

Python:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.misc import electrocardiogram
from scipy.signal import find_peaks
from pytictoc import TicToc
t = TicToc()
x = electrocardiogram()[2000:4000]
t.tic() # start clock
peaks, _ = find_peaks(x, height=0)
plt.plot(x)
plt.plot(peaks, x[peaks], "x")
plt.plot(np.zeros_like(x), "--", color="gray")
plt.show()
t.toc() # ferma l'orologio e stampa il tempo trascorso
np.savetxt('FindPeaks.out',(x), delimiter=',')
```

Matlab:

```
load electrocardiogram.out;
y=electrocardiogram;
tic % start clock
plot(1:length(y),y,'-k')
height=0;
x=1:length(y);
peak=0;
for k = 2:length(x)-1
if y(k) > y(k-1)
if y(k) > y(k+1)
if y(k)>height
peak = peak + 1;
P(peak,:)=[x(k) y(k)];
end
end
end
hold on
for k=1:length(P)
text(P(k,1)-12,P(k,2),'x','color',[1 0 0])
end
grid
hold off
toc % stop clock and print elapsed time
```



Python: 5 righe; 0.01 sec

Matlab: 19 righe: 0.007 sec **come scritto** (1.5 sec utilizzando la funzione findpeaks.m di Matlab)
Qui il vantaggio va a Python, perché la sua funzione "find_peaks" è superiore a quella di Matlab.

Approssimazione ai Quadrati minimi interattiva

(In base al codice di Chris Ostrouchov su https://chrisostrouhov.com/post/peak_fit_xrd_python/)

```

Python:
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
from pytictoc import TicToc
t = TicToc()
def g(x, A, μ, σ):
    return A / (σ * math.sqrt(2 * math.pi)) * np.exp(-(x-μ)**2 / (2*σ**2))
def f(x):
    return np.exp(-(x-2)**2) + np.exp(-(x-6)**2/10) + 1/(x**2 + 1)
A = 100.0 # intensità
μ = 4.0 # media
σ = 4.0 # larghezza del picco
n = 500 # Numero di dati nel segnale
x = np.linspace(-10, 10, n)
y = g(x, A, μ, σ) + np.random.randn(n)

t.tic() # start clock
def cost(parameters):
    g_0 = parameters[:3]
    g_1 = parameters[3:6]
    return np.sum(np.power(g(x, *g_0) + g(x, *g_1) - y, 2)) / len(x)
initial_guess = [5, 10, 4, -5, 10, 4]
result = optimize.minimize(cost, initial_guess)
g_0 = [250.0, 4.0, 5.0]
g_1 = [20.0, -5.0, 1.0]
x = np.linspace(-10, 10, n)
y = g(x, *g_0) + g(x, *g_1) + np.random.randn(n)
fig, ax = plt.subplots()
ax.scatter(x, y, s=1)
ax.plot(x, g(x, *g_0))
ax.plot(x, g(x, *g_1))
ax.plot(x, g(x,*g_0) + g(x,*g_1))
ax.plot(x, y)
t.toc() # ferma l'orologio e stampa il tempo trascorso
np.savetxt('SavedFromPython.out', (x,y), delimiter=',') # Salva il segnale in un file

```

L'approssimazione iterativa è più complessa degli esempi precedenti. Sia Python che Matlab hanno funzioni di ottimizzazione ("optimize.minimize" nel codice Python, sopra; la funzione nativa "fminsearch" nel codice Matlab, sotto). La funzione di ottimizzazione in Python è più flessibile e può utilizzare [diversi metodi di ottimizzazione](#). Matlab utilizza il [metodo Nelder-Mead](#), che viene utilizzato di seguito; vedi pag. 195, ma c'è anche un [toolbox di ottimizzazione](#) opzionale che fornisce metodi alternativi se necessario.

Matlab:

```

clear
clf
load SavedFromPython.out
x=SavedFromPython(1,:);
y=SavedFromPython(2,:);
start=[-5 3 6 3];
format compact
global PEAKHEIGHTS
tic
FitResults=fminsearch(@(lambda)(fitfunction(lambda,x,y)),start);
NumPeaks=round(length(start)./2);
for m=1:NumPeaks
A(m,:)=shapefunction(x,FitResults(2*m-1),FitResults(2*m));
end
model=PEAKHEIGHTS'*A;
plot(x,y,'- r',x,model)
hold on;
for m=1:NumPeaks

```

```

plot(x,A(:, :) *PEAKHEIGHTS(m)) ;
end
hold off
toc % stop clock and print elapsed time
function err = fitfunction(lambda,t,y)
global PEAKHEIGHTS
A = zeros(length(t),round(length(lambda)/2));
for j = 1:length(lambda)/2
A(:,j) = shapefunction(t,lambda(2*j-1),lambda(2*j))';
end
PEAKHEIGHTS = A\y';
z = A*PEAKHEIGHTS;
err = norm(z-y');
end
function g = shapefunction(x,a,b)
g = exp(-((x-a)./(0.6005612.*b)).^2); % Expression for peak shape
end

```

Python: 16 righe; 0.02-0.08 sec

Matlab: 25 righe; 0.01-0.03 sec

L'algoritmo Matlab qui è lo stesso utilizzato dalle funzioni peakfit.m (pag. 384) e ipf.m (pag. 405). I tempi impiegati sia da Matlab che da Python variano con il campione di rumore casuale, ma in questo esempio Matlab ha un vantaggio in termini di velocità di calcolo, che può (o meno) essere significativa nella particolare applicazione.

Un'eccellente introduzione passo passo all'approssimazione iterativa della curva in Python è quella di [Emily Grace Ripka](#).

La conclusione è che Matlab è leggermente più veloce per la maggior parte di queste applicazioni di elaborazione del segnale, ma questo vantaggio potrebbe essere controbilanciato dal fatto che Python è gratuito e open-source, e questo è un forte argomento a suo favore. E Python è certamente più veloce del clone gratuito di Matlab, Octave. Si consiglia questo libro: se si ha un background informatico o si è seguito qualche corso, probabilmente si preferirà Python. Altri potrebbero trovare Matlab più facile da approcciare. Entrambi i linguaggi sono validi, ma è stato usato principalmente Matlab, perché ha così tanti toolbox ben sviluppati e perché funziona e sembra più rifinito e raffinato rispetto a Python/ Spyder o Octave. Dopotutto, *Mathworks Inc* si guadagna da vivere vendendo il costosissimo Matlab, mentre Python è gratuito. Ci deve essere una ragione.

Nota: i chatbot possono convertire programmi da un linguaggio all'altro, ad esempio Matlab in Python o *viceversa*.

Per ulteriori letture, vedere <https://realpython.com/matlab-vs-python/#further-reading>.

Intelligenza Artificiale ed Elaborazione dei Segnali

IA come assistente di un programmatore.

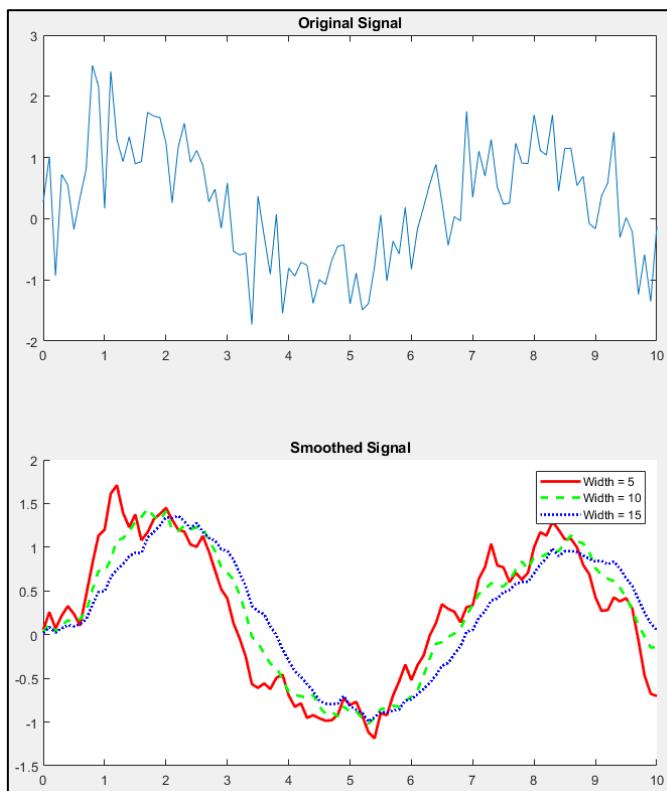
Nel 2022, la società di ricerca e distribuzione dell'intelligenza artificiale OpenAI, ha introdotto un modello conversazionale "large-language" chiamato "[ChatGPT](#)", che è stato addestrato su 8 miliardi di pagine di testo, quasi tutti i libri mai pubblicati, tutti di Wikipedia e siti Web selezionati. (La si può provare gratuitamente su <https://chat.openai.com/>). La versione attuale, a metà del 2024, è ChatGPT-4o. Da quell'introduzione, ci sono stati molti altri chatbot concorrenti, come [Gemini](#) di Google, [CoPilot](#) di Microsoft e [Claude](#) di Anthropic. La forza di questi chatbot sta nell'interpretazione e nella scrittura del linguaggio. Ad esempio, i chatbot come questi sono

abbastanza bravi in compiti semplici come suggerire possibili titoli per documenti, discorsi o proposte su cui sta lavorando. Basta inserire tutto o una parte di ciò che si è scritto. Possono anche parafrasare e sottolineare, il che può essere utile per scrivere riassunti o abstract ridotti. La loro base di conoscenze è straordinariamente ampia e spesso possono rispondere a domande molto specifiche su argomenti tecnici. Ad esempio, i chatbot sono spesso migliori di Google nel rispondere a domande specifiche su programmi hardware o software che altrimenti costringerebbero a sfogliare pagine e pagine di documentazione.

Ma l'intelligenza artificiale può andare ben oltre questo; possono scrivere codice in diversi linguaggi comunemente usati dagli scienziati, come Matlab, Python, Wolfram Mathematica, C/C++, R, Pascal, Fortran, HTML, JavaScript, Macro di Excel, ecc.. Esistono ora diversi servizi AI particolarmente orientati allo sviluppo di codice, tra cui [GitHub Copilot](#), [Replit](#), [Amazon CodeWhisperer](#), [Codex](#) e [Tabine](#). Sulla base di test limitati, ChatGPT può generare codice funzionante in Matlab o Python per applicazioni semplici, *se descrivi adeguatamente il compito*. Nel gennaio 2023, ho eseguito una serie di test in cui ho chiesto a ChatGPT di scrivere il codice per diverse attività di elaborazione del segnale che avevo precedentemente codificato in Matlab. Si scopre che il codice di ChatGPT funziona per alcune semplici attività di elaborazione, se la descrizione è sufficientemente completa, ma per attività più complesse il suo codice spesso non funziona affatto o non fa quello che ci si aspetta. È alquanto fuorviante che, anche nei casi in cui il codice non funziona, sia presentato in buono stile, solitamente con commenti esplicativi per ogni riga, rientri appropriati, esempi d'uso e persino avvertimenti che il codice potrebbe non riuscire in determinate circostanze (ad esempio, divisione per zero, ecc.).

Un semplice esempio in cui ChatGPT funziona bene è scrivere una “funzione che restituisca l'indice e il valore dell'elemento del vettore x che è più vicino allo scalare val . Se più di un elemento di x è ugualmente vicino a val , restituisce vettori di indici e valori”. Questa semplice funzione ma utile viene eseguita dalla mia funzione [val2ind.m](#) in Matlab. Il [codice di ChatGPT](#) è funzionalmente identico alla mia versione ma è migliore in termini di stile: suddivide la mia riga 3 nelle sue parti componenti, include commenti esplicativi per ogni riga e persino fornisce un esempio di utilizzo (anche se per essere onesti il mio codice include tre esempi).

Un altro esempio di successo è l'*Algoritmo di Caruana* (pagina 165), che è un modo veloce per stimare i parametri di un picco di un segnale che è localmente Gaussiano vicino il suo massimo. Ho chiesto a ChatGPT di creare una funzione che "accetta due vettori x e y che si avvicinano a un picco



campionato digitalmente, prende il logaritmo naturale di y , approssima una parabola ai dati risultanti, quindi calcola la posizione, FWHM e l'altezza della Gaussiana dai coefficienti dell'approssimazione parabolica". In questo caso ChatGPT esegue tutta l'algebra richiesta e crea un codice funzionalmente identico alla mia versione codificata manualmente.

È stato chiesto sia a ChatGPT che a CoPilot di Microsoft di “Scrivere uno script Matlab che crei un grafico con due subplot orizzontali, quella superiore che mostra un segnale con diversi picchi e qualche rumore casuale, e quella inferiore che mostra i

risultati dell'applicazione di tre diverse larghezze a quel segnale, ciascuno mostrato con uno stile e un colore di linea diversi, con una legenda che identifica ciascun grafico e la larghezza del suo "smoothing". Il risultato è uno [script funzionante](#) che genera l'immagine mostrata qui, proprio come richiesto, sebbene CoPilot includa solo una delle larghezze dello smoothing nella sua legenda. (Notare che i chatbot sono costretti a fare delle scelte per diverse cose che non sono state esattamente specificate, inclusa la forma dei picchi, la frequenza di campionamento e la lunghezza del segnale e le tre larghezze dello smoothing. Inoltre, aggiunge titoli ad entrambe le subplot, anche se non ho specificato questo dettaglio). È particolarmente utile che, se ad esempio si vuol generare un equivalente Python, si può semplicemente dire "*Come posso farlo in Python*" e verrà creato uno [script Python](#) funzionante che importa le librerie richieste e genera un [grafico quasi identico](#). (Lo stesso potrebbe valere per gli altri linguaggi che conosce, ma non l'ho testato). Si può anche dare un programma scritto in uno dei suoi linguaggi e chiedergli di convertirlo in un altro.

Un'altra query che ha creato un codice funzionale ben strutturato e facilmente modificabile è "Crea un segnale costituito da un picco Gaussiano rumoroso, determinane l'altezza, la posizione e la larghezza del picco mediante l'approssimazione iterativa della curva, ripeti il processo 10 volte con la stessa Gaussiana ma con diversi campioni indipendenti di rumore casuale, poi calcolare la media e la deviazione standard di altezze, posizioni e larghezze dei picchi". [Codice Matlab](#). [Codice Python](#).

Chiaramente chiedere a chatbot di eseguire attività di routine come queste è rapido e conveniente, soprattutto se si sta creando il codice equivalente in più di un linguaggio; sputa il codice più velocemente di quanto si riesca a digitare. Per progetti più grandi e complessi, si potrebbe suddividere il codice in parti o funzioni più piccole che il chatbot possa gestire separatamente e *si possono testare* separatamente, quindi combinarle in seguito secondo necessità.

Un chatbot presenta sempre i suoi risultati ben formattati, con ortografia e grammatica corrette, cosa che molte persone interpretano come un "atteggiamento sicuro". Ciò ispira fiducia nei risultati, ma proprio come per le persone, *sicuro di sé* non significa sempre *corretto*. Ci sono diversi avvertimenti importanti:

Innanzitutto, il codice generato da un chatbot non è necessariamente univoco; se gli si chiede di ripetere il compito, a volte si otterrà un codice diverso (a meno che l'attività non sia così semplice che esiste un solo modo possibile per eseguirlo correttamente). Questo non è necessariamente un difetto; spesso esiste più di un modo per codificare una funzione per uno scopo particolare. Se il codice richiede la definizione di variabili, i nomi di tali variabili saranno scelti da un chatbot e non saranno sempre gli stessi tra una prova all'altra. Inoltre, a meno che non si specifichi il *nome* della funzione stessa, il chatbot sceglierà anche quel nome, in base a ciò che fa la funzione).

In secondo luogo, e cosa ancora più importante, potrebbe esserci più di un modo per *interpretare la richiesta*, a meno che sia stata formulata con molta attenzione in modo che sia inequivocabile. Prendiamo l'esempio dello smoothing dei dati (pagina 38). A prima vista, questo è un processo semplice. Supponiamo di chiedere una funzione che esegua uno smoothing a slittamento-della-media di larghezza n e lo applichi m volte. Come verrà interpretata tale richiesta? Se si dice semplicemente "...applica una media mobile di n punti ai dati y e ripetila m volte", si otterrà [questo codice](#), che fa quanto chiesto ma probabilmente non quello che si vuole. Lo scopo dell'applicazione dello smoothing ripete è applicarlo nuovamente *ai dati precedentemente trattati*, non ai dati *originali*, come fa questo codice. La regola generale è che n passaggi di uno smoothing di larghezza m risultano in uno smoothing con ponderazione centrale di larghezza $n*m-n+1$. Si otterrà quanto desiderato chiedendo a ChatGPT una funzione che "applica uno smoothing a media mobile di n -punti a y e poi lo ripete sui dati già precedentemente trattati con lo smoothing m volte", una differenza piccola ma fondamentale, che risulta in [questo codice](#), mentre il codice precedente

restituisce un risultato di uno smoothing singolarmente indipendentemente dal valore di m .

In terzo luogo, potrebbero esserci dettagli non specificati o effetti collaterali che potrebbero richiedere una gestione, come l'aspettativa che il numero di dati nel segnale *dopo* l'elaborazione debba essere lo stesso di *prima* dell'elaborazione. Nel caso dello smoothing, ad esempio, c'è anche la questione di cosa fare con i primi n e gli ultimi n punti, per i quali non ci sono abbastanza dati per calcolare uno smoothing completo (vedere pagina 41). Esiste anche il requisito che il funzionamento dello smoothing sia costruito in modo tale da non spostare le posizioni dell'asse x delle caratteristiche del segnale, il che è fondamentale in molte applicazioni scientifiche. Gli algoritmi di smoothing codificati dall'uomo, come [fastsmooth](#), considerano tutti questi dettagli.

Un altro esempio di dettagli non specificati è la misura dell'intera larghezza a metà del massimo (FWHM) dei picchi con smoothing di qualsiasi profilo. La funzione che ho scritto per questa attività è "[halfwidth.m](#)". È stata usata la sua descrizione come query ChatGPT: "...una funzione che accetta una serie temporale x,y e calcola l'intera larghezza a metà del massimo del picco in y più vicino a x_0 . Se x_0 viene omesso, calcola la semi-larghezza dal valore del massimo di y ".

L'intelligenza artificiale ha creato "[computeFWHM.m](#)", che funziona bene se la frequenza di campionamento è sufficientemente elevata. Tuttavia, la versione dell'intelligenza artificiale non riesce a eseguire l'interpolazione quando i punti a metà intensità rientrano *tra* i punti dati e quindi è imprecisa quando la frequenza di campionamento è bassa, *perché lo si è specificato*.

"[CompareFWHM.m](#)" confronta entrambe le funzioni su alcuni dati sintetici con frequenza di campionamento regolabile.

Un altro problema tecnico riguarda la pratica comune di Matlab di salvare le funzioni scritte come file separati su disco e poi richiamare successivamente la funzione salvata da uno script che si sta scrivendo, affidandosi a Matlab per trovarla nel path. Se si chiede a ChatGPT di convertire il nuovo script in un altro linguaggio, si devono incorporare le funzioni esterne direttamente nel codice (Matlab R2016b o successivo).

Un compito più impegnativo è l'approssimazione iterativa di picchi rumorosi sovrapposti (pagina 195). A ChatGPT è stato chiesto di "approssimare la somma di n funzioni Gaussiane ai dati forniti in x e in y , date le stime iniziali delle posizioni e delle larghezze di ciascuna Gaussiana, restituendo posizione, larghezza e altezza di tutte le Gaussiane più prossime". ChatGPT ha creato il bel codice di "[iterativefitGaussians.m](#)". L'equivalente codificato manualmente più vicino nella mia cassetta degli attrezzi era [fitshape2.m](#) (pagina 195); entrambi i codici funzionano, hanno all'incirca la stessa lunghezza e richiedono argomenti di input e output simili. Raramente esiste una risposta univocamente corretta ai problemi di approssimazione iterativa, ma la differenza di prestazioni tra questi due codici è istruttiva. Lo script "[DemoiterativefitGaussians2.m](#)" confronta queste due funzioni per un segnale simulato con tre picchi rumorosi i cui parametri reali sono impostati nelle righe 13-15. Per un caso di test "facile", con poca sovrapposizione dei picchi, [entrambi i codici funzionano bene](#). Ma se i picchi si sovrappongono in modo significativo, il codice di ChatGPT [non riesce a fornire una buona approssimazione](#) ma il codice [fitshape2.m](#) funziona. La differenza è probabilmente dovuta alle diverse funzioni di minimizzazione impiegate (lsqcurvefit.m vs fminsearch.m).

Peak fitter iterativi codificati dall'uomo più completi, come la funzione [peakfit.m](#) o il suo equivalente interattivo [ipf.m](#) (pagina 384), sono molto più complessi e contengono diverse migliaia di righe di codice. Sono stati sviluppati in modo incrementale nel tempo e applicati a molti tipi diversi di segnali, con molte funzioni aggiunte e correzioni suggerite dagli utenti. Anche descrivere completamente tali programmi per un chatbot sarebbe, nella migliore delle ipotesi, noioso. Non è realistico aspettarsi che un chatbot sostituisca completamente tali sforzi.

ChatGPT non può scrivere script o app Matlab Live perché non sono file di testo. Tuttavia, si può convertire facilmente uno script generato da ChatGPT in un Live Script, come descritto a pagina 360.

Nei programmi codificati manualmente vengono impiegate molte più riflessioni ed esperienze rispetto a quelli generati dall'intelligenza artificiale. Un programmatore umano esperto conosce la gamma tipica di applicazioni e anticipa i problemi e i limiti tipici, soprattutto quando si applicano al campo di applicazione specifico, come i segnali generati da diversi tipi di strumentazione scientifica. Naturalmente, le IA conoscono molto bene i linguaggi informatici specifici e le loro capacità e funzioni, il che può essere molto utile, soprattutto se si sta ricodificando un algoritmo in un linguaggio nuovo o meno familiare. Ma le IA non sostituiscono l'esperienza umana. Inutile dire che si *deve* testare qualsiasi codice fornito da un chatbot, così come si deve testare il proprio codice.

Sembra ovvio che in futuro i servizi di intelligenza artificiale come i chatbot saranno molto più capaci e più ampiamente disponibili, probabilmente su abbonamento. Lo sviluppo è in corso e milioni di utenti hanno già effettuato l'accesso al server ChatGPT con un account gratuito.

[Come utilizzare ChatGPT per codificare qualsiasi linguaggio di programmazione](#)

[Utilizzo di ChatGPT come assistente alla programmazione](#)

[Come utilizzare ChatGPT per scrivere codice: cosa può e non può fare](#)

Fogli di calcolo per le Curve di Calibrazione Analitiche

Background

In chimica analitica, l'accurata misura quantitativa della composizione dei campioni, ad esempio mediante vari tipi di spettroscopia, richiede solitamente che il metodo sia calibrato utilizzando campioni standard di composizione nota. Questo viene più comunemente, ma non necessariamente, fatto con campioni di soluzione e standard disciolti in un solvente adatto, per la facilità della preparazione e diluizione di miscele accurate e omogenee di campioni e standard in forma di soluzione. Nel metodo della curva di calibrazione (pag. 329), vengono preparate e misurate una serie di soluzioni standard esterne con diverse concentrazioni. Una retta o una curva viene approssimata ai dati e l'equazione risultante viene utilizzata per convertire le letture dei campioni ignoti in concentrazione. I vantaggi di questo metodo sono che (a) gli errori casuali nella preparazione e nella lettura delle soluzioni standard vengono mediati sui diversi standard, e (b) la non linearità nella curva di calibrazione può essere rilevata ed evitata (diluendo nel range lineare) o compensata (utilizzando metodi di approssimazione non lineari della curva).

Di seguito ci sono sette diversi modelli di fogli di calcolo compilabili per eseguire la calibrazione dell'approssimazione della curva [curve fitting] e i calcoli delle concentrazioni per metodi analitici che utilizzano la curva di calibrazione. Tutto quello che serve è digitare (o incollare) le concentrazioni delle soluzioni standard e le loro letture strumentali (ad esempio, assorbanze, intensità o qualsiasi metodo si stia utilizzando) e le letture strumentali delle incognite. Il foglio di calcolo disegna automaticamente e approssima i dati (pag. 154), poi utilizza l'equazione di quella curva per convertire le letture dei campioni ignoti in concentrazione. Si possono aggiungere ed

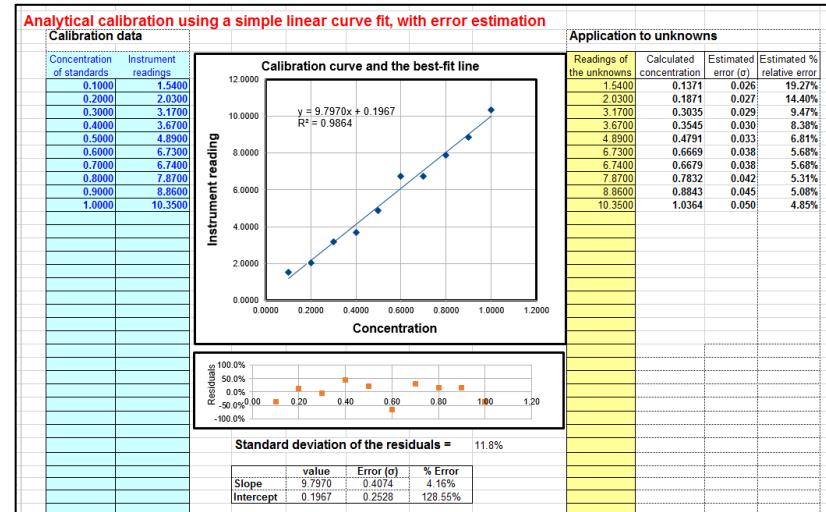
eliminare punti di calibrazione a piacimento, per correggere errori o per rimuovere valori anomali; il foglio automaticamente aggiorna i calcoli e il disegno.

Compilazione dei fogli di calcolo per diversi metodi di calibrazione

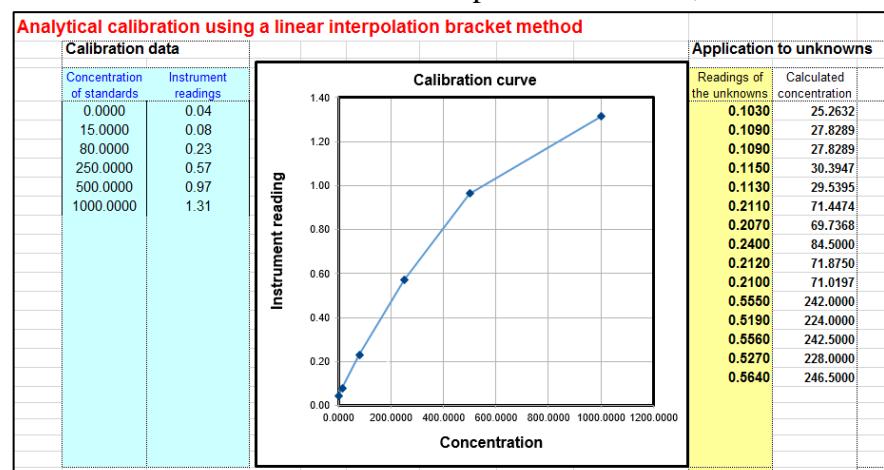
Un'approssimazione del primo ordine (una retta) della misura del segnale A (asse y) rispetto alla concentrazione C (asse x).

L'equazione del modello è $A = \text{pendenza} * C + \text{intercetta}$. Questo è il metodo più comune e diretto ed è quello da utilizzare se *si sa* che la risposta dello strumento è lineare. Questa approssimazione utilizza le equazioni descritte ed elencate a pagina 169. Sono necessari almeno *due* punti sulla curva di calibrazione. La concentrazione di campioni ignoti è data da $(A - \text{intercetta}) / \text{pendenza}$ dove A è il segnale misurato e

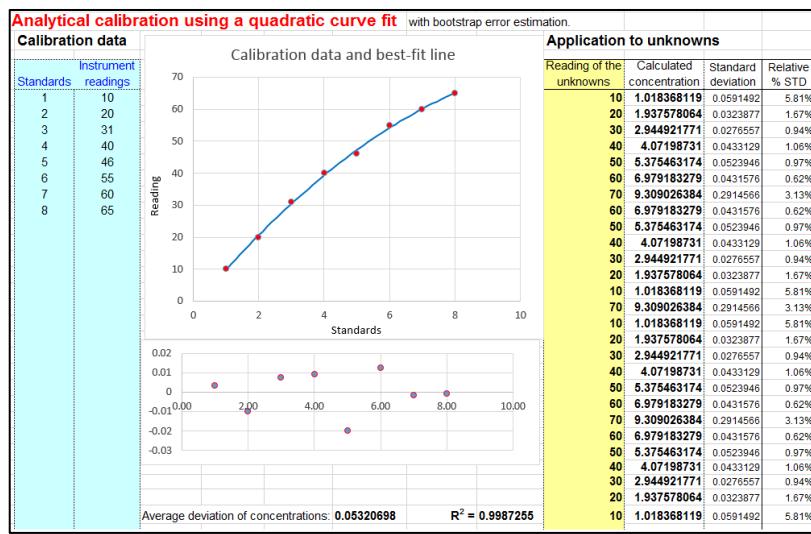
pendenza e *intercetta* provengono dall'approssimazione del primo ordine. Volendo utilizzare questo metodo di calibrazione per i propri dati, scaricare nel formato [Excel](#) o [Calc](#) di OpenOffice. Visualizzare le equazioni per i quadrati minimi [lineare](#).



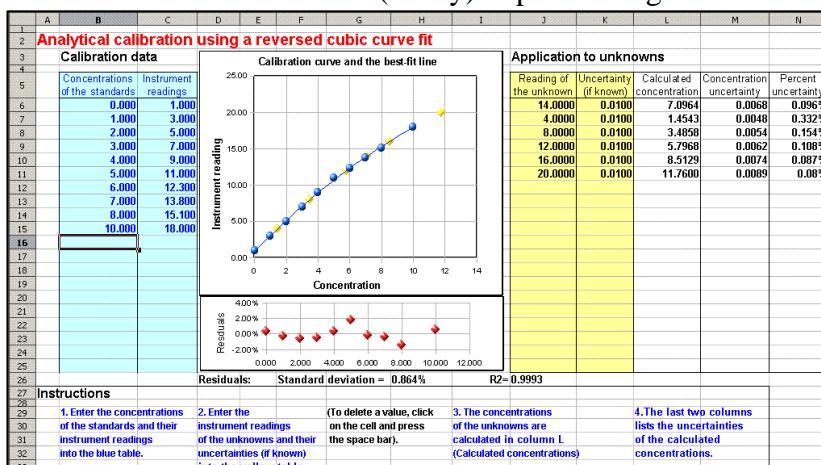
Calibrazione dell'interpolazione lineare. Nel metodo dell'interpolazione lineare, a volte chiamato metodo delle parentesi, il foglio di calcolo esegue una [interpolazione lineare](#) tra i due standard appena sopra e appena sotto ciascun campione ignoto, anziché eseguire un'approssimazione ai quadrati minimi sull'intero set di calibrazione. La concentrazione del campione C_x viene calcolata da $C_{1s} + (C_{2s} - C_{1s}) * (S_x - S_{1s}) / (S_{2s} - S_{1s})$, dove S_{1x} e S_{2x} sono le letture del segnale fornite dai due standard che sono appena sopra e appena sotto il campione ignoto, C_{1s} e C_{2s} sono le concentrazioni di queste due soluzioni standard e S_x è il segnale fornito dalla soluzione campione. Questo metodo può essere utile se nessuno dei metodi dei minimi quadrati può approssimare adeguatamente l'intero intervallo di calibrazione (ad esempio, se contiene due segmenti lineari con pendenze diverse). Funziona abbastanza bene se gli standard sono sufficientemente ravvicinati in modo che la risposta effettiva del segnale non si discosti in modo significativo dalla linearità tra gli standard. Tuttavia, questo metodo non gestisce bene la dispersione casuale nei dati di calibrazione a causa del rumore, perché non calcola una "approssimazione migliore" attraverso più punti di calibrazione come fanno i metodi dei minimi quadrati. Scaricare un template in formato [Excel \(.xls\)](#).



Un'approssimazione quadratica del segnale misurato **A** (asse y) rispetto alla concentrazione **C** (asse x). L'equazione del modello è $\mathbf{A} = a\mathbf{C}^2 + b\mathbf{C} + c$. Questo metodo può compensare la non linearità nella risposta dello strumento alla concentrazione. Questa approssimazione utilizza le equazioni descritte ed elencate a pagina 169. Sono necessari almeno *tre* punti sulla curva di calibrazione. La concentrazione di campioni ignoti viene calcolata risolvendo questa equazione per **C** utilizzando la classica "formula quadratica", ovvero $\mathbf{C} = (-b + \sqrt{b^2 - 4a(c - A)}) / (2a)$, dove **A** = segnale misurato e *a*, *b* e *c* sono i tre coefficienti dell'approssimazione quadratica. Volendo utilizzare questo metodo di calibrazione per i propri dati, scaricare nel formato [Excel](#) o [Calc](#) di OpenOffice. Visualizzare le equazioni per i [quadrati minimi](#) quadratici. La versione alternativa [CalibrationQuadraticB.xlsx](#) calcola la deviazione standard della concentrazione (colonna **L**) e la deviazione standard relativa percentuale (colonna **M**) utilizzando il [metodo bootstrap](#). Sono necessari almeno 5 standard affinché il calcolo dell'errore funzioni. Se si ottiene un "#NUM!" o "#DIV/0" nelle colonne **L** o **M**, basta premere il tasto **F9** per ricalcolare il foglio di calcolo. Esiste anche un template quadratico [e inverso](#) e un [esempio](#), che è analogo all'inversa della cubica (#5 di seguito).



Un'approssimazione cubica inversa della concentrazione **C** (asse y) rispetto al segnale misurato **A** (asse x). L'equazione del modello è $\mathbf{C} = a\mathbf{A}^3 + b\mathbf{A}^2 + c\mathbf{A} + d$. Questo metodo viene talvolta utilizzato per compensare una non linearità più complessa rispetto all'approssimazione quadratica. Una "approssimazione inversa" inverte il normale ordine degli assi, approssimando la concentrazione in funzione del segnale misurato. Lo scopo è quello di evitare la necessità di [risolvere un'e-](#)
[quazione cubica](#) quando l'equazione di calibrazione viene risolta per **C** e utilizzata per convertire i segnali misurati delle incognite nella concentrazione. Questa trasformazione delle coordinate è una scorciatoia, comunemente eseguita nell'approssimazione della curva dei minimi quadrati, almeno da chi non è uno statistico, per evitare confusione matematica quando l'equazione di approssimazione viene risolta per la concentrazione e utilizzata per convertire le letture dello strumento in valori di concentrazione. Tuttavia, questo metodo inverso teoricamente non è ottimale, come dimostrato per il caso quadratico di [simulazione Monte-Carlo](#) nel foglio di calcolo [NormalVsReversedQuad-Fit2.ods](#) ([Screenshot](#)), e dovrebbe essere utilizzato solo se la curva di calibrazione sperimentale è così non lineare da non poter essere approssimata con altri mezzi più semplici. L'approssimazione cubica inversa viene eseguita utilizzando la funzione [LINEST](#) su Sheet3. Sono necessari almeno quattro punti sulla curva di calibrazione. La concentrazione dei campioni noti viene calcolata direttamente da $a\mathbf{A}^3 + b\mathbf{A}^2 + c\mathbf{A} + d$, dove **A** è il segnale misurato e *a*, *b*, *c* e *d* sono i quattro coefficienti dell'approssimazione cubica. I calcoli sono mostrati e spiegati meglio nel template [CalibrationCu-](#)



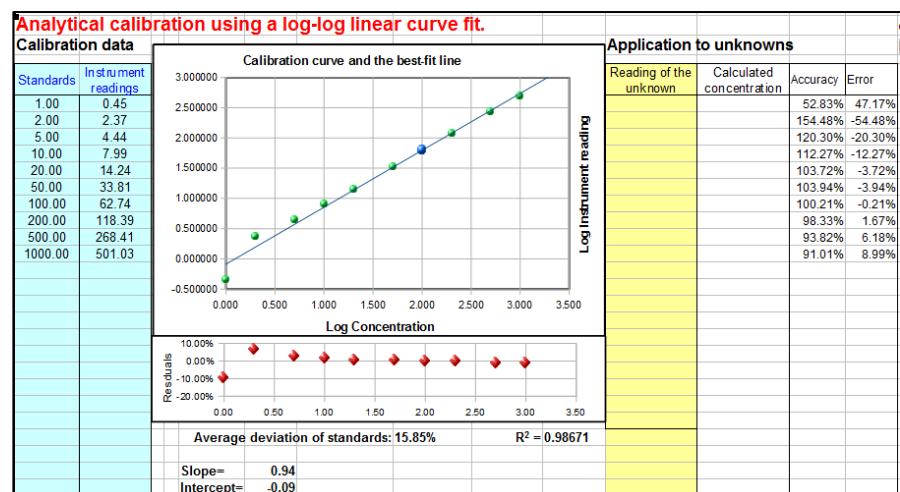
quando l'equazione di calibrazione viene risolta per **C** e utilizzata per convertire i segnali misurati delle incognite nella concentrazione. Questa trasformazione delle coordinate è una scorciatoia, comunemente eseguita nell'approssimazione della curva dei minimi quadrati, almeno da chi non è uno statistico, per evitare confusione matematica quando l'equazione di approssimazione viene risolta per la concentrazione e utilizzata per convertire le letture dello strumento in valori di concentrazione. Tuttavia, questo metodo inverso teoricamente non è ottimale, come dimostrato per il caso quadratico di [simulazione Monte-Carlo](#) nel foglio di calcolo [NormalVsReversedQuad-Fit2.ods](#) ([Screenshot](#)), e dovrebbe essere utilizzato solo se la curva di calibrazione sperimentale è così non lineare da non poter essere approssimata con altri mezzi più semplici. L'approssimazione cubica inversa viene eseguita utilizzando la funzione [LINEST](#) su Sheet3. Sono necessari almeno quattro punti sulla curva di calibrazione. La concentrazione dei campioni noti viene calcolata direttamente da $a\mathbf{A}^3 + b\mathbf{A}^2 + c\mathbf{A} + d$, dove **A** è il segnale misurato e *a*, *b*, *c* e *d* sono i quattro coefficienti dell'approssimazione cubica. I calcoli sono mostrati e spiegati meglio nel template [CalibrationCu-](#)

[bic5Points.xls](#) (schermata), che è impostato per una calibrazione a 5 punti, con i dati campione già inseriti. Per espandere questo modello a un numero maggiore di punti di calibrazione, seguire esattamente questi passaggi: selezionare la **riga 9** (cliccare sull'etichetta della riga "9"), fare clic con il tasto destro e selezionare **Inserisci** e ripetere per ogni ulteriore punto di calibrazione richiesto. Poi si seleziona nella **riga 8** le colonne da **D** a **K** e si trascinano o si copiano verso il basso per riempire le righe appena create. Questo creerà tutte le equazioni richieste e modificherà la funzione LINEST in O16-R20. Esiste anche un altro template, [CalibrationCubic.xls](#), che utilizza alcuni "trucchi" del foglio di calcolo per *rilevare automaticamente il numero di punti di calibrazione* immessi e regolare i calcoli di conseguenza; lo si può scaricare in formato [Excel](#) o OpenOffice [Calc](#).

Calibrazione log-log. Nella calibrazione log-log, il logaritmo del segnale misurato **A** (asse y) viene disegnato rispetto al logaritmo della concentrazione **C** (asse x) e i dati di calibrazione vengono approssimati a un modello lineare o quadratico, come in #1 e #2 sopra. La concentrazione di

campioni sconosciuti si ottiene prendendo il logaritmo delle letture dello strumento, calcolando i logaritmi corrispondenti delle concentrazioni dall'equazione di calibrazione, quindi prendendo l'anti-log per ottenere la concentrazione. (Questi ulteriori passaggi non introducono ulteriori errore matematici, poiché le conversioni del log e dell'anti-log possono essere eseguite rapidamente e senza errori significativi dal computer).

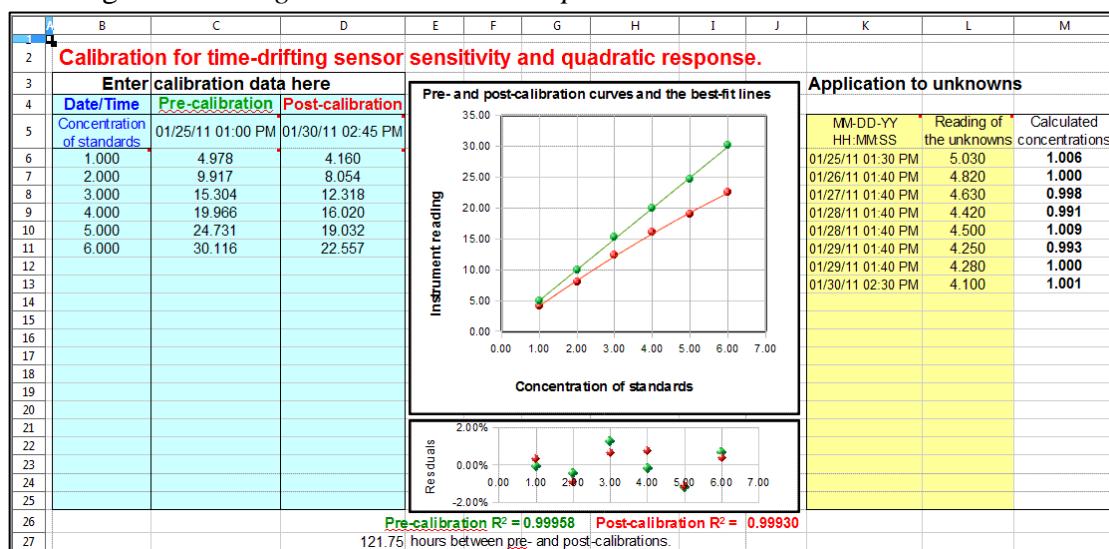
La calibrazione log-log viene talvolta usata per *dati con un intervallo di valori molto ampio* perché distribuisce l'errore di approssimazione relativo in modo più uniforme tra i punti di calibrazione, impedendo ai punti di calibrazione più grandi di dominare e causare errori eccessivi nei punti bassi. In alcuni casi (ad esempio le [relazioni della Legge di Potenza](#)) una relazione non lineare tra segnale e concentrazione può essere completamente linearizzata da una trasformazione log-log. (Alcuni laboratori ufficiali regolamentati dal governo operano secondo regole che [non consentono l'uso di approssimazioni dei minimi quadrati non lineari ai dati di calibrazione](#), partendo dal presupposto che una risposta non lineare sia un sintomo di un guasto dell'apparecchiatura che dovrebbe essere corretto. Ma non vietano i calcolo coi logaritmi, quindi l'uso della trasformazione log-log potrebbe essere d'aiuto in questi casi). Tuttavia, a causa dell'uso dei logaritmi, il set di dati non deve *contenere valori zero o negativi*.



Per utilizzare questo metodo di calibrazione per i propri dati, scaricare i template per log-log lineare ([Excel](#) o [Calc](#)) o il log-log quadratico ([Excel](#) o [Calc](#)).

Calibrazione con correzione della deriva. Tutti i metodi sopra riportati presuppongono che la calibrazione dello strumento sia *stabile nel tempo* e che la calibrazione (solitamente eseguita prima della misura dei campioni) rimanga valida mentre vengono misurati i campioni ignoti. In alcuni casi, tuttavia, strumenti e sensori possono *deviare*, cioè, la *pendenza* e/o l'*intercetta* delle loro curve di calibrazione, e anche la loro *linearità*, possono gradualmente cambiare nel tempo dopo la calibra-

zione iniziale. È possibile verificare questa deriva misurando di nuovo gli standard *dopo* che i campioni sono stati analizzati, per determinare quanto sia diversa la seconda curva di calibrazione dalla prima. Se la differenza non è troppo grande, è ragionevole supporre che la deriva sia approssimativamente lineare col tempo, cioè che i parametri della curva di calibrazione (intercetta, pendenza e curvatura) siano cambiati linearmente in funzione del tempo tra le due calibrazioni. È quindi possibile correggere la deriva se si registra il *tempo* quando viene eseguita ogni calibrazione e quando viene misurato ogni campione ignoto. Il foglio di calcolo per la correzione della deriva (CalibrationDriftingQuadratic): calcola un'approssimazione quadratica per le curve pre e post-calibrazione, quindi utilizza l'interpolazione lineare per stimare i parametri della curva di calibrazione per ogni campione separato in base al tempo in cui è stato misurato. Il metodo funziona perfettamente solo se la deriva è lineare col tempo (un'ipotesi ragionevole se la quantità di deriva non è troppo grande), ma in ogni caso è *meglio che assumere semplicemente l'assenza di deriva*. Per



utilizzare questo metodo di calibrazione per i propri dati, scaricarli nel formato [Excel](#) o OpenOffice [Calc](#). (Vedere le istruzioni a pag. 444)

Calcolo degli errori. In molti casi, è importante calcolare il probabile errore nei valori di concentrazione calcolati (colonna K) causato da una calibrazione imperfetta. Questo è discusso a pagina 159, "[Affidabilità dei risultati dell'approssimazione della curva](#)". Lo spreadsheet della calibrazione lineare (scaricabile in formato [Excel](#) o OpenOffice [Calc](#)) esegue un classico calcolo algebrico di propagazione degli errori (pagina 159) sull'equazione che calcola la concentrazione dal segnale ignoto e la pendenza e l'intercetta della curva di calibrazione. Lo spreadsheet della calibrazione quadratica (Scaricare in formato [Excel](#) o OpenOffice [Calc](#)) esegue un calcolo [bootstrap](#) (pagina 162). È necessario disporre di almeno 5 punti di calibrazione affinché questi calcoli dell'errore siano anche minimamente affidabili; più sono e meglio è. Questo perché tali metodi richiedono un campione rappresentativo di deviazioni dalla linea di calibrazione ideale. Se la linea di calibrazione approssima esattamente i punti, l'errore calcolato sarà zero.

Calibration data	
Concentration of standards	Instrument readings
1	1.83
2	2.59
5	4.21
10	8.17
20	14.56
50	33.83
100	63.14
200	118.43
500	269.2
1000	500.57

Confronto dei metodi di calibrazione

Per confrontare questi vari metodi di calibrazione, si prenderà un set di dati reali e lo si sottoporrà a cinque diversi metodi di approssimazione della curva di calibrazione. Il set di dati, mostrato a lato, ha 10 punti dati che coprono un ampio intervallo (1000) di concentrazioni. Oltre tale intervallo, le letture dello strumento non sono linearmente proporzionali alla concentrazione. Questi dati vengono

utilizzati per costruire una curva di calibrazione, che viene quindi approssimata utilizzando tre diversi modelli, utilizzando i template descritti sopra, e poi le equazioni delle approssimazioni, risolte per la concentrazione, vengono utilizzate per calcolare la concentrazione di ogni standard in base a quella equazione di calibrazione. Per ciascun metodo, si calcola la differenza percentuale relativa tra la concentrazione effettiva di ogni standard e le concentrazioni calcolate dalle curve di calibrazione. Quindi si calcola la media di quegli errori per ogni metodo. L'obiettivo di questo esercizio è determinare quale metodo fornisce l'errore medio più basso per tutti i 10 standard *in questo set di dati*.

I tre metodi utilizzati e i relativi errori percentuali sono: (1) [lineare](#), errore 196% ; (2) [quadratico](#), errore del 50% e (3) [log-log lineare](#), errore del 20%. [ComparisonOfCalibrations.xlsx](#) riassume i risultati. Per questo set di dati, il metodo migliore è il log-log lineare, ma ciò non significa che questo metodo sarà sempre il migliore in ogni situazione. Questi dati di calibrazione non sono lineari e coprono una gamma molto ampia di valori x (concentrazioni), il che rappresenta una sfida per la maggior parte dei metodi di calibrazione.

Istruzioni per l'utilizzo dei template per la calibrazione

1. Scaricare e aprire il foglio di lavoro per la calibrazione desiderata tra quelli [elencati sopra](#) (pag. 439).
2. Immettere le concentrazioni degli standard e le letture dello strumento (ad esempio, l'assorbanza) nella tabella blu a sinistra. Lasciare il resto della tabella vuoto. È necessario disporre di almeno due punti sulla curva di calibrazione (tre punti per il metodo quadratico o quattro punti per il metodo cubico), compreso il bianco (standard a concentrazione zero). Se si dispone di più letture di strumenti per uno standard, è meglio inserire ciascuna di esse come standard separato con la stessa concentrazione, piuttosto che inserire la media. Il foglio di calcolo attribuisce automaticamente più peso agli standard che hanno più di una lettura.
3. Immettere le letture dello strumento (ad esempio, l'assorbanza) delle incognite nella tabella gialla a destra. Si possono avere fino a 20 incognite. (Se si dispone di più letture di strumenti per un'incognita, è meglio inserirle come incognite separate, piuttosto che fare la media, in modo da poter vedere quanta variazione nella concentrazione calcolata viene prodotta dalla variazione nella lettura dello strumento).
4. Le concentrazioni delle incognite vengono calcolate automaticamente e visualizzate nella colonna K. Se si modifica la curva di calibrazione, eliminando, cambiando o aggiungendo più standard di calibrazione, le concentrazioni vengono ricalcolate automaticamente.

Per l'approssimazione lineare (CalibrationLinear.xls), se si dispone di tre o più punti di calibrazione, la deviazione standard stimata della pendenza e dell'intercetta verrà calcolata e visualizzata nelle celle **G36** e **G37**, e la deviazione standard (SD) risultante di ciascuna concentrazione verrà visualizzata nelle righe **L** (SD assoluta) e **M** (SD percentuale relativa). Questi calcoli della deviazione standard sono stime della variabilità delle pendenze e delle intercette che si potrebbero ottenere se si ripete la calibrazione più e più volte nelle stesse condizioni, assumendo che le deviazioni dalla linea retta siano dovute a *variabilità casuale* e non a un errore sistematico causato dalla non linearità. Se le deviazioni sono casuali, di volta in volta saranno leggermente diverse, causando la variazione della pendenza e dell'intercetta da misura a misura. Tuttavia, se le deviazioni sono causate da non linearità sistematica, saranno le stesse da misura a misura, nel qual caso queste previsioni della deviazione standard non saranno rilevanti e sarebbe meglio utilizzare un'approssimazione polinomiale come un quadratico o cubico. L'affidabilità di queste stime della

deviazione standard dipende anche dal numero di punti nell'approssimazione della curva; migliorano con la radice quadrata del numero di punti.

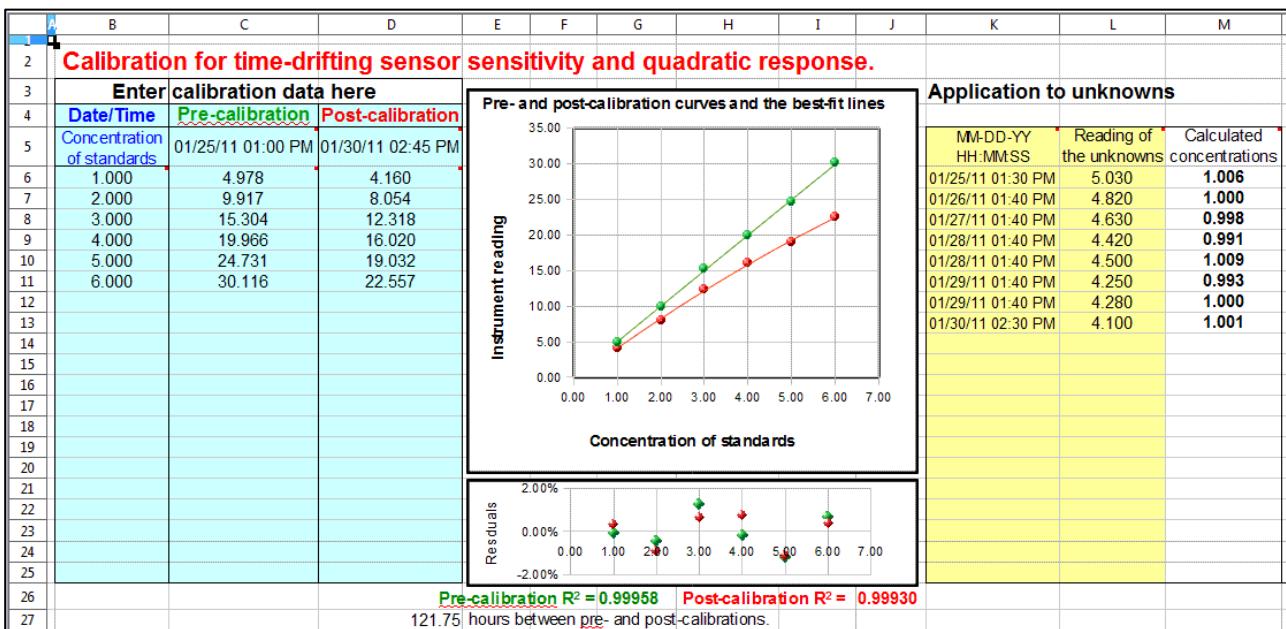
5. È possibile rimuovere qualsiasi punto dall'approssimazione della curva eliminando i valori X e Y corrispondenti nella tabella. Per eliminare un valore; fare clic con il tasto destro sulla cella e cliccare su "Delete Contents" o "Clear Contents". Lo spreadsheet ricalcola automaticamente e il grafico viene aggiornato; in caso contrario, premere F9 per ricalcolare. (Nota: il foglio di calcolo della calibrazione cubica deve avere punti di calibrazione contigui senza celle vuote nell'intervallo di calibrazione).

6. Lo spreadsheet della calibrazione lineare calcola anche il coefficiente di determinazione, R^2 , che è un indicatore della "bontà dell'approssimazione", nella cella **C37**. R^2 è 1.0000 quando l'approssimazione è perfetta ma è inferiore quando l'approssimazione è imperfetta. Più ci si avvicina a 1.0000 meglio è.

7. Un "grafico dei residui" viene visualizzato appena sotto il grafico di calibrazione (ad eccezione del metodo di interpolazione). Questo mostra la differenza tra la curva di calibrazione più adatta e le letture effettive degli standard. Più piccoli sono questi errori, più la curva approssima gli standard di calibrazione. (La deviazione standard di questi errori viene calcolata e visualizzata anche sotto il grafico dei residui; minore è questa deviazione standard, meglio è).

Si può dire molto guardando la forma del diagramma dei residui: se i punti sono sparsi in modo casuale sopra e sotto lo zero, significa che l'approssimazione della curva è *la migliore possibile*, dato il rumore casuale nei dati. Ma se il grafico dei residui ha una forma regolare, ad esempio una curva a forma di U, allora significa che c'è una mancata corrispondenza tra l'approssimazione della curva e la forma effettiva della curva di calibrazione; suggerendo che si potrebbe provare un'altra tecnica di approssimazione della curva (ad esempio, un'approssimazione quadratica o cubica piuttosto che lineare) o che le condizioni sperimentali vengano modificate per produrre una forma della curva di calibrazione sperimentale meno complessa.

8. **Calibrazione con correzione della deriva.** Se si usa il foglio di calcolo per la *calibrazione con correzione della deriva*, si devono misurare *due* curve di calibrazione, una *prima* e una *dopo* aver eseguito i campioni e si devono registrare i dati e il tempo di ciascuna curva di calibrazione misurata. Immettere le concentrazioni degli standard nella colonna **B**. Inserire le letture dello strumento per la prima (pre-) calibrazione nella colonna **C** e la data/ora di questa calibrazione nella cella **C5**; inserire le letture dello strumento per la post-calibrazione nella colonna **D** e la data/ora della calibrazione nella cella **D5**. Il formato per la voce di data/ora è **Mese-Giorno-Anno Ore:Minuti:Secondi**, per esempio, 6-2-2011 13:30:00 per il 2 giugno 2011, 1:30 PM (13:30 su un orologio a 24-ore). Nota: se eseguono entrambe le calibrazioni nello stesso giorno, si può tralasciare la data e inserire semplicemente l'ora. Nel grafico, la curva di pre-calibrazione è in **verde** e quella di post-calibrazione in **rosso**. Poi, per ogni campione ignoto misurato, si inserisce la data/ora (nello stesso formato) nella colonna **K** e la lettura dello strumento per quella nota nella colonna **L**. Il foglio di calcolo calcola le concentrazioni del campione, correggendo la deriva, nella colonna **M**. Nota: La versione 2.1 di questo foglio di calcolo (luglio 2011) consente diversi set di concentrazioni per le pre e post calibrazioni. Basta elencare tutte le concentrazioni utilizzate nella colonna "Concentration of standards" colonna (**B**) e inserire le letture dello strumento corrispondenti nelle colonne **C o D**, o *entrambe*. Se non si utilizza una particolare concentrazione per una delle calibrazioni, lasciare in bianco la lettura dello strumento.



Questa figura mostra un'applicazione del foglio di calcolo di calibrazione [quadratica con correzione della deriva](#), a un esperimento di telerilevamento. In questa dimostrazione, le calibrazioni e le misure sono state effettuate in un periodo di diversi giorni. La pre-calibrazione (colonna C) è stata eseguita con sei standard (colonna B) il 25/01/2011 alle 13:00. Otto campioni ignoti sono stati misurati nei cinque giorni seguenti (colonne L e M), e la post-calibrazione (colonna D) è stata eseguita dopo l'ultima misura il 30/01/2011 alle 14:45. Il grafico al centro mostra la curva di pre-calibrazione in verde e la curva di post-calibrazione in rosso. Come si può vedere, il sensore (o lo strumento) si è spostato in quel periodo di tempo, la sensibilità (pendenza della curva di calibrazione) è diminuita del 28% e la curvatura è diventata notevolmente meno lineare (concava verso il basso). Ciò potrebbe essere stato causato in questo caso dall'accumulo di sporcizia e dalla crescita di alghe sul sensore col passare del tempo. Qualunque sia la causa, entrambe le curve di pre e post calibrazione si approssimano molto bene alle equazioni di calibrazione quadratiche, come indicato dal grafico dei residui e dai coefficienti di determinazione (R²) con "3 nove" indicati sotto i grafici. Gli otto campioni "ignoti" misurati per questo test (tabella gialla) erano *lo stesso campione misurato ripetutamente* - uno standard di concentrazione di 1.00 unità - ma si può vedere che il campione ha dato letture dello strumento inferiori (colonna L) ad ogni misura (colonna K), a causa della deriva. Infine, le concentrazioni corrette per la deriva calcolate dal foglio di calcolo (colonna M a destra) sono tutte molto vicine a 1.00, con una deviazione standard dello 0.6%, a dimostrazione che la correzione della deriva funziona bene, entro i limiti del rumore casuale nelle letture dello strumento e subordinatamente all'assunzione che la deriva nei parametri della curva di calibrazione sia lineare con il tempo tra la pre e la post-calibrazione.

Domande frequenti (tratte da email e query sui motori di ricerca)

1. Domanda: Qual è lo scopo della curva di calibrazione?

Risposta: La maggior parte degli strumenti analitici genera un segnale di uscita elettrico come una corrente o una tensione. Una curva di calibrazione stabilisce la relazione tra il segnale generato da uno strumento di misurazione e la concentrazione della sostanza misurata. Diversi composti ed elementi chimici danno segnali diversi. Quando viene misurato un campione sconosciuto, il segnale di tale campione viene convertito in concentrazione utilizzando la curva di calibrazione.

2. Domanda: Come si crea una curva di calibrazione?

Risposta: Si prepara una serie di "soluzioni standard" della sostanza che si intende misurare, se ne

misura il segnale (es. l'assorbanza, se si sta facendo spettrofotometria di assorbimento) e si disegna la concentrazione sull'asse x e il segnale misurato per ogni standard sull'asse y. Si traccia una linea retta il più vicino possibile ai punti sulla curva di calibrazione (o una curva regolare se non si approssima ad una linea retta), in modo che il maggior numero possibile di punti si trovi proprio sopra o vicino alla curva.

3. Domanda: *Come si utilizza una curva di calibrazione per predire la concentrazione di un campione sconosciuto? Come si determina la concentrazione da un grafico di calibrazione non lineare?*

Risposta: Lo si può fare in due modi, graficamente e matematicamente. Graficamente, si traccia una linea orizzontale dal segnale dell'ignoto sull'asse y fino alla curva di calibrazione e poi verso il basso fino all'asse della concentrazione (x) fino alla concentrazione dell'ignoto. Matematicamente, approssimare un'equazione ai dati di calibrazione e risolvere l'equazione per la concentrazione in funzione del segnale. Poi, per ogni ignoto, si inserisce il segnale in questa equazione e si calcola la concentrazione. Ad esempio, per un'equazione lineare, l'equazione di approssimazione dei minimi quadrati della curva è **Segnale** = *pendenza* * **Concentrazione** + *intercetta*, dove *pendenza* e *intercetta* vengono determinati da un'[approssimazione dei minimi quadrati](#) lineare (primo ordine) ai dati di calibrazione. Risolvendo questa equazione per la **Concentrazione** si ottiene **Concentrazione** = (**Segnale** - *intercetta*) / *pendenza*, dove **Segnale** è il segnale letto (ad esempio, l'assorbanza) della soluzione sconosciuta. ([Cliccare qui](#) per uno spreadsheet OpenOffice da riempire che fa questo lavoro. [Visualizzare la schermata](#)).

4. Domanda: *Come faccio a sapere quando utilizzare un'approssimazione della curva con una linea retta e quando utilizzare una curva di tipo quadratico o cubica?*

Risposta: Approssimare una linea retta ai dati di calibrazione e guardare il grafico dei "residui" (le differenze tra i valori y nei dati originali e i valori y calcolati dall'equazione di approssimazione). Le deviazioni dalla linearità saranno molto più evidenti nel grafico dei residui che in quello della curva di calibrazione. ([Cliccare qui](#) per uno spreadsheet OpenOffice da riempire che fa questo lavoro. [Visualizzare la schermata](#)).

Se i residui sono sparsi in modo casuale lungo tutta la linea dell'approssimazione, significa che le deviazioni sono causate da errori casuali come il rumore dello strumento o da errori casuali volumetrici o procedurali; in tal caso è possibile utilizzare un'approssimazione lineare (lineare). Se i residui hanno una forma regolare, come una forma a "U", significa che la curva di calibrazione è curva e si dovrebbe usare un'approssimazione non lineare, come un'[approssimazione quadratico o cubica](#). Se il grafico residuo ha una forma a "S", probabilmente si dovrebbe usare un'approssimazione cubica. (Se si esegue la spettrofotometria di assorbimento, vedere [Confronto dei Metodi di Approssimazione della Curva nella Spettroscopia di Assorbimento](#)).

5. Domanda: *Cosa succede se la curva di calibrazione è lineare a basse concentrazioni ma si abbassa alle concentrazioni più alte?*

Risposta: In questo caso non è possibile utilizzare un'approssimazione lineare della curva, ma se la curvatura non è troppo evidente, si potrebbe essere in grado di ottenere un buon risultato con un'[approssimazione quadratico o cubica](#). In caso contrario, si può suddividere l'intervallo di concentrazione in due regioni e approssimare una curva lineare alla regione lineare inferiore e una curva quadratico o cubica alla regione non lineare superiore.

6. Domanda: *Qual è la differenza tra una curva di calibrazione e una linea di approssimazione migliore? Qual è la differenza tra un'approssimazione lineare e una curva di calibrazione?*

Risposta: La curva di calibrazione è una relazione misurata sperimentalmente tra concentrazione e segnale. Non si conosce mai veramente la *vera* curva di calibrazione; la si può solo *stimare* in pochi punti misurando una serie di soluzioni standard. Poi si disegna una linea o una curva regolare che

attraversi il più possibile i punti, con alcuni punti un po' più alti della linea e altri un po' più bassi. Questo è ciò si intende con "approssimazione migliore" ai dati. La curva di calibrazione effettiva potrebbe non essere perfettamente lineare, quindi un'approssimazione lineare non è sempre la migliore. Un'approssimazione quadratica o cubica potrebbe essere migliore se la curva di calibrazione mostra una curvatura uniforme e regolare.

7. Domanda: *Perché la linea inclinata non attraversa tutti i punti di un grafico?*

Risposta: Questo accadrà solo se (1) si è dei perfetti sperimentatori, (2) si ha uno strumento perfetto e (3) si sceglie l'equazione di approssimazione della curva perfetta per i dati. Non succederà. Ci sono *sempre* dei piccoli errori. Il metodo di approssimazione dei minimi quadrati produce un'approssimazione *migliore*, non *perfetta*, ai dati di calibrazione per una data forma della curva (lineare, quadratica, o cubica). Si presume che i punti che non cadono sulla curva lo facciano a causa di errori casuali o perché la forma effettiva della curva di calibrazione non corrisponde all'equazione di approssimazione della curva.

C'è un trucco per far passare la curva attraverso tutti i punti, e cioè usare *troppto pochi standard di calibrazione*: per esempio, se si usano solo *due* punti per un'approssimazione in linea retta, la retta passerà ovviamente attraverso quei due punti *sempre*. Allo stesso modo, se si utilizzano solo *tre* punti per un'approssimazione quadratica, la migliore approssimazione quadratica passerà attraverso questi tre punti e se si usano solo *quattro* punti per un'approssimazione cubica, questa passerà attraverso quei quattro punti. Ma questo non è molto raccomandato, perché se uno dei punti di calibrazione è fuori posto per un errore esagerato, l'approssimazione della curva *sarà comunque perfetta e non ci saranno indicazioni* sul fatto che qualcosa non va. *In realtà, si devono usare più standard in modo da sapere quando qualcosa è andato storto.*

8. Domanda: *Cosa succede quando la lettura dell'assorbanza è maggiore di una qualsiasi delle soluzioni standard?*

Risposta: Se si usa un'equazione di approssimazione alla curva, si otterrà comunque un valore di concentrazione calcolato per *qualsiasi* lettura del segnale, anche al di sopra dello standard più elevato. Tuttavia, è rischioso farlo, perché in realtà non si conosce con certezza quale sia la forma della curva di calibrazione al di sopra dello standard più elevato. Potrebbe proseguire dritto, o potrebbe curvare in qualche modo inaspettato - come si fa ad esserne sicuri? È meglio aggiungere un altro standard all'estremità superiore della curva di calibrazione.

9. Domanda: *Qual è la differenza tra l'utilizzo di un unico standard e più standard?*

Risposta: Il metodo a standard unico è quello più semplice e veloce, ma è accurato solo se si sa che la curva di calibrazione è lineare. L'utilizzo di più standard ha il vantaggio che qualsiasi non linearità nella curva di calibrazione può essere rilevata ed evitata (diluendola nell'intervallo lineare) o compensandola (utilizzando metodi di approssimazione della curva non lineare). Inoltre, gli errori casuali nella preparazione e nella lettura delle soluzioni standard sono mediati su diversi standard, il che è meglio che "mettere tutte le uova nello stesso paniere" con un unico standard. D'altra parte, un ovvio svantaggio del metodo a standard multipli è che richiede molto più tempo e utilizza più materiale standard rispetto al metodo con uno standard unico.

10. Domanda: *Qual è la relazione tra la sensibilità in analisi e la pendenza della curva standard?*

Risposta: La sensibilità è definita come la pendenza della curva (calibrazione) standard.

11. Domanda: *Come si crea una curva di calibrazione in Excel o in OpenOffice?*

Risposta: Si mette la concentrazione degli standard in una colonna e i loro segnali (ad esempio, le assorbanze) in un'altra colonna. Poi si crea un grafico a dispersione XY, mettendo la concentrazione sull'asse X (orizzontale) e il segnale sull'asse Y (verticale). I punti si disegnano solo con simboli,

senza le linee tra i punti. Per calcolare un'approssimazione dei minimi quadrati, si può o inserire le [equazioni dei minimi quadrati](#) nello spreadsheet, oppure si può utilizzare la funzione LINEST nativa sia in [Excel](#) che in [OpenOffice Calc](#) per calcolare il polinomio e le altre approssimazioni curvilinee dei minimi quadrati. Per degli esempi di fogli di calcolo OpenOffice che rappresentano graficamente e approssimano le curve di calibrazione, vedere [Fogli di calcolo per le Curve di Calibrazione Analitiche](#).

12. Domanda: *Qual è la differenza nell'uso di una curva di calibrazione nella spettrometria di assorbimento rispetto ad altri metodi analitici come la spettroscopia di fluorescenza o di emissione?*

Risposta: L'unica differenza sono le unità del segnale. Nella spettroscopia di assorbimento si utilizza l'*assorbanza* (perché è quella più lineare con la concentrazione) e nella spettroscopia di fluorescenza (o di emissione) si utilizza l'*intensità della fluorescenza (o emissione)*, che solitamente è lineare con la concentrazione (tranne a volte ad alte concentrazioni). I metodi di approssimazione della curva e di calcolo della concentrazione sono sostanzialmente gli stessi.

13. Domanda: *Se la soluzione obbedisce alla legge di Beer, è meglio usare una curva di calibrazione piuttosto che un unico standard?*

Risposta: Potrebbe non fare molta differenza in entrambi i casi. Se si riconosce, da misure precedenti, che la soluzione obbedisce alla legge di Beer esattamente sullo stesso spettrofotometro e nelle condizioni in uso, è possibile utilizzare un unico standard (sebbene sia meglio se tale standard fornisca un segnale prossimo al segnale campione massimo previsto o comunque a qualsiasi segnale che fornisca il miglior rapporto segnale/rumore - un'assorbanza prossima a 1,0 nella spettroscopia di assorbimento). L'unico vero vantaggio di più standard in questo caso è che gli errori casuali nella preparazione e nella lettura delle soluzioni standard vengono mediati su più standard, ma lo stesso effetto si può ottenere più semplicemente creando più copie dello stesso singolo standard (per mediare gli errori volumetrici casuali) e leggerli separatamente (per calcolare la media degli errori casuali della lettura dei segnali). E se gli errori di lettura del segnale sono molto inferiori quelli volumetrici, si può ripetutamente misurare una *singola* soluzione standard per mediare gli errori casuali delle misure.

14. Domanda: *Qual è l'effetto sulla misura della concentrazione se il monocromatore non è perfetto?*

Risposta: Se la calibrazione della lunghezza d'onda viene disattivata per un po', non si avrà alcun effetto significativo se l'impostazione del monocromatore non viene modificata tra la misura degli standard e il campione ignoto; la pendenza della curva di calibrazione sarà diversa, ma le concentrazioni calcolate saranno OK. (Ma se qualcosa cambia la lunghezza d'onda tra il momento in cui si misurano gli standard e il tempo in cui misurano i campioni, si verificherà un errore). Se la lunghezza d'onda viene influenzata della luce parassita o se la risoluzione è scarsa (la banda passante spettrale è troppo grande), la curva di calibrazione può essere influenzata negativamente. Nella spettroscopia di assorbimento, la luce diffusa e la scarsa risoluzione possono causare una non linearità, che richiede un metodo di approssimazione della curva non lineare. Nella spettroscopia di emissione, la luce diffusa e la scarsa risoluzione possono provocare un'interferenza spettrale che può provocare errori analitici significativi.

15. Domanda: *Cosa significa se l'intercetta dell'approssimazione della curva di calibrazione non è zero?*

Risposta: Idealmente, l'intercetta sull'asse y della curva di calibrazione (il segnale a concentrazione zero) dovrebbe essere zero, ma ci sono diversi motivi per cui potrebbe non essere così. (1) Se c'è una sostanziale dispersione casuale nei punti di calibrazione sopra e sotto la linea di approssimazione migliore, è probabile che l'intercetta diversa da zero sia semplicemente dovuta a un errore casuale. Preparando un altro set di standard, quella curva avrebbe un'intercetta diversa,

positiva o negativa. Non c'è niente che si possa fare al riguardo, a meno che non si sia in grado di ridurre l'errore casuale degli standard e dei campioni. (2) Se il profilo della curva di calibrazione non corrisponde a quello dell'approssimazione, è molto probabile che si otterrà ogni volta un'intercetta diversa da zero. Ad esempio, se la curva di calibrazione si inclina all'aumentare della concentrazione e si utilizza un'approssimazione con una linea retta (lineare), l'intercetta sarà positiva (ovvero, la linea di approssimazione della curva avrà un'intercetta positiva sull'asse y, anche se la curva di calibrazione effettiva passa per lo zero). Questo è un artefatto dovuto alla cattiva scelta dell'approssimazione della curva; se si nota che succede, si provi un profilo diverso della curva (quadratico o cubico). (3) Se lo strumento non è "azzerato" correttamente, in altre parole, se lo strumento fornisce una lettura diversa da zero quando viene misurata la [soluzione vuota](#). In questo caso si hanno tre possibilità: si può azzerare lo strumento (se possibile); è possibile sottrarre il segnale della soluzione vuota da tutte le letture standard e i campioni; oppure si può semplicemente lasciare che l'approssimazione della curva sottragga l'intercetta (se la procedura di approssimazione della curva calcola l'intercetta e la si tiene nella soluzione di quell'equazione, ad esempio, Concentrazione = $(\text{Segnale} - \text{intercetta}) / \text{pendenza}$).

16. Domanda: *Come si può ridurre la dispersione casuale dei punti di calibrazione sopra e sotto la linea di approssimazione migliore?*

Risposta: Errori casuali come questo potrebbero essere dovuti a errori volumetrici casuali (piccoli errori nei volumi utilizzati per preparare la soluzione standard diluendo dalla soluzione concentrata [stack] o aggiungendo i reagenti) o potrebbero essere dovuti a errori casuale di lettura del segnale dello strumento, o ad entrambi. Per ridurre l'errore volumetrico, si usa un'attrezzatura volumetrica più precisa e si fa un po' di pratica per perfezionare la propria tecnica (ad esempio, usando la propria tecnica per erogare acqua pura e pesarla su una bilancia analitica precisa). Per ridurre l'errore di lettura del segnale, regolare le condizioni dello strumento (ad esempio, lunghezza d'onda, lunghezza del percorso, larghezza della fenditura, ecc.) per il miglior rapporto segnale-rumore e mediare le diverse letture di ogni campione o standard.

17. Domanda: *Cosa sono le interferenze? Che effetto hanno le interferenze sulla curva di calibrazione e sull'accuratezza della misura della concentrazione?*

Risposta: Quando un metodo analitico viene applicato a campioni complessi del mondo reale, ad esempio la determinazione di farmaci nel siero del sangue, può verificarsi un errore di misura a causa delle *interferenze*. Le interferenze sono errori di misura causati da componenti chimici nei campioni che influenzano il segnale misurato, ad esempio contribuendo con i propri segnali o riducendo o aumentando il segnale dall'analita. Anche se il metodo è ben calibrato ed è in grado di misurare accuratamente soluzioni di analita puro, possono verificarsi errori di interferenza quando il metodo viene applicato a campioni complessi del mondo reale. Un modo per correggere le interferenze è utilizzare gli "standard a matrice abbinata", soluzioni standard preparate per contenere *tutto ciò che i campioni reali contengono*, tranne che hanno concentrazioni note di analita. Ma questo è molto difficile e costoso da fare esattamente, quindi viene fatto ogni sforzo per ridurre o compensare le interferenze in altri modi. Per ulteriori informazioni sui tipi di interferenze e sui metodi per compensarle, vedere [Confronto dei metodi di calibrazione analitica](#).

18. Domanda: *Quali sono le fonti di errore nella preparazione di una curva di calibrazione?*

Risposta: Una curva di calibrazione è un grafico del segnale analitico (ad esempio, assorbanza, nella spettrofotometria di assorbimento) rispetto alla concentrazione delle soluzioni standard. Pertanto, le principali fonti di errore sono quelli nelle concentrazioni standard e quelli nei loro segnali misurati. Gli errori di concentrazione dipendono principalmente dalla precisione della vetreria volumetrica (matracci volumetrici, pipette, dispositivi di erogazione della soluzione) e dalla

precisione del loro utilizzo da parte delle persone che preparano le soluzioni. In generale, l'accuratezza e la precisione della manipolazione di grandi volumi superiori a 10 mL sono maggiori di quella di volumi inferiori inferiori a 1 mL. La vetreria volumetrica può essere calibrata pesando l'acqua su una bilancia analitica precisa (è possibile controllare la densità dell'acqua a varie temperature e poi calcolare il volume esatto dell'acqua dal suo peso misurato); ciò consentirebbe di etichettare ciascuna delle boccette, ecc., con il loro volume effettivo. Ma la precisione può ancora essere un problema, soprattutto a volumi inferiori, ed è molto dipendente dall'operatore. Ci vuole pratica per diventare bravi a gestire piccoli volumi. L'errore di misura del segnale dipende in larga misura dal metodo strumentale utilizzato e dalla concentrazione dell'analita; può variare da circa lo 0.1% in condizioni ideali al 30% vicino al limite di rilevamento del metodo. La media delle misure ripetute può migliorare la precisione rispetto al rumore casuale. Per migliorare il rapporto segnale-rumore a basse concentrazioni, è possibile modificare le condizioni, come cambiare la larghezza della fenditura del monocromatore o la lunghezza del percorso di assorbimento o utilizzando un altro metodo strumentale (come un forno di grafite piuttosto che un atomizzatore a fiamma nelle misure dell'assorbimento atomico).

19. Domanda: *Come si può trovare l'errore in una quantità specifica utilizzando il metodo di approssimazione dei minimi quadrati? Come si può stimare l'errore nella pendenza e nell'intercetta calcolate?*

Risposta: Quando si utilizza un'approssimazione dei minimi quadrati semplice (del primo ordine), la migliore linea di approssimazione è specificata da due sole quantità: la *pendenza* e l'*intercetta*. L'*errore casuale* nella pendenza e nell'intercetta (in particolare, la loro [deviazione standard](#)) può essere stimato matematicamente dalla misura in cui i punti di calibrazione si discostano dalla linea approssimazione. Le equazioni per fare ciò sono fornite [qui](#) e sono implementate nello "[spreadsheet per la calibrazione lineare](#) con errore di calcolo". È importante rendersi conto che questi calcoli dell'errore sono solo stime, perché presumono che il set di dati di calibrazione sia rappresentativo di tutti i set di calibrazione che si otterrebbero se si ripetesse la calibrazione un numero elevato di volte - in altre parole, l'ipotesi è che gli errori casuali (errori di misurazione volumetrici e di segnale) nel tuo particolare set di dati siano tipici. Se gli errori casuali sono piccoli quando si esegue la curva di calibrazione, si otterrà una curva di calibrazione apparentemente *buona*, ma anche le stime dell'errore casuale nella pendenza e nell'intercetta saranno *basse*. Se gli errori casuali sono grandi, si otterrà una curva di calibrazione apparentemente *cattiva*, e le stime dell'errore casuale nella pendenza e nell'intercetta saranno troppo *alte*. Queste stime dell'errore possono essere particolarmente scarse quando il numero di punti in una curva di calibrazione è piccolo; l'accuratezza delle stime aumenta se il numero di punti aumenta, ma ovviamente la preparazione di molte soluzioni standard richiede tempo e denaro. La conclusione è che ci si può aspettare che queste previsioni di errore da una singola curva di calibrazione siano molto approssimative; potrebbero essere facilmente sfasati di un fattore due o più, come dimostrato dalla simulazione "Propagazione dell'errore nel metodo della curva di calibrazione lineare" ([scaricare la versione OpenOffice](#)).

20. Come si può stimare l'errore nelle concentrazioni calcolate delle incognite?

Risposta: È possibile utilizzare la pendenza e l'intercetta dall'approssimazione dei minimi quadrati per calcolare la concentrazione di una soluzione sconosciuta misurandone il segnale e calcolando **(Segnale - intercetta) / pendenza**, dove **Segnale** è la lettura del segnale (ad esempio, l'assorbanza) della soluzione ignota. Gli errori in questa concentrazione calcolata possono quindi essere stimati secondo le normali regole di propagazione dell'errore: in primo luogo, l'errore in **(Segnale - intercetta)** viene calcolato dalla regola per addizione e sottrazione; secondo, l'errore in **(Segnale - intercetta) / pendenza** viene calcolato dalla regola di moltiplicazione e divisione. Le equazioni per fare ciò sono fornite [qui](#) e sono implementate nello "[spreadsheet per la calibrazione lineare](#) con

errori di calcolo". È importante rendersi conto che questi calcoli dell'errore sono solo stime, per il motivo fornito nella Domanda #19 sopra, specialmente se il numero di punti in una curva di calibrazione è piccolo, come dimostrato dalla simulazione "Propagazione dell'errore nel metodo della curva di calibrazione lineare" ([scaricare la versione OpenOffice](#)).

21. Qual è il valore minimo accettabile del coefficiente di determinazione (R^2) per la calibrazione?

Risposta: Dipende dalla precisione richiesta. Come regola generale, se è necessaria una precisione di circa lo 0.5%, è necessario un R^2 di 0.9998; e un errore dell'1% è sufficiente, un R^2 di 0.997 andrà bene; e se un errore del 5% è accettabile, un R^2 di 0.97 andrà bene. La conclusione è che l' R^2 deve essere molto vicino a 1.0 per ottenere risultati quantitativi in chimica analitica.

Catalogo delle funzioni, script e spreadsheet per il signal processing

Questo è un elenco di funzioni, script, file di dati e fogli di calcolo utilizzati in questo libro, raccolti in base all'argomento, con brevi descrizioni. Se si sta leggendo questo libro online, su un computer connesso ad Internet, si può eseguire un **Ctrl-Click** su qualsiasi link e selezionare "Save link as..." per scaricarli nel proprio computer. Ci sono circa 200 file-m Matlab/Octave (funzione e script dimostrativi); inserirli nel "path" di Matlab o di Octave in modo da poterli utilizzare come qualsiasi altra funzionalità nativa. ([Differenza tra script e funzioni](#)). Per visualizzare la guida inclusa in queste funzioni e script, digitare "help <nome>" al prompt dei comandi (dove "<nome>" è il nome dello script o della funzione).

Molte delle figure in questo libro sono schermate del software in azione. Spesso le schermate visualizzano un numero di versione precedente a quella corrente, ovvero la data in cui sono state realizzate per la prima volta; quasi certamente c'è una versione più recente che include funzioni aggiuntive. Se non si è certi di avere tutte le versioni più recenti, il modo più semplice per aggiornare tutte le funzioni, script, strumenti, fogli di calcolo e file di documentazione è scaricare l'ultimo [file ZIP dell'archivio](#) (circa 400 MByte), poi click-destro sul file zip e selezionare "Extract all". Quindi visualizzare il contenuto della cartella estratta *per data* e, col "drag and drop", inserire o re-inserire i file aggiornati in una cartella nel path di Matlab/Octave. I file ZIP contengono *tutti* i file utilizzati di questo sito web *in un'unica directory*, quindi si possono cercare per nome o ordinarli per data per determinare quali sono cambiati dall'ultimo download.

Se tentando di eseguire uno di questi script o funzione e si riceve l'errore "missing function", si cerchi l'elemento mancante in questo catalogo, scaricandolo nel proprio path e riprovando.

Alcune di queste funzioni sono state richieste dagli utenti, suggerite dai termini di ricerca di Google o corrette ed ampliate sulla base del feedback degli utenti; si potrebbe quasi considerare questo un progetto software internazionale "crowd-sourced". *Desidero esprimere il mio ringraziamento e apprezzamento a tutti coloro che hanno fornito suggerimenti utili, corretti errori, e soprattutto a coloro che mi hanno inviato i dati del loro lavoro per testare i miei programmi. Questi contributi hanno davvero aiutato a correggere i bug e ad espandere le capacità dei programmi.*

Funzioni per il profilo dei picchi (per Matlab e [Octave](#))

La maggior parte di queste funzioni di forma richiede *tre* argomenti di input: il vettore della variabile indipendente ("x"), la posizione del picco, "pos", e la sua larghezza, "wid", solitamente l'intera larghezza a metà del massimo. Le funzioni contrassegnate con '*forma variabile*' richiedono un quarto argomento di input aggiuntivo



che determina la forma esatta del picco. I profili sigmoidale ed esponenziale (funzione alfa, impulso esponenziale, sigmoide in salita, sigmoide in discesa, Gompertz, FourPL e OneMinusExp) hanno nomi di variabili diversi.

[Gaussiana](#) `y = gaussian(x, pos, wid)`

[Gaussiana esponzialmente allargata](#) (forma variabile)

[Triangle-broadened Gaussian](#) (forma variabile)

[bifurcated Gaussian](#) (forma variabile)

[Flattened Gaussian](#) (forma variabile)

[Clipped Gaussian](#) (forma variabile)

[Lorentzian](#) (ovvero 'Cauchy') `y = lorentzian(x, pos, wid)`

[exponentially-broadened Lorentzian](#) (forma variabile)

[Clipped Lorentzian](#) (forma variabile)

[Mix Gaussiana/Lorentziana](#) (forma variabile)

[Profilo Voigt](#) (forma variabile)

[lognormal](#)

[sech2pulse](#) (intermedio tra Gaussiano e Lorentziano) NEW Lo script [Sech2ShapeComparison.m](#) confronta le forme dell'impulso Gaussiano, Lorentziano e sech2 ([grafico](#)). NEW

<https://terpconnect.umd.edu/~toh/spectrum/logistic.m> [distribuzione](#) logistica (per la *funzione* logistica, vedere [up-sigmoid](#))

[Pearson 5](#) (forma variabile)

[funzione alfa](#)

[impulso esponenziale](#)

[plateau](#) (forma variabile, prodotto simmetrico di sigmoide e sigmoide discendente, simile a [Flat-topped Gaussian](#))

[Breit-Wigner-Fano resonance \(BWF\)](#) (forma variabile)

[triangle](#)

[exponentially-broadened triangle](#) (variable shape)

[Gaussian/Triangle blend](#) (forma variabile)

[rectanglepulse](#)

[tsallis distribution](#) (forma variabile, simile alla Pearson 5)

[up-sigmoid](#) (*function* logistica o "Forma-a-S"). Sigmoide semplice verso l'altro.

[Sigmoide verso il basso](#) ("A-forma-di-Z") Semplice sigmoide discendente.

[Gompertz](#), logistica a 3 parametri, una sigmoidale a forma variabile:

`y=Bo*exp(-exp((Kh*exp(1)/Bo)*(L-t) +1))`

[FourPL](#), 4-parameter logistic, `y = maxy+ (miny-maxy) ./ (1+(x./ip).^slope)`

[OneMinusExp](#), Asymptotic rise to flat plateau: `g = 1-exp(-wid.* (x-pos))`

[peakfunction.m](#), una funzione che genera molti diversi tipi di picchi specificati da un numero.

[modelpeaks](#), una funzione che simula un segnale temporale multi-picco costituito da un qualsiasi numero di picchi della stessa forma. La sintassi è `model= modelpeaks(x, NumPeaks, peakshape, Heights, Positions, Widths, extra)`, dove 'x' è il vettore della variabile indipendente, 'NumPeaks' è il

numero dei picchi, 'peakshape' è il numero del profilo del picco, 'Heights' è il vettore delle altezze dei picchi, 'Positions' è il vettore delle posizioni dei picchi, 'Widths' è il vettore delle larghezze dei picchi e 'extra' è un il parametro aggiunto del profilo richiesto dalle forme allargate esponenzialmente, Pearson, Mix Gaussiana/Lorentziana, BiGaussiana e BiLorentziana. Digitare 'help modelpeaks'. Per creare picchi rumorosi, usare una delle seguenti funzioni di rumore per creare del rumore casuale da aggiungere all'array modelpeaks.

[modelpeaks2](#), una funzione che simula un segnale temporale multi-picco costituito da un qualsiasi numero di picchi di forma diversa. La sintassi è `y=modelpeaks2(t, Shape, Height, Position, Width, extra)` dove 'shape' è un vettore di numeri di profili e gli altri argomenti di input sono gli stessi di `modelpeaks.m`. Digitare 'help modelpeaks2'.

[ShapeDemo](#) mostra 16 forme di picco elementari graficamente, mostrando i picchi di forma variabile come linee multiple. (Grafico a pagina 411)

[SignalGenerator.m](#) è uno script che utilizza la funzione `modelpeaks.m` precedente per creare e disegnare un segnale realistico generato dal computer costituito da più picchi su una linea di base variabile con del rumore casuale variabile. È possibile modificare le righe contrassegnate con “<<<” per modificare il carattere dei picchi del segnale, della linea di base e del rumore.

Aritmetica del Segnale

[stdev.m](#) Funzione della deviazione standard compatibile con Octave e Matlab (perché la normale funzione `std.m` nativa si comporta in modo diverso in Matlab e in Octave). [rsd.m](#) è la deviazione standard relativa (la deviazione standard divisa per la media).

[PercentDifference.m](#) Una semplice funzione che calcola la differenza percentuale tra due numeri o vettori, cioè, `100 .* (b-a) ./ a`, dove a e b possono essere scalari o vettori.

[halfwidth e tenthwidth:](#) `[FWHM, slope1, slope2, hwhm1, hwhm2] = halfwidth(x, y, xo)` utilizza l'interpolazione lineare tra i punti per calcolare la FWHM approssimativa (larghezza intera a metà del massimo) di qualsiasi picco uniforme il cui massimo è a $x=x_0$, ha una linea di base zero e scende al di sotto della metà dell'altezza massima su entrambi i lati. Non accurato se il picco è rumoroso o scarsamente campionato. Se si forniscono argomenti di output aggiuntivi, vengono restituite anche le pendenze del bordo iniziale e finale, `slope1` e `slope2` e le semi-larghezze del bordo iniziale e finale a metà del massimo, `hwhm1` e `hwhm2`, rispettivamente. Se `x0` viene omesso, determina la metà della larghezza del picco più grande. Esempio: `x0=500; width=100; x=1:1000; y=exp(-1.*((x-x0)/(0.60056120439323.*width)).^2); halfwidth(x, y, xo)`. La funzione analoga `[twidth,slope1,slope2,hwhm1,hwhm2] = tenthwidth(x,y,xo)` calcola l'intera larghezza a 1/10 del massimo e, solo per controllo, [hundredwidth](#), `[hwidth, slope1, slope2] = hundredwidth(x, y, xo)`, calcola l'intera larghezza a 1/100 del massimo.

[MeasuringWidth.m](#) è uno script che confronta due metodi per misurare l'intera larghezza a metà del massimo di un picco: approssimazione Gaussiana (utilizzando [peakfit.m](#)) e interpolazione diretta (utilizzando `halfwidth.m`). I due metodi concordano esattamente per una Gaussiana senza rumore accuratamente campionata su una linea di base zero, ma danno risposte leggermente diverse se una qualsiasi di queste condizioni non è soddisfatta. La funzione `halfwidth` funziona bene per qualsiasi forma di picco con smoothing finemente campionata su una linea di base zero, ma la funzione `peakfit` è migliore nel resistere al rumore casuale e può correggere alcuni tipi di linee di base e ha un'ampia selezione di forme di picco da utilizzare come modello. Vedere il file di help.

[IQRorange.m](#), stima la deviazione standard di un insieme di numeri dividendo il suo “[intervallo interquartile](#)” (IQR) per 1.34896, un'alternativa al normale calcolo della deviazione standard che funzio-

na meglio per calcolare la dispersione (diffusione) di un set di dati contenente valori anomali. Esenzialmente è la deviazione standard con i valori anomali rimossi. La sintassi è `b = IQrange(a)`.

rmnan(a), che sta per "ReMove Not A Number", rimuove i NaN ("Not a Number") e gli Inf ("Infinite") dai vettori, sostituendoli con i numeri reali più vicini e stampando il numero di modifiche (se presenti). Lo si usa per impedire che le operazioni successive si interrompano in caso di errore.

rmz(a) RiMuove gli Zero dai vettori, sostituendoli con i numeri reali più vicini diversi da zero e stampando il numero di modifiche (se effettuate). Si usa per rimuovere gli zeri dai vettori che verranno successivamente usati come denominatore di una divisione.

[a,changes]=**nht(a,b)**: "no higher than" [non maggiore di] sostituisce tutti i numeri nel vettore a che sono maggiori dello scalare b con b. Optionalmente "changes" restituisce il numero di modifiche. La funzione simile [a,changes]=**nlt(a,b)**, "no lower than" [non minore di], sostituisce tutti i numeri nel vettore a che sono minori dello scalare b con b. Optionalmente "changes" restituisce il numero di modifiche.

makeodd(a): Rende gli elementi del vettore "a" i successivi numeri interi dispari più alti. Ciò può essere utile nel calcolare le larghezze uniformi per garantire che lo smoothing non sposti il massimo dei picchi. Per esempio, `makeodd([1.1 2 3 4.8 5 6 7.7 8 9]) = [1 3 3 5 5 7 9 9 9]`

condense(y,n), funzione per ridurre la lunghezza del vettore y sostituendo ogni gruppo di n valori successivi con la loro media. La funzione simile **condensem.m** funziona per le matrici. Si usa per ricampionare un segnale sovra-campionato. Menzionato in Smoothing (pag. 38) e in iSignal (pag. 379)

val2ind(x,val), restituisce l'indice e il valore dell'elemento del vettore x che è più vicino a val. Esempio: `if x=[1 2 4 3 5 9 6 4 5 3 1], allora val2ind(x,6)=7 e val2ind(x,5.1)=[5 9]`. Questo è utile per accedere a sottoinsiemi di set di dati x, y; ad esempio, la sequenza di codice `x1=7;x2=8; irange = val2ind(x,x1):val2ind(x,x2); xx=x(irange); yy=y(irange); plot(xx,yy)` isolerà il sottoinsieme xx, yy e lo trasporterà solo nell'intervallo di valori x da 7 a 8. Per alcuni altri esempi di come questo può essere utilizzato, vedere [pagina 242](#).

testcondense.m è uno script che dimostra l'effetto del calcolo della media del boxcar usando la funzione condense.m per ridurre il rumore senza modificarne il colore. Mostra che riduce il rumore misurato, rimuovendo le componenti ad alta frequenza, ne risulta un tempo di esecuzione dell'approssimazione più veloce e un errore di approssimazione inferiore, ma non una misurazione più accurata dei parametri del picco.

NumAT(m,threshold): "Numbers Above Threshold": Conta il numero di elementi adiacenti nel vettore 'm' che sono maggiori o uguali al valore scalare 'threshold'. Restituisce una matrice che elenca ogni gruppo di valori adiacenti, il loro indice iniziale, il numero di elementi in quel gruppo, la somma di quel gruppo e la media. Digitare "help NumAT" e provare gli esempi.

isOctave.m Restituisce "true" se questo codice viene eseguito da Octave. Restituisce "false" se questo codice viene eseguito da MATLAB o da qualsiasi altra variante di MATLAB. Utile in quei pochi casi in cui c'è una piccola differenza tra la sintassi o il funzionamento delle funzioni Matlab e Octave, come ad esempio [trypoly\(x,y\)](#), [tablestats.m](#) e [trydatatrans.m](#).

Rappresentazione grafica dei dati. Gli script Matlab/Octave [plotting.m](#) e [plotting2.m](#) mostrano come disegnare più segnali utilizzando matrici e sub-grafici (piccoli grafici in una singola finestra).

Gli script [realtimeplotautoscale.m](#) e [realtimeplotautoscale2.m](#) mostrano la stampa (Cliccare [per un grafico animato](#)).

[plotit](#), versione 2, è una funzione facile da usare per disegnare dati x, y in matrici o in vettori separati. Sintassi:

`[coef,RSquared,StdDevs,BootResults]=plotit(xi,yi,polyorder,datastyle,fits tyle)`. Può anche approssimare i polinomi ai dati e calcolare gli errori. [Click qui](#) o digitare "help plotit" al prompt di Matlab/Octave per degli esempi.

[plotxrange](#) estrae e disegna i valori dei vettori x, y solo per i valori x compresi tra x1 e x2. Restituisce i valori estratti nei vettori xx,yy e l'intervallo dei valori dell'indice in irange. Ignora i valori di x1 e x2 al di fuori dell'intervallo di x.

[segplot](#), syntax `[s,xx,yy]=segplot(x,y,NumSegs,seg)`, divide y in "NumSegs" segmenti di uguale lunghezza e disegna i dati x, y con segmenti contrassegnati da linee verticali, ciascuno etichettato con un piccolo numero di segmento nella parte inferiore. Restituisce un vettore 's' di indici di segmento e il sottoinsieme xx,yy, di valori nel numero di segmento 'seg'. Se è incluso il quarto argomento di input, 'seg', disegna solo questo segmento.

Segnali e Rumore

[whitenoise](#), [pinknoise](#), [bluenoise](#) [propnoise](#), [sqrtnoise](#), [bimodal](#): diversi tipi di rumore casuale che potrebbero essere riscontrati nelle misure fisiche. Digitare "help whitenoise", ecc., per l'help e gli esempi.

[noisetest.m](#) è una funzione Matlab/Octave autonoma per mostrare diversi tipi di rumore. Disegna i picchi Gaussiani con quattro diversi tipi di rumore aggiunto con la stessa deviazione standard: rumore bianco costante; rumore rosa costante (1/f); rumore bianco proporzionale; e rumore bianco a radice quadrata, poi approssima un modello Gaussiano a ciascun set di dati rumorosi e calcola la media e la deviazione standard dell'altezza, della posizione, della larghezza e dell'area del picco per ciascun tipo di rumore. Vedere [pagina 22](#). Vedere anche [NoiseColorTest.m](#).

[SubtractTwoMeasurements.m](#) è uno script dimostrativo Matlab/Octave della misura del rumore e del rapporto segnale/rumore di una forma d'onda stabile sottraendo due misure della forma d'onda del segnale, m1 e m2 e calcolando la deviazione standard della differenza. Il segnale deve essere stabile tra le misure (eccetto per il rumore casuale). La deviazione standard del rumore misurato è data da $\text{sqrt}((\text{std}(m1-m2).^2)/2)$.

[NoiseColorTest.m](#), una funzione che dimostra l'effetto dello smoothing del rumore bianco, rosa e blu. Visualizza un grafico dei cinque tipi di colore del rumore sia [prima](#) che [dopo](#) lo smoothing, nonché i loro [spettri di frequenza](#). Tutti i campioni di rumore hanno una deviazione standard di 1.0 prima dello smoothing. È possibile modificare la larghezza dello smoothing e modificando le righe 6 e 7.

[CurvefitNoiseColorTest.m](#), una funzione che mostra l'effetto del rumore bianco, rosa e blu sulla curva che si approssimando a un singolo picco Gaussiano.

[RANDtoRANDN.m](#) è uno script che mostra come l'espressione $1.73*(\text{RAND}() - \text{RAND}() + \text{RAND}() - \text{RAND}())$ approssima numeri casuali normalmente distribuiti con media zero e una deviazione standard di 1. Vedere pagina 22.

[RoundingError.m](#). Uno script che mostra il rumore della digitalizzazione (arrotondamento) e mostra che l'aggiunta di rumore e quindi la media d'insieme di più segnali può ridurre il rumore complessivo.

vo nel segnale. Questo è un raro esempio in cui l'aggiunta di rumore è vantaggiosa. Vedere pagina 300.

[DigitizedSpeech.m](#), una dimostrazione audio/grafica dell'errore di arrotondamento sul parlato digitalizzato. Inizia con una registrazione audio della frase "Testing, one, two, three", precedentemente registrata a 44000 Hz e salvata in formato WAV (link per il [download](#)), arrotondando progressivamente l'ampiezza dei dati a 8 bit (256 passi), 4 bit (16 passi) e 1 bit (2 passi), poi lo stesso con rumore bianco casuale aggiunto prima dell'arrotondamento (2 passi + rumore), disegna le forme d'onda e riproduce i suoni risultanti, dimostrando sia l'effetto degradante dell'arrotondamento che il notevole miglioramento causato dall'aggiunta di rumore. Vedere pagina 300.

[CentralLimitDemo.m](#), script che dimostra che le variabili casuali uniformi più indipendenti sono combinate, la distribuzione di probabilità diventa sempre più vicina alla normale (Gaussian). Vedere [pagina 22](#)

[EnsembleAverageDemo.m](#) è uno script Matlab/Octave che mostra la media dell'insieme per migliorare il rapporto segnale/rumore di un segnale molto rumoroso. [Click per il grafico](#). Lo script richiede che la funzione "[gaussian.m](#)" venga scaricata e inserita nel path Matlab/Octave, oppure si può usare qualsiasi altra [funzione di forma del picco](#), come [lorentzian.m](#) o [rectanglepulse.m](#).

[EnsembleAverageDemo2.m](#) è uno script Matlab/Octave che mostra l'effetto del *rumore di ampiezza*, del *rumore di frequenza* e del *rumore di fase* sulla media d'insieme di una forma d'onda sinusoidale.

[EnsembleAverageFFT.m](#) è uno script Matlab/Octave che mostra l'effetto del *rumore di ampiezza*, del *rumore di frequenza* e del *rumore di fase* sulla media d'insieme di una forma d'onda sinusoidale. Mostra che: (a) la media dell'insieme riduce il rumore bianco nel segnale ma non il rumore di frequenza o quello di fase, (b) la media dell'insieme della trasformata di Fourier ha lo stesso effetto della media dell'insieme del segnale stesso e (c) l'effetto del rumore di fase viene ridotto se si esegue la media d'insieme degli spettri di potenza. [EnsembleAverageFTFGaussian.m](#) fa lo stesso per un segnale di un picco Gaussiano, dove la variazione nell'ampiezza del picco è il rumore di frequenza e la variazione nella posizione del picco è il rumore di fase.

[iPeakEnsembleAverageDemo.m](#) è una dimostrazione in un unico file della funzione iPeak. In questo esempio, il segnale contiene un modello ripetuto di due picchi Gaussiani sovrapposti di larghezza 12, con un rapporto di altezza 2:1. Questi schemi si verificano a intervalli casuali e il livello di rumore è circa il 10% dell'altezza media del picco. Utilizzando la funzione della media dell'insieme di iPeak (**Shift-E**), è possibile fare la media dei pattern e migliorare significativamente il rapporto segnale/rumore. Vedere [pagina 25](#).

[PeriodicSignalSNR.m](#) è uno script Matlab/Octave che mostra la stima dell'ampiezza del segnale picco-picco e la radice quadrata media e il rapporto segnale-rumore di una forma d'onda periodica, stimando il rumore osservando i periodi di tempo in cui il suo inviluppo scende al di sotto di una soglia. Vedere [pagina 22](#).

[LowSNRdemo.m](#) è uno script che confronta diversi metodi di misura dei picchi con rapporti segnale/rumore molto bassi. Crea un singolo picco, con forma, altezza, posizione e larghezza regolabili, aggiunge un rumore casuale bianco costante in modo che il rapporto segnale/rumore varia da 0 a 2, quindi misura l'altezza e la posizione del picco con ciascun metodo e calcola l'errore medio. Vengono confrontati quattro metodi: (1) la misura da picco a picco del segnale con smoothing e del background; (2) un metodo di ricerca dei picchi basato su [findpeakG](#); (3) [approssimazione dei minimi quadrati iterativa](#) (INLS) basata sulla funzione [peakfit.m](#); e (4) [approssimazione dei minimi quadrati vincolata classica](#) (CLS) basata sulla funzione [cls2.m](#). Vedere [l'appendice: Quanto si può scendere in basso? Prestazioni con rapporti segnale/rumore molto bassi](#).

[RandomWalkBaseline.m](#) simula un picco Gaussiano con posizione e larghezza variabili casualmente sovrapposto a una linea di base "random walk" alla deriva. Confrontare con [WhiteNoiseBaseline.m](#). Vedere [pagina 310](#).

[AmplitudeModulation.m](#) è uno script Matlab/Octave di simulazione della modulazione e del rilevamento sincrono, che mostra la capacità di riduzione del rumore. Vedere pagina 312.

[DerivativeNumericalPrecisionDemo.m](#). Funzione autonoma che dimostra come i *limiti della precisione numerica* del computer influiscano sulle derivate dalla prima alla quarta di una banda Gaussiana con smoothing ("senza rumore"), mostrando sia le forme d'onda (nella finestra 1) che i loro spettri di frequenza (nella finestra 2). Il limite della precisione numerica del computer crea rumore casuale a frequenze molto alte, che viene enfatizzato dalla differenziazione, e dalla derivata quarta in cui il rumore sommerge il segnale alle frequenze più basse. La maggior parte del rumore può essere rimossa con uno smoothing p-spline (tre passaggi di uno slittamento-della-media) con un rapporto di smoothing di 0,2. Con dati sperimentali reali, anche la più piccola quantità di rumore nei dati originali sarebbe molto maggiore. Vedere pagina 332.

[RegressionNumericalPrecisionTest.m](#) è uno script Matlab/Octave che mostra come i *limiti della precisione numerica* del computer influiscano sui Quadrati Minimi Classici (regressione multilineare) di due picchi Gaussiani sovrapposti "senza rumore" molto ravvicinati. Utilizza tre diverse formule matematiche del calcolo dei minimi quadrati che danno risultati diversi quando vengono raggiunti i limiti di precisione numerica del computer. Ma praticamente, è improbabile che la differenza tra questi metodi si veda; anche il più piccolo bit di rumore casuale aggiunto (riga 15) o instabilità del segnale produce un errore molto maggiore. Usato a pagina 332.

[RegressionADCbitsTest.m](#). Dimostrazione dell'effetto della risoluzione del convertitore analogico-digitale (definita dal numero di bit nella riga 9) sui minimi quadrati classici (regressione multilineare) di due picchi Gaussiani sovrapposti ravvicinati. Normalmente, il rumore casuale (riga 10) produce un errore maggiore della risoluzione dell'ADC. Usato a pagina 332.

[CreateSimulatedSignal.m](#). Script che crea un segnale multi-picco simulato destinato a corrispondere a un segnale sperimentale, utilizzando un elenco di picchi nel segnale sperimentale nella matrice P, oltre al rumore e alla linea di base aggiunti. Utilizzato per testare l'accuratezza dei metodi di rilevamento del picco e di misura dell'area con quel tipo di segnale.

Smoothing

[fastsmooth](#), una versatile funzione per un rapido smoothing di dati. La sintassi è **SmoothY=fastsmooth(Y,w, type, ends)**. Vedere pagina 38. Nota: [Greg Pittam](#) ha pubblicato una modifica della funzione fastsmooth che tollera i NaN (Not a Number) nei dati ([nanfastsmooth\(Y,w,type,tol\)](#)) ed un'altra versione per lo smoothing di dati "angoli" ([nanfastsmoothAngle\(Y,w,type,tol\)](#)). [Cliccare per un esempio animato](#).

[SegmentedSmooth.m](#), funzione segmentata di smoothing a larghezza multipla basata sull'algoritmo di fastsmooth. La sintassi è **SmoothY = SegmentedSmooth(Y,smoothwidths,type,ends)**. Questa funzione divide Y in diversi segmenti di uguale lunghezza in base alla lunghezza del vettore 'smoothwidths', poi esegue lo smoothing di ciascun segmento con una larghezza di smoothing definita dagli elementi sequenziali del vettore 'smoothwidths' e dal tipo di smoothing, 'type'. Digitare "help SegmentedSmooth" per gli esempi. [DemoSegmentedSmooth.m](#) mostra l'operazione ([click per il grafico](#)). Vedere pagina 38.

[medianfilter](#), funzione di filtro basata sulla mediana per eliminare gli spike artefatti. La sintassi è **mY=medianfilter(y, Width)**, dove "Width" è il numero di punti negli spike che si desidera eliminare. Digitare "help medianfilter" al prompt dei comandi.

[killspikes.m](#) è una funzione di filtro basata su una soglia per eliminare gli spike artefatti. La sintassi è **fy= killspikes(x, y, threshold, width)**. Ogni volta che trova un salto positivo o

negativo nei dati tra $y(n)$ e $y(n+1)$ che eccede "threshold", sostituisce i successivi "width" punti con un segmento interpolato linearmente che si estende da $x(n)$ a $x(n+width+1)$, Vedere [killspike-sdemo](#). Digitare "help killspikes" al prompt dei comandi.

[testcondense.m](#) è uno script che mostra l'effetto della media 'boxcar' utilizzando la funzione [condense.m](#), che esegue una funzione di media boxcar non sovrapposta, per ridurre il rumore senza modificare il colore. Mostra che riduce il rumore misurato, rimuovendo le componenti ad alta frequenza, ottenendo un tempo esecuzione più veloce e un errore più basso dell'approssimazione, ma sfortunatamente non una misura più accurata dei parametri del picco.

[SmoothWidthTest.m](#) è uno script Matlab/Octave che mostra l'effetto dello smoothing sull'altezza del picco, sul rumore bianco casuale e sul rapporto segnale/rumore del segnale rumoroso di un picco. Produce un'animazione che mostra l'effetto di larghezze progressivamente più ampie dello smoothing, poi disegna un grafico dell'altezza del picco, del rumore e del rapporto segnale/rumore rispetto al rapporto di smoothing. [Cliccare per un'animazione GIF](#). È possibile modificare la forma, *shape*, e la larghezza, *width*, del picco nella riga 8 e il tipo, *type*, di smoothing nella riga 9: 1=rettangolo; 2=triangolo; 3=p-spline. Lo script richiede che la funzione "[gaussian.m](#)" venga scaricata e inserita nel path di Matlab/Octave, oppure si può utilizzare qualsiasi altra [funzione per la forma del picco](#), come [lorentzian.m](#) o [rectanglepulse.m](#), ecc.

[SmoothExperiment.m](#), script molto semplice che mostra l'effetto dello smoothing sulla posizione, la larghezza e l'altezza di un singolo picco Gaussiano. Richiede che le funzioni [fastsmooth.m](#) e [peakfit.m](#) siano presenti nel path. Vedere pagina 52.

[smoothdemo.m](#), funzione auto-contenuta che confronta le prestazioni e la velocità di quattro tipi di [operazioni di smoothing](#): (1) slittamento della media, (2) triangolare, (3) p-spline (equivalente a tre passaggi dello slittamento della media), e (4) il Savitzky-Golay. Queste operazioni di smoothing vengono applicate a un singolo picco Gaussiano rumoroso. L'altezza del picco del picco con smoothing, la deviazione standard del rumore con smoothing e il rapporto segnale/rumore vengono tutti misurati in funzione dell'ampiezza dello smoothing. Vedere pagina 52.

[SmoothOptimization.m](#), script che mostra perché non è necessario lo smoothing dei dati prima dell'approssimazione dei minimi quadrati della curva; confronta l'effetto dello smoothing sul rapporto segnale/rumore dell'altezza di un picco Gaussiano rumoroso, utilizzando tre diversi metodi di misura. Richiede che le funzioni [fitgauss2.m](#), [gaussfit.m](#), [gaussian.m](#), e [fminsearch.m](#) siano presenti nel path. Vedere pagina 226.

[SmoothVsCurvefit.m](#), confronto della misure dell'altezza del picco prendendo il massimo del segnale con smoothing e approssimando la curva ai dati originali senza smoothing. Richiede [peakfit.m](#) e [gaussian.m](#) nel path.

[DemoSegmentedSmooth.m](#) mostra il funzionamento di [SegmentedSmooth.m](#) con un segnale costituito da picchi rumorosi di larghezza variabile che diventano progressivamente più larghi. Richiede [SegmentedSmooth.m](#) e [gaussian.m](#) nel path.

[DeltaTest.m](#). Un semplice script Matlab/Octave che dimostra che la forma di qualsiasi algoritmo di smoothing può essere determinata applicando quello smoothing a una *funzione delta*, un segnale composto da tutti zeri tranne un punto. Il risultato è detto *funzione di risposta all'impulso*.

[iSignal](#) (pagina 379) esegue diversi tipi di smoothing, segmentato, filtro mediano e rimozione degli spike (oltre alla differenziazione, lo sharpening, le misure dei quadrati minimi per la posizione, l'altezza, la larghezza e l'area del picco, dell'ampiezza del segnale e del rumore, spettri di frequenza nelle regioni selezionate del segnale e rapporto segnale/rumore dei picchi). Link al file m: [isignal.m](#). [Cliccare qui per scaricare il file ZIP "iSignal8.zip". Cliccare per un esempio animato.](#)

Lo script [RealTimeSmoothTest.m](#) mostra lo smoothing in tempo reale, disegnando i dati originali senza come una linea nera e quelli con smoothing in rosso. In questo caso lo script pre-calcola i dati simulati nella riga 28 e poi accede ai dati punto per punto nel ciclo di elaborazione (righe 30-51). Il numero totale di punti è controllato da 'maxx' nella riga 17 (inizialmente impostato a 1000) e la larghezza dello smoothing(in punti) è controllata da 'SmoothWidth' nella riga 20. [Grafico animato](#).

Il Tool di Smoothing (link per il download: [DataSmoothing mlx](#)) è un [Live Script](#) interattivo in grado di applicare diversi tipi di smoothing ai dati sperimentali archiviati su disco. Può eseguire la rimozione degli spike, lo smoothing della media mobile con un massimo di 5 passaggi, il filtraggio passa-basso di Savitsky-Golay e Fourier e la rimozione del rumore con wavelet (che richiede il Matlab Wavelet Toolkit). Cfr. pagina 55.

Differenziazione e sharpening

[deriv](#), [deriv2](#), [deriv3](#), [deriv4](#), [derivxy](#) e [secdervxy](#), semplici funzioni per calcolare le derivate di serie temporali senza smoothing. Vedere [pagina 71](#).

[SmoothDerivative.m](#) combina la differenziazione con lo smoothing. La sintassi è SmoothedDeriv = SmoothedDerivative(x, y, DerivativeOrder, w, type, ends) dove 'DerivativeOrder' determina l'ordine di derivazione (da 0 a 5), 'w' è la larghezza dello smoothing, 'type' determina la modalità di smoothing, e 'ends' indica come le "estremità" del segnale (i primi w/2 punti e gli ultimi w/2) devono essere gestiti.

[SlopeAnimation.m](#) è una dimostrazione [animata](#) in Matlab/Octave che mostra che la derivata prima di un segnale è la pendenza della tangente al segnale in ciascun punto.

[sharpen](#), sharpening del picco col metodo delle derivate pari. La sintassi è SharpenedSignal = sharpen(signal, factor1, factor2, SmoothWidth). Vedere pagina 83. Demo correlate: [SegmentedSharpen.m](#), [DemoSegmentedSharpen.m \(grafico\)](#), [SharpenedGaussianDemo.m \(grafico\)](#), [SharpenedGaussianDemo4terms.m \(grafico\)](#), [SharpenedLorentzianDemo.m \(grafico\)](#), [Sharpened-LorentzianDemo4terms.m](#).

[symmetrize.m](#) converte i picchi ampliati esponenzialmente in picchi simmetrici mediante [l'aggiunta o la sottrazione ponderata della derivata prima](#). La sintassi è ySym = symmetrize(t, y, factor, smoothwidth, type, ends), dove t, y sono i vettori dei dati originali, 'factor' è il fattore di ponderazione della derivata e 'smoothwidth', 'type', 'ends' sono gli [argomenti di SegmentedSmooth](#) per lo smoothing della derivata. Per eseguire una simmetrizzazione *segmentata*, "factor" e "smoothwidth" devono essere vettori. (Nella versione 2, solo la derivata subisce internamente lo smoothing, non l'intero segnale simmetrizzato). [SymmetrizeDemo.m](#) esegue tutti e cinque gli esempi nel file di help symmetrize.m, ciascuno in una finestra separata.

La simmetrizzazione con la derivata prima può essere seguita da un'applicazione di sharpening con le derivate pari per un ulteriore sharpening del picco, come dimostrato per una singola Gaussiana modificata esponenzialmente (EMG) dalla funzione auto-contenuta di Matlab/Octave [EMGplusfirstderivative.m](#) e, per una Lorentziana esponenzialmente modificata (EML), da [EMLplusfirstderivative.m](#). In entrambe queste demo, la finestra "Figure 1" mostra la [simmetrizzazione](#) e la finestra "Figure 2" mostra che il picco simmetrizzato può essere ulteriormente ristretta da ulteriori sharpening con le derivate [2^a](#) e [4^a](#). [SymmetizedOverlapDemo.m](#) mostra l'ottimizzazione della simmetrizzazione della derivata prima per la misura delle aree di due Gaussiane allargate esponenzialmente sovrapposte. Una simmetrizzazione esponenziale *doppia* viene eseguita dalla funzione [DEMSymm.m](#). Lo dimostra lo script [DemoDEMSymm.m](#) e le sue due varianti (1, 2), che crea due picchi esponenziali doppi sovrapposti da originali Gaussiane, poi chiama la funzione DEMSymm.m per eseguire la simmetrizzazione, utilizzando una tecnica di bracketing [raggruppamento] più e meno a tre livelli per aiutare a determinare i valori migliori dei due fattori di ponderazione andando per tentativi. La funzione interattiva *iSignal* (pagina 379) può eseguire la simmetrizzazione della derivata prima in modo interattivo, premendo i tasti per aumentare e diminuire il "fattore" mentre si osserva l'effetto sul segnale. Lo script [AsymmetricalOverlappingPeaks.m](#) mostra l'uso della simmetrizzazione della derivata prima e

l'approssimazione della curva per analizzare un picco complesso "misterioso". Vedere pagina 358).

[ProcessSignal](#), una funzione multiuso a riga di comando Matlab/Octave che include smoothing, differenziazione, sharpening e filtro mediano sul set di dati temporali x,y (vettori colonna o riga). Come [iSignal](#), senza i controlli del disegno e i tasti interattivi. Digitare "help ProcessSignal". Restituisce il segnale processato come vettore che ha la stessa forma di x, indipendentemente dalla forma di y. La sintassi è `Processed = Processed=ProcessSignal(x, y, DerivativeMode, w, type, ends, Sharpen, factor1, factor2, Symsize, Symfactor, SlewRate, MedianWidth)`.

[derivdemo1.m](#), una funzione che mostra le forme di base delle derivate. Vedere pagina 60.

[DerivativeShapeDemo.m](#) è una funzione che mostra le derivate prime di 16 diverse forme di picco. ([Grafico](#))

[derivdemo2.m](#), una funzione che dimostra l'effetto della larghezza del picco sull'ampiezza delle derivate. Vedere pagina 60.

[derivdemo3.m](#), una funzione che dimostra l'effetto dello smoothing sulla derivata *prima* di un segnale rumoroso. Vedere pagina 60.

[derivdemo4.m](#), una funzione che dimostra l'effetto dello smoothing sulla derivata *seconda* di un segnale rumoroso. Vedere pagina 60.

[DerivativeDemo.m](#) è una funzione demo auto-contenuta di Matlab/Octave che usa [ProcessSignal.m](#) e [plotit.m](#) per mostrare un applicazione della differenziazione all'analisi quantitativa di un picco sepolto in un background instabile (es. come in diverse forme di spettroscopia). Lo scopo è quello di ricavare la misura dell'altezza del picco che varia linearmente con quella effettiva del picco ed è minimamente influenzata dal background e dal rumore. Per avviarlo, basta digitare DerivativeDemo al prompt dei comandi. È possibile modificare molte delle variabili interne (ad esempio, Noise, BackgroundAmplitude) per rendere il problema più difficile o più facile. Si noti che, anche se l'ampiezza della derivata è numericamente inferiore al segnale originale (perché ha unità diverse), il rapporto segnale/rumore della derivata è migliore e il segnale della derivata è linearmente proporzionale all'altezza effettiva del picco, nonostante l'interferenza di grandi variazioni del background e del rumore casuale. Vedere pagina 71.

[iSignal](#) o [isignal octave](#) (pagina 379) è una funzione interattiva che include *la differenziazione e lo smoothing* per i segnali di serie temporali, fino alla derivata 5^a, includendo automaticamente il tipo di smoothing richiesto. Delle semplici sequenze di tasti consentono di regolare i parametri dello smoothing (il tipo, l'ampiezza e la gestione delle estremità) mentre se ne osserva dinamicamente l'effetto sul segnale. Può anche eseguire una *simmetrizzazione interattiva* e lo *sharpening* di picchi espansi esponenzialmente mediante la tecnica dell'addizione della derivata prima (pagina 77). [Cliccare qui per scaricare il file ZIP "iSignal8.zip"](#). [Cliccare per un esempio animato](#).

[demoisignal.m](#) per Matlab è uno script a esecuzione automatica che mostra molte delle caratteristiche di [iSignal](#) (e richiede che l'ultima versione di iSignal e la versione 6 di [plotit.m](#), siano presenti nel path di Matlab). Mostra pan e zoom, smoothing, differenziazione, spettro di frequenza, misura dei picchi e calibrazione della spettroscopia derivativa (in combinazione con [plotit.m](#) versione 6).

[iSignalDeltaTest](#) è uno script Matlab/Octave che mostra la risposta in frequenza (spettro di potenza) delle funzioni di smoothing e differenziazione di [iSignal](#) applicandole alla [funzione delta](#). Modificando il tipo di smoothing, l'ampiezza e l'ordine di derivazione si vedrà come cambia lo spettro della potenza.

Lo script [RealTimeSmoothFirstDerivative.m](#) mostra la differenziazione con smoothing in tempo reale, utilizzando un semplice algoritmo di differenza adiacenti (riga 47) e tracciando i dati originali

come una linea nera e quelli della derivata prima in rosso. Lo script [RealTimeSmoothSecondDerivative.m](#) calcola la derivata *seconda* con smoothing utilizzando un algoritmo di differenza centrale (riga 47). Entrambi gli script pre-calcolano i dati simulati nella riga 28 per poi accedervi punto per punto nel ciclo di elaborazione (righe 31-52). In entrambi i casi il numero massimo di punti è impostato nella riga 17 e la larghezza dello smoothing è impostata nella riga 20.

Lo script [RealTimePeakSharpening.m](#) mostra lo sharpening dei picchi in tempo reale utilizzando la tecnica della derivata seconda. Utilizza dati simulati pre-calcolati nella riga 30 e poi accede ai dati punto per punto nel ciclo di elaborazione (righe 33-55). In entrambi i casi il numero massimo di punti è impostato nella riga 17, la larghezza dello smoothing è impostata nella riga 20 e il fattore di ponderazione (K1) è impostato nella riga 21. In questo esempio l'ampiezza dello smoothing è di 101 punti, che tiene conto del ritardo nel con sharpening rispetto all'originale.

Analisi delle armoniche

[FrequencySpectrum.m](#) (sintassi `fs=FrequencySpectrum(x,y)`) restituisce la parte reale dello spettro di potenza di Fourier di x, y come matrice.

[PlotFrequencySpectrum.m](#) disegna lo spettro di frequenza o periodogramma del segnale x,y su coordinate lineari o logaritmiche. La sintassi è `PowerSpectrum= PlotFrequencySpectrum(x, y, plotmode, XMODE, LabelPeaks)` . . Digitare "help PlotFrequencySpectrum" per i dettagli. Provare questo esempio:

```
x= [0:.01:2*pi]'; y=sin(200*x)+randn(size(x));
subplot(2,1,1); plot(x,y); subplot(2,1,2);
PowerSpectrum=PlotFrequencySpectrum(x,y,1,0,1);
```

[CompareFrequencySpectrum.m](#). Uno script che confronta due segnali (pannello superiore) e i loro spettri di frequenza (pannello inferiore) con il segnale originale mostrato in blu e il segnale modificato in verde. plotmode: =1:lineare, =2:semilog X, =3:semilog Y; =4: log-log). XMODE: =0 per spettro di frequenza (x è la frequenza); =1 per il periodogramma (x è il tempo). Definire la modifica del segnale nella riga 15. Si può caricare un segnale memorizzato in formato .mat o creare un segnale simulato per il test. Si deve avere [PlotFrequencySpectrum.m](#) nel path.

[PlotSegFreqSpect.m](#) è uno spettro di Fourier segmentato (sintassi `PSM=(x,y, NumSegments, MaxHarmonic, LogMode)`) divide y in 'NumSegments' segmenti di uguale lunghezza, moltiplica ciascuno per una finestra di Hanning apodizzante, calcola lo spettro di potenza di ogni segmento, restituisce la matrice dello spettro di potenza (PSM) e disegna il risultato delle prime 'MaxHarmonic' componenti di Fourier come diagramma a contorno. Vedere pagina 99 per un esempio della sua applicazione a un segnale completamente sepolto da un eccesso di rumore e segnali di disturbo.

[iSignalDeltaTest](#) è uno script Matlab/Octave che mostra la *risposta in frequenza* (spettro di potenza) delle funzioni di smoothing e differenziazione di [iSignal](#) applicandole a una [funzione delta](#). Modificando il tipo di smoothing, l'ampiezza e l'ordine di derivazione si vedrà come cambia lo spettro della potenza.

[SineToDelta.m](#). Un'animazione dimostrativa ([grafica animata](#)) che mostra la forma d'onda e lo spettro di potenza di un'onda sinusoidale pulsante rettangolare di durata variabile (il cui spettro di potenza è una funzione "sinc") che cambia continuamente da una onda sinusoidale pura a un estremo (dove il suo spettro di potenza è una funzione delta) a un impulso a punto singolo all'altro estremo (dove il suo spettro di potenza è una linea piatta). [GaussianSineToDelta.m](#) è simile, tranne per il fatto che mostra un'onda sinusoidale pulsata *Gaussiana*, il cui spettro di potenza è una funzione Gaussiana, ma che è la stessa ai due estremi della durata dell'impulso ([grafica animata](#)).

[isignal.m](#) o [isignaloctave.m](#), (pag. 379) serve per l'elaborazione generica del segnale interattiva che comprende una [Modalità Spettro di Frequenza](#), attivata e disattivata dal tasto **Shift-S** ; calcola lo spettro di frequenza del segmento del segnale visualizzato nella finestra superiore e lo visualizza nella finestra inferiore (in rosso). È possibile utilizzare i tasti Pan e Zoom per regolare la regione del

segna da visualizzare o premere **Ctrl-A** per selezionare l'intero segnale. Si preme ancora **Shift-S** per tornare alla modalità normale. Vedere [pagina 87](#) per un esempio a riguardo. [Cliccare per un esempio animato.](#)

[iPower](#), un dimostrativo interattivo, controllato da tastiera, dello spettro di potenza, utile per insegnare e apprendere gli spettri di potenza di diversi tipi di segnali, l'effetto della durata del segnale e della frequenza di campionamento. Le singole sequenze di tasti consentono di selezionare il tipo di segnale (inclusi 12 segnali diversi), la durata totale del segnale, la frequenza di campionamento e le variabili globali f_1 e f_2 che vengono utilizzate in modi diversi nei diversi segnali. Quando viene premuto il tasto **Enter** il segnale (y) viene inviato al dispositivo audio WAVE di Windows. Premere **K** per visualizzare un elenco di tutti i comandi da tastiera. (link al file m: [ipower.m](#)). [Presentazione degli esempi](#).

Lo script [RealTimeFrequencySpectrumWindow.m](#) calcola e disegna lo spettro di frequenza di Fourier di un segnale. Carica i dati simulati in tempo reale da un "file .mat" (nella riga 31) e poi accede a quei dati punto per punto nel ciclo di elaborazione 'for'. Una variabile critica in questo caso è "WindowWidth" (riga 37), il numero di punti presi per calcolare ogni spettro di frequenza. Se il flusso di dati è un segnale audio, è anche possibile riprodurre il suono attraverso il sistema audio del computer sincronizzato con la visualizzazione degli spettri di frequenza (impostare "PlaySound" a 1).

Convoluzione e deconvoluzione di Fourier

[ExpBroaden](#), funzione di ampliamento esponenziale. La sintassi è $\mathbf{yb} = \text{ExpBroaden}(\mathbf{y}, t)$. Esegue una convoluzione del vettore y con un decadimento esponenziale della costante di tempo t . Menzionato alle pagine 32 e 405.

[GaussConvDemo.m](#), uno script che mostra che una Gaussiana di altezza unitaria, convoluta con Fourier con una Gaussiana centrata sullo zero della stessa larghezza è una Gaussiana con un'altezza di $1/\sqrt{2}$, una larghezza di $\sqrt{2}$ e con area uguale alla Gaussiana originale. Quando si esegue questo script, il pannello superiore mostra la convoluzione e quello inferiore mostra come recuperare la y originale dal risultato convoluto ([grafico](#)). Facoltativamente, è possibile aggiungere rumore nella riga 9 per mostrare come la convoluzione attenua il rumore e come la deconvoluzione di Fourier lo ripristina. Richiede gaussian.m nel path.

[CombinedDerivativesAndSmooths.txt](#). Coefficienti di convoluzione per calcolare le derivate dalla prima alla quarta, con smoothing rettangolari, triangolari e P-spline.

[Convolution.txt](#), semplici esempi di vettori di convoluzione di soli numeri interi per smoothing e differenziazione.

[deconvolutionexample.m](#), un semplice script di esempio che mostra l'uso della funzione di deconvoluzione di Fourier, 'deconv', di Matlab. Vedere pagina 111.

[DeconvDemo.m](#), uno script demo di deconvoluzione di Fourier con un segnale contenente quattro Gaussiane ampliate da una funzione esponenziale ([grafico](#)). [DeconvDemo2.m](#) è uno script simile per una singola Gaussiana ([grafico](#)). [DeconvDemo3.m](#) mostra la deconvoluzione di una funzione di convoluzione *Gaussiana* da un impulso rettangolare ([grafico animato](#)). [DeconvDemo4.m](#) ([grafico animato](#)) mostra una "auto-deconvoluzione" applicata ad un segnale costituito da un picco Gaussiano che viene ampliato dallo strumento di misura, e un tentativo di recuperare l'ampiezza del picco originale. [DeconvDemo5.m](#) ([grafico](#)) mostra un tentativo di risolvere *due* picchi sottostanti ravvicinati che sono *completamente irrisolti* nel segnale osservato. Vedere pagina 299. Delle variazioni includono le versioni con [picchi Lorentziani](#) e uno con una [funzione di convoluzione triangolare](#).

[deconvgauss.m](#). $ydc = \text{deconvgauss}(x, y, w)$ esegue la deconvoluzione di una funzione Gaussiana di larghezza ' w ' dal vettore y , restituendo il risultato deconvoluto.

[LorentzianSelfDeconvDemo.m](#). Dimostrazione dell'auto-deconvoluzione Lorentziana. Richiede le funzioni lorentzian, halfwidth e fastsmooth.

[deconvexp.m](#). ydc=deconvexp(y,tc) esegue la deconvoluzione di una funzione esponenziale della costante di tempo 'tc' dal vettore y, restituendo il risultato deconvoluto.

[SegExpDeconv\(x,y,tc\)](#) è una versione segmentata [dideconvexp.m](#); divide x,y in un numero di segmenti di uguale lunghezza definiti dalla lunghezza del vettore 'tc', poi ogni segmento viene deconvoluto con un decadimento esponenziale della forma $\exp(-x./t)$ dove 't' è il corrispondente elemento del vettore 'tc'. È possibile utilizzare qualsiasi numero e sequenza di valori t. Utile quando l'ampiezza del picco e/o la coda esponenziale dei picchi varia per la durata del segnale. [SegExpDeconvPlot.m](#) è lo stesso tranne per il fatto che disegna i segnali originali e deconvoluti e mostra le divisioni tra i segmenti mediante linee magenta verticali. [SegGaussDeconv.m](#) e [SegGaussDeconvPlot.m](#) sono uguali tranne per il fatto che eseguono una deconvoluzione Gaussiana simmetrica (centrata sullo zero). [SegDoubleExpDeconv.m](#) e [SegDoubleExpDeconvPlot.m](#) eseguono una deconvoluzione esponenziale simmetrica (centrata sullo zero).

[P=convdeconv\(x,y,vmode,smode,vwidth,DAdd\)](#), per Matlab o Octave, esegue la convoluzione Gaussiana, Lorentziana o esponenziale e la deconvoluzione del segnale in x,y.

[iSignal 8.3](#) (pagina 379) ha un tasto **Shift-V** che visualizza il menù delle operazioni di convoluzione e deconvoluzione di Fourier che consentono di eseguire la convoluzione di una funzione Gaussiana o esponenziale col segnale, oppure la deconvoluzione di una funzione Gaussiana o esponenziale dal segnale e consentono di regolare l'ampiezza in modo interattivo. [Cliccare qui per scaricare il file ZIP "iSignal8.zip"](#)

Tool per la convoluzione dei dati. Il Live Script interattivo [DeconvoluteData mlx](#) può eseguire l'autodeconvoluzione di Fourier sui dati archiviati nel disco. Vedere pagina 120.

Filtro di Fourier

[FouFilter](#), funzione del filtro di Fourier, con passa-banda, passa-basso, passa-alto o notch (escludi-banda) variabile. La sintassi è `[ry,fy,ffilter,ffy] =FouFilter(y, samplingtime, centerfrequency, frequencywidth, shape, mode)`. Versione 2, marzo 2019. Vedere pagina 122.

[SegmentedFouFilter.m](#) è una versione segmentata di FouFilter.m che applica diverse frequenze centrali e larghezze ai diversi segmenti del segnale. La sintassi è la stessa di FouFilter.m tranne per il fatto che i due argomenti di input "centerFrequency" e "FilterWidth" devono essere vettori con i valori di centerFrequency di filterWidth per ogni segmento. Il segnale viene diviso in diversi segmenti uguali determinato dalla lunghezza di centerFrequency e filterWidth, che devono essere uguali in lunghezza. Digitare "help SegmentedFouFilter" per l'help e gli esempi.

[iFilter](#), filtro interattivo di Fourier. (link al file m: [ifilter.m](#)), che utilizza i tasti pan e zoom per controllare la frequenza centrale e la larghezza del filtro (pag. 380). [Cliccare qui per un esempio animato](#). Scegliere tra i filtri passa basso, passa alto, passa banda, elimina banda, passa pettine [comb] armonico o elimina pettine [comb] armonico. [Cliccare qui per guardare o scaricare un video mp4](#) di iFilter mentre filtra un segnale rumoroso di un codice Morse, con audio (guardare il titolo della figura durante la riproduzione del video). Le versioni Octave usano i tasti < e > (con e senza shift).

[MorseCode.m](#) è uno script che utilizza iFilter per dimostrare le capacità e le limitazioni del filtro di Fourier. Crea un'onda sinusoidale pulsata a frequenza fissa che trasmette "SOS" in codice Morse (dit-dit-dit/dah-dah-dah/dit-dit-dit), aggiunge rumore bianco casuale in modo che l'SNR sia molto scarso (circa 0.1 in questo esempio), poi utilizza un filtro passa banda di Fourier sintonizzato sulla frequenza del segnale, per isolare il segnale dal rumore. Man mano che la larghezza di banda si riduce, il rapporto segnale/rumore inizia a migliorare e il segnale emerge dal rumore fino a quando non diventa chiaro, ma se la larghezza di banda è troppo stretta, il tempo di risposta del gradino è troppo lento per ottenere dei "dit" e "dah" distinti. Usare i tasti "?" e " " per regolare la larghezza di banda. (Il tempo di risposta al gradino è inversamente proporzionale alla larghezza di banda).

Premere 'P' o la barra spaziatrice per ascoltare il suono. È necessario installare [iFilter.m](#) nel path di Matlab. [Guardare su YouTube](#) all'indirizzo <https://youtu.be/agjs1-mNkmY>. (guardare la spiegazione nel titolo della figura durante la riproduzione del video).

[TestingOneTwoThree.wav](#) è l'audio di 1.58 secondi della frase "Testing, one, two, three", registrata a una frequenza di campionamento di 44000 Hz e salvata in formato WAV. Quando viene caricata in `iFilter(v=wavread('TestingOneTwoThree.wav'))` impostato sulla modalità passa banda e sintonizzato su un segmento stretto che è ben al di sopra della gamma di frequenza della maggior parte del segnale, potrebbe sembrare che questa banda passante manchi la maggior parte delle componenti di frequenza nel segnale, ma anche in questo caso il discorso è intelligibile, a dimostrazione della notevole capacità del sistema orecchio-cervello di accontentarsi di un segnale altamente compromesso. Premere P o spazio per ascoltare l'output del filtro. Diverse impostazioni del filtro cambieranno il [timbro](#) del suono. Vedere pagina 380. Cliccare per il [grafico](#).

Lo script [RealTimeFourierFilter.m](#) è una dimostrazione di un [filtro di Fourier](#) in tempo reale. Come gli [altri script di signal processing in tempo reale](#), questo pre-calcola un segnale simulato a partire dalla riga 38, poi accede ai dati punto per punto (righe 56, 57) e divide il flusso di dati in segmenti per calcolare ciascuna sezione filtrata. In questa dimostrazione, un filtro [passa banda](#) viene utilizzato per rilevare un'onda sinusoidale da 500 Hz ('f' nella riga 28) che si verifica nel terzo centrale di un segnale molto rumoroso (riga 32), da circa 0.7 sec a 1.3 sec. La frequenza centrale del filtro (CenterFrequency) e la larghezza (FilterWidth) sono impostate nelle righe 46 e 47.

Wavelet ed eliminazione del rumore

[Morelet.m](#) mostra l'applicazione della trasformata wavelet per rivelare le componenti di un segnale complicato. Codice scritto da Michael X. Cohen, in "A better way to define and describe Morlet wavelets for time-frequency analysis", *NeuroImage*, Volume 199, 1 October 2019, Pagine 81-86.

[MorletExample2.m](#) crea e analizza il segnale dei "picchi sepolti" costituito da tre componenti: una coppia di picchi Gaussiani deboli che sono le componenti desiderabili del segnale, una forte interferenza da parte di un'onda sinusoidale a frequenza variabile e un eccesso di rumore bianco. I picchi gaussiani sono invisibili nel segnale originale.

Misura dell'area del picco

[PerpDropAreas.m](#) `[AreaVector]=PerpDropAreas(x,y,startx,endx,MaxVector)` misura le aree dei picchi in x, y, iniziando un valore x di startX e terminando a endX, con le posizioni dei picchi specificate nel vettore MaxVector, che può essere di qualsiasi lunghezza. Utilizza il metodo del punto intermedio. Restituisce le aree nel vettore PDMeasAreas e gli indici del punto medio nel secondo argomento output opzionale.

[HeightAndArea.m](#) è uno script dimostrativo che utilizza [measurepeaks.m](#) per misurare i picchi in [segnali](#) generati al computer costituiti da una serie di picchi Gaussiani con larghezze gradualmente crescenti che vengono sovrapposte a una linea di base curva con in più del rumore bianco casuale. [Disegna il segnale](#) e i [singoli picchi](#) e confronta la posizione effettiva, le altezze e le aree di ciascun picco con quelle misurate da [measurepeaks.m](#) utilizzando i metodi dell'altezza assoluta del picco, della differenza picco-valle, del dislivello perpendicolare e del taglio tangente [tangent skim]. Stampa una [tabella](#) della differenza percentuale relativa tra i valori effettivi e quelli misurati per ciascun picco e l'errore medio per tutti i picchi.

[measurepeaks.m](#) rileva automaticamente i picchi in un segnale, simile a `findpeaksSG`. Restituisce una [tabella](#) con il numero del picco, la posizione, l'altezza assoluta, la differenza picco-valle, l'area col taglio verticale e l'area col taglio tangente di ciascun picco. Può [disegnare il segnale](#) e i [singoli picchi](#) se l'ultimo (il 7^o) argomento di input è 1. Digitare "help measurepeaks" e provare i sette esempi o eseguire [HeightAndArea.m](#) per eseguire un test sull'accuratezza della misura dell'altezza e dell'area del picco con segnali che hanno più picchi con rumore, sfondo e alcune sovrapposizioni.

Lo script [testmeasurepeaks.m](#) eseguirà tutti gli esempi con una pausa di 1 secondo tra ciascuno (richiede measurepeaks.m e gaussian.m nel path).

Lo script [SharpenedOverlapDemo.m \(grafico\)](#) mostra l'effetto dello sharpening sulle [misure dell'area col taglio verticale](#) di due picchi Gaussiani sovrapposti con altezza, separazione e larghezza regolabili, calcolando la percentuale di differenza tra l'area misurata sul segnale del picco sovrapposto rispetto alle aree reali dei picchi isolati.

[SharpenedOverlapCalibrationCurve.m](#) è uno script che simula la misura quantitativa di un mix di *tre* picchi Gaussiani sovrapposti. Lo sharpening delle derivate pari (la linea rossa nei grafici del segnale) viene utilizzata per migliorare la risoluzione dei picchi per consentire la misura dell'area col taglio verticale. Una linea retta viene approssimata alla curva di calibrazione e viene calcolato l' R^2 , per mostrare (1) la linearità della risposta e (2) l'indipendenza dei picchi adiacenti sovrapposti. Deve contenere gaussian.m, derivxy.m, autopeaks.m, val2ind.m, halfwidth.m, fastsmooth.m e plotit.m nel path.

[ComparePDAreas.m](#) confronta l'effetto dell'elaborazione digitale sulle aree di un insieme di picchi misurati con il metodo del taglio verticale. La sintassi è `[P1, P2, coef, R2] = ComparePDAreas(x, orig, processed, PeakSensitivity)`, dove x=variabile indipendente (p.es., il tempo); orig = valori y del segnale originale; processed = valori y del segnale processato; P1 = tabella dei picchi del segnale originale; P2 = tabella dei picchi del segnale processato; PeakSensitivity = numero approssimativo di picchi che si approssimerebbero all'intero intervallo dell'asse x (numeri più grandi > più picchi rilevati). Visualizza un grafico a dispersione delle aree originali rispetto alle aree calcolate per ciascun picco e restituisce le tabelle dei picchi, rispettivamente P1 e P2, e i valori di pendenza, intercetta e di R^2 , che idealmente dovrebbero essere 1,0 e 1, se l'elaborazione non ha alcun effetto sull'area del picco.

[iSignal](#) (pagina 379) è la funzione Matlab scaricabile che esegue varie funzioni di signal processing descritte in questo tutorial, inclusa la misura manuale, una alla volta, dell'area del picco utilizzando la regola di Simpson e il metodo del taglio verticale. Cliccare per visualizzare o click destro > Save link as... [qui](#), oppure è possibile scaricare il [file ZIP](#) con i dati di esempio per il test. La GIF animata iSignalAreaAnimation.gif ([clicare per vedere](#)) mostra iSignal che applica il metodo del taglio verticale a una serie di quattro picchi di area uguale. (Guardare nel pannello inferiore come gli intervalli di misura, contrassegnati dalle linee magenta tratteggiate verticali, vengono posizionati al minimo della valle su entrambi i lati di ciascuno dei quattro picchi). Ha anche un "peak fitter" integrato, attivato dal tasto **Shift-F**, basato su [peakfit.m](#), che misura le aree di un picco sovrapposto di forma nota. Esiste anche una funzione *automatica* di ricerca dei picchi basata sulla funzione [autopeaks](#), attivata dai tasti **J** o **Shift-J**, che visualizzano una [tabella](#) che elenca il numero del picco, la posizione, l'altezza assoluta, la differenza picco-valle, l'area di taglio verticale e l'area taglio tangente di ogni picco nel segnale.

[peakfit](#), una funzione della riga di comando per l'approssimazione di più picchi mediante metodo iterativo non lineare dei quadrati minimi. Misura la posizione, l'altezza, la larghezza e l'area dei picchi sovrapposti e ha diversi modi per [correggere le linee di base diverse da zero](#). Per ottenere i migliori risultati, è necessario che la forma dei picchi siano tra quelle [elencate qui](#).

[PeakCalibrationCurve.m](#) è una simulazione Matlab/Octave della calibrazione di un sistema di iniezione del flusso o cromatografia che produce segnali di picchi correlati a una concentrazione o ampiezza in esame ('amp'). La funzione [measurepeaks.m](#) viene utilizzata per determinare l'altezza assoluta del picco, la differenza picco-valle, l'area col taglio verticale e l'area col taglio tangente. Lo script Matlab/Octave [PeakShapeAnalyticalCurve.m](#) mostra che, per un singolo picco isolato la cui forma è costante e indipendente dalla concentrazione, se viene utilizzata il profilo errato, le altezze del picco misurate dall'approssimazione della curva saranno imprecise, ma tale errore sarà esattamente lo stesso per i campioni ignoti e gli standard di calibrazione noti, quindi l'errore verrà "annullato" e le concentrazioni misurate saranno accurate, a condizione che si utilizzi lo stesso modello impreciso sia per gli standard noti che per campioni ignoti. Vedere pagina 329.

[PowerTransformTest.m](#) è un semplice script che mostra il [metodo di potenza](#) dello sharpening dei picchi per aiutare a ridurre la sovrapposizione dei picchi. Gli script [PowerMethodGaussian.m](#) e [PowerMethodLorentzian.m](#) confrontano i metodi di potenza con la deconvoluzione, rispettivamente per il picco Gaussiano e Lorentziano. [PowerMethodCalibrationCurve](#) è una variante di [PeakCalibrationCurve.m](#) che valuta il [metodo di potenza](#) nel contesto di un'iniezione di flusso o di una misura cromatografica. La funzione autonoma [PowerMethodDemo.m](#) mostra il metodo di potenza per misurare l'area di una piccola protuberanza del picco che è parzialmente sovrapposto da un picco di interferenza molto più forte ([Grafico](#)). Dimostra anche l'effetto del rumore casuale, dell'attenuazione e di qualsiasi background non corretto sotto i picchi.

<https://terpconnect.umd.edu/~toh/spectrum/functions.htmlAsymmetricalAreaTest.m>. Test dell'accuracy dei metodi di misura dell'area per un picco asimmetrico, confrontando (A) la stima Gaussiana, (B) la triangolazione, (C) il metodo del taglio verticale e dell'approssimazione della curva mediante (D) una Gaussiana ampliata esponenzialmente e (E) due Gaussiane sovrapposte. Deve avere le seguenti funzioni nel path Matlab/Octave: gaussian.m, expgaussian.m, findpeaksplot.m, findpeaksTplot.m, autopeaks.m e peakfit.m. Lo script correlato [AsymmetricalAreaTest2.m](#) confronta le deviazioni standard di quegli stessi metodi con campioni di rumore randomizzati.

[SumOfAreas.m](#). Mostra che anche i picchi drasticamente non Gaussiani possono approssimarsi a un massimo di cinque componenti Gaussiane sovrapposte e che l'area totale delle componenti si avvicina all'area del picco non Gaussiano all'aumentare del numero di componenti ([grafico](#)). Nella maggior parte dei casi sono necessarie solo poche componenti per ottenere una buona stima dell'area del picco.

I minimi quadrati lineare

[TestLinearFit effect of number of points.txt](#). Effetto della dimensione del campione sulle stime dell'errore dei minimi quadrati dalla Simulazione Monte Carlo, dalla propagazione algebrica degli errori e dal metodo bootstrap, utilizzando lo script Matlab [TestLinearFit.m](#).

[LeastSquaresCode.txt](#). Semplice pseudo-codice per calcolare l'approssimazione dei minimi quadrati del primo ordine di y rispetto a x, inclusi Slope e Intercept e la deviazione standard prevista della pendenza (SDslope) e dell'intercetta (SDintercept).

[CalibrationQuadraticEquations.txt](#). Semplice pseudo-codice per calcolare l'approssimazione dei minimi quadrati del secondo ordine di y rispetto a x, inclusi i termini costanti, x e x2.

[plotit](#), versione 2, (precedentemente chiamato 'plotfit'), è una funzione per disegnare dati x,y in matrici o in vettori separati. Opzionalmente approssima i dati con un polinomio di ordine n se n è incluso come terzo argomento di input. Nella **versione 6** la sintassi è **[coef, RSquared, StdDevs] = plotit(x, y)** o facoltativamente **plotit(x, y, n, data-**

`style, fitstyle)`, dove `datastyle` e `fitstyle` sono stringhe opzionali che specificano lo stile e il colore della linea e del simbolo, nella convenzione standard di Matlab. Per esempio, `plotit(x,y,3,'or','-g')` disegna i dati come cerchi rossi e l'approssimazione come una linea continua verde (il valore predefinito è punti rossi e un linea blu, rispettivamente). Plotit restituisce i coefficienti più adatti '`coeff`', in potenze decrescenti di `x`, le deviazioni standard di tali coefficienti '`StdDevs`' nello stesso ordine e il valore `R` al quadrato. Digitare "help plotit" al prompt dei comandi per le opzioni della sintassi. Vedere pagina 172. Questa funzione funziona in Matlab o Octave e ha una routine di bootstrap nativa che calcola le stime di errore dei coefficienti (STD e % RSD di ciascun coefficiente) con il metodo bootstrap e restituisce i risultati nella matrice "BootResults" (di dimensione 5 x polyorder + 1). Il calcolo viene attivato includendo un argomento di output 4th, ad es. `[coef, RSquared, StdDevs, BootResults] = plotit(x,y,polyordine)`. Funziona per qualsiasi ordine polinomiale intero positivo. La variante `plotfit` anima il processo di bootstrap per scopi didattici. La variazione `logplotfit` disegna e approssima log(`x`) rispetto a log(`y`), per i dati che seguono una power law relationship o che coprono un intervallo numerico molto ampio.

[RSquared.m](#) Calcola la R^2 (R-quadro o coefficiente di correlazione) sia in Matlab che in Octave. La sintassi `RS=RSquared(polycoeff, x,y)`.

[trypoly\(x,y\)](#) approssima i dati in `x,y` con una serie di polinomi dal grado 1 a `length(x)-1` e restituisce i coefficienti di determinazione (R^2) di ogni approssimazione come vettore, consentendo di valutare come i polinomi di vari ordini si approssimano ai dati. Per disegnare come grafico a barre, scrivere `bar(trypoly(x,y)); xlabel('Polynomial Order'); ylabel('Coefficient of Determination (R2)')`. [Cliccare per un esempio](#). Vedere la funzione correlata [testnumpeaks.m](#).

[trydatatrans\(x,y,polyorder\)](#) prova 8 diverse trasformazioni di dati semplici sui dati `x,y`, approssima i dati trasformati a un polinomio di ordine '`polyorder`', visualizza i risultati [graficamente in array 3 x 3 di piccoli grafici](#) e restituisce tutti i valori R^2 in un vettore.

[LinearFiMC.m](#), uno script che confronta la deviazione standard della pendenza e dell'intercetta per un'approssimazione dei minimi quadrati del primo ordine calcolato dalla simulazione di numeri casuali di 1000 ripetizioni con le previsioni fatte da equazioni algebriche in forma chiusa. Vedere pagina 159.

[TestLinearFit.m](#), uno script che confronta la deviazione standard della pendenza e dell'intercetta per un'approssimazione dei minimi quadrati del primo ordine calcolato dalla simulazione di numeri casuali di 1000 ripetizioni alle previsioni fatte da equazioni algebriche in forma chiusa e al metodo di campionamento bootstrap. Diversi modelli di rumore possono essere selezionati commentando/rimuovendo il commento dal codice nelle righe 20-26. Vedere pagina 159.

[GaussFitMC.m](#), una funzione che dimostra la simulazione Monte Carlo della misura dell'altezza, della posizione e della larghezza di un picco Gaussiano `x,y` rumoroso. Vedere pagina 165.

[GaussFitMC2.m](#), una funzione che mostra la misura dell'altezza, della posizione e della larghezza di un picco Gaussiano `x,y` rumoroso, confrontando l'approssimazione parabolica `gaussfit` con l'approssimazione iterativa `fitgaussian`. Vedere pagina 165.

[SandPfrom1950.mat](#) è un file MAT contenente il valore giornaliero dell'[indice di borsa S&P 500](#) rispetto al tempo dal 1950 fino a settembre 2016. Questi dati sono utilizzati da [FitSandP.m](#) uno script Matlab/Octave che esegue un'approssimazione dei minimi quadrati dell'[equazione dell'interesse composto](#) al valore giornaliero, `V`, dell'[Indice di borsa S&P 500](#) rispetto al tempo, `T`, dal 1950 a settembre 2016, con due metodi: (1) il [metodo di approssimazione della curva iterativo](#) e (2) prendendo il [logaritmo dei valori](#) e approssimandoli a una linea retta. [SnPsimulation.m](#). Script Matlab/Octave che simula l'indice del mercato azionario S&P 500 aggiungendo rumore casuale proporzionale ai dati calcolati dall'[equazione dell'interesse composto](#) con un rendimento percentuale annuo noto, poi approssima l'equazione a quei dati sintetici rumorosi per i due metodi citati sopra. Vedere pagina [319](#).

[gaussfit.m](#) [**Height**, **Position**, **Width**]=**gaussfit(x,y)**. Questa funzione prende il logaritmo naturale di y, approssima una parabola (quadratica) ai dati (x, ln(y)), poi calcola la posizione, la larghezza e l'altezza della Gaussiana dai tre coefficienti dell'approssimazione quadratica.

[lorentzfit.m](#) [**Height**, **Position**, **Width**]=**lorentzfit(x,y)**. Questa funzione prende il reciproco di y, approssima una parabola (quadratica) ai dati (x, 1/y), poi calcola la posizione, la larghezza e l'altezza del Lorentziano dai tre coefficienti dell'approssimazione quadratica.

[OverlappingPeaks.m](#) è uno script demo che mostra come utilizzare gaussfit.m come un modo rapido per misurare due picchi Gaussiani parzialmente sovrapposti. Richiede un'attenta selezione delle regioni ottimali dei dati intorno alla parte superiore di ciascun picco (righe 15 e 16). Si provi a cambiare la posizione e l'altezza relative del secondo picco o ad aggiungere rumore (riga 3) e si osservi come influiscono sulla precisione. Questa funzione necessita delle funzioni gaussian.m e gaussfit.m nel path. I metodi iterativi funzionano molto meglio in questi casi, ma sono più lenti.

Ricerca e Misura dei Picchi

[allpeaks.m](#). **allpeaks (x,y)** Un semplicissimo 'peak detector', per insiemi di dati x,y, che elenca ogni valore di y che ha altri valori y più bassi da entrambi i lati; [allvalleys.m](#) è lo stesso per gli avvallamenti, elenca ogni valore y che ha valori y *più alti* da entrambi i lati. Una versione correlata, allpeaksw.m, stima anche la larghezza dei picchi.

[peaksat.m](#). (**peaks above threshold**) elenca ogni valore y che (a) ha valori y più bassi da entrambi i lati e (b) è che sia maggiore di una data soglia. Restituisce una matrice $2 \times n$ P con i valori x e y di ciascun picco, dove n è il numero di picchi rilevati. Una versione correlata, peaksatw.m, stima anche l'ampiezza dei picchi. La variante [peaksatG.m](#) ("Peaks Above Threshold/Gaussian" [Picchi sopra la soglia/Gaussiana]) esegue inoltre un'approssimazione dei minimi quadrati alla parte superiore di ciascun picco rilevato per stimarne la larghezza e l'area.

[findpeaksx.m](#), **P=findpeaksx(x,y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth, smoothtype)** è una semplice funzione a riga di comando per individuare e contare i picchi positivi in un insieme rumoroso di dati. È un'alternativa alla funzione findpeaks nel Signal Processing Toolkit. Rileva i picchi, cercando gli attraversamenti per lo zero della derivata prima filtrata con smoothing, che superano SlopeThreshold e le cui ampiezze superano AmpThreshold e restituisce un elenco (nella matrice P) contenente il numero, la posizione e l'altezza di ciascun picco. Può trovare e contare oltre 10.000 picchi al secondo in segnali molto grandi. Digitare "help findpeaksx.m". Vedere [PeakFindingandMeasurement.htm](#). La variante [findpeaksxw.m](#) misura anche la larghezza dei picchi. Vedere lo script demo [demofindpeaksxw.m](#).

[findpeaksG.m](#) e [findvalleys.m](#) cercano automaticamente i picchi o gli avvallamenti in un segnale misurandone posizione, altezza, larghezza e area col 'curve fitting'. La sintassi è **P=**
findpeaksG(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth, smoothtype). Restituisce una matrice contenente i parametri per ciascun picco rilevato. Per i picchi con profilo Lorentziano, invece, usare [findpeaksL.m](#). Cfr. pagina 227. Esistono molte varianti ed estensioni basate su questa funzione elementare. Vedere pagina 230.

[findpeaksplot.m](#) è una semplice variante di findpeaksG.m che disegna anche i dati x,y e il numero dei picchi sul grafico (se ne ha trovati). Sintassi: **findpeaksplot(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth, smoothtype)**

[OnePeakOrTwo.m](#) è uno script demo che crea un segnale che può essere interpretato sia come un picco a x=3 su una linea di base curva che come due picchi a x=.5 e a x=3, a seconda del contesto. In questa demo, la funzione findpeaksG.m function è stata chiamata due volte, con due diversi valori di SlopeThreshold.

[iPeak](#) (pagina 245) o la sua versione per Octave ipeakoctave.m, cerca, e misura, automaticamente più picchi in un segnale. (m-file link: [ipeak.m](#)). Seguire le [Istruzioni passo-passo animate](#). Il file ZIP [ipeak8.zip](#) contiene diversi script demo (ipeakdemo.m, ipeakdemo1.m, ecc.) che illustrano i vari aspetti della funzione iPeak e di come si possa usare efficacemente. [testipeak.m](#) è uno script che verifica la corretta installazione e funzionamento di iPeak eseguendo rapidamente [tutti gli otto esempi](#) e [le sei demo](#) per iPeak. Si presume che [ipeakdata.mat](#) sia stato caricato nello spazio di lavoro di

Matlab. [Cliccare per la presentazione degli esempi](#). La sintassi è `P=ipeak(DataMatrix, PeakD, AmpT, SlopeT, SmoothW, FitW, xcenter, xrange, MaxError, positions, names)`

[findpeaksSG.m](#) è una variante *segmentata* di [findpeaksG](#) con la stessa sintassi, tranne per il fatto che i parametri di rilevamento del picco possono essere *vettori*, dividendo il segnale in regioni ottimizzate per picchi di diverse larghezze. La sintassi è `P = findpeaksSG(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype)`. Funziona meglio di [findpeaksG](#) quando le larghezze dei picchi variano notevolmente lungo il segnale. Lo script [TestPrecisionFindpeaksSG.m](#) mostra l'applicazione. [Grafico](#). Vedere pagina 326.

[findpeaksSGw.m](#) è come sopra eccetto che usa il *wavelet denoising* (pag. 130) anziché lo smoothing. Come argomento di input accetta il livello della wavelet anziché che l'ampiezza uniforme. Lo script [TestPrecisionFindpeaksSGvsW.m](#) confronta la precisione e l'accuratezza della misura della posizione e dell'altezza del picco.

[autofindpeaks.m](#) (e [autofindpeaksplot.m](#)) sono simili a [findpeaksSG.m](#) tranne per il fatto si può *traslasciare il rilevamento dei parametri dei picchi* scrivendo semplicemente “`autofindpeaks(x,y)`” o `autofindpeaks(x,y,n)` dove *n* è la capacità dei picchi, all'incirca il numero di picchi che si approssimerebbero alla registrazione del segnale (un *n* maggiore, cerca più picchi ma stretti; *n* più piccolo cerca pochi picchi ma larghi). Stampa anche l'elenco degli argomenti di input da utilizzare con una qualsiasi delle funzioni [findpeaks...](#) Nella versione 1.1, si può chiamare [autofindpeaks](#) con gli argomenti di output [P,A] e restituisce i parametri di rilevamento dei picchi calcolati come un vettore di riga a 4 elementi A, che si può poi passare ad altre funzioni simili a [measurepeaks](#), dando effettivamente a tale funzione la capacità di calcolare i parametri di rilevamento del picco da un singolo numero *n*. Per esempio:

```
x=[0:.1:50];  
y=5+5.*sin(x)+randn(size(x));  
[P,A]=autofindpeaks(x,y,3);  
P=measurepeaks(x,y,A(1),A(2),A(3),A(4),1);
```

Digitare "help [autofindpeaks](#)" ed eseguire gli esempi. Lo script [testautofindpeaks.m](#) esegue tutti gli esempi nel file help, inoltre disegna i dati e numera i picchi (come [autofindpeaksplot.m](#)). [Animazione grafica](#).

[\[M,A\]=autopeaks.m](#) e [autopeaksplot.m](#). Rilevamento dei picchi e misura di altezza e area per picchi di forma arbitraria in dati x,y di serie temporali. La sintassi è `[P, DetectionParameters] = autofindpeaks(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype)`, ma come [autofindpeaks.m](#), i parametri di rilevamento del picco SlopeThreshold, AmpThreshold, smoothwidth, peakgroup e smoothtype si possono omettere e la funzione calcolerà i valori iniziali stimati. Utilizza l'algoritmo [measurepeaks.m](#) per la misura, restituisce una [tabella](#) nella matrice M contenente numero del picco, posizione, altezza assoluta, differenza picco-valle, area col taglio verticale e col taglio tangente per ciascun picco. Facoltativamente restituisce i parametri di rilevamento del picco che calcola nel vettore A. L'uso della semplice sintassi `M=autopeaks(x,y)` funziona bene in alcuni casi, altrimenti provare `M=autopeaks(x,y,n)`, utilizzando diversi valori per *n* (il numero approssimativo di picchi che si approssimerebbero alla registrazione del segnale) fino a quando non rileva i picchi che si desidera misurare. Per il controllo più preciso sul rilevamento dei picchi, è possibile specificare tutti i parametri di rilevamento digitando `M=autopeaks(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup)`; [autopeaksplot.m](#) è lo stesso ma [disegna anche il segnale](#) e contrassegna [i singoli picchi](#) (in blu) con il massimo (cerchi rossi), i punti di avvallamento (magenta) e le linee tangenti (ciano). Lo script [testautopeaks.m](#) esegue tutti gli esempi nel file help [autopeaks](#), con una pausa ci 1 secondo tra ciascuno, stampando i risultati nella finestra di comando e inoltre disegnando e numerando i picchi (Finestra 1) e ogni singolo picco (Finestra 2); richiede [gaussian.m](#) e [fastsmooth.m](#) nel path. [iSignal](#) (pagina 379) ha na funzione di ricerca del picco basata sulla funzione [autopeaks](#), attivata dai tasti **J** o **Shift-J**, che visualizza una [tabella](#) con numero del

picco, posizione, altezza assoluta, differenza picco-valle, area col taglio perpendicolare e col taglio tangente per ciascun picco nel segnale.

[findpeaksG2d.m](#) è una variante di findpeaksSG che può essere utilizzata per individuare i picchi positivi *e le sporgenze* in una serie x-y temporale. Rileva i picchi nel negativo della derivata seconda del segnale, cercando pendenze discendenti nella derivata terza che superino SlopeThreshold. Vedere [TestFindpeaksG2d.m](#). Sintassi: `P = findpeaksG2d(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype)`

[measurepeaks.m](#) rileva automaticamente i picchi in un segnale, come [findpeaksSG](#). `M = measurepeaks(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth, plots)`. Restituisce una [tabella M](#) con numero del picco, posizione, altezza assoluta, differenza picco-valle, area col taglio verticale e col taglio tangente per ciascun picco. Può [disegnare il segnale](#) e i [singoli picchi](#) se l'ultimo (il 7°) argomento di input è 1. Digitare "help measurepeaks" e provare i sette esempi presenti lanciare [HeightAndArea.m](#) per eseguire un test sull'accuratezza della misura dell'altezza e dell'area con segnali che hanno più picchi, del rumore, un background e alcuni dei picchi sono sovrapposti. Generalmente, i suoi valori per l'area col taglio verticale sono migliori per quei picchi che non hanno un background, anche se sono leggermente sovrapposti, mentre i valori per l'area col taglio tangente sono migliori per i picchi isolati su un background diritto o leggermente curvo. Nota: questa funzione usa lo [smoothing](#)(specificato dall'argomento di input SmoothWidth) solo per il *rilevamento* dei picchi; esegue le misure sui dati y *originali senza smoothing*. In alcuni casi, può essere utile eseguire comunque un filtraggio dei dati y prima di chiamare measurepeaks.m, utilizzando una qualsiasi funzione liscia di smoothing. Lo script [testmeasurepeaks.m](#) eseguirà tutti gli esempi nel file help di measurepeaks con una pausa di 1 secondo tra ciascuno (richiede measurepeaks.m e gaussian.m nel path). [Animazione grafica](#). Le funzioni correlate [wmeasurepeaks.m](#) e [testwmeasurepeaks.m](#) usano il *wavelet denoising* (pag. 130) anziché lo smoothing.

[findpeaksT.m](#) e [findpeaksTplot.m](#) sono varianti di findpeaks che misurano i parametri costruendo un triangolo attorno a ciascun picco con i lati tangentì ai suoi lati. [Esempio grafico](#).

[findpeaksb.m](#) è una variante di findpeaksG.m che misura in modo più accurato i parametri utilizzando l'approssimazione iterativa dei minimi quadrati della curva basato su [peakfit.m](#). Questo produce valori dei parametri migliori di quelli di findpeaks da solo, perché approssima l'intero picco, non solo la parte superiore, e perché dispone di 33 diversi [profili di picco](#) e per la sottrazione del background (lineare o quadratica). Funziona meglio con picchi isolati che non si sovrappongono. La sintassi è `P = findpeaksb(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, window, PeakShape, extra, BASELINEMODE)`. I primi sette argomenti di input sono esattamente gli stessi della funzione [findpeaksG.m](#); se si è utilizzato findpeaks o iPeak (pag. 245) per cercare e misurare i picchi nei segnali, si possono utilizzare gli stessi valori di input per findpeaksb.m. Lo script dimostrativo [DemoFindPeaksb.m](#) mostra come funziona findpeaksb3 con più picchi sovrapposti. Digitare "help [findpeaksb](#)" al prompt dei comandi. Vedere [PeakFindingandMeasurement.htm](#). Confrontarlo con i correlati findpeaksfit.m e findpeaksb3, di seguito. [Cliccare per la presentazione degli esempi](#).

[findpeakssb.m](#) è una variante segmentata di findpeaksb.m. Ha la stessa sintassi di findpeaksb.m, `P = findpeakssb(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, window, PeakShape, extra, NumTrials, BASELINEMODE)`, eccetto che gli argomenti di input SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, window, width, PeakShape, extra, NumTrials, BaselineMode e fixedparameters, possono essere tutti opzionalmente scalari o vettori con una voce per ciascun segmento, allo stesso modo di [findpeaksSG.m](#). Restituisce una matrice P che elenca il numero, la posizione, l'altezza, la larghezza, l'area, l'errore di approssimazione percentuale e lo "R2" di ciascun picco rilevato. [DemoFindPeaksSb.m](#) mostra questa funzione creando una serie di picchi Gaussiani le cui larghezze aumentano di un fattore di 25 volte e che vengono sovrapposti ad una linea di base curva con rumore bianco casuale gradualmente crescente; vengono utilizzati quattro segmenti, modificando il rilevamento e i valori di approssimazione della curva in modo che tutti i picchi vengano misurati accuratamente. [Grafico](#). [Stampa](#). Vedere pagina 326.

[findpeaksb3.m](#) è una variante di findpeaksb.m che approssima ciascun picco rilevato *insieme al picco seguente e al precedente* trovato da findpeaksG.m. Gestisce meglio i picchi sovrapposti rispetto a

`findpeaksb.m` e gestisce un numero maggiore di picchi meglio di `findpeaksfit.m`, ma *approssima solo i picchi trovati da findpeaks*. La sintassi è `P=findpeaksb3(x,y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, PeakShape, extra, NumTrials, BASELINEMODE, ShowPlots)`. I primi sette argomenti di input sono esattamente gli stessi della funzione `findpeaksG.m`; se si è utilizzato `findpeaks` o `iPeak` (pagina 245) per cercare e misurare i picchi nei segnali, si possono utilizzare gli stessi valori di input per `findpeaksb3.m`. Lo script dimostrativo `DemoFindPeaksb3.m` mostra come funziona `findpeaksb3` con più picchi sovrapposti.

`findpeaksfit.m` è essenzialmente una combinazione seriale di `findpeaksG.m` e `peakfit.m`. Utilizza il numero di picchi trovati da `findpeaks` e le loro posizioni e ampiezze come input per la funzione `peakfit.m`, che poi approssima l'intero segnale col profilo specificato. Tratta picchi non Gaussiani e sovrapposti meglio della sola `findpeaks`. Tuttavia, approssima solo i picchi trovati da `findpeaks`. La sintassi è `[P, FitResults, LowestError, BestStart, xi, yi] = findpeaksfit(x, y, SlopeThreshold, AmpThreshold, smoothwidth, peakgroup, smoothtype, peakshape, extra, NumTrials, BaselineMode, fixedparameters, plots)`. I primi sette argomenti di input sono esattamente gli stessi della funzione `findpeaksG.m`; se si è utilizzato `findpeaks` o `iPeak` (pagina 245) per cercare e misurare i picchi nei segnali, si possono utilizzare gli stessi valori di input per `findpeaksfit.m`. I restanti sei argomenti di input di `findpeaksfit.m` sono per la funzione `peakfit`; se è stata usata `peakfit.m` o `ipf.m` (pagina 405) per approssimare i picchi nei segnali, allora si possono utilizzare gli stessi valori per gli argomenti di input per `findpeaksfit.m`. Digitare "help `findpeaksfit`" per ulteriori informazioni. Vedere pagina 227. [Cliccare per un esempio animato](#).

`peakstats.m` usa lo stesso algoritmo di `findpeaksG.m`, ma calcola e restituisce una tabella di statistiche riassuntive degli intervalli dei picchi (l'intervallo sull'asse x tra i picchi rilevati adiacenti), altezze, larghezze e aree, elencando i valori massimo, minimo, medio, e la deviazione standard percentuale di ciascuno e, facoltativamente, la visualizzazione del grafico dei dati x, t con i picchi numerati nella Window 1, la tabella delle statistiche dei picchi nella finestra di comando e gli istogrammi degli intervalli, delle altezze, delle larghezze e delle aree dei picchi nella [Window 2](#). digitare "help `peakstats`". Vedere pagina 227. La versione 2, marzo 2016, aggiunge mediana e modalità.

`tablestats.m` (`PS=tablestats(P, displayit)`) è simile a `peakstats.m` tranne per il fatto che accetta come input una tabella dei picchi P come quella generata da `findpeaksG.m`, `findvalleys.m`, `findpeaksL.m`, `findpeaksb.m`, `findpeaksplot.m`, `findpeaksnr.m`, `findpeaksGSS.m`, `findpeaksLSS.m` o `findpeaksfit.m`, qualsiasi funzione che restituisca una tabella di picchi con almeno 4 colonne elencando il numero del picco, altezza, larghezza e area. Calcola gli intervalli dei picchi (l'intervallo dell'asse x tra i picchi adiacenti rilevati) e la deviazione standard massima, minima, media e percentuale di ciascuno e, facoltativamente, visualizza gli istogrammi degli intervalli dei picchi, delle altezze, delle larghezze e delle aree nella [Window 2](#). L'ultimo argomento opzionale `displayit = 1` se si devono visualizzare gli istogrammi, altrimenti no.

`findpeaksnr.m` è una variante di `findpeaksG.m` che calcola inoltre il [rapporto segnale/rumore](#) (SNR) di ciascun picco e lo restituisce nella 5^a colonna della tabella dei picchi. L'SNR viene calcolato come il rapporto tra l'altezza del picco e la radice quadrata del residuo medio (la differenza tra i dati effettivi e i minimi quadrati si approssima alla parte superiore del picco). Vedere [PeakFindingandMeasurement.htm](#).

`findpeaksE.m` è una variante di `findpeaksG.m` che stima inoltre l'errore percentuale relativo di approssimazione di ciascun picco (assumendo un profilo Gaussiano) e lo restituisce nella 6^a colonna della tabella dei picchi.

`findpeaksGSS.m` e `findpeaksLSS.m`, per picchi Gaussiani e Lorentziani rispettivamente, sono varianti di `findpeaksG.m` e `findpeaksL.m` che inoltre calcolano l'1% della posizioni iniziale e finale restituendole nella 6^a e 7^a colonne della tabella dei picchi. Vedere [PeakFindingandMeasurement.htm](#).

[findsquarepulse.m](#) (sintassi `S=findsquarepulse(t, y, threshold)`) individua gli impulsi rettangolari nel segnale t, y che superano un valore y di "threshold" e ne determina il tempo di inizio, l'altezza media (relativa alla linea di base adiacente) e la larghezza. [DemoFindsquare.m](#) crea un segnale di prova e chiama findsquarepulse.m per testarlo.

[findsteps.m](#) `P= findsteps(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, peakgroup)` individua i passaggi transitori positivi in dati rumorosi di serie temporali x-y, calcolando la derivata prima di y che supera SlopeThreshold, calcola l'altezza del gradino come differenza tra i valori massimo e minimo di y su un numero di punti dati pari a "Peakgroup". Restituisce l'elenco (P) con il numero del passo, le posizioni x e y della parte inferiore e superiore nonché l'altezza di ogni passo rilevato; "SlopeThreshold" e "AmpThreshold" regolano la sensibilità del passo; con valori più alti verranno trascurate le caratteristiche più piccole. Aumentando "SmoothWidth" vengono ignorati i falsi impulsi stretti e piccoli provocati dal rumore casuale o da "interferenze" nell'acquisizione dei dati. Vedere [findsteps.png](#) per un esempio reale. [findstepsplot.m](#) disegna il segnale e il numero di picchi.

[idpeaks](#), funzione per l'identificazione dei picchi. La sintassi è `[IdentifiedPeaks, AllPeaks] = idpeaks(DataMatrix, AmpT, SlopeT, sw, fw, maxerror, Positions, Names)`. Individua e identifica i picchi in DataMatrix che corrispondono alla posizione dei picchi nella matrice "Positions" con nomi corrispondenti in "Names". Digitare "help idpeaks" per ulteriori informazioni. Scaricare ed estrarre [idpeaks.zip](#) per un esempio funzionante o vedere l'**Esempio 8** a pagina 244.

[idpeaktable.m](#) `[IdentifiedPeaks]=idpeaktable(P, maxerror, Positions, Names)`. Confronta le posizioni dei picchi trovati nella tabella "P" con un database di picchi noti, sotto forma di un array di celle delle posizioni massime note ("Positions") e un array di celle di nomi corrispondenti ("Names"). Se la posizione di un picco trovato nel segnale è più vicina a uno di quelli noti per meno dell'errore massimo specificato ("maxerror"), quel picco viene considerato una corrispondenza e vengono immessi la posizione, il nome, l'errore e l'ampiezza del picco nella matrice di celle di output "IdentifiedPeaks". La tabella dei picchi può essere restituita da una qualsiasi delle funzioni di ricerca o approssimazione dei picchi, con una riga per ogni picco e colonne per il numero, la posizione e l'altezza del picco come prime tre colonne.

[demoipeak.m](#) è un semplice script demo che genera un segnale rumoroso con picchi, chiama iPeak, poi stampa una tabella dei parametri effettivi e un elenco dei picchi rilevati e misurati da iPeak per il confronto. Prima di eseguire questa demo, è necessario scaricare e mettere nel path di Matlab, [ipeak.m](#) (pag. 245). La versione Octave è [demoipeakoctave.m](#).

[DemoFindPeak.m](#), uno script dimostrativo che utilizza la funzione findpeaksG su dati sintetici rumorosi. Numera i picchi e ne stampa i parametri nella finestra di comando. Richiede che [gaussian.m](#) e [findpeaksG.m](#) siano presenti nel path. Vedere pagina 227.

[TestFindpeaksG2d.m](#). Script dimostrativo per findpeaks2d.m, mostra che questa funzione è in grado di localizzare i picchi risultanti in "protuberanze" che non producono un massimo distinto nel segnale originale. Rileva i picchi nel negativo della derivata seconda del segnale filtrata con smoothing (indicata come linea tratteggiata nella figura). Richiede gaussian.m, findpeaksG.m, findpeaksG2d.m, fastsmooth.m e peakfit.m nel path. [Grafico](#). Utilizza anche i risultati di [TestFindpeaksG2d](#) come valori "start" per l'approssimazione iterativa utilizzando [peakfit.m](#), che richiede più tempo per il calcolo ma fornisce risultati più accurati, specialmente per larghezza e area:

[DemoFindPeakSNR](#) è una variante di DemoFindPeak.m che usa [findpeaksnr.m](#) per calcolare il rapporto segnale/rumore (SNR) del segnale di ciascun picco e lo restituisce nella 5^a colonna.

[triangulationdemo.m](#) è una funzione demo ([schermata](#)) che confronta [findpeaksG](#) (che determina i parametri del picco con l'approssimazione ad una Gaussiana al centro di ciascun picco) con [findpeaksT](#), che determina i parametri col metodo della costruzione del triangolo (disegnando un triangolo attorno a ciascun picco con i lati tangenti ai suoi lati). Esegue il confronto con 4 diverse forme di picco: Gaussiana semplice, Gaussiana biforcata, Gaussiana esponenzialmente espansa, e la Breit-Wigner-Fano). In alcuni casi, il metodo di costruzione del triangolo può essere più accurato del metodo Gaussiano se la forma del picco è asimmetrica.

[findpeaksfitdemo.m](#), uno script dimostrativo di findpeaksfit che trova e approssima automaticamente i picchi in un insieme di 150 segnali, ognuno dei quali può avere da 1 a 3 picchi Lorentziani rumorosi in posizioni variabili. Richiede l'installazione delle funzioni [findpeaksfit.m](#) e [lorentzian.m](#). Questo script è stato utilizzato per generare l'animazione GIF [findpeaksfit.gif](#).

[FindpeaksComparison.m](#). Quale usare: [findpeaksG](#), [findpeaksb](#), [findpeaksb3](#) o [findpeaksfit](#)? Questo script confronta tutte e quattro le funzioni applicate a un segnale generato dal computer con più picchi con tipi e quantità variabili di linea di base e rumore casuale. (Richiede tutte queste funzioni, più [modelpeaks.m](#), [findpeaksG](#) e [findpeaksL.m](#), nel path di Matlab/Octave. Digitare "help FindpeaksComparison" per i dettagli). [I risultati vengono visualizzati graficamente](#) nelle Window 1, 2 e 3 e stampati in una [tabella di parametri con l'accuratezza e il tempo impiegato per ciascun metodo](#). È possibile modificare le righe nello script contrassegnate da <<< per modificare il numero, il carattere e l'ampiezza dei picchi, della linea di base e del rumore. (Regolare i parametri per rendere il segnale simulato simile a quello sperimentale per scoprire quale metodo funziona meglio per il proprio tipo di segnale). Il metodo migliore dipende principalmente dalla forma e dall'ampiezza della linea di base e dall'entità della sovrapposizione dei picchi.

[iPeakEnsembleAverageDemo.m](#) è uno script dimostrativo per la funzione della media d'insieme di iPeak. In questo esempio, il segnale contiene un modello ripetuto di due picchi Gaussiani sovrapposti, a 12 punti di distanza, entrambi di larghezza 12, con un rapporto di altezza 2:1. Questi schemi capitano a intervalli casuali in tutto il segnale registrato e il livello di rumore casuale è circa il 10% dell'altezza media del picco. Utilizzando la funzione della media dell'insieme di iPeak (**Shift-E**), i pattern possono essere mediati e migliorare il rapporto segnale/rumore.

[ipeakdata.mat](#), set di dati dimostrativi di [idpeaks.m](#) o per la funzione di identificazione del picco di iPeak; include uno spettro atomico ad alta risoluzione e una tabella delle lunghezze d'onda di emissione note. Vedere pagina 227.

Quale usare: iPeak o Peakfit? Provare queste funzioni demo di Matlab che confrontano [iPeak.m](#) (pag. 245) con [peakfit.m](#) (pag. 227) per segnali con [pochi picchi](#) e quelli con [molti picchi](#) e questo mostra come regolare iPeak per rilevare [picchi larghi o stretti](#). Queste sono demo autonome che includono tutte le sotto-funzioni richieste. Basta piazzarle nel path e digitare il nome sulla riga di comando. Si possono scaricare tutte assieme in [idemos.zip](#). Non richiedono argomenti di input o output.

[SpikeDemo1.m](#) e [SpikeDemo2.m](#) sono script Matlab/Octave che dimostrano come due misurano picchi (picchi molto stretti) molto disturbati da altri segnali. Vedere pagina 296.

[PowerTransformTest.m](#) è un semplice script che mostra il [metodo di potenza](#) dello sharpening dei picchi per aiutare a ridurre la sovrapposizione dei picchi. [PowerMethodCalibrationCurve](#) è una variante di [PeakCalibrationCurve.m](#) che valuta il metodo della potenza nel contesto di un'iniezione di flusso o di una misura cromatografica. [powertest2](#) è una funzione autonoma che mostra il metodo della potenza per misurare l'area del piccolo picco sporgente ([Grafico](#)).

Lo script [realtimepeak.m](#) mostra un semplice rilevamento dei picchi in tempo reale basato sul passaggio dello zero della derivata, utilizzando i clic del mouse per simulare i dati. Ogni volta che il clic del mouse forma un picco (ovvero, va su e poi di nuovo giù), il programma registrerà ed etichetterà il picco sul grafico (come illustrato a lato) e ne stamperà i valori x e y . In questo caso, un picco è definito come qualsiasi punto che ha punti adiacenti di ampiezza inferiore su entrambi i lati, determinato dai cicli "for" annidati nelle righe 31-36. Lo script più sofisticato [RealTimeSmoothedPeakDetectionGauss.m](#) utilizza la tecnica descritta a [pagina 227](#) che individua i picchi positivi in un insieme rumoroso di dati che aumentano sopra una soglia di ampiezza impostata, esegue un'approssimazione dei minimi quadrati di una funzione Gaussiana alla parte superiore del picco dei dati originali, calcola la posizione, l'altezza e la larghezza (FWHM) di ciascun picco da quell'approssimazione e stampa ogni picco trovato nella finestra di comando. ([Grafico animato](#)).

[AreasOfIsolatedPeaks.m](#). Script per dimostrare la misura delle aree di picchi isolati sovrapposti a una linea di base variabile, con il metodo trapezoidale, utilizzando la funzione trapz nativa.

NEW Il tool per il rilevamento dei picchi. [PeakDetection mlx \(pagina 245\)](#) è un [Live Script](#) interattivo per il rilevamento e la misura dei picchi, inclusa una selezione di rilevatori, [smoothing](#), [simmetrizzazione](#), [sharpening](#) e [approssimazione della curva](#), con cursori interattivi e menù a discesa per controllarli in modo interattivo.

Spettroscopia Multicomponente

[cls.m](#) è una funzione dei minimi quadrati classici utilizzabile per approssimare un modello generato dal computer, costituito da un numero qualsiasi di picchi di forma, larghezza e posizione noti, ma di altezza sconosciuta, ad un segnale x,y rumoroso. La sintassi è `heights= cls(x,y, NumPeaks, PeakShape, Positions, Widths, extra)` dove x e y sono i vettori del segnale misurato (ad esempio x potrebbe essere la lunghezza d'onda e y potrebbe essere l'assorbanza a ciascuna lunghezza d'onda), "NumPeaks" è il numero di picchi, "PeakShape" è il numero della forma del picco (1=Gaussiana, 2=Lorentziana, 3=logistica, 4=Pearson, 5=Gaussiana espansa esponenzialmente; 6=Gaussiane di larghezze uguali; 7=Lorentziane di larghezze uguali; 8=Gaussiana espansa esponenzialmente di uguale larghezza, 9=impulso esponenziale, 10=sigmoide, 11=Gaussiana a larghezza fissa, 12=Lorentziana a larghezza fissa; 13=Mix Gaussiana/Lorentziana; 14=BiGaussiana, 15=BiLorentziana), 'Positions' è il vettore delle posizioni dei picchi lungo l'asse x (una voce per ogni picco), 'Widths' è il vettore delle larghezze in unità x (una voce per ogni picco) e 'extra' è un parametro addizionale richiesto dai profili espansi esponenzialmente di Pearson, mix Gaussiana/Lorentziana, BiGaussiana e BiLorentziana. `cls.m` restituisce un vettore della altezze di ciascun picco misurato. Vedere [clsdemo.m](#). (Nota: questo metodo è ora incluso nel peak fitter iterativo non lineare [peakfit.m](#) (pagina 227) col numero di profilo 50. Vedere lo script dimostrativo [peakfit9demo.m](#))

La funzione [cls2.m](#) è simile a [cls.m](#), tranne per il fatto che misura anche la linea di base (che si presume sia piatta) e restituisce un vettore contenente il background B e altezze dei picchi misurate H per ogni picco, p.es. $[B \ H1 \ H2 \ H3\dots]$.

[RegressionDemo.m](#), script che dimostra la classica procedura dei minimi quadrati per uno spettro di assorbimento simulato di una miscela di 5 componenti a 100 lunghezze d'onda. Richiede che [gaussian.m](#) sia presente nel path. Vedere pagina 179.

[clsdemo.m](#) uno script dimostrativo che crea un segnale rumoroso, lo approssimazione utilizzando il metodo dei minimi quadrati classici con [cls.m](#), calcola la precisione delle altezze misurate, quindi ripete il calcolo dei [quadrati minimi iterativi](#) utilizzando [peakfit.m](#) (pagina 227) per un confronto. (Questo script richiede [cls.m](#), [modelpeaks.m](#) e [peakfit.m](#) nel path di Matlab/Octave).

[CLVsINLS.m](#) è uno script che confronta il metodo dei minimi quadrati classico (CLS) con tre diverse varianti del metodo iterativo (INLS) per misurare le altezze di tre picchi Gaussiani in un segnale rumoroso di test, dimostrando che minore è il numero di parametri sconosciuti, più veloce e preciso è il calcolo dell'altezza del picco.

Approssimazione iterativa non lineare e approssimazione dei picchi

[gaussfit](#), funzione che esegue l'approssimazione dei minimi quadrati di una singola funzione Gaussiana a un insieme di dati x,y, restituendo l'altezza, la posizione e la larghezza della Gaussiana più adatta. La sintassi è [Height, Position, Width] = [gaussfit\(x, y\)](#). La funzione simile, [lorentzfit.m](#), esegue il calcolo per un profilo Lorentziano. Vedere pagina 164. La funzione simile, [plotgaussfit](#), fa la stessa cosa di [gaussfit.m](#) ma disegna anche i dati e l'approssimazione. Il set di dati non può contenere valori zero o negativi.

[bootgaussfit](#) è una versione espansa di [gaussfit](#) che fornisce grafici opzionali e la stima degli errori. La sintassi è [Height, Position, Width, BootResults] = [bootgaussfit\(x, y, plots\)](#). Se plots=1, disegna i dati originali come punti rossi e la Gaussiana, approssimazione migliore, come una linea. Se viene fornito il 4° argomento di output (BootResults) calcola le stime di errore dei parametri con il metodo [bootstrap](#).

[fitshape2.m](#), sintassi [**Positions**, **Heights**, **Widths**, **FittingError**] = [fitshape2\(x, y, start\)](#), è una funzione Matlab/Octave *semplificata d'uso generico* per approssimare più tipi di profili sovrapposti ai dati contenuti nel vettore delle variabili x e y. Il modello è una combinazione lineare di un numero qualsiasi di funzioni di base definite matematicamente come una funzione di x, con due variabili che il programma determinerà indipendentemente per ogni picco, posizioni e larghezze, oltre alle altezze (cioè, i pesi della somma ponderata). Si deve fornire il vettore iniziale di prima ipotesi 'start', nella forma [posizione1 larghezza1 posizione2 larghezza2 ...ecc.], che specifica la prima ipotesi di posizione e larghezza per ciascuna componente (una coppia di posizione e larghezza per ciascun picco nel modello). La funzione restituisce i parametri del modello 'best-fit' nei vettori **Positions**, **Heights**, **Widths**, e calcola l'errore percentuale tra i dati e il modello in **FittingError**. Inoltre disegna i dati come punti e il modello approssimato come una linea. La cosa interessante di questa funzione è che *l'unica parte che definisce la forma del modello è l'ultima riga*. In fitshape2.m, quella riga contiene l'espressione per un *picco Gaussiano di altezza unitaria*, ma o si può cambiare in *qualsiasi espressione o algoritmo* che calcola g come una funzione di x con due parametri ignoti 'pos' e 'wid' (posizione e larghezza, rispettivamente, per le forme dei picchi, ma potrebbero rappresentare qualsiasi cosa per altri tipi di funzione, come l'impulso esponenziale, sigmoidale, ecc.); tutto il resto nella funzione [fitshape.m](#) può rimanere lo stesso. Ciò rende fitshape una buona piattaforma per sperimentare diverse espressioni matematiche come modelli proposti per approssimare i dati. Ci sono anche altre due varianti di questa funzione per i modelli con *una* variabile iterata più l'altezza del picco ([fitshape1.m](#)) e *tre* variabili iterate più l'altezza del picco ([fitshape3.m](#)). Ciascuna ha esempi illustrativi contenuti nel file di help integrato (digitare "help <nomefile>"). Una versione alternativa è [FitMultipleShapes2](#), che consente di specificare una qualsiasi delle 16 funzioni del profilo del picco in base al numero. Digitare "help FitMultipleShapes2" per gli esempi. Sintassi:
[Positions, Heights, Widths, FittingError] = FitMultipleShapes2(x, y, shape, start, m)

[peakfit](#) (pagina 227) una versatile funzione a riga di comando per l'approssimazione di più picchi mediante i minimi quadrati iterativi non lineari. Una "[Scelta della settimana](#)" di Matlab File Exchange. La sintassi completa è [FitResults, GOF, baseline, coeff, BestStart, xi, yi, BootResults] = [peakfit\(signal, center, window, NumPeaks, peakshape, extra, NumTrials, start, BASELINEMODE, fixedwidth, plots, bipolar, minwidth\)](#). Digitare "help peakfit". Cfr. pagina 384. Rispetto alla funzione fitshape.m descritta in precedenza, [peakfit.m](#) ha un gran numero di profili *integrati* di picco selezionati per numero, non richiede (sebbene si possano fornire) la prima ipotesi della posizione e della larghezza di ciascuna componente, e dispone di funzionalità per la correzione del background e altre funzioni utili per migliorare la qualità e stimare l'affidabilità delle approssimazioni. Verificare l'installazione sul computer eseguendo lo script [autotestpeakfit.m](#), che esegue l'intera sfilza di test delle approssimazioni senza interruzioni, stampando ciò che sta facendo e i risultati, controllando se l'errore di approssimazione è maggiore del previsto e stampa un AVVISO se capita. Impiega 17

secondi girando in Matlab 9.9 2020b su una macchina 3.5Ghz i7 con Windows 10. Vedere lo [storico delle versioni](#), pagina 384, per una breve descrizione delle nuove funzionalità in ciascuna versione di peakfit.m dalla 3.7 ad oggi.

[testnumpeaks](#)(x,y,peakshape,extra,NumTrials,MaxPeaks). Semplice test per stimare il numero del profilo dei picchi necessari per approssimare un set di dati x,y. Approssima dati x,y, col profilo "peakshape", con un opzionale parametro extra, "extra", con NumTrials ripetizioni per ogni approssimazione, fino ad un massimo di "MaxPeaks" profili, mostra ciascuna approssimazione e disegna l'errore di approssimazione rispetto al numero di profili. Se due o più numeri di picchi danno circa lo stesso errore, è meglio prendere il numero più piccolo.

[SmoothVsFit.m](#) è uno script dimostrativo che confronta l'approssimazione iterativa dei minimi quadrati con due metodi più semplici per la misura dell'altezza di un singolo picco di larghezza e posizione incerta e con un pessimo rapporto segnale-rumore di 1. L'accuratezza e la precisione dei metodi vengono confrontate. [SmoothVsFitArea.m](#) fa la stessa cosa per la misura dell'area del picco. Vedere pagina 165.

[ipf.m](#) (pag. 405) è un'approssimazione interattiva di più picchi (link al file m: [ipf.m](#)). Utilizza i minimi quadrati iterativi non lineari per approssimare un numero qualsiasi di picchi sovrapposti della stessa forma o di forme di picco diverse ad un set di dati x-y. [Demoipf.m](#) è uno script dimostrativo per ipf.m, con un generatore di segnali simulato incluso. I valori reali delle posizioni, delle altezze e delle larghezze dei picchi simulati vengono visualizzati nella finestra dei comandi di Matlab, per il confrontarli con i FitResults ottenuti mediante l'approssimazione del picco. [Cliccare per le istruzioni passo-passo animate](#). Si può anche scaricare un [file ZIP](#) contenente ipf.m più degli esempi e demo. [Cliccare per un esempio animato](#).

[SmallPeak.m](#) è una dimostrazione di diverse tecniche di approssimazione della curva applicate al difficile problema di misurare l'altezza di un piccolo picco strettamente sovrapposto e completamente oscurato da un picco molto più grande. Confronta le approssimazioni iterative di modelli non vincolati, di uguale larghezza e con posizione fissa (utilizzando [peakfit.m](#), pagina 227) con una approssimazione dei minimi quadrati classica in cui *solo* le altezze dei picchi sono ignote (utilizzando [cls.m](#)). Allargare le quattro finestre in modo da poter osservare la notevole differenza di stabilità dei vari metodi. Una tabella finale degli errori relativi all'altezza percentuale del picco mostra che più ci sono i vincoli, migliori sono i risultati (ma *solo se i vincoli sono giustificati*). Vedere pagina 315.

[BlackbodyDataFit.m](#), uno script che mostra l'approssimazione dei minimi quadrati iterativa dell'*equazione del corpo nero* a uno spettro misurato di un corpo incandescente, allo scopo di stimarne la temperatura del colore. Vedere pagina 199.

[Demofitgauss.m](#) uno script che mostra l'approssimazione iterativa di una *singola* funzione Gaussiana ad un insieme di dati, utilizzando la funzione fminsearch. Richiede che [gaussian.m](#) e fmsearch.m (nel pacchetto "Optim 1.2.1")

siano installate [Demofitgaussb.m](#) e [fitgauss2b.m](#) illustrano una modifica di questa tecnica per approssimare lo spostamento della linea di base ([Demofitlorentzianb.m](#) e [fitlorentzianb.m](#) per i picchi Lorentziani). Questa modifica è ora inclusa in peakfit.m (versione 4.2 e successive), ipf.m (versione 9.7 e successive), findpeaksb.m (versione 3 e successive) e findpeaksfit, (versione 3 e successive). Cfr. pagina 199.

[Demofitgauss2.m](#) uno script che mostra l'approssimazione iterativa, di *due* funzioni Gaussiane sovrapposte, a un insieme di dati, utilizzando la funzione fminsearch. Richiede che [gaussian.m](#) e fmsearch.m (nel pacchetto "Optim 1.2.1") siano installate. Demofitgauss2b.m è l'estensione per la linea di base corretta. Vedere pagina 199.

[VoigtFixedAlpha.m](#) e [VoigtVariableAlpha.m](#) mostrano due modi diversi per approssimare i picchi con *forme variabili*, come i profili Voigt, Pearson, mix Gauss-Lorentz e le versioni biforcate ed esponenzialmente espande, che sono definite non solo da posizione, altezza e larghezza del picco, ma anche da un parametro aggiuntivo "shape" che ottimizza la forma del picco. Se quel parametro è *uguale* per tutti i picchi in un gruppo, può essere passato come argomento di input aggiuntivo alla funzione shape, come mostrato in [VoigtFixedAlpha.m](#). Se il parametro shape può essere *diverso* per ogni picco nel gruppo e deve essere determinato per iterazione (proprio come la posizione e la larghezza), la routine deve essere modificata per adattarsi a *tre*, anziché *due*, variabili iterate, come mostrato in [VoigtVariableAlpha.m](#). Sebbene l'*errore di approssimazione* è *basso* con le variabili alfa, il tempo di esecuzione è più lungo e i *valori alfa così determinati non sono molto stabili*, rispetto al rumore nei dati e ai valori di prima ipotesi, soprattutto per picchi multipli. Vedere pagina 195. Lo script [VoigtShapeFittingDemonstration.m](#) usa peakfit.m versione 9.5 per approssimare un singolo profilo Voigt e per calcolare la componente della larghezza Gaussiana, della larghezza Lorentziana, e l'alfa. Calcola il profilo Voigt teorico e aggiunge rumore casuale per il realismo.

[VoigtShapeFittingDemonstration2.m](#) fa lo stesso per due profili Voigt sovrapposti, utilizzando entrambi i modelli con alfa fisso e alfa variabile (numeri di forma 20 e 30). (Richiede voigt.m, halfwidth.m e peakfit.m nel path).

[Demofitmultiple.m](#). Mostra un approssimazione iterativa ad un set di picchi rumorosi generati dal computer di diversi tipi, conoscendo solo il tipo e i parametri del profilo variabile di ciascun picco. I parametri iterati sono forma, altezza, posizione e larghezza per tutti i picchi. Richiede le funzioni [fitmultiple.m](#) e [peakfunction.m](#). [Visualizzare la schermata](#). Cfr. pagina 195.

[BootstrapIterativeFit.m](#), una funzione che mostra la stima bootstrap della variabilità di un'approssimazione iterativa dei minimi quadrati a un singolo picco Gaussiano rumoroso. La sintassi è: **BootstrapIterativeFit(TrueHeight, TruePosition, TrueWidth, NumPoints, Noise, NumTrials)**. Vedere pagina 162.

[BootstrapIterativeFit2.m](#), una funzione che mostra la stima bootstrap della variabilità di un'approssimazione iterativa dei minimi quadrati a due picchi Gaussiani rumorosi. La sintassi è: **BootstrapIterativeFit2(TrueHeight1, TruePosition1, TrueWidth1, TrueHeight2, TruePosition2, TrueWidth2, NumPoints, Noise, NumTrials)**. Vedere pagina 162.

[DemoPeakfitBootstrap.m](#). Funzione dimostrativa autonoma per peakfit.m (pagina 227), con generatore di segnale integrato. Dimostra la stima dell'errore di bootstrap. Vedere pagina 162.

[DemoPeakfit.m](#), Script dimostrativo (per peakfit.m) che genera un segnale con picchi sovrapposti, aggiunge rumore, lo approssima con peakfit.m, poi calcola l'accuratezza e la precisione delle misure dei parametri di picco. Richiede che [peakfit.m](#) sia presente nel path. Vedere pagina 402.

[peakfit9demo](#). Dimostra la regressione multilineare (profilo 50) disponibile in peakfit.m versione 9 (richiede modelpeaks.m e peakfit.m nel path di Matlab). Crea un segnale rumoroso di tre picchi di forme, posizioni e larghezze note, ma altezze sconosciute. Confronta la regressione multilineare nella Finestra 1 con i minimi quadrati non lineari iterativi non vincolati nella Finestra 2.

[DemoPeakFitTime.m](#) è un semplice script che mostra come utilizzare peakfit.m per applicare l'*approssimazione multipla della curva ad un segnale che cambia nel tempo*. Il segnale contiene due picchi Gaussiani rumorosi in cui la posizione del secondo picco aumenta col tempo e gli altri parametri rimangono costanti (eccetto il rumore). ([cliccare per eseguire l'animazione](#)).

[isignal](#) (pagina 379) può essere utilizzato come funzione a riga di comando in Octave, ma le sue caratteristiche interattive funzionano solo in Matlab. La sintassi è `isignal(DataMatrix, xcenter, xrange, SmoothMode, SmoothWidth, ends, DerivativeMode, Sharpen, Sharp1, Sharp2, SlewRate, MedianWidth)`.

[testpeakfit.m](#), uno script di test che mostra i 36 diversi esempi a pagina 405. Utilizzare per verificare che peakfit e le relative funzioni siano presenti nel path. [autotestpeakfit.m](#) fa lo stesso senza fermarsi tra le funzioni e attendere la pressione di un tasto (impiega circa 17 secondi per essere eseguito).

NEW Il tool Live Script di approssimazione dei picchi, simile agli altri strumenti di Live Script per [smoothing](#), [deconvoluzione](#), [rilevamento del picco](#), è descritto a pagina 427. Si basa sulla funzione `peakfit.m` e può approssimare una varietà di profili di picchi con posizione iniziale specificata opzionale, correzione della linea di base e simmetrizzazione dell'ampliamento esponenziale.

Approssimazioni di più picchi con profili diversi. [ShapeTestS.m](#) e [ShapeTestA.m](#) testano i dati degli argomenti di input `x,y`, supponendo che sia un singolo picco isolato, lo approssimano con diversi modelli candidati utilizzando `peakfit.m`, disegnano ciascuna approssimazione in una finestra separata, e stampano una tabella con gli errori di approssimazione nella finestra di comando.

[ShapeTestS.m](#) prova sette diversi modelli simmetrici candidati di picco e [ShapeTestA.m](#) prova sei diversi modelli asimmetrici candidati. Quello con l'errore di approssimazione più basso (e R^2 più prossimo a 1.000) è probabilmente il candidato migliore. Provare gli esempi dei file di help. Ma attenzione: se c'è troppo rumore nei dati, i risultati possono risultare fuorvianti.

[WidthTest.m](#) è uno script che dimostra che vincolare alcuni dei parametri di un modello di approssimazione a valori fissi, se tali valori sono accuratamente noti, migliora l'accuratezza della misurazione degli altri parametri, anche se aumenta l'errore di approssimazione. Richiede l'installazione delle funzioni [GL.m](#) e [peakfit.m](#) (versione 7.6 o successiva) nel path di Matlab/Octave.

Lo script [NumPeaksDemo.m](#) mostra un modo per tentare di stimare il numero minimo di picchi del modello necessari per approssimare un insieme di dati, tracciando l'errore di approssimazione rispetto al numero di picchi del modello e cercando il punto in cui l'errore di approssimazione raggiunge il minimo. Questo script crea un segnale rumoroso generato al computer contenente 3, 4, 5 o 6 picchi Lorentziani in esame selezionati dall'utente e utilizza `peakfit.m` per approssimare i dati a una serie di modelli contenenti da 1 a 10 picchi. Il numero corretto di picchi è o l'approssimazione con l'errore più basso o, se due o più approssimazioni hanno circa lo stesso errore, quello col minor numero di picchi, come in [questo esempio](#), che in realtà ha 4 picchi. Se i dati sono molto rumorosi, tuttavia, la determinazione diventa inaffidabile. (Per rendere questa demo più vicina al proprio tipo di dati, si potrebbe cambiare il Lorentziano in Gaussiano o qualsiasi altro profilo, o cambiare l'ampiezza del picco, il numero di punti o il livello di rumore). Questo script richiede che `peakfit.m` e le [funzioni di profilo appropriate](#) (`gaussian.m`, `lorentzian.m`, ecc.) siano presenti nel path. La funzione [testnumpeaks.m](#) fa questo per i dati utente `x,y`.

Test sui tempi di Peakfit. Si tratta di una serie di script che mostrano come il tempo di esecuzione della funzione [peakfit.m](#) varia con la forma del picco ([PeakfitTimeTest2.m](#) e [PeakfitTimeTest2a.m](#), con il numero di picchi nel modello ([PeakfitTimeTest.m](#)) e col numero di punti nella regione approssimata ([PeakfitTimeTest3.m](#)). Questo problema è trattato a pagina 420.

[TwoPeaks.m](#) è un semplice script di 8 righe che confronta `findpeaksG.m` e `peakfit.m` con un segnale costituito da due picchi rumorosi. `findpeaksG.m` e `peakfit.m` devono stare nel path di Matlab/Octave.

[peakfitVSfindpeaks.m](#) esegue un confronto diretto dell'accuratezza di `findpeaksG` rispetto a `peakfit`. Questo script genera [quattro picchi molto rumorosi](#) di diverse altezze e larghezze, poi applica `findpeaksG.m` e `peakfit.m` per misurare i picchi e confronta i risultati. I picchi rilevati da `findpeaks`

vengono etichettati come "Peak 1", "Peak 2", ecc. Se si esegue questo script più volte, si scoprirà che entrambi i metodi funzionano bene la maggior parte del tempo, con peakfit che fornisce errori minori nella maggior parte dei casi, ma a volte findpeaks perderà il primo picco (più basso) e raramente rileverà un picco extra che non c'è se il segnale è molto rumoroso.

[CaseStudyC.m](#) è una funzione dimostrativa Matlab/Octave autonoma che mostra l'applicazione di diverse tecniche descritte in questo sito per la misura quantitativa di un picco sepolto in un background instabile, situazione che può verificarsi nell'analisi quantitativa di varie forme di spettroscopia e telerilevamento. Vedere [Caso di Studio C.](#)

[GaussVsExpGauss.m](#) Confronto tra i vari modelli di Gaussiane esponenzialmente espanso non vincolate, profili 31 e 39. Il profilo 31 ([expgaussian.m](#)) crea la forma eseguendo una convoluzione di Fourier di una Gaussiana specificata mediante un decadimento esponenziale della costante di tempo specificata, mentre la forma 39 ([expgaussian2.m](#)) usa un'espressione matematica per la forma finale così prodotta. Entrambi danno lo *stesso profilo* ma sono parametrizzati in modo diverso. Il profilo 31 riporta l'altezza e la posizione del picco come quella della Gaussiana originale prima dell'ampliamento, mentre la forma 39 riporta l'altezza del picco del risultato ampliato. Il profilo 31 riporta la larghezza come FWHM della Gaussiana originale e il profilo 39 riporta la deviazione standard (sigma) di quella Gaussiana. Il profilo 31 riporta il fattore esponenziale sul *numero di punti* e il profilo 39 riporta il *reciproco della costante di tempo* in unità di tempo. Vedere le Finestre [2](#) e [3](#). Si devono avere [peakfit.m](#) (versione 8.4) [gaussian.m](#), [expgaussian.m](#), [expgaussian2.m](#), [findpeaksG.m](#) e [halfwidth.m](#) nel path di Matlab/Octave. [DemoExpgaussian.m](#) è uno script che fornisce un'esplorazione, più dettagliata, dell'effetto dell'ampliamento esponenziale su un picco Gaussiano (richiede: gaussian.m, expgaussian.m, halfwidth.m, val2ind.m e peakfit.m nel path di Matlab/Octave).

[AsymmetricalOverlappingPeaks.m](#) è uno script a più passaggi che dimostra l'uso di una combinazione di simmetrizzazione con la derivata prima, prima dell'approssimazione della curva per analizzare un complesso picco misterioso. Vedere pagina 358).

Funzioni interattive operanti da tastiera

Le funzioni interattive **ipeak**, **isignal**, **ipf** e **ifilter** vengono eseguite nella finestra "Figure" e utilizzano un semplice insieme di comandi da tastiera a carattere singolo, anziché pulsanti, menù o cursori sullo schermo, per ridurre l'ingombro dello schermo, ridurre al minimo il sovraccarico, massimizzare la velocità di elaborazione e consentire di esplorare i dati e provare vari approcci in modo semplice e rapido. Tutte hanno diversi comandi da tastiera in comune: condividono tutte lo stesso set di *pan* e *zoom* per regolare la porzione del segnale visualizzata nel pannello superiore. (Ci sono anche versioni *Octave* di tutti questi, **ipeakoctave**, **isignaloctave**, **ipfoctave** e **ifilteroctave**, utilizzano tutti i tasti diversi per le regolazioni del pan e dello zoom rispetto alle versioni Matlab). Tutte le versioni usano il tasto **K** per visualizzare l'elenco dei comandi da premere. Doppio-click sulla barra del titolo della figura per espanderla a schermo intero e visualizzare meglio le caratteristiche dei piccoli segnali. Tutte usano il tasto **T** per scorrere tra le modalità di correzione della linea di base. Tutte usano i tasti **Shift-Ctrl-S**, **Shift-Ctrl-F** e **Shift-Ctrl-P** per trasferire il segnale corrente tra **iSignal**, **ipf** e **iPeak**, rispettivamente. Per rendere più semplice il trasferimento delle impostazioni da una di queste funzioni ad altre funzioni correlate, tutte usano il tasto **W** per stampare la sintassi delle altre funzioni correlate, con le impostazioni di pan e zoom e altri argomenti di input numerici specificati, pronti per essere copiati, incollati e modificati nei propri script o re-inseriti nella finestra di comando. Ad esempio, è possibile convertire un'operazione di approssimazione della curva eseguita in **ipf.m** nella riga di comando della funzione **peakfit.m**; oppure è possibile convertire un'operazione di ricerca dei picchi eseguita in **ipeak.m** in una funzione **findpeaksG.m** o **findpeaksb.m** a riga di comando. Il tasto **W** è utile con segnali che richiedono un'elaborazione del segnale diversa in diverse regioni sull'asse x, consentendo di creare una serie di funzioni a riga di comando per ciascuna regione che, se eseguite in sequenza, elaborano rapidamente ogni segmento del segnale in modo appropriato e si possono ripetere facilmente per un numero qualsiasi di altri esempi dello stesso tipo di segnale. Per regolare parametri variabili in modo continuo, questi programmi utilizzano *copie di*

tasti adiacenti per aumentare o diminuire ciascun parametro a passi, spesso col tasto shift si controlla la dimensione di tale passo.

Spettrofotometria di Assorbimento Quantitativo Iperlineare

[tfit.m](#), una funzione Matlab/Octave autonoma a riga di comando che mostra un [metodo computazionale](#) per l'analisi quantitativa mediante spettroscopia di assorbimento a lunghezza d'onda che utilizza la convoluzione e l'approssimazione iterativa della curva per correggere la non linearità. La sintassi è `tfit(TrueAbsorbance)`. [TFitStats.m](#) è uno script che mostra la riproducibilità del metodo. [TFitCalCurve.m](#) confronta le curve di calibrazione per i metodi a lunghezza d'onda singola, a regressione semplice a regressione pesata e TFit. [TFit3.m](#) è una funzione demo per una miscela di 3 componenti assorbenti; la sintassi è `TFit3(TrueAbsorbanceVector)`, p.es., `TFit3([3 .2 5])`. Scaricare tutti questi come un unico file [ZIP](#). [Cliccare per un esempio animato](#). [TFitDemo.m](#) è un 'esploratore' *interattivo* da tastiera per il metodo Tfit, applicato alla misura di una singola componente con un picco di assorbimento Lorentziano (o Gaussiano), con controlli che consentono di regolare la vera assorbanza ("Peak A"), l'ampiezza spettrale del picco di assorbimento ("AbsWidth"), l'ampiezza spettrale della funzione dello strumento ("InstWidth"), la luce diffusa e il livello del rumore ("Noise") in modo continuo osservando gli effetti graficamente e numericamente. Vedere pagina 268. [Cliccare per un esempio animato](#). Queste funzioni e script funzionano anche nell'ultima versione di Octave.

File MAT (per Matlab e [Octave](#)) e i file Testo (.txt)

[DataMatrix2](#) è un segnale di test generato al computer composto da 16 picchi Gaussiani simmetrici con l'aggiunta di rumore bianco casuale. Utilizzabile per testare la funzione `peakfit.m`. Vedere pagina 220.

[DataMatrix3](#) è un segnale di test generato al computer composto da 16 picchi Gaussiani, con rumore bianco casuale, ampliati esponenzialmente con una costante di tempo di 33 unità dell'asse x. Vedere pagina 220.

[udx.txt](#): un file di testo contenente la matrice 2 x 1091 composta da due picchi Gaussiani con intervalli di campionamento differenti. Viene utilizzato come esempio nello [Smoothing](#) e nel [Curve Fitting](#).

[TimeTrial.txt](#), un file di testo che confronta la velocità di vari compiti di signal processing, utilizzando le seguenti diverse configurazioni software:

- (a) Matlab 2020b su Windows 10, 64-bit, 3.6 GHz, core i7, 16 GBytes RAM
- (b) Matlab 2009a, su una vecchia macchina Windows
- (c) Matlab 2017b Home, su una vecchia macchina Windows
- (d) Matlab Online, R2018b, in Google Chrome
- (e) Matlab Mobile (su un recente iPad)
- (f) Octave 6.2.0 su Windows 10, 64-bit, 3.6 GHz, core i7, 16 GBytes RAM

Il codice Matlab/Octave che lo genera è [TimeTrial.m](#), che esegue tutte le attività una dopo l'altra e stampa i tempi impiegati dalla macchina corrente più i tempi precedentemente registrati per ciascuna attività su ciascuna dei cinque sistemi software. [TimeTrial.xlsx](#) riassume i confronti tra Matlab e Octave.

[Readability.txt](#). Rapporto sull'analisi della leggibilità in lingua inglese di [IntroToSignalProcessing.pdf](#) eseguita da http://www.online-utility.org/english/readability_test_and_improve.jsp

Spreadsheet (per Excel e OpenOffice Calc)

Note. Questi fogli di calcolo sono autonomi e quindi non si basano su file esterni. Vi si possono inserire i propri dati utilizzando il tab Data e/o il Copia-Incolla.

Se appare una barra gialla nella parte superiore della finestra del foglio di lavoro, cliccare sul pulsante "Enable Editing".

Se il browser cambia l'estensione di questi fogli di calcolo in .zip quando vengono scaricati, rinominarli con le loro estensioni originali (.ods, .xls o .xlsx) prima di eseguirli.

Questi fogli di calcolo non hanno celle protette, quindi nulla ne impedisce la modificare accidentale delle formule. Ciò significa che si possono modificare qualsiasi aspetto di questi fogli di calcolo per i propri scopi, cosa che si è invitati a fare. Se si sbaglia, basta usare la funzione Undo (**Ctrl-Z**) o se ne può scaricare un'altra copia.

Numeri casuali e rumore (pag. 22). Gli spreadsheet [RandomNumbers.xls](#) (per Excel) e [RandomNumbers.ods](#) (per OpenOffice) mostrano come creare una colonna di numeri casuali normalmente distribuiti (come il rumore bianco) in uno spreadsheet che ha solo la funzione per i numeri casuali uniformemente distribuiti. Mostra anche come calcolare l'intervallo interquartile, il valore picco-picco e come si confrontano con la deviazione standard. Vedere pagina 22. La stessa tecnica viene utilizzata nel foglio di calcolo [SimulatedSignal6Gaussian.xlsx](#), che calcola e disegna un segnale simulato composto da un massimo di 6 bande Gaussiane sovrapposte più del rumore bianco casuale.

Smoothing (pagina 38). I fogli di calcolo [smoothing.ods](#) (per Open office Calc) e [smoothing.xls](#) (per Microsoft Excel) mostrano uno smoothing rettangolare di 7 punti (slittamento della media) nella colonna C e uno triangolare a 7 punti nella colonna E, applicati ai dati nella colonna A. Ci si possono digitare (o copiare e incollare) tutti i dati che si vogliono nella colonna A. Lo spreadsheet si può estendere con colonne più lunghe trascinando l'ultima riga delle colonne A, C ed E in basso secondo la necessità. È possibile modificare la larghezza dello smoothing cambiando le equazioni nelle colonne C o E. Il foglio di calcolo [MultipleSmoothing.xls](#) per Excel o Calc mostra un metodo più flessibile che consente di definire vari tipi di smoothing digitando qualche numero intero. Gli spreadsheet [UnitGainSmooths.xls](#) e [UnitGainSmooths.ods](#) contengono una raccolta di coefficienti di convoluzione a guadagno unitario per gli smoothing rettangolari, triangolari e P-spline con larghezze da 3 a 29 sia in formato verticale (colonna) che orizzontale (riga). Si possono Copiare e Incollare nei propri spreadsheet. [Convolution.txt](#) elenca alcuni semplici set di coefficienti di numeri interi per eseguire smoothing a passi singoli e multipli. [VariableSmooth.xlsx](#) mostra una tecnica ancora più potente e flessibile, soprattutto per larghezze di smoothing ampie e variabili, che utilizza le funzioni AVERAGE e INDIRECT (pagina 345). Permette di cambiare la larghezza dello smoothing semplicemente cambiando il valore di una singola cella. Vedere pagina 49 per i dettagli. [SegmentedSmoothTemplate.xlsx](#) è un template per lo smoothing segmentato a larghezza multipla, che può applicare larghezze di smoothing diverse per ciascun segmento del segnale, particolarmente utile se le larghezze dei picchi o il livello del rumore varia sostanzialmente lungo il segnale. In questa versione ci sono 20 segmenti. [SegmentedSmoothExample.xlsx](#) è un esempio con dati ([grafico](#)). Un foglio correlato [GradientSmoothTemplate.xlsx](#) ([grafico](#)) esegue uno smoothing con larghezza linearmente crescente (o decrescente) lungo il segnale, dando solo i valori di inizio e fine, genera automaticamente i segmenti necessari.

Differenziazione (pagina 58). [DerivativeSmoothingOO.ods](#) (per OpenOffice Calc) e [DerivativeSmoothing.xls](#) (per Excel) mostrano l'applicazione della differenziazione per misurare l'ampiezza di un picco sepolti in un ampio background curvo. Differenziazione e smoothing vengono entrambe eseguite insieme. Le derivate di ordine superiore vengono calcolate prendendo le derivate delle derivate calcolate in precedenza. [DerivativeSmoothingWithNoise.xlsx](#) è uno spreadsheet correlato che mostra l'effetto drammatico dello smoothing sul rapporto segnale-rumore delle derivate su un segnale rumoroso. Utilizza lo stesso segnale di [DerivativeSmoothing.xls](#), ma aggiunge rumore bianco simulato ai dati Y. Si può controllare la quantità di rumore aggiunto. [SecondDerivativeXY2.xlsx](#),

mostra l'individuazione e la misura dei cambiamenti nella derivata seconda (una misura di curvatura o accelerazione) di un segnale che cambia nel tempo, mostrando l'apparente aumento del rumore causato dalla differenziazione e la misura in cui il rumore può essere ridotto attenuando (in questo caso con due passaggi di uno smoothing triangolare a 5 punti). La derivata seconda filtrata con smoothing mostra un grande picco nel punto in cui l'accelerazione cambia e si appiattisce su entrambi i lati mostrando l'entità dell'accelerazione prima e dopo il cambiamento (2 e 4, rispettivamente). [Convolution.txt](#) elenca semplici insiemi di coefficienti di numeri interi per eseguire la differenziazione e lo smoothing. [CombinedDerivativesAndSmooths.txt](#) elenca gli insiemi di coefficienti per un guadagno unitario che eseguono dalla 1^a alla 4^a derivata con vari gradi di smoothing. Vedere pagina 58.

Sharpening dei picchi (pag. 74). I metodi di sharpening con due derivate (2^a e 4^a) sono disponibili sotto forma di un template vuoto ([PeakSharpeningDeriv.xlsx](#) e [PeakSharpeningDeriv.ods](#)) o con dati di esempio inseriti ([PeakSharpeningDerivWithData.xlsx](#) e [PeakSharpeningDerivWithData.ods](#)). È possibile digitare i valori dei fattori di ponderazione delle derivate K1 e K2 direttamente nelle celle J3 e J4, oppure è possibile immettere l'ampiezza stimata del picco (FWHM in numero di punti) nella cella H4 e il foglio di calcolo calcolerà K1 e K2. È disponibile una versione demo con picchi simulati regolabili ([PeakSharpeningDemo.xlsx](#) e [PeakSharpeningDemo.ods](#)), così come una [versione con pulsanti cliccабili](#) per una comoda regolazione interattiva dei fattori K1 e K2 dell'1% o del 10% per ciascun click. Esiste anche una versione a 20 segmenti in cui è possibile specificare le costanti dello sharpening per ciascuno dei 20 segmenti del segnale ([SegmentedPeakSharpeningDeriv.xlsx](#)). Per le applicazioni in cui le larghezze dei picchi aumentano (o diminuiscono) gradualmente nel tempo, è disponibile anche un template per lo sharpening graduale ([GradientPeakSharpeningDeriv.xlsx](#)) e un esempio con dei dati già inseriti ([GradientPeakSharpeningDerivExample.xlsx](#)); è necessario solo impostare le larghezze del picco iniziale e finale e il foglio di calcolo applicherà i fattori di sharpening richiesti K1 e K2. [Peak-SymmetricalizationDemo.xlsxm](#) ([grafico](#)) mostra la simmetrizzazione delle Gaussiane modificate esponenzialmente (EMG) mediante [l'aggiunta ponderata della derivata prima](#) (e consente inoltre un ulteriore sharpening con la derivata seconda del picco risultante simmetrizzato). C'è anche un template vuoto [PeakSymmetricalizationTemplate.xlsxm](#) ([grafico](#)) e un'applicazione di esempio con dati di esempio già digitati: [PeakSymmetricalizationExample.xlsxm](#). [PeakDoubleSymmetrizationExample.xlsxm](#) esegue la simmetrizzazione di un picco *doppiamente* ampliato esponenzialmente. Dispone di pulsanti per regolare interattivamente le due fasi della ponderazione della derivata prima. Due variazioni ([1](#), [2](#)) includono dati di esempio per due picchi sovrapposti, per i quali le aree dopo la simmetrizzazione sono misurate col metodo del taglio verticale. [ComparisonOfPerpendicularDropAreaMeasurements.xlsx](#) ([grafico](#)) mostra l'effetto del [metodo di sharpening con la potenza](#) sulle misure dell'area col taglio verticale di picchi Gaussiani e Gaussiani espansi esponenzialmente, compreso l'effetto della risoluzione, altezza relativa, rumore casuale, attenuazione e linea di base diversa da zero sul metodo di sharpening normale e con la potenza. [PowerSharpeningTemplate.xlsx](#) è un template vuoto che esegue questo metodo e [PowerSharpeningExample.xlsx](#) è lo stesso con dati di esempio.

Convoluzione (pag. 103). Gli spreadsheets possono essere utilizzati per eseguire la convoluzione "trasla-e-moltiplica" per piccoli set di dati (ad esempio, [MultipleConvolution.xls](#) o [MultipleConvolution.xlsx](#) per Excel e [MultipleConvolutionOO.ods](#) per Calc), che è essenzialmente la stessa tecnica dei fogli di calcolo precedente per lo smoothing e la differenziazione. Utilizzare questo foglio di calcolo per studiare la convoluzione, lo smoothing, la differenziazione e l'effetto di tali operazioni sul rumore e sul rapporto segnale/rumore. (Per set di dati più grandi le prestazioni sono più lente della convoluzione di Fourier, che è molto più facile in Matlab o Octave che nei fogli di calcolo). [Convolution.txt](#) elenca semplici insiemi di coefficienti di numeri interi per eseguire la differenziazione e lo smoothing.

Misura dell'area dei picchi (pag. 126). [EffectOfDx.xlsx](#) mostra che la somma della semplice equazione $\text{sum}(y)^*dx$ misura accuratamente l'area di un picco Gaussiano isolato se ci sono almeno 4 o 5 punti visibilmente al di sopra della linea di base. [EffectOfNoiseAndBaseline.xlsx](#) mostra l'ef-

fetto del rumore casuale e della linea di base diversa da zero, mostrando che l'area è più sensibile alla linea di base diversa da zero rispetto alla stessa quantità di rumore casuale. [PeakSharpeningAreaMeasurementDemo.xlsx \(schermata\)](#) mostra l'effetto dello [sharpening derivativo](#) sulle misure delle aree col taglio verticale di due picchi Gaussiani. Lo sharpening dei picchi riduce il grado di sovrapposizione e può ridurre notevolmente gli errori di misura dell'area commessi dal metodo del *taglio verticale* (pagina 135). I fogli di calcolo elencati in "Sharpening" nella pagina precedente comprendono la misura dell'area dei picchi.

Approssimazione della curva [Curve Fitting] (pagina 154). [LeastSquares.xls](#) e [LeastSquares.odt](#) eseguono approssimazioni dei minimi quadrati a un modello lineare e [QuadraticLeastSquares.xls](#) e [QuadraticLeastSquares.ods](#) fanno lo stesso per un modello quadratico (parabolico). Esistono versioni specifiche di questi spreadsheet che calcolano anche le concentrazioni delle incognite (scaricare il set completo [CalibrationSpreadsheets.zip](#)).

Spettroscopia multicomponente (pag. 179). [RegressionTemplate.xls](#) e [RegressionTemplate.ods \(grafico con dati di esempio\)](#) esegue l'analisi multicomponente utilizzando il [metodo della matriciale](#) per un set di dati *fisso* di 5 componenti, 100 lunghezze d'onda. [RegressionTemplate2.xls](#) utilizza una tecnica di foglio di calcolo più avanzata (pagina 345) che consente al template di *adattarsi automaticamente* a diversi numeri di componenti e lunghezze d'onda. Due esempi mostrano lo stesso template con i dati inseriti per una miscela di 5 componenti misurati a 100 lunghezze d'onda ([RegressionTemplate2Example.xls](#)) e per 2 componenti a 59 lunghezze d'onda ([RegressionTemplate3Example.xls](#)).

Approssimazione del picco (pagina 165). Una serie di fogli di calcolo che utilizzano la funzione [Solver](#) per eseguire [l'approssimazione iterativa non-lineare del picco](#) per più modelli di picchi sovrapposti, è descritta [qui](#). Esistono versioni per forme di picco Gaussiana e Lorentziana, con e senza linea di base, per modelli con 2-6 picchi e 100 lunghezze d'onda (con le istruzioni per la modifica). Tutti questi hanno nomi di file che iniziano con "[CurveFitter...](#)".

Rilevamento e misurazione dei picchi (pag. 227). Lo spreadsheet [PeakAndValleyDetectionTemplate.xlsx](#) (o [PeakAndValleyDetectionExample.xlsx](#) con i dati di esempio), è un semplice rilevatore di picchi e valli che definisce un picco come qualsiasi punto con punti inferiori su entrambi i lati e una valle come qualsiasi punto con punti più alti su entrambi i lati (vedere pagina 464). Lo spreadsheet [PeakDetection.xls](#) implementa un metodo di rilevamento dei picchi con l'attraversamento dello zero della derivata più selettivo descritto a pagina 230. In entrambi i casi, i dati di input x,y sono contenuti nello Sheet1, colonne A e B, a partire dalla riga 9. (Qui, vi si possono incollare i propri dati). Vedere [PeakDetectionExample.xlsx/.xls](#) per un esempio con i dati già incollati. [PeakDetectionDemo2.xls/xlsx](#) è una dimostrazione con una serie di picchi generata al computer e controllata dall'utente. [PeakDetectionSineExample.xls](#) è una demo che genera una sinusoide con un numero di picchi regolabile.

Un'estensione di tale metodo viene eseguita in [PeakDetectionAndMeasurement.xlsx \(schermata\)](#), presupponendo che i picchi siano *Gaussiani* e misura la loro altezza, posizione e larghezza sui dati *senza smoothing* utilizzando una *tecnica dei quadrati minimi*, proprio come "[findpeaksG.m](#)". Il vantaggio di questa tecnica è che elimina la distorsione del picco che potrebbe derivare dallo smoothing dei dati per evitare falsi picchi derivanti da rumore casuale. Per i primi 10 picchi trovati, i dati originali x,y, senza smoothing, vengono copiati rispettivamente negli Sheets da 2 a 11, dove quel segmento di dati è soggetto a un'approssimazione Gaussiana dei minimi quadrati, utilizzando la stessa tecnica di [GaussianLeastSquares.xls](#). I risultati dei parametri Gaussiani più adatti vengono copiati di nuovo in Sheet1, nella tabella nelle colonne **AH-AK**. (Nella sua forma attuale. Lo spreadsheet è limitato a misurare 10 picchi, sebbene possa rilevare un qualsiasi numero di picchi. Inoltre, ha dei valori di larghezza dello smoothing [Smooth Width] e larghezza dell'approssimazione [Fit Width] limitati dai coefficienti della convoluzione da 17 punti). Il foglio di calcolo è disponibile nei formati OpenOffice ([.ods](#)) nonché Excel ([.xls](#)) e ([.xlsx](#)). Sono funzionalmente equivalenti e differiscono so-

lo per dei particolari cosmetici. È disponibile un foglio di calcolo di [esempio](#), con i dati. È disponibile anche una [versione demo](#), con una forma d'onda rumorosa calcolata che è possibile modificare. Vedere pagina 265. Se i picchi nei dati sono troppo sovrapposti, i massimi potrebbero non essere sufficientemente distinti per un rilevamento affidabile. Se il livello di rumore è basso, i picchi possono essere evidenziati artificialmente dalla [tecnica di sharpening derivativa descritta in precedenza](#). Questo è implementato da [PeakDetectionAndMeasurementPS.xlsx](#) e dalla sua versione demo [PeakDetectionAndMeasurementDemoPS.xlsx](#).

Spreadsheets per il Metodo TFit (pag. 268): Spettrofotometria di Assorbimento Quantitativo Iperlineare. [TransmissionFittingTemplate.xls \(schermata\)](#) è un template vuoto per un singolo picco isolato; [TransmissionFittingTemplateExample.xls \(schermata\)](#) è lo stesso template con i dati di esempio inseriti. [TransmissionFittingDemoGaussian.xls \(schermata\)](#) è una dimostrazione con un picco simulato di assorbimento Gaussiano con posizione, larghezza e altezza variabili, più luce diffusa, rumore fotonico e il rumore del rivelatore, visto da uno spettrometro con una funzione di fenditura triangolare. È possibile variare tutti i parametri e confrontare la migliore approssimazione dell'assorbanza con l'altezza del picco reale e all'assorbanza convenzionale $\log(1/T)$.

[TransmissionFittingCalibrationCurve.xls \(schermata\)](#) include una [macro](#) Excel (pagina 308) che costruisce automaticamente una curva di calibrazione e confronta i metodi TFit e il convenzionale $\log(1/T)$, per una serie di 9 concentrazioni standard che è possibile specificare.

Tecniche speciali per gli spreadsheet (pag. 345): “[SpecialFunctions.xlsx](#)” ([Grafico](#)) mostra l'applicazione delle funzioni MATCH, INDIRECT, COUNT, IF e AND quando si gestiscono array di dimensione variabile. “[IndirectLINEST.xls](#)” ([link al Grafico](#)) mostra il vantaggio particolare dell'utilizzo della funzione INDIRECT insieme alle funzioni sugli *array* come INV e LINEST.

Epilogo

Come è nato questo libro.

Durante la mia carriera presso l'Università del Maryland nel Dipartimento di Chimica e Biochimica, ho svolto [ricerche in chimica analitica](#) e sviluppato e tenuto diversi corsi, tra cui un corso universitario di laboratorio di livello superiore in “[Elettronica per Chimici](#)”, che negli anni '80 comprendeva una componente informatica di laboratorio e un esperimento di acquisizione ed elaborazione di dati digitali utilizzando tecniche matematiche e numeriche per l'elaborazione di dati sperimentali da strumenti scientifici. I chimici analitici come me sono fondamentalmente costruttori di strumenti. Agli albori della nostra professione, gli strumenti erano principalmente chimici (ad esempio, reagenti colorati), ma negli anni seguenti includevano strumenti (ad esempio, spettroscopia e cromatografia), e alla fine del 20° secolo strumenti software. Quando il Web è diventato disponibile per la comunità accademica nei primi anni '90, come molti insegnanti, ho messo a disposizione degli studenti un programma, esperimenti e altro materiale di studio per questo e per gli altri miei corsi online.

Quando andai in pensione dall'Università nel 1999, dopo 30 anni di servizio, notai che ricevevo pagine visualizzate sul sito del corso che provenivano dall'esterno dell'Università e dall'estero, in particolare indirizzate all'esperimento di laboratorio sull'elaborazione digitale dei dati che avevo condotto. Si era sviluppato negli anni '80, quando i computer erano relativamente nuovi nei laboratori di chimica. Ho iniziato a ricevere un numero crescente di e-mail con domande, suggerimenti e commenti da persone in campi scientifici molto diversi. Alla fine, ho deciso di fare di questo un progetto di pensionamento a lungo termine e di ampliarlo oltre la chimica e il mio specifico corso. Lo scopo è quello di aiutare gli scienziati ad apprendere e applicare tecniche matematiche di elaborazione dei dati basate su computer, producendo materiale educativo gratuito

spiegando le cose in modo intuitivo piuttosto che con formalismo matematico, con esempi di codifica, software pratico e guida/consulenza su progetti specifici. Per rendere questo lavoro utile ad un pubblico il più ampio possibile, compresi quelli con risorse limitati, sono state fatte diverse scelte:

- a. È tutto gratuito: il libro (in formato elettronico), il software, l'aiuto e la consulenza. -solo la versione cartacea del libro è a pagamento, ed è [disponibile su Amazon](#).
- b. Il libro e la documentazione sono disponibili in più formati: HTML, PDF e DOCX.
- c. La scrittura è al livello dell'11° grado, in uno stile semplice, privo di gergo inutile, modi di dire, figure retoriche, metafore, riferimenti culturali, sarcasmo, ironia e umorismo, tutti elementi che potrebbero essere difficili per chi ha una conoscenza limitata dell'inglese o per i traduttori automatici.
- d. La matematica formale è ridotta al minimo. Mi affido maggiormente alla logica, agli esempi pratici, alla grafica, alle analogie e alle animazioni per spiegare i concetti.
- e. È possibile utilizzare più piattaforme hardware: PC, Mac, Unix e dispositivi portatili.
- f. Vengono utilizzate molteplici piattaforme software: Matlab, Octave, Python, Excel e fogli di calcolo Open Office. Alcune sono gratuite.

Chi ha bisogno di questo software?

Il software non è incluso in tutti gli acquisti di hardware per strumenti scientifici moderni? Questo è vero, soprattutto per coloro che utilizzano strumenti convenzionali in modo standard. Ma molti scienziati stanno lavorando in nuove aree di ricerca per le quali non ci sono strumenti commerciali, o stanno usando sistemi esistenti modificati per i quali non c'è software, o stanno costruendo tipi di strumenti completamente nuovi. In alcuni casi, il software fornito con strumenti commerciali è non è flessibile, non adeguatamente documentato o è difficile da usare. Non tutti i ricercatori o gli operatori scientifici amano programmare, o hanno tempo per farlo, o sono bravi. I programmatori assunti tipicamente non capiscono la scienza e comunque prima o poi vanno avanti e non aggiornano più il loro codice. Il codice ben documentato è più importante che mai. Mi piace scrivere e scrivere codice, quindi questa sembrava essere una nicchia in cui potevo inserirmi.

Organizzazione

Il mio progetto ha cinque parti:

- Un libro, intitolato "A Pragmatic Introduction to Signal Processing", disponibile su [carta](#), Kindle, in DOCX e in PDF [formati online stampabili](#);
- Un [sito Web](#) (dominio .edu), con essenzialmente lo stesso materiale del libro. Non è richiesto alcun accesso né registrazione.
- Software gratuito scaricabile in diversi formati, elencati nel [sito web](#) e a pagina 451.
- Assistenza e consulenza via e-mail (opzionalmente con dati allegati).
- Un [gruppo Facebook](#) e il [Matlab File Exchange](#) per annunci e discussioni pubbliche.

Sebbene il libro completo sia disponibile gratuitamente in formato DOCX e PDF, diversi lettori hanno ritenuto che fosse troppo lungo da stamparsi in proprio e hanno richiesto una versione prestampata, che ora viene [venduta tramite Amazon](#) (ISBN 9798794182446). I materiali, il software, l'aiuto e la consulenza on-line sono tutti gratuiti. Sono disponibili alternative software open-source, ovvero Octave, Python e OpenOffice/LibreOffice.

Metodologia

La mia politica è che il contatto con gli utenti ("clienti") sia avviato solo dai clienti e rigorosamente in forma scritta, in inglese, principalmente tramite e-mail o messaggi sul gruppo Facebook - non al telefono né con *Skype*. Le richieste di comunicazione vocale o diretta video in tempo reale vengono gentilmente declinate. Ciò viene fatto per consentire conversazioni estese tra fusi orari, per preservare le comunicazioni in forma scritta e per evitare problemi di lingua e per mie difficoltà uditive legate all'età (i lettori provengono da almeno 162 paesi diversi). Le comunicazioni scritte via e-mail consentono anche l'uso di app per la traduzione automatica come Google Translate. Inoltre, i clienti possono inviare esempi dei propri dati tramite gli allegati e-mail o tramite Google Drive.

Le informazioni sull'affiliazione del cliente e sulla natura del progetto non sono richieste e sono strettamente a discrezione del cliente. Le informazioni e i dati del cliente vengono mantenuti riservati. In molti casi, non so nulla dell'origine dei loro dati e devo trattarli come numeri astratti. Di solito non conosco l'età, il sesso, la razza, il paese d'origine, il livello di istruzione, l'esperienza o l'occupazione dei clienti a meno che non me lo dicano. Devo cercare indizi tra i loro scritti per valutare il loro livello di conoscenza ed esperienza ed evitare di insultarli da un lato o confonderli dall'altro. Sono tutti il benvenuto.

Ho tentato di ridurre al minimo l'uso di formattazione di fantasia ed effetti speciali sul mio sito web, per renderlo compatibile con i vecchi sistemi operativi e browser. Non è necessario alcun account o registrazione. Non permetto la pubblicità sulle mie pagine web. Riduco al minimo l'uso di video, ma uso semplici animazioni GIF dove risulta utile, in quanto possono essere visualizzate direttamente sulla pagina Web o all'interno della versione ".docx" in [Microsoft Word 365](#), senza scaricare alcun componente o software aggiuntivo. Provo la mia formattazione per assicurarmi che sia visualizzabile su dispositivi mobili (tablet, smartphone).

L'influenza di Internet

Ci sono molti paesi, stati, università, dipartimenti, specialità e riviste differenti, ma un solo Internet globale. La maggior parte, ma non tutte, sono accessibili a chiunque disponga di una connessione Internet e di un computer, tablet o smartphone. Google (o qualsiasi motore di ricerca) guarda (quasi) l'intera Internet, indipendentemente dalla specializzazione accademica, offrendo la possibilità che la necessità di una soluzione sorta in un angolo della mondo accademico venga scoperta da un'esigenza da un'altra parte. Perché, ad esempio, un neuro-scientista, un ricercatore sul cancro, un economista, un linguista o uno studioso di musica, se è per questo, dovrebbero sapere qualcosa del mio lavoro? Sicuramente *non* ne saprebbero nulla, se pubblicassi solo sulle riviste scientifiche della mia specialità, plausibilmente non le leggono. Ma in realtà, tutti quei tipi di ricercatori, e centinaia di altri provenienti da altri campi diversi, hanno trovato il mio lavoro "inciampandoci" effettuando *query sul motore di ricerca*, piuttosto che leggendo pubblicazioni accademiche, e molti di essi lo hanno trovato abbastanza utile da *citarlo nelle loro pubblicazioni*. Nella mia carriera accademica, ho pubblicato ricerche solo su riviste di chimica analitica, che vengono lette principalmente da altri chimici analitici. Al contrario, i miei successi sul Web, le mie e-mail e le citazioni dei miei lavori provengono da una gamma molto più ampia di scienziati, ingegneri, ricercatori, istruttori e studenti che lavorano in università, industria, settori ambientali, medicina, ingegneria, scienze della terra, spazio, militare, finanziaria, agricoltura, comunicazioni e persino lingua e musicologia.

Lo Stile

Volevo che il mio scritto fosse istruttivo, non particolarmente accademico o rigoroso. È sfacciata-mente *pragmatico*, ovvero "Relativamente a questioni di fatto o affari pratici, spesso con l'esclusione di questioni intellettuali o artistiche; pratico anziché idealistico". Per molte persone, la matemati-

ca troppo astratta può costituire un ostacolo alla comprensione. Faccio solo presupposti di base sulle conoscenze pregresse che vanno oltre il consueto livello di specializzazione in scienze universitari: una conoscenza minima di matematica e un livello di lettura di 11° grado (scuola superiore negli Stati Uniti), secondo diversi [indici di leggibilità automatizzati](#). Ho cercato di ridurre al minimo le forme discorsive che potrebbero confondere i traduttori (macchina e umani), e cerco anche di ridurre al minimo l'uso del passivo. Spesso spiego lo stesso concetto più di una volta in contesti diversi perché credo che possa aiutare a "fissare" meglio alcune idee. Una parte importante del mio processo di scrittura è il *feedback dagli utenti*, tramite e-mail, social media, termini dei motori di ricerca, domande, correzioni, ecc. Inoltre, rileggo regolarmente anche le sezioni più vecchie con "occhi nuovi", correggendo gli errori e apportando miglioramenti nel fraseggio. Le domande dei lettori e persino i termini di ricerca su Google possono anche suggerire aree in cui sono possibili miglioramenti.

Per rendere più facile l'accesso, metto a disposizione i miei scritti in più formati: Web (HTML semplice, con grafica e animazioni GIF mute automatiche e una ricerca specifica per il sito); DOCX (Microsoft Word modificabile), la cui ultima versione mostra le animazioni GIF in esecuzione direttamente sulla pagina; PDF (Portable Document Format) per la stampa e [versioni cartacee e Kindle](#), tramite il programma di Amazon [Kindle Direct Publishing](#). Tutti tranne la versione web hanno un sommario dettagliato. Tutti tranne la versione in cartacea e Kindle sono gratuiti.

Un libro cartaceo viene solitamente letto a partire dall'inizio: l'indice e l'introduzione. Ma l'accesso al sito web, soprattutto tramite motori di ricerca (Google, Bing, ecc.), non è correlato all'ordine delle pagine. Ciò è evidente nei dati relativi agli accessi alle pagine web: il sommario e l'introduzione *non* sono i più consultati; infatti, nella maggior parte del tempo non ci sono *affatto visite* al sommario o alle pagine di introduzione. Ciò può causare un problema con la sequenza degli argomenti, che viene parzialmente ridotto includendo, nelle versioni Web e PDF del libro, "hot link" al sommario e al relativo materiale precedente e successivo. (La versione cartacea ha una media di tre riferimenti interni per ogni pagina, più un sommario con oltre 200 voci). Inoltre, per facilitare la comunicazione, ho aggiunto un collegamento "mail-to" a ciascuna pagina nella versione Web che include il mio indirizzo email e *il titolo della pagina come oggetto* (così posso capire dalla riga dell'oggetto dell'email in quale pagina si trovavano quando hanno cliccato sul link della mail).

Criterio di selezione della piattaforma software

Per quanto riguarda le piattaforme software, ho scelto due tipi: gli spreadsheet (pag. 14) e linguaggi script Matlab (pag. 15), Octave (pag. 20) e Python (pag. 427). Tutti hanno il vantaggio di essere multi-piattaforma; funzionano su PC, Mac, Unix, anche su dispositivi mobili (tablet/iPad) e su dispositivi distribuibili in miniatura (ad esempio, Python su Raspberry Pi, pag. 335). Questi sono ambienti di sviluppo noti che hanno grandi comunità di utenti con più contributori ed entrambi sono ampiamente utilizzati nelle applicazioni scientifiche. Tutti hanno un grado di compatibilità con le versioni precedenti che ne consente l'interoperabilità. Octave e Python sono gratuiti; aziende, organizzazioni e campus universitari spesso dispongono di licenze per sito per Excel (o per Microsoft Office, che include Excel) e per Matlab. Queste piattaforme hanno anche il vantaggio di evitare algoritmi segreti, ovvero i loro algoritmi possono essere visualizzati in dettaglio da qualsiasi utente. Il codice è distribuito nei formati "open source" e "open document", che è un normale testo (come i file Matlab ".m" o di Python ".py") o in un formato che potrebbe essere aperto e ispezionato utilizzando anche software libero (ad esempio, gli spreadsheet .xls e .xlsx di Microsoft Excel si possono aprire con OpenOffice o LibreOffice). Per coloro che non possono permettersi software costosi, Python, OpenOffice Calc (pag. 14) e Octave (pag. 20) possono essere scaricati gratuitamente.

La maggior parte dei programmi Matlab/Octave nel mio toolkit di elaborazione del segnale sono "funzioni", che sono essenzialmente pezzi di codice modulari che possono essere assemblati insieme dall'utente in modi diversi, un po' come i mattoncini *Lego*, piuttosto che programmi autonomi con elaborate interfacce grafiche utente, come i programmi commerciali. Le funzioni sviluppate dall'utente come la mia possono essere scaricate e utilizzate da sole, ma possono anche essere utilizzate proprio come le funzioni native fornite col linguaggio, come *componenti per costruire qualcosa di più grande*. Si possono scrivere le proprie funzioni o, se lo si desidera, si possono scaricare e usare le [funzioni scritte da altri](#). Quando si utilizzano le funzioni, si possono semplicemente ignorare i particolari del codice e utilizzare il ben definito input e output standard. Ciò è analogo all'assemblaggio di sistemi di intrattenimento personalizzati o di sicurezza domestica utilizzando componenti commerciali separati, collegati insieme tramite porte e cavi USB e HDMI o connessioni Bluetooth e Wi-Fi tra smartphone, tablet, computer, telecamere di sicurezza, campanelli, termostati e dispositivi elettronici. stampanti/auricolari/altoparlanti, ecc., *il tutto senza preoccuparsi del design interno di ciascun componente*. Utilizzo Matlab/Octave per le sue elevate prestazioni, la sua ampia popolarità e la sua somiglianza con altri linguaggi spesso utilizzati dagli scienziati, come Fortran, Basic e Pascal. Ho anche fornito molti esempi in Python, un altro linguaggio basato su funzioni molto capace e potente. In ogni caso ci sono altri linguaggi che hanno i loro paladini e potrebbero essere state valide alternative, come R, Mathematica, Julia e Scilab. Nell'interesse del tempo e della sanità mentale, mi sono limitato, per il momento, a Matlab/Octave e Python.

Ho cercato di trovare un equilibrio tra costo, velocità, facilità d'uso e curva di apprendimento e di rendere il mio software utilizzabile anche da coloro che non leggono tutta la documentazione, fornendo molti esempi e demo, comprese GIF animate che verranno riprodotte in posizione su qualsiasi browser Web o in Microsoft Word 365. Ogni script o funzione dispone di un help *incluso* nel software. Questi help integrati si possono visualizzare semplicemente digitando "help __" in Matlab/Octave, o "help(__)" in Python, dove __ è il nome dello script o della funzione. Questi file di 'help' contengono non solo istruzioni ma hanno spesso semplici *esempi d'uso* e in molti casi includono riferimenti ad altre funzioni simili. Matlab/Octave e Python (con l'aggiunta di Spyder desktop) hanno un editor per l'ispezione e la modifica del codice, con il rilevamento automatico degli errori. Anche questo non è necessario se l'azione esistente e gli input e gli output forniscono tutto ciò di cui si ha bisogno. (I modelli di fogli di calcolo e i loro esempi e demo hanno anche delle istruzioni integrate e la maggior parte di essi hanno "commenti di cella" a comparsa su determinate celle, contrassegnati da un punto rosso, che appaiono quando il puntatore del mouse viene posizionato su di essi, spiegando la funzione di quella cella).

Risultati

Nel 2016 il sito web ha ricevuto oltre 2 milioni di visualizzazioni e oltre 100.000 download dei miei programmi (attualmente alcune centinaia al mese), dal [sito web](#) dell'autore o dal [Matlab File Exchange](#). Ho ricevuto migliaia di e-mail con commenti, suggerimenti, correzioni, domande, offerte di traduzioni, ecc. I commenti dei lettori sono stati estremamente positivi, persino entusiasti, come indicato da questi estratti letterali di e-mail [riguardo al sito web](#) e al [mio software](#). In effetti, molti di questi commenti sono così entusiasti che ci si chiede: *perché, per un argomento così da nerd?* Dopo tutto, la maggior parte delle persone perde tempo a scrivere agli autori dei siti web. Un fattore è che il numero di utenti di Internet globale è così grande che anche argomenti altamente specializzati possono raccogliere un pubblico considerevole. Come si suol dire, "Un'ampia rete cattura anche i pesci più rari". Ma credo anche che parte del motivo della risposta entusiasta sia che la documentazione del software è spesso scritta male ed è difficile da capire, quindi è necessario uno sforzo maggiore per spiegare meglio il software e come funziona e dove non ci si può aspettare che funzioni. Cerco di essere reattivo, rispondendo a ogni e-mail e agendo in base ai loro suggerimenti e

correzioni. Anche la crescita nei social media è un fattore determinante; per un esempio specifico di questo, dal [Matlab File Exchange](https://blogs.mathworks.com/pick/2016/09/09/most-active-interactive-file-exchange-entry/), vedere <https://blogs.mathworks.com/pick/2016/09/09/most-active-interactive-file-exchange-entry/>.

Impatto

Commenti positivi e molti download sono piacevoli, ma non tutti quelli che scaricano qualcosa la provano nel loro lavoro, e non tutti quelli che la provano la trovano abbastanza preziosa da citarla nelle loro pubblicazioni. In modo molto gratificante, a dicembre 2023, *oltre 750 pubblicazioni avevano citato il sito Web e i programmi*, sulla base di ricerche di *Google Scholar*, che trattavano una gamma straordinariamente ampia di argomenti nel settore, ambiente, medicina, ingegneria, scienze della terra, spazio, militare, finanza, agricoltura, comunicazioni e anche occasionalmente lingua e musicologia. (Queste citazioni sono elencate a partire da pagina 480 nella versione PDF del libro e in <https://terpconnect.umd.edu/~toh/spectrum/citations.pdf>)

Il Futuro

Dove andrà a finire tutto questo nel lungo periodo? Per quanto riguarda il mio progetto di pensionamento, prima o poi passerò ad altri progetti, o passerò oltre, e il mio lavoro svanirà, per unirsi ai terabyte di materiale dimenticato su Internet che, a meno che non venga specificamente cancellato, presumibilmente resterà in giro per sempre, come libri ammuffiti negli scaffali di biblioteche abbandonate. Ma le tecniche di cui ho scritto potrebbero entrare a far parte della formazione di tutti gli studenti di scienze, oppure potranno essere sostituite da metodi più sofisticati, oppure l'elaborazione dei dati scientifici potrà essere affidata alle intelligenze artificiali (AI). Per lo meno, parte della programmazione applicativa sarà probabilmente sostituita o aiutata dall'intelligenza artificiale, come descritto a pagina 434.

Riferimenti

1. Douglas A. Skoog, *Principles of Instrumental Analysis*, 3^a Edizione, Saunders, Philadelphia, 1984. Pagine 73-76.
2. Gary D. Christian and James E. O'Reilly, *Instrumental Analysis*, Second Edition, Allyn and Bacon, Boston, 1986. Pagine 846-851.
3. Howard V. Malmstadt, Christie G. Enke, and Gary Horlick, *Electronic Measurements for Scientists*, W. A. Benjamin, Menlo Park, 1974. Pagine 816-870.
4. Stephen C. Gates and Jordan Becker, *Laboratory Automation using the IBM PC*, Prentice Hall, Englewood Cliffs, NJ, 1989.
5. Muhammad A. Sharaf, Deborah L Illman, and Bruce R. Kowalski, *Chemometrics*, John Wiley and Sons, New York, 1986.
6. Peter Wentzell and Christopher Brown, Signal Processing in Analytical Chemistry, in *Encyclopedia of Analytical Chemistry*, R.A. Meyers (Ed.), p. 9764–9800, John Wiley & Sons, Chichester, 2000 (<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.2407&rep=rep1&type=pdf>)
7. Constantinos E. Efstrathiou, Educational Applets in Analytical Chemistry, Signal Processing, and Chemometrics. (http://www.chem.uoa.gr/Applets/Applet_Index2.htm)
8. A. Felinger, Data Analysis and Signal Processing in Chromatography, Elsevier Science (19 May 1998).
9. Matthias Otto, *Chemometrics: Statistics and Computer Application in Analytical Chemistry*, Wiley-VCH (March 19, 1999). Some parts viewable in [Google Books](#).

10. Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. (Downloadable chapter by chapter in PDF format from <http://www.dspguide.com/pdfbook.htm>). This is a much more general treatment of the topic.
11. Robert de Levie, *How to use Excel in Analytical Chemistry and in General Scientific Data Analysis*, Cambridge University Press; 1 edition (February 15, 2001), ISBN-10:0521644844. [PDF excerpt](#).
12. Scott Van Bramer, Statistics for Analytical Chemistry, <http://science.widener.edu/svb/stats/stats.html>.
13. Taechul Lee, [Numerical Analysis for Chemical Engineers](#).
14. Educational Matlab GUIs, Georgia Institute of Technology. (<http://spfirst.gatech.edu/matlab/>)
15. Jan Allebach, Charles Bouman, and Michael Zoltowski, Digital Signal Processing Demonstrations in Matlab, Purdue University (<http://www.ecn.purdue.edu/VISE/ee438/demos/Demos.html>)
16. Chao Yang, Zengyou He and Weichuan Yu, Comparison of public peak detection algorithms for MALDI mass spectrometry data analysis, <http://www.biomedcentral.com/1471-2105/10/4>
17. Michalis Vlachos, [A practical Time-Series Tutorial with MATLAB](#).
18. Laurent Duval, Leonardo T. Duarte, Christian Jutten, [An Overview of Signal Processing Issues in Chemical Sensing](#).
19. Nicholas Laude, Christopher Atcherley, and Michael Heien, *Rethinking Data Collection and Signal Processing. 1. Real-Time Oversampling Filter for Chemical Measurements*, <https://pubs.acs.org/doi/abs/10.1021/ac302169y>
20. P. E. S. Wormer, Matlab for Chemists, http://www.math.ru.nl/dictaten/Matlab/matlab_diktaat.pdf
21. Martin van Exter, Noise and Signal Processing, <http://molphys.leidenuniv.nl/~exter/SVR/noise.pdf>
22. Scott Sinex, Developer's Guide to Excelets, <http://academic.pgcc.edu/~ssinex/excelets/>
23. R. de Levie, Advanced Excel for scientific data analysis, Oxford University Press, New York (2004)
24. S. K. Mitra, Digital Signal Processing, a computer-based approach, 4th ed, McGraw-Hill, New York, 2011.
25. "Calibration in Continuum-Source AA by Curve Fitting the Transmission Profile", T. C. O'Haver and J. Kindervater, *J. of Analytical Atomic Spectroscopy* 1, 89 (**1986**)
26. "Estimation of Atomic Absorption Line Widths in Air-Acetylene Flames by Transmission Profile Modeling", T. C. O'Haver and Jing-Chyi Chang, *Spectrochim. Acta* 44B, 795-809 (**1989**)
27. "Effect of the Source/Absorber Width Ratio on the Signal-to-Noise Ratio of Dispersive Absorption Spectrometry", T. C. O'Haver, *Anal. Chem.* 68, 164-169 (**1991**).
28. "Derivative Luminescence Spectrometry", G. L. Green and T. C. O'Haver, *Anal. Chem.* 46, 2191 (**1974**).
29. "Derivative Spectroscopy", T. C. O'Haver and G. L. Green, *American Laboratory* 7, 15 (**1975**).
30. "Numerical Error Analysis of Derivative Spectroscopy for the Quantitative Analysis of Mixtures", T. C. O'Haver and G. L. Green, *Anal. Chem.* 48, 312 (**1976**).
31. "Derivative Spectroscopy: Theoretical Aspects", T. C. O'Haver, *Anal. Proc.* 19, 22-28 (**1982**).
32. "Derivative and Wavelength Modulation Spectrometry," T. C. O'Haver, *Anal. Chem.* 51, 91A (**1979**).
33. "A Microprocessor-based Signal Processing Module for Analytical Instrumentation", T. C. O'Haver and A. Smith, *American Lab.* 13, 43 (**1981**).
34. "Introduction to Signal Processing in Analytical Chemistry", T. C. O'Haver, *J. Chem. Educ.* 68 (**1991**)
35. "Applications of Computers and Computer Software in Teaching Analytical Chemistry", T. C. O'Haver, *Anal. Chem.* 68, 521A (**1991**).
36. "The Object is Productivity", T. C. O'Haver, *Intelligent Instruments and Computers* Mar-Apr, **1992**, p 67-70.
37. Analysis software for spectroscopy and mass spectrometry, Spectrum Square Associates (<http://www.spectrumsquare.com/>).
38. *Fityk*, a program for data processing and nonlinear curve fitting. (<http://fityk.nieto.pl/>)

39. Peak fitting in *Origin* (<http://www.originlab.com/index.aspx?go=Products/Origin/DataAnalysis/PeakAnalysis/PeakFitting>)
40. *IGOR Pro 6*, software for signal processing and peak fitting (<http://www.wavemetrics.com/index.html>)
41. [*PeakFIT*, automated peak separation analysis](#), Systat Software Inc.
42. *OpenChrom*, open-source software for chromatography and mass spectrometry. (<http://www.openchrom.net/main/content/index.php>)
43. W. M. Briggs, *Do not smooth times series, you hockey puck!*, <http://wmbriggs.com/blog/?p=195>
44. Nate Silver, *The Signal and the Noise: Why So Many Predictions Fail—but Some Do Not*, Penguin Press, 2012. ISBN 159420411X. A much broader look at "signal" and "noise", aimed at a general audience, but still worth reading.
45. David C. Stone, Dept. of Chemistry, U. of Toronto, [Stats Tutorial - Instrumental Analysis and Calibration](#).
46. Streamlining Digital Signal Processing: A Tricks of the Trade Guidebook, Richard G. Lyons, John Wiley & Sons, 2012.
47. Atomic spectra lines database. <http://physics.nist.gov/PhysRefData/ASD/> and <http://www.astm.org/Standards/C1301.htm>
48. Curve fitting to get overlapping peak areas (<http://matlab.cheme.cmu.edu/2012/06/22/curve-fitting-to-get-overlapping-peak-areas>)
49. Tony Owen, [Fundamentals of Modern UV-Visible Spectroscopy](#), Agilent Corp, 2000.
50. Nicole K. Keppy, Michael Allen, Understanding Spectral Bandwidth and Resolution in the Regulated Laboratory, Thermo Fisher Scientific Technical Note: 51721.
http://www.analiticaweb.com.br/newsletter/02/AN51721_UV.pdf
51. Martha K. Smith, "Common mistakes in using statistics", <http://www.ma.utexas.edu/users/mks/statmistakes/TOC.html>
52. Jan Verschelde, "Signal Processing in MATLAB", <http://homepages.math.uic.edu/~jan/mcs320s07/matlec7.pdf>
53. H. Mark and J. Workman Jr, "Derivatives in Spectroscopy", Spectroscopy 18 (12). p.106.
54. Jake Blanchard, Comparing Matlab to Excel/VBA,
https://blanchard.ep.wisc.edu/PublicMatlab/Excel/Matlab_VBA.pdf
55. Ivan Selesnick, "Least-squares with Examples in Signal Processing", http://eeweb.poly.edu/iselesni/lecture_notes/least_squares/
56. Tom O'Haver, "Is there Productive Life after Retirement?", *Faculty Voice*, University of Maryland, April 24, 2014. DOI: 10.13140/2.1.1401.6005; URL: <https://terpconnect.umd.edu/~toh/spectrum/Retirement.pdf>
57. <http://www.dsprelated.com/>, the most popular independent internet resource for Digital Signal Processing (DSP) engineers around the world.
58. John Denker, "Uncertainty as Applied to Measurements and Calculations", <http://www.av8n.com/physics/uncertainty.htm>. Excellent.
59. T. C. O'Haver, Teaching and Learning Chemometrics with Matlab, *Chemometrics and Intelligent Laboratory Systems* 6, 95-103 (1989).
60. Allen B. Downey, "Think DSP", Green Tree Press, 2014. ([153-page PDF download](#)). Python code instruction using sound as a basis.
61. Purnendu K. Dasgupta, et. al, "Black Box Linearization for Greater Linear Dynamic Range: The Effect of Power Transforms on the Representation of Data", *Anal. Chem.* 2010, 82, 10143–10150.
62. Joseph Dubrovkin, Mathematical Processing of Spectral Data in Analytical Chemistry: A Guide to Error Analysis, Cambridge Scholars Publishing, 2018 and 2019, 379 pages. ISBN 978-1-5275-1152-1. [Link](#).
63. Power Law Approach as a Convenient Protocol for Improving Peak Shapes and Recovering Areas from Partially Resolved Peaks, M. Farooq Wahab, et. al., *Chromatographia* (2018).

<https://doi.org/10.1007/s10337-018-3607-0>.

64. T. C. O'Haver, *Interactive Computer Models for Analytical Chemistry Instruction*, <https://terpconnect.umd.edu/~toh/models/>, **1995**.
65. T. C. O'Haver, *Interactive Simulations of Basic Electronic and Operational Amplifier Circuits*, <https://terpconnect.umd.edu/~toh/ElectroSim>, **(1996)**
66. Signal Processing at Rice University. (<http://dsp.rice.edu/software/>)
67. Steven Pinker, The Sense of Style: The Thinking Person's Guide to Writing in the 21st Century, New York, NY: Penguin, **2004**.
68. Joseph Dubrovkin, <https://www.researchgate.net/profile/Joseph-Dubrovkin>
69. Separations at the Speed of Sensors, D. C. Patel, M. Farooq Wahab, T. C. O'Haver, and Daniel W. Armstrong, *Analytical Chemistry* **2018** 90 (5), 3349-3356, DOI: 10.1021/acs.analchem.7b04944
70. MF Wahab, TC O'Haver, F. Gritti, G. Hellinghausen, and DW Armstrong, "Increasing chromatographic resolution of analytical signals using derivative enhancement approach," *Talanta*, vol. 192, pp. 492–499, **2019**
71. Yuri Kalambet, "Reconstruction of exponentially modified functions", **2019**. DOI: 10.13140/RG.2.2.12482.84160. [Link](#).
72. Yuri Kalambet, Yuri Kozmin, Andrey Samokhin, "Comparison of integration rules in the case of very narrow chromatographic peaks", *Chemometrics and Intelligent Laboratory Systems* 179, May **2018**. DOI: 10.1016/j.chemolab.2018.06.001
73. Yuri Kalambet, et. al., "Reconstruction of chromatographic peaks using the exponentially modified Gaussian function", *Journal of Chemometrics* June **2011**, 25(7):352 - 356. DOI: 10.1002/cem.1343
74. Allen, L. C., Gladney, H. M., Glarum, S. H., J. Chem. Phys. 40, 3135 (**1964**)
75. J. W. Ashley, Charles N. Reilley, "De-Tailing and Sharpening of Response Peaks in Gas Chromatography", *Anal. Chem.*, 37, 6, 626-630, **1965**.
76. M. Johansson, M. Berglund and D. C. Baxter, "Improving accuracy in the quantitation of overlapping, asymmetric, chromatographic peaks by deconvolution: theory and application to coupled gas chromatography atomic absorption spectrometry", *Spectrochimica Acta*, Vol 48B, p. 1393-1409, **1993**.
77. S. Sterlinski, "A Method for Resolution Enhancement of Interfering Peaks in Ge(Li) Gamma-Ray Spectra", *J. of Radioanalytical Chemistry*, 31, 195-226, **1976**.
78. "Importance of academic blogs", Teachers Insurance and Annuity Association of America-College Retirement Equities Fund, New York, NY. <https://careerpurpose.com/industries/education/academic-blogs>.
79. Robi Polikar, The Wavelet Tutorial, <http://web.iitd.ac.in/~sumeet/WaveletTutorial.pdf>
80. C. Valens, "A Really Friendly Guide to Wavelets", <http://agl.cs.unm.edu/~williams/cs530/arfgtw.pdf>
81. Brani Vidakovic and Peter Mueller, "Wavelets for Kids", <http://www.gwavelet.bme.gatech.edu/wp/kidsA.pdf>
82. Amara Graps, "An Introduction to Wavelets" <https://www.eecis.udel.edu/~amer/CISC651/IEEEwavelet.pdf>
83. Muhammad Ryan, "What is Wavelet and How We Use It for Data Science", <https://towardsdatascience.com/what-is-wavelet-and-how-we-use-it-for-data-science-d19427699cef>
84. Michael X. Cohen, "A better way to define and describe Morlet wavelets for time-frequency analysis", *NeuroImage*, Volume 199, 1 October **2019**, Pages 81-86.
85. Wahab M. F, O'Haver T. C., "Wavelet transforms in separation science for denoising and peak overlap detection." *J Sep Sci.* 43 (9-10) 1615–2012 (**2020**). ISSN 1615-9306; <https://doi.org/10.1002/jssc.202000013>
86. G. K. Wertheim, *J. of Electron Spectroscopy and Related Phenomena*, 6 (**1975**) 239-251.

87. R. E. Sturgeon, et. al., Atomization in graphite-furnace atomic absorption spectrometry. Peak-height method vs. integration method of measuring absorbance. *Anal. Chem.* 47, 8, 1240–1249 (1975) <https://doi.org/10.1021/ac60358a039>
88. Sunaina et al, “Calculating numerical derivatives using Fourier transform: some pitfalls and how to avoid them”, *Eur. J. Phys.* 39, 065806, **2018**
89. Sinex, Scott A, Investigating types of errors. *Spreadsheets in Education* 2.1 (**2005**): 115-124.
90. Catherine Perrin, Beata Walczak, and Désiré Luc Massart, “Quantitative Determination of the Components in Overlapping Chromatographic Peaks Using Wavelet Transform”, *Analytical Chemistry* **2001** 73 (20), 4903-4917 DOI: 10.1021/ac010416a
91. F. Gritti, S. Besner, S. Cormier, M. Gilar, Applications of high-resolution recycling liquid chromatography: from small to large molecules, *Journal of Chromatography A* 1524 (**2017**) 108-120.
92. Desimoni E. and Brunetti B., "About Estimating the Limit of Detection by the Signal to Noise Approach", *Pharmaceutica Analytica Acta* 67, 4, **2015**. DOI: 10.4172/2153-2435.100035. [PDF link](#).
93. Royal Society of Chemistry Analytical Methods Committee, “Recommendations for the Definition, Estimation and Use of the Detection Limit”, *Analyst*, Feb. **1987**, vol.112, p. 199.
94. “MATLAB vs Python: Why and How to Make the Switch”, <https://realpython.com/matlab-vs-python/>
95. [MLAB, an advanced mathematical and statistical modeling system](#), by Gary Knott.
96. NIST Engineering Statistics Handbook: <https://www.itl.nist.gov/div898/handbook/index.htm>
97. “Why and How Savitzky–Golay Filters Should Be Replaced”, Michael Schmid, David Rath, and Ulrike Diebold, *ACS Measurement Science Au* **2022** 2 (2), 185-196. DOI: 10.1021/acsmeasurescäu.1c00054
98. Farooq Wahab and Thomas C. O'Haver, “Peak deconvolution with significant noise suppression and stability using a facile numerical approach in Fourier space”, *Chemometrics and Intelligent Laboratory Systems* 235, **2023**. <https://authors.elsevier.com/c/1gVwgcc6MExCW>
99. M.F. Wahab, F. Gritti, T.C. O'Haver, Discrete Fourier transform techniques for noise reduction and digital enhancement of analytical signals, *TrAC, Trends Anal. Chem.*, 143, Article 116354 (**2021**)
100. Aditi Gupta, *et. al.*, [A-TSPD: autonomous-two stage algorithm for robust peak detection in online time series | Cluster Computing \(springer.com\)](#)

Pubblicazioni che citano questo libro, i programmi e/o la documentazione

Updated annually. MLA format. Last updated December 2023.

Se è stato pubblicato un articolo utilizzando questi programmi che si vorrebbe includere in questa lista, si pregha di inviare il documento, o una citazione, a Tom O'Haver a toh@umd.edu

1. Poppi, R. J., Vazquez, P. A., & Pasquini, C. (**1992**). Fast scanning Hadamard spectrophotometer. *Applied Spectroscopy*, 46(12), 1822-1827.
2. Ghatee, M. H., and A. Boushehri. "Modulation of the integrated rate equation of a composite system for the kinetic parameters." *Chemometrics and intelligent laboratory systems* 25.1 (**1994**): 43-49.
3. C.W.K. Chow, D.E. Davey, Dennis Mulcahy, T.C.W. Yeow, Signal enhancement of potentiometric stripping analysis using digital signal processing, *Analytica Chimica Acta* 307(1):15-26, April **1995** DOI: 10.1016/0003-2670(95)00023-S
4. Ringe, Steven A. "Hydrogen-extended defect interactions in heteroepitaxial InP materials and devices." *Solid-State Electronics* 41.3 (**1997**): 359-380.
5. Chow, Christopher WK, David Edward Davey, and D. E. Mulcahy. "Signal filtering of potentiometric stripping analysis using Fourier techniques." *Analytica chimica acta* 338.3 (**1997**): 167-178.

6. Leung, Alexander Kai-man, Foo-Tim Chau, and Jun-bin Gao. "Wavelet transform: a method for derivative calculation in analytical chemistry." *Analytical Chemistry* 70.24 (1998): 5222-5229.
7. Harris, D. C. (1998). "Spektralphotometer". In *Lehrbuch der Quantitativen Analyse* (pp. 695-746). Vieweg+ Teubner Verlag. Link.
8. Keyhani, Ali, Wenzhe Lu, and Gerald T. Heydt. "Neural network based composite load models for power system stability analysis." IEEE International Conference on Computational Intelligence for Measurement Systems and Applications. 2005.
9. Fernández, Mario, and J. Ricardo Pérez-Correa. "Instrumentation for Monitoring SSF Bioreactors." Solid-State Fermentation Bioreactors. Springer Berlin Heidelberg, 2006. 363-374
10. Richard Graham, Ring Laser Gain Media, Thesis, http://ir.canterbury.ac.nz/bitstream/10092/1377/1/thesis_fulltext.pdf (2006)
11. Sheaff, Chrystal N., Delyle Eastwood, and Chien M. Wai. "Increasing selectivity for TNT-based explosive detection by synchronous luminescence and derivative spectroscopy with quantum yields of selected aromatic amines." *Applied spectroscopy* 61.1 (2007): 68-73.
12. Hovick, James W., Michael Murphy, and J. C. Poler. "" Audibilization" in the chemistry laboratory: An introduction to correlation techniques for data extraction." *J. Chem. Educ* 84.8 (2007): 1331.
13. de Aragão, Bernardo José Guilherme, and Younes Messaddeq. "PEAK SEPARATION IN SPECTRAL ANALYSIS." (2007). Link.
14. Ingersoll, Justin Edward. A Regularization Technique for the Analysis of Photographic Data Used in Chemical Release Wind Measurements. ProQuest, 2008.
15. Dinesh, S. "The Effect of Smoothing on the Extraction of Drainage Networks from Simulated Digital Elevation Models." *Journal of Applied Sciences Research* 4.11 (2008): 1356-1360.
16. Jed A. Meltzer, Hitten P. Zaveri, Irina I. Goncharova, Marcello M. Distasio, Xenophon Papademetris, Susan S. Spencer, Dennis D. Spencer and R. Todd Constable, "Effects of Working Memory Load on Oscillatory Power in Human Intracranial EEG", *Cereb. Cortex* (2008) 18 (8): 1843-1855. doi: 10.1093/cercor/bhm213
17. Sheaff, Chrystal N., et al. "Fluorescence detection and identification of tagging agents and impurities found in explosives." *Applied spectroscopy* 62.7 (2008): 739-746.
18. "A regularization technique for the analysis of photographic data used in chemical release wind measurements", JE Ingersoll, 2008, books.google.com
19. "An application of detection function for the eye blinking detection", Pander, T. Przybyla, T.; Czabanski, Human System Interactions 2008 Conference: 25-27 May 2008, Page(s): 287- 291
20. "Isotopically labeled oxygen studies of the NO_x exchange behavior of La₂CuO₄ to determine potentiometric sensor response mechanism" F.M. Van Assche IV, E.D. Wachsman, *Solid State Ionics*, Volume 179, Issue 39, 15 December 2008, Pages 2225–2233
21. "High-speed laryngoscopic evaluation of the effect of laryngeal parameter variation on aryepiglottic trilling." Moisik, Scott R., John H. Esling, and Lise CrevierBuchman. poster, http://www.ncl.ac.uk/linguistics/assets/documents/MoisikEslingBuchman_NewcastlePharyngealsPoster_2009.pdf (2009).
22. Tricas, Marazico, and Juan Ignacio. "Auto configuration dans LTE: procédés de mesure de l'occupation du canal radio pour une utilisation optimisée du spectre.", "Auto configuration in LTE: measuring the occupancy of the radio channel for optimized use of the spectrum" (2009). PDF link.
23. "Early age concrete strength monitoring with piezoelectric transducers by the harmonic frequencies method", Thomas J. Kelleher, 2009. <http://www.engineering.swarthmore.edu/e90/2008/reports/Thomas%20Kelleher.pdf>
24. "Information management for high content live cell imaging", Daniel Jameson, David A Turner, John Ankers, Stephnie Kennedy, Sheila Ryan, Neil Swainston, Tony Griffiths, David G Spiller, Stephen G Oliver, Michael RH White, Douglas B Kell and Norman W Paton, *BMC Bioinformatics* 2009, 10:226 doi:10.1186/1471-2105-10-2263
25. "Human-Computer Systems Interaction: Backgrounds and Applications", edited by Zdzislaw S. Hippe,

26. "Multiplexed DNA detection using spectrally encoded porous SiO₂ photonic crystal particles", SO Meade, MY Chen, MJ Sailor, *Anal. Chem.*, **2009**, 81 (7), pp 2618–2625. DOI: 10.1021/ac802538x
27. "Prolonged stimulus exposure reveals prolonged neurobehavioral response patterns, Brett A. Johnson, Cynthia C. Woo, Yu Zeng, Zhe Xu, Edna E. Hingco, Joan Ong, Michael Leon. *The Journal of Comparative Neurology*, Volume 518, Issue 10, pages 1617–1629, 15 May **2010**
- 28."Alternative Measures of Phonation: Collision Threshold Pressure and Electroglossographic Spectral Tilt: Extra: Perception of Swedish Accents." Enflo, Laura. (**2010**). Full Text.
29. Botcharova, Maria. "Changes in structure of EEG-EMG coherence during brain development: analysis of experimental data and modeling of putative mechanisms." (**2010**) [PDF] from ucl.ac.uk
- 30."Vowel Dependence for Electroglossography and Audio Spectral Tilt", L Enflo, Proceedings of Fonetik, **2010**.
31. "Rapid and accurate detection of plant miRNAs by liquid northern hybridization.", Wang, Xiaosu, Yongao Tong, and Shenghua Wang. International journal of molecular sciences 11.9 (**2010**): 3138-3148.
32. Nusz, G. J. (**2010**). Label-free biodetection with individual plasmonic nanoparticles (Doctoral dissertation, Duke University).
33. Khudaish, Emad A., and Aysha A. Al Farsi. "Electrochemical oxidation of dopamine and ascorbic acid at a palladium electrode modified with in situ fabricated iodine-adlayer in alkaline solution." *Talanta* 80.5 (**2010**): 1919-1925.
34. "Advances in Music Information Retrieval", edited by Zbigniew W. Ras, Alicja Wieczorkowska, Springer, **2010**, page 135.
35. Bilal, M., Sharif, M., Jaffar, M. A., Hussain, A., & Mirza, A. M. (2010, May). Image restoration using modified hopfield fuzzy regularization method. In Future Information Technology (FutureTech), **2010** 5th International Conference on (pp. 1-6). IEEE.
36. Rim, Jung Ho. "Preparation and Characterization of Sources for Ultra-high Resolution Microcalorimeter Alpha Spectrometry." The Pennsylvania State University (**2010**). PDF link.
37. Xiaosu Wang, Yongao Tong and Shenghua Wang, Rapid and Accurate Detection of Plant miRNAs by Liquid Northern Hybridization, *Int. J. Mol. Sci.* **2010**, 11(9), 3138-3148; doi:10.3390/ijms11093138
38. "Radio Frequency Fuel Gauging with Neuro-Fuzzy Inference Engine For Future Spacecrafts". Kumagai, A., Liu, T. I., & Sul, D. In Proceedings of the 10th IASTED, International Conference, **2010** (Vol. 674, No. 020, p. 243).
39. "Automatic Seizure Detection in ECoG by Differential Operator and Windowed Variance," Majumdar, K.K.; Vardhan, P., Neural Systems and Rehabilitation Engineering, IEEE Transactions on, vol.19, no.4, pp.356,365, Aug. **2011**
40. "Genetic algorithm with peaks adaptive objective function used to fit the EPR powder spectrum", Sebastian Grzegorz Żurek, Applied Soft Computing, Volume 11, Issue 1, January **2011**, Pages 1000–1007
41. "Determination of sea conditions for wave energy conversion by spectral analysis", B Yagci, P Wegener, EEE Transactions on Power Delivery, 18(2): 372–376, **2011**.
42. "Push-broom hyperspectral imaging for elemental mapping with glow discharge optical emission spectrometry", G Gamez, D Frey, J Michler - *J. Anal. At. Spectrom.*, **2011**, 65, 85–98
- 43."Dual-order snapshot spectral imaging of plasmonic nanoparticles", Gregory J. Nusz, Stella M. Marinakos, Srinath Rangarajan, and Ashutosh Chilkoti, *Applied Optics*, Vol. 50, Issue 21, pp. 4198-4206 (**2011**) <http://dx.doi.org/10.1364/AO.50.004198>
44. Sugandharaju, Ravi Kumar Chatnaballi. "Gaussian Deconvolution and MapReduce Approach for Chipseq Analysis". Dissertation. University of Cincinnati, **2011**.
45. "Parallel Deconvolution Algorithm in Perfusion Imaging" F Zhu, DR Gonzalez, T Carpenter, Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference, 26-29 July **2011**
46. "Field observations of infragravity waves and their behaviour on rock shore platforms" Edward P.

Beetham, Paul S. Kench, Earth Surface Processes and Landforms, Volume 36, Issue 14, pages 1872–1888, November **2011**

47. "Majority Voting: Material Classification by Tactile Sensing Using Surface Texture", Jamali, N., Sammut, C., IEEE Transactions on Robotics, Volume: 27, Issue: 3, Page(s): 508 - 521, June **2011**
48. Yuan, Yuan, Yishan Luo, and Albert Chung. "VE-LLI-VO: Vessel enhancement using local line integrals and variational optimization." IEEE Transactions on Image Processing 20.7 (**2011**): 1912-1924.
49. "Demand Estimation with Automated Meter Reading in a Distribution Network", Aksela, K. and Aksela, M., J. Water Resour. Plann. Manage., 137(5), 456–467 (**2011**). doi: 10.1061/(ASCE) WR.1943-5452.0000131
50. Ochoa, Jeimy Catherine Millán. Design and Development of a Localization System for a Sensor Network in Collective Symbiotic Organisms. Diss. Universitätsbibliothek der Universität Stuttgart, **2011**.
51. "Genetic algorithm with peaks adaptive objective function used to fit the EPR powder spectrum", Sebastian Grzegorz Żurek, Journal Applied Soft Computing archive. Volume 11, Issue 1, January **2011**, pages 1000-1007
52. Hornung, J. P. (**2011**). Exploring the potential for using deep-sea bamboo corals (*Isidella* sp.) for paleoceanographic reconstructions (Doctoral dissertation).
53. Boll, Marie-Theres. Ein neues Konzept zur automatisierten Bewertung von Fertigkeiten in der minimal invasiven Chirurgie für Virtual-Reality-Simulatoren in GridUmgebungen. Vol. 38. KIT Scientific Publishing, **2011**. Link.
54. "Development of ECG signal interpretation software on Android 2.2, Hermawan, K.; Iskandar, A.A.; Hartono, R.N., "Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2011 2nd International Conference, vol., no., pp.259,264, 8-9 Nov. **2011** doi: 10.1109/ICICI-BME.2011.6108621
55. Choi, Sheng Heng. "Signal processing and amplifier design for inexpensive genetic analysis instruments." (**2011**). <https://era.library.ualberta.ca/files/qr46r139p#.WifTkEqnGUk>
56. Hoffman, Galen Brandt. Direct Write of Chalcogenide Glass Integrated Optics Using Electron Beams. Diss. The Ohio State University, **2011**. Direct link.
57. Bai, Er-Wei, et al. "Detection of radionuclides from weak and poorly resolved spectra using Lasso and subsampling techniques." Radiation Measurements 46.10 (**2011**): 1138-1146.
58. Sugandharaju, Ravi Kumar Chatnahalli. Gaussian Deconvolution and MapReduce Approach for Chipseq Analysis. Diss. University of Cincinnati, **2011**.
59. "Automated peak alignment for nucleic acid capillary electrophoresis data by dynamic programming". Fethullah Karabiber, Kevin Weeks, and Oleg V. Favorov. In Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine (BCB '11). ACM, New York, NY, USA, **2011**. pages 544-546. DOI=10.1145/2147805.2147895 <http://doi.acm.org/10.1145/2147805.2147895>
60. Shin, Sung-Hwan, et al. "Mass estimation of impacting objects against a structure using an artificial neural network without consideration of background noise." Nuclear Engineering and Technology 43.4 (**2011**): 343-354.
61. Taibo, María Luisa Gómez, et al. "Matching needs and capabilities with assistive technology in an amyotrophic lateral sclerosis patient." Accessibility, Inclusion and Rehabilitation using Information Technologies (**2011**): 21.
62. Paul, Ruma R., Victor C. Valgenti, and Min Sik Kim. "Real-time Netshuffle: Graph distortion for on-line anonymization." Network Protocols (ICNP), 2011 19th IEEE International Conference on. IEEE, **2011**.
63. Lopez-Castellanos, V. (**2011**). Ultrawideband time domain radar for time reversal applications (Doctoral dissertation, The Ohio State University).
64. "Electricity gain via integrated operation of turbine generator and cooling tower using local model network." Pan, Tian-Hong, et al. Energy Conversion, IEEE Transactions on 26.1 (**2011**): 245-255.
65. "Dynamic analysis of electronic devices' power signatures, Marcu, M.; Cernazanu, C., " Instrumentation and Measurement Technology Conference (I2MTC), **2012** IEEE International, vol., no., pp.117,122, 13-16 May **2012**. doi: 10.1109/I2MTC.2012.6229562
66. "Experimental comparison among pileup recovery algorithms for digital gamma ray spectroscopy" El-

67. Kwon, Soonil. "Voice-driven sound effect manipulation." International Journal of Human-Computer Interaction 28.6 (2012): 373-382.
68. "Distributed representation of chemical features and tunotopic organization of glomeruli in the mouse olfactory bulb" Limei Maa, Qiang Qiu, Stephen Gradwohl, Aaron Scotta, Elden Q. Yua, Richard Alexandra, Winfried Wiegaebe, and C. Ron Yu, Proceeding of the National Academy of Sciences, April 3, 2012 vol. 109, no. 14, pages 5481-5486.
69. Hofler, Alicia S. Optimization Framework for a Radio Frequency Gun Based\ Injector. Old Dominion University, PhD dissertation, 2012.
70. "A Robust Heart Sound Segmentation and Classification Algorithm using Wavelet Decomposition and Spectrogram." Deng, Yiqi, and Peter J. Bentley. 2012. Full text:
<http://www.peterjbentley.com/heartworkshop/challengepaper3.pdf>
71. "Detecting STR peaks in degraded DNA samples". Marasco, E., Ross, A., Dawson, J., Moroose, T., & Ambrose, T. Proc. of 4th International Conference on Bioinformatics and Computational Biology (BICoB), (Las Vegas, USA), March 2012. Full text:
http://www.cse.msu.edu/~rossarun/pubs/RossDNAEnhancement_BICoB2011.pdf
72. "Saccades detection in optokinetic nystagmus-a fuzzy approach." PANDER, Tomasz, et al. , Journal of Medical Informatics & Technologies 19 (2012): 33-39.
73. "Grain-size properties and organic-carbon stock of Yedoma Ice Complex permafrost from the Kolyma lowland, northeastern Siberia", J Strauss, L Schirrmeister, S Wetterich, Andreas Borchers, Sergei P. Davydov, Global Biogeochemical Cycles, Volume 12, 2012.
74. "An Early Prediction of Cardiac Risk using Augmentation Index Developed based on a Comparative Study." Manimegalai, P., Delpha Jacob, and K. Thanushkodi. , International Journal of Computer Applications 50 (2012). Abstract.
75. "Determinação Da Estabilidade Oxidativa De Biocombustíveis," Bruno A. F. Vitorino, Franz H. Neff, Elmar U. K. Melcher, Antonio M. N. Lima, Anais do XIX Congresso Brasileiro de Automática, CBA 2012.
<http://www.eletrica.ufpr.br/anais/cba/2012/Artigos/100018.pdf>
76. "Efficacy of Differential Operators in Brain Electrophysiological Signal Processing: A Case Study in Epilepsy." Majumdar, Kaushik, and Pratap Vardhan. 2012 Full text.
77. Snider, W. (2012). Electro-optically Tunable Microring Resonators for Non-Linear Frequency Modulated Waveform Generation (Doctoral dissertation, Texas A & M University).
78. "9.0 Experimental-Two-Dimensional GCxGC." Technologies towards the Development of a Lab-on-a-Chip GCxGC for Environmental Research (2012). Full text. A Thesis by Jaydene Halliday, BSc MRSC
79. "BaNa: A hybrid approach for noise resilient pitch detection," He Ba; Na Yang; Demirkol, I.; Heinzelman, W., Statistical Signal Processing Workshop (SSP), 2012 IEEE, vol., no., pp.369,372, 5-8 Aug. 2012. doi: 10.1109/SSP.2012.6319706
80. Tripathi, Ashish. THE NEW IMAGE PROCESSING ALGORITHM FOR\ QUALITATIVE AND QUANTITATIVE STM DATA ANALYSIS. Diss. 2012.
81. Skelton, Martin. "Diffusion of Innovation System Elements-A Novel Method to Study Technology Development and Its Application to Wind Power." (2012). [PDF] from chalmers.se
82. Pander, T., et al. "A new method of saccadic eye movement detection for optokinetic nystagmus analysis." Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE. IEEE, 2012.
83. Mahmoud, I. I., M. S. El_Tokhy, and H. A. Konber. "Pileup recovery algorithms for digital gamma ray spectroscopy." Journal of Instrumentation 7.09 (2012): P09013.
84. Zhu, Fan, et al. "Parallel perfusion imaging processing using GPGPU." Computer methods and programs Pagina | 497

in biomedicine 108.3 (2012): 1012-1021.

85. Cuss, C. W., and Celine Guéguen. "Determination of relative molecular weights of fluorescent components in dissolved organic matter using asymmetrical flow fieldflow fractionation and parallel factor analysis." *Analytica chimica acta* 733 (2012): 98- 102.
86. Olugboji, Oluwafemi A., and Jack M. Hale. "Development of Damage Reconstruction Techniques from Impulsive Events Based on Measurements Made Remotely." ASME 2012 International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers, 2012.
87. Dickson, B., and M. Craig. "Deconvolving gamma-ray logs by adaptive zone refinement." *Geophysics* 77.4 (2012): D159-D169.
88. "SmartBells: RFID-Enhanced System to Monitor Free Weight Exercises. "Chaudhri, Rohit, and Gaetano Borriello. 2012 Full text.
89. "Diffusion of Innovation System Elements-A Novel Method to Study Technology Development and Its Application to Wind Power." Skelton, Martin. (2012). Fulltext.
90. Grotenhuis, Michael Gary. "An Overview of the Maximum Entropy Method of Image Deconvolution." A University of Minnesota-Twin Cities "Plan B" Master's paper, 2012.
91. "On comet attitude determination of Rosetta lander Philae through nonlinear optimal system identification." Caputo, Gianluca. (2012). Full text.
92. Valadares¹, D. C., Vitorino, B. A., Neta, M. L. N., Batista, E. S., Santos, M. V., Neff, F. H., & Melcher, E. N. (2012). System for Analysis of the Biodiesel Quality.
93. Mukhopadhyay, C. K., et al. "Acoustic emission during fracture toughness tests of SA333 Gr. 6 steel." *Engineering Fracture Mechanics* 96 (2012): 294-306.
94. Huang, Zifang. "Knowledge-Assisted Sequential Pattern Analysis: An Application in Labor Contraction Prediction." (2012). PDF link.
95. van de Voort, Frederik R., and David Pinchuk. "System and Method for Determining Base Content of a Hydrophobic Fluid." U.S. Patent Application 13/171,566.
96. Hoerndl, Frédéric J., et al. "Kinesin-1 regulates synaptic strength by mediating the delivery, removal, and redistribution of AMPA receptors." *Neuron* 80.6 (2013): 1421-1437.
97. Brockie, Penelope J., et al. "Cornichons control ER export of AMPA receptors to regulate synaptic excitability." *Neuron* 80.1 (2013): 129-142.
98. Žáčík, Michal. Šumová spektroskopie pro biologii. Diss. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2013.
99. Phillips, James William, and Yi Jin. "Systems and methods for modulating the electrical activity of a brain using neuro-EEG synchronization therapy." U.S. Patent No. 8,465,408. 18 Jun. 2013.
100. Moon, Jim, et al. "Body-worn vital sign monitor." U.S. Patent No. 8,364,250. 29 Jan. 2013.
101. Hao, Manzhao, et al. "Corticospinal Transmission of Tremor Signals by Propriospinal Neurons in Parkinson's Disease." *PloS one* 8.11 (2013): e79829.
102. McCOMBIE, Devin, Marshal Dhillon, and Matt Banet. "Method for measuring patient motion, activity level, and posture along with PTT-based blood pressure." U.S. Patent No. 8,475,370. 2 Jul. 2013.
103. Banet, Matt, Devin McCombie, and Marshal Dhillon. "Body-worn monitor for measuring respiration rate." U.S. Patent No. 8,545,417. 1 Oct. 2013.
104. Banet, Matt, and Jim Moon. "Body-worn vital sign monitor." U.S. Patent No. 8,591,411. 26 Nov. 2013.
105. McCombie, Devin, et al. "Alarm system that processes both motion and vital signs using specific

- heuristic rules and thresholds." U.S. Patent No. 8,594,776. 26 Nov. **2013**.
106. Banet, Matt, Marshal Dhillon, and Devin McCombie. "Body-worn system for measuring continuous non-invasive blood pressure (cNIBP)." U.S. Patent No. 8,602,997. 10 Dec. **2013**.
107. Moon, Jim, et al. "Body-worn pulse oximeter." U.S. Patent No. 8,437,824. 7 May **2013**.
108. Cheng, Chunmei, et al. "Remote sensing estimation of Chlorophyll and suspended sediment concentration in turbid water based on spectral separation." Optik-International Journal for Light and Electron Optics 124.24 (**2013**): 6815-6819.
109. Phillips, James William, and Yi Jin. "Systems and methods for neuro-EEG synchronization therapy." U.S. Patent No. 8,585,568. 19 Nov. **2013**.
110. Khvostichenko, Daria S., et al. "An X-ray transparent microfluidic platform for screening of the phase behavior of lipidic mesophases." Analyst 138.18 (**2013**): 5384- 5395.
111. "A signal alignment method based on DTW with new modification", Karabiber, F.; Bilgisayar Muhendisligi Bolumu; Balcilar, M. Signal Processing and Communications Applications Conference (SIU), **2013** 21st, 24-26 April 2013. ISBN: 978-1-4673-5562-9; DOI: 10.1109/SIU.2013.6531176
112. "An automated signal alignment algorithm based on dynamic time warping for capillary electrophoresis data", Turkish Journal of Electrical Engineering & Computer Sciences, Fethullah KARABİBER, 21, (**2013**), 851-863. Full text: pdf
113. "Traditional Asymmetric Rhythms: A Refined Model of Meter Induction Based On Asymmetric Meter Templates", Fouloulis, Thanos, Aggelos Pikrakis, and Emiliros Cambouropoulos, Proceedings of the Third International Workshop on Folk Music Analysis (FMA2013). **2013**. ISBN 978-90-70389-78-9
114. "Comparison of two methods for measuring γ -H2AX nuclear fluorescence as a marker of DNA damage in cultured human cells: applications for microbeam radiation therapy." Anderson, D., et al. , Journal of Instrumentation 8.06 (**2013**): C06008. Full text PDF.
115. Ayodeji, Olugboji Oluwafemi, Jonathan Yisa Jiya, and Jack M. Hale. "Event Reconstruction by Digital Filtering." Advances in Signal Processing 1.3 (**2013**): 48-56.
116. "A conserved aromatic residue regulating photosensitivity in short-wavelength sensitive cone visual pigments". Kuemmel, C. M., Sandberg, M. N., Birge, R. R., & Knox, B. E. Biochemistry, 52(30), 5084-5091 (**2013**).
117. "Measurement of The Lightweight Rotor Eigenfrequencies And Tuning Of Its\ Model Parameters," Luboš SMOLÍK, Michal HAJŽMAN, Transactions of the VŠB – Technical University of Ostrava, Mechanical Series, No. 1, **2013**, vol. LIX. Full English text.
118. "Investigation of the phase separation of PNIPAM using infrared spectroscopy together with multivariate data analysis." Munk, Tommy, et al. , Polymer 54.26 (**2013**): 6947-6953. Abstract.
119. "Phase separation in $In_xGa_{1-x}N$ ($0.10 < x < 0.40$). " Belyaev, K. G., Raklin, M. -V., Jmerik, V. N., Mizerov, A. M., Kuznetsova, Y. V., Zamoryanskaya, M. V., ... & Toropov, A. A. (2013). Physica Status Solidi (c), 10 (3), 527-531.
120. "Cortic muscular Transmission of Tremor Signals by Propriospinal Neurons in Parkinson's Disease." Hao, Manzhao, et al. , PloS one 8.11 (**2013**): e79829.
121. "Sickle-shaped voxel approach to enhance automatic reclaiming operation using bucket wheel reclamer," Maung Thi Rein Myo; Tien-Fu Lu, Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on, vol., no., pp.1700,1705, 19- 21 June **2013**. doi: 10.1109/ICIEA.2013.6566642
122. "Review of software tools for design and analysis of large-scale MRM proteomic datasets." Colangelo, Christopher M., et al., Methods 61.3 (**2013**): 287-298.

123. Carabetta, Valerie J., et al. "A complex of YlbF, YmcA and YaaT regulates sporulation, competence and biofilm formation by accelerating the phosphorylation of Spo0A." *Molecular microbiology* 88.2 (2013): 283-300.
124. Web, N. L. P. M. L., and Andrew Rosenberg. "Ensemble Methods." (2013).
125. Cannon, Robert William, "Automated Spectral Identification of Materials using Spectral Identity Mapping", 2013, Master of Science in Chemistry, Cleveland State University, College of Sciences and Health Professions.
126. MS Freeman, ZI Cleveland, Y Qi, Enabling hyperpolarized ^{129}Xe MR spectroscopy and imaging of pulmonary gas transfer to the red blood cells in transgenic mice expressing human hemoglobin", *Magnetic Resonance in Medicine*, Volume 70, Issue 5, pages 1192–1199, November 2013
127. SMOLÍK, Luboš, and Michal HAJ ŽMAN. "MEASUREMENT OF THE LIGHTWEIGHT ROTOR EIGENFREQUENCIES AND TUNING OF ITS MODEL PARAMETERS. Transactions of the VSB – Technical University of Ostrava, Mechanical Series No. 1, 2013, vol. LIX article No. 1942
128. Kumssa, Aida Meredassa. "Tablet User Interface Evaluation for a Portable Ultrasound System and Real-time Doppler Spectrum Processing." (2013).
129. Circuit level defects in the developing neocortex of Fragile X mice, J Tiago Gonçalves, James E Anstey, Peyman Golshani, Carlos Portera-Cailliau, *Nature Neuroscience* 16, 903–909 (2013) doi:10.1038/nn.3415
130. A Baradarani, J Sadler, JRB Taylor, High-resolution blood flow imaging through the skull, *Electronics Letters*, vol. 40, no. 13, 2014, pp. 798–799.
131. Singh, R. (2014). Tune Measurement at GSI SIS-18: Methods and Applications (Doctoral dissertation, Technische Universität).
132. Pander, Tomasz, et al. "An automatic saccadic eye movement detection in an optokinetic nystagmus signal." *Biomedical Engineering/Biomedizinische Technik* 59.6 (2014): 529-543.
133. "Demonstration of Large Coupling-Induced Phase Delay in Silicon Directional Cross-Couplers," Westerveld, W.J.; Pozo, J.; Leinders, S.M.; Yousefi, M.; Urbach, H.P., *Selected Topics in Quantum Electronics, IEEE Journal of*, vol.20, no.4, pp.1,6, July-Aug. 2014, doi: 10.1109/JSTQE.2013.2292874
134. "Probabilistic peak detection for first-order chromatographic data", M. Lopatka, G. Vivo-Truyols, M.J. Sjerps, *Analytical Chemica Acta*, 2014 DOI: <http://dx.doi.org/10.1016/j.aca.2014.02.015>
135. "A recursive algorithm for optimizing differentiation." Mashreghi, Ali, and Hadi Sadoghi Yazdi. *Journal of Computational and Applied Mathematics* 263 (2014): 1-13.
136. Cade, D. E. (2014). Detection, classification and ecology of acoustic scattering layers (Doctoral dissertation).
137. Grubišić, Vladimir, et al. "Heterogeneity of myotubes generated by the MyoD and E12 basic helix-loop-helix transcription factors in otherwise non-differentiation growth conditions." *Biomaterials* 35.7 (2014): 2188-2198.
138. "Comparison of Signal Smoothing Techniques for Use in Embedded System for Monitoring and Determining the Quality of Biofuels", Dalton Cézane Gomes Valadares, Rute Cardoso Drebes, Elmar Uwe Kurt Melcher, Sérgio de Brito Espínola, Joseana Macêdo Fechine Régis de Araújo, *Applied Mechanics and Materials*, Vols. 448-453, pages 1679-1688, Trans Tech Publications, Switzerland, 2014. DOI: 10.4028/www.scientific.net/AMM.448-453.1679
139. "Characterization of Integrated Optical Strain Sensors Based on Silicon Waveguides," Westerveld, W.J.; Leinders, S.M.; Muilwijk, P.M.; Pozo, J.; van den Dool, T.C.; Verweij, M.D.; Yousefi, M.; Urbach, H.P., *Selected Topics in Quantum Electronics, IEEE Journal of*, vol.20, no.4, pp.1,10, July-Aug. 2014. doi: 10.1109/JSTQE.2013.2289992
140. "Gaussian-function-based deconvolution method to determine the penetration ability of petrolatum oil into in vivo human skin using confocal Raman microscopy", Chun-Sik Choe, Jürgen Lademann, and Maxim

141. "Borosilicate Glass Containing Bismuth and Zinc Oxides as a Hot Cell Material for Gamma-Ray Shielding". H. A. Saudi, H. A. Sallam, K. Abdullah. Physics and Materials Chemistry. **2014**; 2(1):20-24. doi: 10.12691/pmc-2-1-4.
142. "Theta-Burst Stimulation of Hippocampal Slices Induces Network-Level Calcium Oscillations and Activates Analogous Gene Transcription to Spatial Learning", Graham K. Sheridan, Emad Moeendarbary, Mark Pickering, John J. O'Connor, and Keith J. Murphy, PLOS One, June 20, **2014**. DOI: 10.1371/journal.pone.0100546
143. Mahmoud, Imbabay I., and Mohamed S. El_Tokhy. "Development of coincidence summing and resolution enhancement algorithms for digital gamma ray spectroscopy." Journal of Analytical Atomic Spectrometry 29.8 (**2014**): 1459-1466.
144. M. Rahmat, W. Maulina, Isnaeni, Miftah, N. Sukmawati, E. Rustami, M. Azis, K.B. Seminar, A.S. Yuwono, Y.H. Cho, H. Alatas, Development of a novel ozone gas sensor based on sol-gel fabricated photonic crystal, Sensors and Actuators A: Physical, Volume 220, 1 December **2014**, Pages 53–61
145. "Bacteria-instructed synthesis of polymers for self-selective microbial binding and labelling", E. Peter Magennis, Francisco Fernandez-Trillo, Cheng Sui, Sebastian G. Spain, David J. Bradshaw, David Churchley, Giuseppe Mantovani, Klaus Winzer & Cameron Alexander, Nature Materials 13, 748–755 (**2014**) doi:10.1038/nmat3949 (<http://www.nature.com/nmat/journal/v13/n7/extref/nmat3949-s1.pdf>)
146. A COMPUTERIZED DATABASE FOR BULLET COMPARISON BY CONSECUTIVE MATCHING, Ashley Chu, David Read and David Howitt, Federally funded grant report, U.S. Department of Justice, Document No. 247771, July **2014**. (<http://www.crime-scene-investigator.net/computerized-database-for-bullet-comparisonby-consecutive-matching.pdf>)
147. Cade David E., Benoit-Bird Kelly J., (**2014**), An automatic and quantitative approach to the detection and tracking of acoustic scattering layers, Limnology and Oceanography: Methods, 12, doi: 10.4319/lom.2014.12.742.
148. Blake, Phillip, et al. "Diffraction in nanoparticle lattices increases sensitivity of localized surface plasmon resonance to refractive index changes." Journal of Nanophotonics 8.1 (**2014**): 083084-083084.
149. Sprinkhuizen, Sara M., Jerome L. Ackerman, and Yi Qiao Song. "Influence of bone marrow composition on measurements of trabecular microstructure using decay due to diffusion in the internal field MRI: Simulations and clinical studies." Magnetic Resonance in Medicine 72.6 (**2014**): 1499-1508.
150. Canlas, Reich Rechner D., Carlo Noel E. Ochotorena, and Elmer P. Dadios, "Fuzzy-genetic photoplethysmograph peak detection." Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management(HNICEM), 2014 International Conference on. IEEE, **2014**.
151. Duenas, J. A., et al. "Dependency on the silicon detector working bias for proton-deuteron particle identification at low energies." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 714 (**2013**): 48-52.
152. Sterling, Ryan, and Nathaniel Todd. "USING NEURAL SIGNALS TO PROVIDE INPUT FOR COMPUTING APPLICATIONS IN AUTONOMOUS PROSTHETICS." [PDF] from 136.142.82.187
153. Wang, Xiao, Yi-Qing Ni, and Ke-Chang Lin. "Comparison of statistical counting methods in SHM-based reliability assessment of bridges." Journal of Civil Structural Health Monitoring: 1-12.
154. González-Sáiz, J. M., et al. "Modulation of the phenolic composition and colour of red wines subjected to accelerated ageing by controlling process variables", Food chemistry 165 (**2014**): 271-281.
155. Kurniawan, Itmy Hidayat, and Sahat Simbolon. "Deteksi dan Pengukuran Spektra dalam Analisis Spektrografi Emisi dengan Pengolahan Citra." Jurnal Nasional Teknik Elektro dan Teknologi Informasi

(JNTETI) 3.1 (2014).

156. Souri, Zoha. EEG-BASED ASSESSMENT OF DRIVER'S PERCEPTION OF TRAFFIC ENVIRONMENT. Diss. Lamar University, **2014**.
157. Lin, Junfang, et al. "Novel method for quantifying the cell size of marine phytoplankton based on optical measurements." *Optics express* 22.9 (**2014**): 10467-10476.
158. Hammonds Jr, James S., Kimani A. Stancil, and Charlezetta E. Stokes. "Quality factor temperature dependence of a surface phonon polariton resonance cavity." *Applied Physics Letters* 105.11 (**2014**): 114107.
159. Mall, U., et al. "Characterization of lunar soils through spectral features extraction in the NIR." *Advances in Space Research* 54.10 (**2014**): 2029-2040.
160. Bucur, R. V. "Structure of the Voltammograms of the Platinum-Black Electrodes: Derivative Voltammetry and Data Fitting Analysis." *Electrochimica Acta* 129 (**2014**): 76-84.
161. Teixeira, Carlos Esteves. "Sobre a teoria da difração de raios-X em estruturas tridimensionais." (**2014**). [PDF] from ufmg.br
162. Moon, Jim, et al. "Cable system for generating signals for detecting motion and measuring vital signs." U.S. Patent No. 8,738,118. 27 May **2014**.
163. Thompson, D. Brian, et al. "A Comparison of R-line Photoluminescence of Emeralds from Different Origins." *The Journal of Gemmology* 34.4 (**2014**): 334.
164. Oliveira, Raphael Rocha de. "Modelos de calibração multivariada por NIRS para a predição de características de qualidade da carne bovina." (**2014**). PDF] from ufg.br
165. Kirley, M. P. (**2014**). Electrical conductivity of metal surfaces at terahertz frequencies (Doctoral dissertation, The University of Wisconsin-Madison).
166. Anderson, Danielle L., et al. "Spatial and temporal distribution of γH2AX fluorescence in human cell cultures following synchrotron-generated X-ray microbeams: lack of correlation between persistent γH2AX foci and apoptosis." *Synchrotron Radiation* 21.4 (**2014**).
167. Maxfield, Dane Arthur. KINESIN-1 REGULATES SYNAPTIC STRENGTH BY MEDIATING DELIVERY, REMOVAL AND REDISTRIBUTION OF AMPARS. Diss. The University of Utah, **2014**.
168. Zou, Xiaoyu, Magneto-optical properties of ferromagnetic nanostructures on modified nanosphere templates. Thesis, CALIFORNIA STATE UNIVERSITY, LONG BEACH, **2014**, 87 pages; 1591619
169. A Carrasco, TA Brown, SG Lomber, Spectral and Temporal Acoustic Features Modulate Response Irregularities within Primary Auditory Cortex Columns, *PloS one*, **2014**, DOI: 10.1371/journal.pone.0114550
170. Sirotin, Yevgeniy B., Martín Elias Costa, and Diego A. Laplagne. "Rodent ultrasonic vocalizations are bound to active sniffing behavior." *Frontiers in behavioral neuroscience* 8 (**2014**).
171. Luo, Changtong, et al. "Wave system fitting: A new method for force measurements in shock tunnels with long test duration." *Mechanical Systems and Signal Processing* (**2015**).
172. Bleeker, J. V. (**2015**). Relating phase separation and thickness mismatch in model lipid membranes (Doctoral dissertation).
173. Möbius, Klaus, et al. "Möbius–Hückel topology switching in an expanded porphyrin cation radical as studied by EPR and ENDOR spectroscopy." *Physical Chemistry Chemical Physics* 17.9 (**2015**): 6644-6652.
174. Tariq, Humera, and SM Aqil Burney. "Low Level Segmentation of Brain MR Slices and Quantification Challenges.", NCMCS'15 (**2015**). Link.
175. Nystad, Helle Emilia. Comparison of Principal Component Analysis and Spectral Angle Mapping for

Identification of Materials in Terahertz Transmission Measurements. Diss. Master's thesis, Norwegian University of Technology and Science, **2015**.

176. Hahn, Christian, et al. "Adjusting rheological properties of concentrated microgel suspensions by particle size distribution." *Food Hydrocolloids* 49 (**2015**): 183-191.
177. Chiuchiú, D. "Time-dependent study of bit reset." *EPL (Europhysics Letters)* 109.3 (**2015**): 30002.
178. Taghizadeh, Mohammad Taghi, Nazanin Yeganeh, and Mostafa Rezaei. "The investigation of thermal decomposition pathway and products of poly (vinyl alcohol) by TG FTIR." - *Journal of Applied Polymer Science* 132.25 (**2015**).
179. P Sevusu, Real-time air quality measurements using mobile platforms, **2015**, Thesis, [PDF] from rutgers.edu
180. D. S. Khvostichenko, J. D. D. Ng, S. L. Perry, M. Menon, P. J. A. Kenis, Effects of detergent β -octyglucoside and phosphate salt solutions on phase behavior of monoolein mesophases, [PDF] from researchgate.net
181. Mahmoud, Imbabay I., and Mohamed S. El_Tokhy. "Advanced signal separation and recovery algorithms for digital x-ray spectroscopy." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 773 (**2015**): 104-113.
182. Morrow, Justin D. Surface Microstructure and Properties of Pulsed Laser Micro Melted S7 Tool Steel. The University of Wisconsin-Madison, **2015**.
183. Ubnoske, Stephen M., et al. "Role of nanocrystalline domain size on the electrochemical double-layer capacitance of high edge density carbon nanostructures." *MRS Communications* (**2015**): 1-6.
184. MUHAMMAD MUFTI AZIS, "Experimental and kinetic studies of H₂ effect on lean exhaust after treatment processes: HC-SCR and DOC" CHALMERS UNIVERSITY OF TECHNOLOGY Göteborg, Sweden **2015**
185. Umesh Rudrapatna, S., et al. "Measurement of distinctive features of cortical spreading depolarizations with different MRI contrasts." *NMR in Biomedicine* 28.5 (**2015**): 591-600.
186. Kühbach, Markus, Brüggemann, Thiemo, Molodov, Konstantin, Gottstein, Günter. "On a Fast and Accurate In-Situ Measuring Strategy for Recrystallization Kinetics and Its Application to an Al-Fe-Si Alloy", *Metallurgical and Materials Transactions A*, March **2015**, Volume 46, Issue 3, pp 1337-1348
187. D. Y. Lipatov, Y. R. Shaltaeva, V. V. Belyakov, A. V. Golovin, V. S. Pershenkov, V. V. Shurenkov, D. Y. Yakovlev, "Modeling of IMS Spectra in Medical Diagnostic Purposes", 3rd International Conference on Nanotechnologies and Biomedical Engineering, Volume 55 of the series IFMBE Proceedings, **2015**, pp 404-408
188. Y. Meerten, Y. Swolfs, J. Baets, L. Gorbatikh, I. Verpoebucurst, "Penetration impact testing of self-reinforced composites", Composites Part A: Applied Science and Manufacturing, Volume 68, January **2015**, Pages 289–295
189. Ivanov, I, Optimal filtering of synchronized current phasor measurements in a steady state, 2015 IEEE International Conference on Industrial Technology (ICIT), Pages 1362 - 1367, 17-19 March **2015**
190. L Farge, J Boisse, J Dillet, S André, Wide angle X ray scattering study of the lamellar/fibrillar transition for a semi crystalline polymer deformed in tension in relation with the evolution of volume strain, *Journal of Polymer Science B*, Volume 53, Issue 20, 15 October **2015**, Pages 1470–1480
191. Patrick Schloth, Precipitation in the high strength AA7449 aluminium alloy: implications on internal stresses on different length scales, Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, June **2015**.
192. Guzman, P. (**2015**). Studying the Physical Stability of BSA at the Bulk Solution and Oil/Water Interface

(Doctoral dissertation, University of Otago).

193. FlavonQ: An Automated Data Processing Tool for Profiling Flavone and Flavonol Glycosides with Ultra-High-Performance Liquid Chromatography–Diode Array Detection–High Resolution Accurate Mass–Mass Spectrometry, Mengliang Zhang, Jianghao Sun, and Pei Chen*, *Anal. Chem.*, **2015**, 87 (19), pp 9974–9981,
DOI: 10.1021/acs.analchem.5b02624
194. Schulze, H. Georg, and Robin FB Turner. "Development and Integration of Block Operations for Data Invariant Automation of Digital Preprocessing and Analysis of Biological and Biomedical Raman Spectra." *Applied spectroscopy* 69.6 (**2015**): 643-664.
195. Hutchison, Richard Stephen. Novel high refractive index, thermally conductive additives for high brightness white LEDs. Diss. Rensselaer Polytechnic Institute, **2015**.
196. Magnotti, G., et al. "Raman spectra of methane, ethylene, ethane, dimethyl ether, formaldehyde and propane for combustion applications." *Journal of Quantitative Spectroscopy and Radiative Transfer* 163 (**2015**): 80-101.
197. Chen, Rex Chin-Hao. "Spectral and Temporal Interrogation of Cerebral Hemodynamics Via High-Speed Laser Speckle Contrast Imaging." (**2015**).
198. Maistry, N. (**2015**). Investigating the concept of Fraunhofer lines as a potential method to detect corona in the wavelength region 338nm-405nm during the day (Doctoral dissertation).
199. Parker, Michael J. Coupling Nuclear Induced Phonon Propagation with Conversion Electron Moessbauer Spectroscopy. No. AFIT-ENP-MS-15-J-054. AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ENGINEERING AND MANAGEMENT, **2015**.
200. Maistry, Nattele. Investigating the concept of Fraunhofer lines as a potential method to detect corona in the wavelength region 338nm-405nm during the day. Diss. **2015**.
201. Liu, Yanping, et al. "Applications of Savitzky-Golay Filter for Seismic Random Noise Reduction." *Acta Geophysica* (**2015**).
202. Kojimoto, N. C. (**2015**). Ultrasonic inspection methods for defect detection and process control in roll-to-roll flexible electronics manufacturing (Doctoral dissertation, Massachusetts Institute of Technology).
203. Sheehan, Terry L., and Richard A. Yost. "What's the most meaningful standard for mass spectrometry: instrument detection limit or signal-to-noise ratio" *Current Trends Mass Spectrometry* 13 (**2015**): 16-22.
204. Bleeker, J. V. (**2015**). Relating phase separation and thickness mismatch in model lipid membranes (Doctoral dissertation).
205. Ilewicz, Witold, et al. "Comparison of baseline estimation algorithms for chromatographic signals." Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on. IEEE, **2015**.
206. Massimi, Federico. Sviluppo di metodi integrati basati sulle tecniche di nanoindentazione e del fascio ionico focalizzato (FIB) per la caratterizzazione, risolta nello spazio, delle proprietà meccaniche dei materiali", ArcAdiA." (**2015**). <http://hdl.handle.net/2307/5329>
207. Swoboda, Daniel Maximilian, et al. "A Comprehensive Study of Simple Digital Filters for Botball IR Detection Techniques." PDF link.
208. CE Funes, EF Cromwell System and method for determining a baseline measurement for a biological response curve, US Patent App. 13/308,021, 2
209. AD Beyene, R Bluffstone, Z Gebreegziabher, The Improved Biomass Stove Saves Wood, But How Often Do People Use It?, [TXT] from worldbank.com

210. Raunio, Saida. "IMMUNOASSAY TEST FOR A QVANTITATIVE DETERMINATION OF HELICOBACTER PYLORI ANTIBODY IN BLOOD DONORS" (**2015**). PDFAlt, Daniel M. Design and Commissioning of a 16.1 MHz Multiharmonic Buncher for the ReAccelerator at NSCL. ProQuest, **2016**.
211. Coy, A., Rankine, D., Taylor, M., Nielsen, D. C., & Cohen, J. (**2016**). Increasing the accuracy and automation of fractional vegetation cover estimation from digital photographs. *Remote Sensing*, 8(7), 474.
212. Nguyen, Tuan Ngoc. "An algorithm for extracting the PPG Baseline Drift in realtime." (**2016**). PDF link.
213. Maitre, Léa. "Metabonomic and epidemiological analyses of maternal parameters and exposures during pregnancy and their influence on fetal growth amongst the INMA birth cohort." (**2016**). PDF link.
214. Lipatov, D. Y., et al. "Modeling of IMS Spectra in Medical Diagnostic Purposes." 3rd International Conference on Nanotechnologies and Biomedical Engineering. Springer Singapore, **2016**.
215. Tong, Xia, et al. "Recursive Wavelet Peak Detection of Analytical Signals." *Chromatographia* 79.19-20 (**2016**): 1247-1255.
216. Wang, Xing. Effects of Interfaces on Properties of Cladding Materials for Advanced Nuclear Reactors. The University of Wisconsin-Madison, **2016**. PDF link.
217. Dang, Hue, Marian Dekker, Jason Farquhar, and Tom Heskes. "Processing and analyzing functional near-infrared spectroscopy data." (**2016**).
218. Damavandi, H. G. (**2016**). Data analytics, interpretation and machine learning for environmental forensics using peak mapping methods (Doctoral dissertation, The University of Iowa).
219. Gill, Ruby K., et al. "The effects of laser repetition rate on femtosecond laser ablation of dry bone: a thermal and LIBS study." *Journal of biophotonics* 9.1-2 (**2016**): 171-180.
220. Performance evaluation and optimization of X-ray stress measurement for nickel aluminium bronze based on the Bayesian method. *Journal of Applied Crystallography*, **2016** – scripts.iucr.org
221. Top-down modulation of stimulus drive via beta-gamma cross-frequency interaction. CG Richter, WH Thompson, CA Bosman, P Fries - bioRxiv, **2016** – biorxiv.org
222. Azpúrua, Marco A., Marc Pous, and Ferran Silva. "Decomposition of Electromagnetic Interferences in the Time-Domain." (**2016**).
223. Barros, Rodrigo Emanoel de Britto Andrade. SISTEMA DE INTERROGAÇÃO DE REDES DE BRAGG: PRIMEIROS PASSOS NA CRIAÇÃO DE UM PROTÓTIPO. Diss. Universidade Federal do Rio de Janeiro, **2016**.
224. Li, Yuanlu, et al. "A novel signal enhancement method for overlapped peaks with noise immunity." *Spectroscopy Letters* 49.4 (**2016**): 285-293.
225. Hatterschide, Joshua. "Retroviral-RNA Structure and Function: Investigating the role of aminoacyl-tRNA synthetases and retroviral-RNA structural elements in the initiation of reverse transcription." (**2016**).
226. Guizani, Chamseddine, et al. "Biomass char gasification by H₂O, CO₂ and their mixture: Evolution of chemical, textural and structural properties of the chars." *Energy* 112 (**2016**): 133-145.
227. Wagner, C. F. (**2016**). Transition from transparency to hole-boring in relativistic laser-solid interactions at the Texas Petawatt (Doctoral dissertation).
228. Bocage, E., and S. Hillson. "Disturbances and noise: Defining furrow form enamel hypoplasia." *American journal of physical anthropology* 161.4 (**2016**): 744-751
229. Merla, Yu, et al. "Extending battery life: A low-cost practical diagnostic technique for lithium-ion batteries." *Journal of Power Sources* 331 (**2016**): 224-231.
230. Besemer, Matthieu, et al. "Identification of Multiple Water-Iodide Species in Concentrated NaI Solutions Based on the Raman Bending Vibration of Water." *The Journal of Physical Chemistry A* 120.5

(2016): 709-714.

231. Cairone, Fabiana, Salvina Gagliano, and Maide Bucolo. "Experimental study on the slug flow in a serpentine microchannel." *Experimental Thermal and Fluid Science* 76 (2016): 34-44.
232. Davison, Adrian K., et al. "Objective Micro-Facial Movement Detection Using FACS-Based Regions and Baseline Evaluation." arXiv preprint arXiv:1612.05038 (2016).
233. Ninh, Giang Nguyen, et al. "Radioisotope identification method for poorly resolved gamma-ray spectrum of nuclear security concern." *AIP Conference Proceedings*. Vol. 1704. No. 1. AIP Publishing, 2016.
234. Brachi, Paola, et al. "Pseudo-component thermal decomposition kinetics of tomato peels via isoconversional methods." *Fuel Processing Technology* 154 (2016): 243-250.
235. Lee, Hansol, et al. "Flow suppressed hyperpolarized ¹³C chemical shift imaging using velocity optimized bipolar gradient in mouse liver tumors at 9.4 T.", *Magnetic resonance in medicine* (2016).
236. Wu, B., et al. "Novel application of differential thermal voltammetry as an in-depth state-of-health diagnosis method for lithium-ion batteries." PDF file.
237. Creese, Andrew J., and Helen J. Cooper. "Separation of cis and trans Isomers of Polyproline by FAIMS Mass Spectrometry." *Journal of The American Society for Mass Spectrometry* 27.12 (2016): 2071-2074.
238. Kvyetnyy, Roman, et al. "Improving the quality perception of digital images using modified method of the eye aberration correction." *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments* 2016. Vol. 10031. *International Society for Optics and Photonics*, 2016.
239. Myers, G. A., Turner, L. G., Morgan, Q., & Pearce, J., "Raman Spectroscopy-detecting SO_x and NO_x in the Precipice Sandstone". (2016)
240. Pancholi, Manthan, et al. "Relative Translation and Rotation Calibration Between Optical Target and Inertial Measurement Unit." International Conference on Sensor Systems and Software. Springer, Cham, 2016.
241. Ferriss, Elizabeth, Terry Plank, and David Walker. "Site-specific hydrogen diffusion rates during clinopyroxene dehydration." *Contributions to Mineralogy and Petrology* 171.6 (2016): 55.
242. Roy, Sujan Kumar, Wei-Ping Zhu, and Benoit Champagne. "Single channel speech enhancement using subband iterative Kalman filter." Circuits and Systems (ISCAS), 2016 IEEE International Symposium on. IEEE, 2016.
243. Langaas, Gjertrud Louise. "Measurements of radioactivity in plant and soil samples taken near a nuclear power plant." (2016). PDF link.
244. Benigni, Paolo, and Francisco Fernandez-Lima. "Oversampling selective accumulation trapped ion mobility spectrometry coupled to FT-ICR MS: fundamentals and applications." *Analytical chemistry* 88.14 (2016): 7404-7412.
245. Geiger, Matthew, and Michael T. Bowser. "Effect of fluorescent labels on amino acid sample dimensionality in two dimensional nLC× μFFE separations." *Analytical chemistry* 88.4 (2016): 2177-2187.
246. Aldokhail, A. M. (2016). Automated Signal to Noise Ratio Analysis for Resonance Imaging Using a Noise Distribution Model (Doctoral dissertation, University of Toledo).
247. Fasching, Joshua, et al. "Automated coding of activity videos from a study." Robotics and Automation (ICRA), 2016 IEEE International Conference. IEEE, 2016.
248. Bleeker, J. V., Cox, P. A., Foster, R. N., Litz, J. P., Blosser, M. C., Castner, D. G., & Keller, S. L. (2016). Thickness Mismatch of Coexisting Liquid Phases in Non-Canonical Lipid Bilayers. *The journal of physical chemistry. B*, 120(10), 2761.
249. Joshi, Bijal, and Nitu Anil Kumar. "Computationally efficient data rate mismatch compensation for telephony clocks." U.S. Patent No. 9,514,766. 6 Dec. 2016.
250. Vallet, Aurélien, et al. "A multi-dimensional statistical rainfall threshold for deep landslides based on groundwater recharge and support vector machines." *Natural Hazards* 84.2 (2016): 821-849.

251. Wang, Lili, Paul DeRose, and Adolfas K. Gaigalas. "Assignment of the number of equivalent reference fluorophores to dyed microspheres." *J. Res. Nat. Ins. Stand. Technol.* 121 (2016): 269-286.
252. Skaret, H. B. (2016). The Arctic Sea Ice-Melting During Summer or not Freezing in Winter? (Master's thesis, The University of Bergen). PDF link.
253. Tuan T. Tran, et. al., Synthesis of Ge_{1-x}Sn_x alloys by ion implantation and pulsed laser melting: Towards a group IV direct bandgap material, *Journal of Applied Physics* 119(18):183102, 2016, DOI: 10.1063/1.4948960
363. Choi, Jae Sung, et al. "A New Automated Cell Counting Program by Using Hough Transform-Based Double Edge." *Advances in Computer Science and Ubiquitous Computing*. Springer, Singapore, 2016. 712-716.
253. Van der Rest, Guillaume, Human Rezaei, and Frédéric Halgand. "Monitoring Conformational Landscape of Ovine Prion Protein Monomer Using Ion Mobility Coupled to Mass Spectrometry." *Journal of The American Society for Mass Spectrometry* 28.2 (2017): 303-314.
254. Mirsafavi, Rustin Y., et al. "Detection of papaverine for the possible identification of illicit opium cultivation." *Analytical Chemistry* 89.3 (2017): 1684-1688.
255. Myers, Grant A., Kelsey Kehoe, and Paul Hackley. "Analysis of Artificially Matured Shales with Confocal Laser Scanning Raman Microscopy: Applications to Organic Matter Characterization." Unconventional Resources Technology Conference (URTEC), 2017.
256. Torres, Andrei BB, José Adriano Filho, Atslands R. da Rocha, Rubens Sonsol Gondim, and José Neuman de Souza. "Outlier detection methods and sensor data fusion for precision agriculture", 2017, PDF link.
257. Desmet, F., Lesaffre, M., Six, J., Ehrlé, N., & Samson, S. (2017). Multimodal analysis of synchronization data from patients with dementia. In ESCOM 2017.
258. Seeber, Renato, and Alessandro Ulrici. "Analog and digital worlds: Part 2. Fourier analysis in signals and data treatment." *ChemTexts* 3.2 (2017): 8.
259. Mustafa, M. A., et al. "Nonintrusive Freestream Velocity Measurement in a Large-Scale Hypersonic Wind Tunnel." *AIAA Journal* (2017).
260. Suárez-Cortés, Pablo, et al. "Ned-19 inhibition of parasite growth and multiplication suggests a role for NAADP mediated signaling in the asexual development of Plasmodium falciparum." *Malaria Journal* 16.1 (2017): 366.
261. Catalbas, M. C., & Dobrisek, S., 3D Moving Sound Source Localization via Conventional Microphones. *Elektronika ir Elektrotechnika*, 23(4), 63-69. (2017).
262. Du, Zhenhui, et al. "High-sensitive carbon disulfide sensor using wavelength modulation spectroscopy in the mid-infrared fingerprint region." *Sensors and Actuators B: Chemical* 247 (2017): 384-391.
263. Hamilton, N. E., Mahjoub, R., Laws, K. J., & Ferry, M. (2017). A blended NPT/NVT scheme for simulating metallic glasses. *Computational Materials Science*, 130, 130-137.
264. Sun, Y. C., Huang, C., Xia, G., Jin, S. Q., & Lu, H. B. (2017). Accurate wavelength calibration method for compact CCD spectrometer. *JOSA A*, 34(4), 498-505.
265. Mikhailov, I. F., et al. "Rapid diagnostics of urinary iodine using a portable EDXRF spectrometer." *Journal of X-Ray Science and Technology Preprint* (2017): 1-7. PDF link.
266. Bianchi, Davide, et al. "A wavelet filtering method for cumulative gamma spectroscopy used in wear measurements." *Applied Radiation and Isotopes* 120 (2017): 51-59.
267. Xiong, Zheng, et al. "Automated Phase Segmentation for Large-Scale X-ray Diffraction Data Using a Graph-Based Phase Segmentation (GPhase) Algorithm." *ACS Combinatorial Science* 19.3 (2017): 137-144
268. Jiménez-Carvajal, C., et al. "Weighing lysimetric system for the determination of the water balance during irrigation in potted plants." *Agricultural Water Management* 183 (2017): 78-85.

269. Acciarri, R., et al. "Noise Characterization and Filtering in the MicroBooNE Liquid Argon TPC." arXiv preprint arXiv:1705.07341 (**2017**). PDF link.
270. Mathault, Jessy, Hamza Landari, Frederic Tessier, Paul Fortier, and Amine Miled. "Biological Modeling Challenges in a Multiphysics Approach." Circuits and Systems (MWSCAS), **2017** IEEE 60th International Midwest Symposium
271. Weiss, Charles J. "Scientific Computing for Chemists: An Undergraduate Course in Simulations, Data Processing, and Visualization." Journal of Chemical Education 94.5 (**2017**): 592-597.
272. Kianifar, Rezvan, et al. "Automated Assessment of Dynamic Knee Valgus and Risk of Knee Injury During the Single Leg Squat." *IEEE Journal of Translational Engineering in Health and Medicine* 5 (**2017**): 1-13.
273. Willem deGroot, A., et al. "Molecular Structural Characterization of Polyethylene." Handbook of Industrial Polyethylene and Technology: Definitive Guide to Manufacturing, Properties, Processing, Applications and Markets (**2017**): 139.
274. Mertens, Andreas, and Josef Granwehr. "Two-dimensional impedance data analysis by the distribution of relaxation times." *Journal of Energy Storage* 13 (**2017**): 401-408.
275. Wu, Yingwen, and Long Chen. "Comparison of spectra processing methods for SERS based quantitative analysis." Information, Cybernetics and Computational Social Systems (ICCSS), 2017 4th International Conference on. IEEE, **2017**.
276. Dehnavi, Sahar, Yasser Maghsoudi, and Mohammadjavad Valadanzej. "Using spectrum differentiation and combination for target detection of minerals." *International Journal of Applied Earth Observation and Geoinformation* 55 (**2017**): 9-20.
278. Jia, Zhenhua, et al. "HB-phone: a bed-mounted geophone-based heartbeat monitoring system: demo abstract." Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks. ACM, **2017**.
279. Gozé, Perrine, et al. "Effects of ozone treatment on the molecular properties of wheat grain proteins." *Journal of Cereal Science* 75 (**2017**): 243-251.
280. Giron-Sierra, Jose Maria. "Periodic Signals." Digital Signal Processing with Matlab Examples, Volume 1. Springer Singapore, **2017**. 3-28.
281. Shojaosadati, Seyed Abbas, Sajjad Naeimipour, and Ahmad Fazeli. "FTIR Investigation of secondary structure of Reteplase inclusion bodies produced in Escherichia coli in terms of urea concentration (Spring 2017)." *Iranian Journal of Pharmaceutical Research* (**2017**).
282. Peng, Jiyu, et al. "Rapid Identification of Varieties of Walnut Powder Based on Laser-Induced Breakdown Spectroscopy." (**2017**): 19-28. Abstract.
283. Sun, Lili, et al. "Comprehensive evaluation of chemical stability of Xuebijing injection based on multiwavelength chromatographic fingerprints and multivariate chemometric techniques." *Journal of Liquid Chromatography & Related Technologies* 40.14 (**2017**): 715-724.
284. Thompson, D. Brian, et al. "Photoluminescence Spectra of Emeralds from Colombia, Afghanistan, and Zambia." *Gems & Gemology* 53.3 (**2017**): 296-311.
285. Choorat, P., et al. "Applied integral intensity projection to find the numbers of the parking spots." Knowledge and Smart Technology (KST), **2017** 9th International Conference on. IEEE, 2017.
286. Phillips, James William, and Yi Jin. "Devices and methods of low frequency magnetic stimulation therapy." U.S. Patent No. 9,649,502. 16 May **2017**.
287. Augustyns, Valérie, et al. "Evidence of tetragonal distortion as the origin of the ferromagnetic ground state in γ - Fe nanoparticles." *Physical Review B* 96.17 (**2017**):174410.
288. Sprague-Klein, Emily A., et al. "Observation of Single Molecule Plasmon-Driven Electron Transfer in Isotopically Edited 4, 4-Bipyridine Gold Nanosphere Oligomers." *Journal of the American Chemical Society* 139.42 (**2017**): 15212-15221.
289. Mohan, Varun, and Prashant K. Jain. "Spectral Heterogeneity of Hybrid Lead Halide Perovskites

Demystified by Spatially Resolved Emission." *The Journal of Physical Chemistry C* 121.35 (2017): 19392-19400.

290. Cuss, Chad W., Iain Grant-Weaver, and William Shotyk. "AF4-ICPMS with the 300 Da Membrane to Resolve Metal-Bearing "Colloids"< 1 kDa: Optimization, Fractogram Deconvolution, and Advanced Quality Control." *Analytical Chemistry* 89.15 (2017): 8027-8035.
291. Shi, Xiaoyu, et al. "Super-Resolution Microscopy Reveals That Disruption of Ciliary Transition Zone Architecture Is a Cause of Joubert Syndrome." *bioRxiv* (2017): 142042.
292. Robinson, M. T., et al. "Photocatalytic photosystem I/PEDOT composite films prepared by vapor-phase polymerization." *Nanoscale* 9.18 (2017): 6158-6166.
293. Ros Martí, Marc. Deep convolutional neural network architecture for effective Image analysis. MS thesis. Universitat Politècnica de Catalunya, 2017.
294. Jackson, Philip J., et al. "Identification of protein W, the elusive sixth subunit of the Rhodopseudomonas palustris reaction center-light harvesting 1 core complex." *Biochimica et Biophysica Acta (BBA)-Bioenergetics* (2017).
295. Johnson, Alexander C., and Michael T. Bowser. "High-Speed, Comprehensive, Two-Dimensional Separations of Peptides and Small Molecule Biological Amines Using Capillary Electrophoresis Coupled with Micro Free Flow Electrophoresis." *Analytical chemistry* 89.3 (2017): 1665-1673.
296. Toose, Peter, et al. "Radio-frequency interference mitigating hyperspectral L band radiometer." *Geoscientific Instrumentation, Methods and Data Systems* 6.1 (2017): 39.
297. Pajankar, Ashwin. "Filters and Their Application." Raspberry Pi Image Processing Programming. Apress, 2017. 99-110.
298. Taraszewski, Michał, and Janusz Ewertowski. "Complex experimental analysis of rifle-shooter interaction." *Defence Technology* (2017).
299. Manlises, Cyrel Ontimare, et al. "Characterization of an ISFET with Built-in Calibration Registers through Segmented Eight-Bit Binary Search in Three-Point Algorithm Using FPGA." *Journal of Low Power Electronics and Applications* 7.3 (2017):19.
300. Kim, Geonha, et al. "Soil sampling strategies for site assessments in petroleum contaminated areas." *Environmental geochemistry and health* 39.2 (2017): 293-305.
301. Lanevski, Dmitri, Koit Mauring, and Eric Tkaczyk. "Interference filter tilting to detect a polycyclic aromatic hydrocarbon at the second harmonic of wavelength modulation frequency." *Applied Optics* 56.11 (2017): 3155-3161.
302. Hong, Tae-Kee, Iason Rusodimos, and Myung-Hoon Kim. "Higher order derivative voltammetry for reversible and irreversible electrode processes under spherical diffusion." *Journal of Electroanalytical Chemistry* 785 (2017): 255-264.
303. Root, Katharina, et al. "Insight into Signal Response of Protein Ions in Native ESI-MS from the Analysis of Model Mixtures of Covalently Linked Protein Oligomers." *Journal of The American Society for Mass Spectrometry* (2017): 1-13.
304. Du, Zhenhui, et al. "High-sensitive carbon disulfide sensor using wavelength Modulation spectroscopy in the mid-infrared fingerprint region." *Sensors and Actuators B: Chemical* 247 (2017): 384-391.
305. Elzanfaly, Eman S., et al. "Zero and second derivative synchronous fluorescence spectroscopy for the quantification of two non-classical β lactams in pharmaceutical vials: Application to stability studies." *Luminescence* (2017).
306. Ferraz de Menezes, Rebeca, et al. "Fs laser ablation of teeth is temperature limited and provides information about the ablated components." *Journal of Biophotonics* (2017).
307. Huang, Yi-Fan, et al. "Label-free, ultrahigh-speed, 3D observation of bidirectional and correlated intracellular cargo transport by coherent brightfield microscopy." *Nanoscale* 9.19 (2017): 6567-6574.
308. Mahmud, Akib. "Hardware in the Loop (HIL) Rig Design and Electrical Architecture." (2017).

309. Beyerl, Thomas, et al. *Reducing Complexity in Routing of Non-Standard Intersections, to Aid in Autonomous Vehicle Navigation*. No. 2017-01-0103. SAE Technical Paper, **2017**.
310. Lee, Hansol, et al. "Flow-suppressed hyperpolarized ¹³C chemical shift imaging using velocity-optimized bipolar gradient in mouse liver tumors at 9.4 T." *Magnetic resonance in medicine* 78.5 (2017): 1674-1682.
311. Haines, Grant E., and S. Laurie Sanderson. "Integration of swimming kinematics and ram suspension feeding in a model American paddlefish, *Polyodon spathula*." *Journal of Experimental Biology* (2017): jeb-166835.
312. Zhang, Huajun, and Y. I. N. G. Ning. "Method for Analyzing Mixture Components." U.S. Patent Application 15/120,974, filed March 2, **2017**.
313. Soto Morras, Marta. "Implementation and Analysis of Real Time Optical Flow Solutions for GPU architectures." (**2017**).
314. Pajankar, Ashwin. *Raspberry Pi Image Processing Programming*. Apress, **2017**.
315. Vintila, Florentin, Thomas C. Kübler, and Enkelejda Kasneci. "Pupil response as an indicator of hazard perception during simulator driving." *Journal of Eye Movement Research* 10.4 (2017): 3.
316. Humera Tariq, Abdul Muqeet, S.M.Aqil Burney, Humera Azam, "Otsu's Segmentation....", *J. Theoretical and Applied Information Technology*, Vol.95. No 22, **2017**
- 317 Liu, Yu, et al. "Supersonic transient magnetic resonance elastography for quantitative assessment of tissue elasticity." *Physics in Medicine & Biology* 62.10 (2017): 4083.
318. Manar M. Ouda, et. Al., Development of Pileup Recovery Algorithms by Peak Detection Method of Digital Gamma Ray Spectroscopy, 34th National Radio Science Conference (NRSC), **2017**. [Link to full paper](#).
319. Xu, Jun-Li, Aoife A. Gowen, and Da-Wen Sun. "Time series hyperspectral chemical imaging (HCI) for investigation of spectral variations associated with water and plasticizers in casein-based biopolymers." *Journal of Food Engineering* 218 (2018): 88-105.
320. Smith, Brad C., Bachana Lomsadze, and Steven T. Cundiff. "Optimum repetition rates for dual-comb spectroscopy." *Optics express* 26.9 (2018): 12049-12056.
321. Butler, C. W., et al. "Neurons Specifically Activated by Fear Learning in Lateral Amygdala Display Increased Synaptic Strength." *eNeuro* 5.3 (2018).
322. Pukhlyakova, Ekaterina, et al. " β -Catenin-dependent mechanotransduction dates back to the common ancestor of Cnidaria and Bilateria." *Proceedings of the National Academy of Sciences* 115.24 (2018): 6231-6236.
323. Cheng, Jie. *Peak Detection to Count Gold Nanoparticles Translocations in Nanopipette*. Diss. UC Santa Cruz, **2018**.
324. Bonde, Amelie, et al. "VVRRM: Vehicular Vibration-Based Heart RR-Interval Monitoring System." *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*. ACM, **2018**.
325. Paige, Cristen, et al. "Characterizing the Normative Voice Tremor Frequency in Essential Vocal Tremor." *JAMA Otolaryngology–Head & Neck Surgery* (2018).
326. Myers, Grant A., Kelsey Kehoe, and Paul Hackley. "Development of Raman Spectroscopy as a Thermal Maturity Proxy in Unconventional Resource Assessment." *Unconventional Resources Technology Conference, Houston, Texas, 23-25 July 2018*. Society of Exploration Geophysicists, American Association of Petroleum Geologists, Society of Petroleum Engineers, **2018**.
327. Taraszewski, Michal, and Janusz Ewertowski. "Small-Caliber Grenade Projectile Applicable to Individual Grenade Launchers." *Defence Science Journal* 68.5 (2018).
328. Trinh, N. D., et al. "Double differential neutron spectra generated by the interaction of a 12 MeV/nucleon ³⁶S beam on a thick natCu target." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 896 (2018): 152-164.
329. Swainsbury, David JK, et al. "Probing the local lipid environment of the Rhodobacter sphaeroides cytochrome bc₁ and Synechocystis sp. PCC 6803 cytochrome b₆f complexes with styrene maleic acid." *Biochimica et Biophysica Acta (BBA)-Bioenergetics* 1859.3 (2018): 215-225.

330. Reynes, Julien, et al. "Experimental constraints on hydrogen diffusion in garnet." *Contributions to Mineralogy and Petrology* 173.9 (2018): 69.
331. Omer, Muhammad, and Elise C. Fear. "Automated 3D method for the construction of flexible and reconfigurable numerical breast models from MRI scans." *Medical & biological engineering & computing* 56.6 (2018): 1027-1040.
332. Klein, Tobias, et al. "Influence of Liquid Structure on Fickian Diffusion in Binary Mixtures of n-Hexane and Carbon Dioxide Probed by Dynamic Light Scattering, Raman Spectroscopy, and Molecular Dynamics Simulations." *The Journal of Physical Chemistry B* (2018).
333. Kielar, A., T. Deschamps, R. Jokel, and J. A. Meltzer. "Abnormal language-related oscillatory responses in primary progressive aphasia." *NeuroImage: Clinical* 18 (2018): 560-574.
334. Prodanov, Milana, et al. "Software Module for Processing EEG Signals in a Biofeedback Based System." *2018 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE, 2018.
335. Fratini, Marta, et al. "Surface Immobilization of Viruses and Nanoparticles Elucidates Early Events in Clathrin-Mediated Endocytosis." *ACS infectious diseases* (2018).
336. Siliņš, Kaspars. *Plasma Enhanced Chemical-and Physical-Vapor Depositions Using Hollow Cathodes*. Diss. Acta Universitatis Upsaliensis, 2018.
- 337 Schito, Andrea, and Sveva Corrado. "An automatic approach for characterization of the thermal maturity of dispersed organic matter Raman spectra at low diagenetic stages." *Geological Society, London, Special Publications* 484 (2018): SP484-5.
338. Krystal T. Vasquez, et. al., Low-pressure gas chromatography with chemical ionization mass Spectrometry for quantification of multifunctional organic compounds in the atmosphere, *Atmos. Meas. Tech.* 2018. [PDF](#).
339. Pushkarsky, I., Tseng, P., Black, D., France, B., Warfe, L., Koziol-White, C. J., ... & Damoiseaux, R. (2018). Elastomeric sensor surfaces for high-throughput single-cell force cytometry (vol 2, pg 124, 2018).
340. Ismail, Omar, et al. "The Way to Ultrafast, High-Throughput Enantioseparations of Bioactive Compounds in Liquid and Supercritical Fluid Chromatography." *Molecules* 23.10 (2018): 2709.
338. Hellinghausen, Garrett, M. Farooq Wahab, and Daniel W. Armstrong. "Improving visualization of trace components for quantification using a power law-based integration approach." *Journal of Chromatography A* 1574 (2018): 1-8.
339. Khundadze, Nana, et al. "On our way to sub-second separations of enantiomers in high-performance liquid chromatography." *Journal of Chromatography A* 1572 (2018): 37-43.
344. Roy, Daipayan, et al. "Frontiers in Ultrafast Chiral Chromatography." *LC• GC Europe* (2018): 308.
345. Maddalena, Riccardo, Christopher Hall, and Andrea Hamilton. "Effect of silica particle size on the formation of calcium silicate hydrate using thermal analysis." *Thermochimica Acta* (2018).
346. Darweesh, Samar Ahmed, et al. "Advancement and Validation of New Derivative Spectrophotometric Method for Individual and Simultaneous Estimation of Diclofenac sodium and Nicotinamide." *Oriental Journal of Chemistry* 34.3 (2018).
347. Li, Yuanlu, and Min Jiang. "Spatial-fractional order diffusion filtering." *Journal of Mathematical Chemistry* 56.1 (2018): 257-267.
348. Huang, Dian, et al. "High-Speed Live-Cell Interferometry: A New Method for Quantifying Tumor Drug Resistance and Heterogeneity." *Analytical chemistry* 90.5 (2018): 3299-3306.
349. Wu, Rihan, et al. "Demonstration of time-of-flight technique with all-optical modulation and MCT detection in SWIR/MWIR range." *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*. Vol. 10799. International Society for Optics and Photonics, 2018.
350. Pontremoli, Carlotta, et al. "Insight into the interaction of inhaled corticosteroids with human serum albumin: A spectroscopic-based study." *Journal of pharmaceutical analysis* 8.1 (2018): 37-44.
351. Zhao, Chenjiang. *Signal Processing: Peak Detection*. Diss. UC Santa Cruz, 2018.

352. Coelho, Alan A. "Deconvolution of instrument and $K\alpha_2$ contributions from X-ray powder diffraction patterns using nonlinear least-squares with penalties." *Journal of Applied Crystallography* 51.1 (2018): 112-123.
353. Al-gawwam, Sarmad, and Mohammed Benissa. "Robust Eye Blink Detection Based on Eye Landmarks and Savitzky-Golay Filtering." *Information* 9.4 (2018): 93.
354. Yilmaz, Cagatay Murat, Cemal Kose, and Bahar Hatipoglu. "A Quasi-probabilistic distribution model for EEG Signal classification by using 2-D signal representation." *Computer methods and programs in biomedicine* 162 (2018): 187-196.
355. Gou, Yonggang, et al. "Motion parameter estimation and measured data correction derived from blast-induced vibration: new insights." *Measurement* (2018).
356. Hakala, Teemu, et al. "Direct Reflectance Measurements from Drones: Sensor Absolute Radiometric Calibration and System Tests for Forest Reflectance Characterization." *Sensors (Basel, Switzerland)* 18.5 (2018).
357. Mihálik, A., R. Ďuríkovič, and M. Sejč. "Application of Motion Capture Attributes to Individual Identification under Corridor Surveillance." *Journal of Applied Mathematics, Statistics and Informatics* 14.1 (2018): 37-56.
358. Kianifar, Rezvan, and Dana Kulic. "Automatic assessment of the squat quality and risk of knee injury in the single leg squat." U.S. Patent Application 15/826,259, filed October 11, 2018.
359. Parziale, Nick J., et al. "Amplification and Structure of Streamwise-Velocity Fluctuations in Four Shock-Wave/Turbulent Boundary-Layer Interactions." *2018 Fluid Dynamics Conference*. 2018.
360. Simon, David M., and Mark T. Wallace. "Integration and Temporal Processing of Asynchronous Audiovisual Speech" *Journal of cognitive neuroscience* 30.3 (2018): 319-337.
361. Richter, Craig G., Richard Coppola, and Steven L. Bressler. "Top-down beta oscillatory signaling conveys behavioral context in early visual cortex." *Scientific reports* 8.1 (2018): 6991.
362. Dinç, Erdal, and Zehra Yazan. "Wavelet transform-based UV spectroscopy for pharmaceutical analysis" *Frontiers in Chemistry* 6 (2018).
363. Bartussek, Jan, and Fritz-Olaf Lehmann. "Sensory processing by motoneurons: a numerical model for low-level flight control in flies." *Journal of The Royal Society Interface* 15.145 (2018): 20180408.
364. Ben Hendrickson, Ralf Widenhorn, Paul R. DeStefano and Erik Bodegom, Detection and Reconstruction of Random Telegraph Signals Using Machine Learning, *Image Processing (ICIP)*, Athens, 2018, pp. 2441-2445. [Link](#).
365. Oeltzscher, Georg, et al. "Hadamard editing of glutathione and macromolecule-suppressed GABA." *NMR in Biomedicine* 31.1 (2018): e3844.
364. Nocco, Mallika A., Matthew D. Ruark, and Christopher J. Kucharik. "Apparent electrical conductivity predicts physical properties of coarse soils." *Geoderma* 335 (2019): 1-11.
366. Лубов, Д. П., М. В. Катков, и Ю. В. Першин. "Вольт-амперные характеристики коммерческих сегнетоэлектрических конденсаторов: отклонения от модели Прейзаха." << ԳԱԱ Տեղեկագիր. Ֆիզիկա 53.1 (2018): 86-95. (Machine translation: Voltage – ampere characteristics of commercial ferroelectric capacitors: deviations from the Preisach model. Armenian NAS RA Bulletin: Physics).
367. Thanos Papanicolaou, Achilleas G. Tsakiris, Micah A Wyssmann, Casey Kramer, Boulder Array Effects on Bedload Pulses and Depositional Patches, *Journal of Geophysical Research: Earth Surface* 123(11), 2018.
368. Manuja Sharma, et. al., "Optical pH measurement system using a single fluorescent dye for assessing susceptibility to dental caries", *Journal of Biomedical Optics* 24(01):1, 2019.
369. Mustafa, Muhammad A., David Shekhtman, and Nick J. Parziale. "Single-Laser Krypton Tagging Velocimetry Investigation of Supersonic Air and N₂ Boundary-Layer Flows over a Hollow Cylinder in a Shock Tube." *Physical Review Applied* 11.6 (2019): 064013.[Link](#).
370. Karl Auerswald, Franziska K. Fischer, Tanja Winterrath, Robert Brandhuber, "Rain erosivity map for Germany derived from contiguous radar rain data", *Hydrology and Earth System Sciences* 23(4):1819-1832, April 2019, DOI: 10.5194/hess-23-1819-2019
371. Martin Leblanc, et. al., Actinide mixed oxide conversion by advanced thermal denitration route, *Journal*

372. Sujan Kumar Roy and Kuldip K. Paliwal, An Iterative Kalman Filter with Reduced-Biased Kalman Gain for Single Channel Speech Enhancement in Non-stationary Noise Condition, *International Journal of Signal Processing Systems* Vol. 7, No. 1, March 2019. DOI: 10.18178/ijsp.7.1.7-13
373. J. Chen, C. Yang, H. Zhu & Y. Li, Adaptive signal enhancement for overlapped peaks based on weighting factor selection, *Spectroscopy Letters*, Volume 52, 2019. DOI: 10.1080/00387010.2018.1556219
374. Delfino, I., S. Cavella, and M. Lepore. "Scattering-based optical techniques for olive oil characterization and quality control." *Journal of Food Measurement and Characterization* 13.1 (2019): 196-212.
375. Zhang, G. W., et al. "Decomposition of overlapped ion mobility peaks by sparse representation." *International Journal of Mass Spectrometry* 436 (2019): 147-152.
376. Demetriou, Demetris. *An Investigation into Nonlinear Random Vibrations based on Wiener Series Theory*. Diss. University of Cambridge, 2019.
377. McLaren, Timothy I., René Verel, and Emmanuel Frossard. "The structural composition of soil phosphomonoesters as determined by solution 31P NMR spectroscopy and transverse relaxation (T2) experiments." *Geoderma* 345 (2019): 31-37.
378. Yan, Qi, Rui Yang, and Jiwu Huang. "Detection of Speech Smoothing on Very Short Clips." *IEEE Transactions on Information Forensics and Security* (2019).
379. Zhang, Weifang, et al. "The Analysis of FBG Central Wavelength Variation with Crack Propagation Based on a Self-Adaptive Multi-Peak Detection Algorithm." *Sensors* 19.5 (2019): 1056.
380. Fayolle, Clemence, Mélody Labrune, and Jean-Philippe Berteau. "Raman spectroscopy investigation shows that mineral maturity is greater in CD-1 than in C57BL/6 mice distal femurs after sexual maturity." *Connective Tissue Research* (2019).
381. Catlow, C. Richard A., et al. "Synthesis, characterisation and water-gas shift activity of nano-particulate mixed-metal (Al, Ti) cobalt oxides.", 2019. [researchgate.net](#).
383. Hansen, Lars N., et al. "Low-temperature plasticity in olivine: Grain size, strain hardening, and the strength of the lithosphere." *Journal of Geophysical Research: Solid Earth* (2019).
384. Shao, Xueguang, et al. "High order derivative to investigate the complexity of the near infrared spectra of aqueous solutions." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 213 (2019): 83-89.
385. Chores, Yael, et al. "Long-term griffon vulture population dynamics at Gamla Nature Reserve." *The Journal of Wildlife Management* 83.1 (2019): 135-144.
386. DeFelice, Mialy M., et al. "NF-κB signaling dynamics is controlled by a dose-sensing autoregulatory loop." *Sci. Signal.* 12.579 (2019): eaau3568. [PDF](#).
387. Xiaoxiao Ge, et al., Mechanism studies and fabrication for the incorporation of carbon into Al alloys by the electro-charging assisted process, *Carbon*. Page: 203-212, April 2019. DOI: 10.1016/j.carbon.2019.04.049
388. Natalia Molinero. Et. al, "The human gallbladder microbiome is related to the physiological state and the biliary metabolic profile", *Microbiome* 7(1), 2019. DOI: 10.1186/s40168-019-0712-8
389. Ci Song and Tao Pei, "Decomposition of Repulsive Clusters in Complex Point Processes with Heterogeneous Components", *International Journal of Geo-Information* 8(8):326, 2019. DOI: 10.3390/ijgi8080326
390. Stephanie Zaleski, et. al., "Application of Fiber Optic Reflectance Spectroscopy for the Detection of Historical Glass Deterioration", *Journal of the American Ceramic Society*, June 2019. DOI: 10.1111/jace.16703
391. M A Mustafa and Nick Parziale, "Proper Orthogonal Decomposition of Streamwise-Velocity Fluctuations in a Compression-Corner Shock-Wave/Turbulent Boundary-Layer Interaction", Conference: 32nd International Symposium on Shock Waves (ISSW32), June 2019, DOI: 10.3850/978-981-11-2730-4_0473-cd

392. Antonio Matus-Vargas, et. al, "Aerodynamic Disturbance Rejection Acting on a Quadcopter Near Ground", Conference: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), March **2019**. DOI: 10.1109/CoDIT.2019.8820321
393. Wenqi Cai, "Modeling and Experimental Study of the Vibration Effects in Urban Free-Space Optical Communication Systems", *IEEE Photonics Journal* PP(99):1-1, October **2019**. DOI: 10.1109/JPHOT.2019.2945695
394. Wei, Lingxiao, et al. "I know what you see: Power side-channel attack on convolutional neural network accelerators." Proceedings of the 34th Annual Computer Security Applications Conference. ACM, **2018**.
395. Shewcraft, Ryan A., et al. "Coherent neuronal dynamics driven by optogenetic stimulation in the primate brain." *bioRxiv* (**2019**): 437970.
396. Burke, I. V., and William Johan. "A robust and automated deconvolution algorithm of peaks in spectroscopic data." (**2019**).
397. Tu, Shen, et al. "Enhanced Formation of Solvent-Shared Ion Pairs in Aqueous Calcium Perchlorate Solution toward Saturated Concentration or Deep Supercooling Temperature and Its Effects on the Water Structure." *The Journal of Physical Chemistry B* 123.45 (**2019**): 9654-9667.
398. Noori, Ansara, et al. "Portable Device for Continuous Sensing with Rapidly Pulsed LEDs—Part 1: Rapid On-the-fly Processing of Large Data Streams using an Open-Source Microcontroller with Field Programmable Gate Array." *Measurement* (**2019**).
399. Kim, Tae Hyong, et al. "Machine learning-based pre-impact fall detection model to discriminate various types of fall." *Journal of biomechanical engineering* 141.8 (**2019**): 081010.
400. Suresh, P. S., Niranjan Kumar Sura, and K. Shankar. "Landing Response Analysis on High-Performance Aircraft Using Estimated Touchdown States." *SAE International Journal of Aerospace* 12.1 (**2019**): 23-40.
401. Muirhead, David K., et al. "Raman Spectroscopy: an effective thermal marker in low temperature carbonaceous fold-thrust belts." *Geological Society, London, Special Publications* 490 (**2019**): SP490-2019.
402. Moreira, Mateus Perrisé, Manuel Castro Carneiro, and Andrey Linhares Bezerra de Oliveira. "Desenvolvimento de um programa para modelagem da curva de titulação de traços de carbonato em solução de hidróxido de lítio concentrado em sistema fechado." (**2019**).
403. Paruzzo, Federico Maria. *New Approaches to NMR Crystallography*. No. THESIS. EPFL, **2019**.
404. Paruzzo, Federico M., and Lyndon Emsley. "High-resolution ^1H NMR of powdered solids by homonuclear dipolar decoupling." *Journal of Magnetic Resonance* 309 (**2019**): 106598.
405. Alzamil, Yasser. Optimising the quantitative analysis in functional pet brain imaging. Diss. Cardiff University, **2019**.
406. Walker, Patrick William, "War without oversight: Challenges to the development of autonomous weapons systems.", Thesis, University of Buckingham (**2019**).
407. Yang, Guofeng, et al. "Multiple Constrained Reweighted Penalized Least Squares for Spectral Baseline Correction." *Applied Spectroscopy* (**2019**): 0003702819885002.
408. Zhang, Jing, Shuai Chen, and Guoxiang Sun. "Spectral and chromatographic overall analysis: An insight into chemical equivalence assessment of traditional Chinese medicine." *Journal of Chromatography A* (**2019**): 460556.
409. Richter, Craig G., et al. "Brain rhythms shift and deploy attention." *bioRxiv* (**2019**): 795567.
410. Chen, Weiqi, et al. "An automated microfluidic system for the investigation of asphaltene deposition and dissolution in porous media." *Lab on a Chip* 19.21 (**2019**): 3628-3640.
411. Mukherjee, Soma, Soumi Betal, and Asoke Prasun Chattopadhyay. "Dual sensing and synchronous fluorescence spectroscopic monitoring of Cr^{3+} and Al^{3+} using a luminescent Schiff base: Extraction and DFT studies." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* (**2019**): 117837.
412. Roy Daipayan, and Daniel W. Armstrong. "Fast super/subcritical fluid chromatographic enantioseparations on superficially porous particles bonded with broad selectivity chiral selectors relative to fully porous particles." *Journal of Chromatography A* 1605 (**2019**): 360339.

413. Kang, Yuhao, and Azriel Z. Genack. "Time delay in a disordered topological system." *arXiv preprint arXiv:1912.05151* (2019).
414. Chankvetadze, Bezhana. "Recent trends in preparation, investigation and application of polysaccharide-based chiral stationary phases for separation of enantiomers in high-performance liquid chromatography." *TrAC Trends in Analytical Chemistry* (2019): 115709.
415. de Paula Pedroza, Ricardo Henrique. *Development of methods based on NIR and Raman spectroscopies together with chemometric tools for the qualitative and quantitative analysis of gasoline*. MS thesis. The University of Bergen, 2019.
416. Wolf, Moritz, et al. "Synthesis, characterisation and water–gas shift activity of nano-particulate mixed-metal (Al, Ti) cobalt oxides." *Dalton Transactions* 48.36 (2019): 13858-13868.
417. Welch, Christopher J. "High throughput analysis enables high throughput experimentation in pharmaceutical process research." *Reaction Chemistry & Engineering* 4.11 (2019): 1895-1911.
418. Mironov, N. A., et al. "Methods for Studying Petroleum Porphyrins." *Petroleum Chemistry* 59.10 (2019): 1077-1091.
419. Yang, Guofeng, et al. "Spectral features extraction based on continuous wavelet transform and image segmentation for peak detection." *Analytical Methods* (2019).
420. Briggs, Tokini Kiki. *An Auto-Picking Algorithm for the Detection of Clay Seams in Potash Mines Using GPR Data*. Diss. Faculty of Graduate Studies and Research, University of Regina, 2019.
421. Hong, Ning, et al. "High-Speed Rail Suspension System Health Monitoring Using Multi-Location Vibration Data." *IEEE Transactions on Intelligent Transportation Systems* (2019).
422. Hu, Jennifer F., et al. "Sequencing-based quantitative mapping of the cellular small RNA landscape." *bioRxiv* (2019): 841130.
423. Elsayad, Kareem. "Spectral Phasor Analysis for Brillouin Microspectroscopy." *Frontiers in Physics* 7 (2019): 62.
424. Huang, Weinan, et al. "Morphology and cell wall composition changes in lignified cells from loquat fruit during postharvest storage." *Postharvest Biology and Technology* 157 (2019): 110975.
425. Kupka, Teobald, et al. "Local aromaticity mapping in the vicinity of planar and nonplanar molecules." *Magnetic Resonance in Chemistry* 57.7 (2019): 359-372.
426. Parigger, Christian G. "Measurements of Gaseous Hydrogen–Nitrogen Laser-Plasma." *Atoms* 7.3 (2019): 61.
427. Venara, J., et al. "Design and development of a portable β -spectrometer for 90Sr activity measurements in contaminated matrices." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* (2019).
428. Gallmeier, Esther A., et al. "Real time monitoring of the chemistry of hydroxylamine nitrate and iron as surrogates for nuclear materials processing." *Separation Science and Technology* (2019): 1-9.
429. Hong, Ning, et al. "High-Speed Rail Suspension System Health Monitoring Using Multi-Location Vibration Data." *IEEE Transactions on Intelligent Transportation Systems* (2019).
430. Kielar, Aneta, et al. "Slowing is slowing: Delayed neural responses to words are linked to abnormally slow resting state activity in primary progressive aphasia." *Neuropsychologia* 129 (2019): 331-347.
431. Wu, Wenchang, et al. "Diffusivities in 1-Alcohols Containing Dissolved H₂, He, N₂, CO, or CO₂ Close to Infinite Dilution." *The Journal of Physical Chemistry B* 123.41 (2019): 8777-8790.
432. Park, Sungchan, K. A. N. G. Jooyoung, and Jungho Kim. "Ultrasound imaging apparatus and method for controlling the same." U.S. Patent Application 10/2 47,824, filed April 2, 2019.
433. Renda, Fioranna, et al. "kSHREC 'Delta' reflects the shape of kinetochore rather than intrakinetochore tension." *BioRxiv* (2019): 811075.
434. Santiago, Ruben, et al. "Methanol-Promoted Oxidation of Nitrogen Oxide (NO x) by Encapsulated Ionic Liquids." *Environmental science & technology* 53.20 (2019): 11969-11978.
435. Hu, Jennifer Fan. A systems-level view of the tRNA epitranscriptome: defining the role of tRNA

abundance, stability, and modifications in the bacterial stress response. Diss. Massachusetts Institute of Technology, **2019**.

436. Weaver, Jordan S., Veronica Livescu, and Nathan A. Mara. "A comparison of adiabatic shear bands in wrought and additively manufactured 316L stainless steel using nano-indentation and electron backscatter diffraction." *Journal of Materials Science* 55.4 (**2020**): 1738-1752.

437. Li, Dongmei, Zhiwei Zhu, and Da-Wen Sun. "Visualization of the in-situ distribution of contents and hydrogen bonding states of cellular level water in apple tissues by confocal Raman microscopy." *Analyst* (**2020**).

438. Tsai, Chong-Bin, Wei-Yu Hung, and Wei-Yen Hsu. "A Fast and Effective System for Analysis of Optokinetic Waveforms with a Low-Cost Eye Tracking Device." *Healthcare*. Vol. 9. No. 1. Multidisciplinary Digital Publishing Institute, **2020**.

439. Elsayad, Kareem. "Spectral phasor analysis for Brillouin microspectroscopy." *Frontiers in Physics* 7 (**2019**): 62.

440. Takis, Panteleimon G., et al. "SMolESY: An Efficient and Quantitative Alternative to On-Instrument Macromolecular 1 H-NMR Signal Suppression." *Chemical Science* (**2020**).

441. Pipathanapoompron, Thalerungsak, et al. "Magnetic reader testing for asymmetric oscillation noise." *Journal of Magnetism and Magnetic Materials* (**2020**): 167064.

442. Ito, Motohiro, et al. "Evaluation of cone-beam computed tomography over a small field of view in a water bath based on the modulation transfer function with repeating-edge oversampling." *Journal of Oral Science* (**2020**): 20-0479.

443. Li, Tianjun, Long Chen, and Xiliang Lu. "An Alternating Direction Minimization based denoising method for extracted ion chromatogram." *Chemometrics and Intelligent Laboratory Systems* 206 (**2020**): 104138.

444. Zhang, Jing, Shuai Chen, and Guoxiang Sun. "Spectral and chromatographic overall analysis: An insight into chemical equivalence assessment of traditional Chinese medicine." *Journal of Chromatography A* 1610 (**2020**): 460556.

445. McPherson, David L., Richard Harris, and David Sorensen. "Functional Neuroimaging of the Central Auditory System." *Advances in Audiology and Hearing Science: Volume 1: Clinical Protocols and Hearing Devices* (**2020**): 327.

446. Hebert, Michael J., and David H. Russell. "Tracking the Structural Evolution of 4-Aminobenzoic Acid in the Transition from Solution to the Gas Phase." *The Journal of Physical Chemistry B* 124.11 (**2020**): 2081-2087.

447. Tang, Hui, et al. "On 2D-3D Image Feature Detections for Image-To-Geometry Registration in Virtual Dental Model." *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, **2020**.

448. Bhatia, Siddharth, and Karthikeyan Vasudevan. "Comparative proteomics of geographically distinct saw-scaled viper (Echis carinatus) venoms from India." *Toxicon: X* 7 (**2020**): 100048.

449. Resentini, Alberto, et al. "Zircon as a provenance tracer: Coupling Raman spectroscopy and UPb geochronology in source-to-sink studies." *Chemical Geology* 555 (**2020**): 119828.

450. Ajemigbitse, Moses A., Fred S. Cannon, and Nathaniel R. Warner. "A rapid method to determine 226Ra concentrations in Marcellus Shale produced waters using liquid scintillation counting." *Journal of Environmental Radioactivity* 220 (**2020**): 106300.

451. Muirhead, D. K., et al. "Raman spectroscopy: an effective thermal marker in low temperature carbonaceous fold-thrust belts." *Geological Society, London, Special Publications* 490.1 (**2020**): 135-151.

452. Quilfen, Y., and B. Chapron. "On denoising satellite altimeter measurements for high-resolution geophysical signal analysis." *Advances in Space Research* (**2020**).

453. Wu, Dan, Pol Mac Aonghusa, and Donal O'Shea. "Correlation of National and Healthcare Workers COVID-19 Infection Data; Implications for Large-scale Viral Testing Programs." *medRxiv* (**2020**).

454. Battini, Davide, et al. "Modeling Approach and Finite Element Analyses of a Shape Memory Epoxy-Based Material." *Conference of the Italian Association of Theoretical and Applied Mechanics*. Springer, Cham, **2019**.

455. Wu, Billy, et al. "An Energy Storage Device Monitoring Technique." U.S. Patent Application No. 16/088,016, **2020**.
456. Ge, X., et al. "Electrical and structural characterization of nano-carbon–aluminum composites fabricated by electro-charging-assisted process." *Carbon* 173: 115-125, **2021**.
456. Hsu, Gee-Sern Jison, et al. "A deep learning framework for heart rate estimation from facial videos." *Neurocomputing* 417 (**2020**): 155-166.
457. Hammonds, James S., and Kimani A. Stancil. "Phonon effect based nanoscale temperature measurement." U.S. Patent No. 10,520,374. 31 Dec. **2019**.
458. Weaver, Jordan S., Veronica Livescu, and Nathan A. Mara. "A comparison of adiabatic shear bands in wrought and additively manufactured 316L stainless steel using nanoindentation and electron backscatter diffraction." *Journal of Materials Science* 55.4 (**2020**): 1738-1752.
459. Gallagher, John Barry, Vishnu Prahalad, and John Aalders. "Inorganic and black carbon hotspots constrain blue carbon mitigation services across tropical seagrass and temperate tidal marshes." *bioRxiv* (**2020**).
460. Wu, Wenchang, et al. "Mutual and Thermal Diffusivities as well as Fluid-Phase Equilibria of Mixtures of 1-Hexanol and Carbon Dioxide." *The Journal of Physical Chemistry B* 124.12 (**2020**): 2482-2494.
461. Busa, William, Phillip H. Coelho, and Paul Getchel. "Lateral flow immunoassay test reader and method of use." U.S. Patent No. 10,823,746. 3 Nov. **2020**.
462. Dubrovkin, Joseph. *Mathematical processing of spectral data in analytical chemistry: A guide to error analysis*. Cambridge Scholars Publishing, **2019**.
463. Bonnin-Pascual, Francisco, and Alberto Ortiz. "UWB-Based Self-Localization Strategies: A Novel ICP-Based Method and a Comparative Assessment for Noisy-Ranges-Prone Environments." *Sensors* 20.19 (**2020**): 5613.
464. Chaumel, Júlia, et al. "Co-aligned chondrocytes: Zonal morphological variation and structured arrangement of cell lacunae in tessellated cartilage." *Bone* (**2020**): 115264.
465. Bruendl, Stefan A., et al. "A New Emulation Platform for Real-time Machine Learning in Substance Use Data Streams." *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, **2020**.
466. Cuyt, Annie, and Wen-shin Lee. "Parameter spectral analysis: scale and shift." *arXiv preprint arXiv:2008.02125* (**2020**).
467. Li, Wenda, et al. "A novel processing methodology for traffic-speed road surveys using point lasers." *IEEE Transactions on Intelligent Transportation Systems* (**2019**).
468. Li, Yuanlu, Kun Li, and Qiyu Lu. "Applying segmentation and classification to improve performance of smoothing." *Digital Signal Processing* 109: 102913, **2021**.
469. Rutt, Daryl, et al. "Importance of Accurate and Detailed Data Processing of Laser Mapping in Coke Drum." *Pressure Vessels and Piping Conference*. Vol. 58943. American Society of Mechanical Engineers, **2019**.
470. Al-Mbaideen, Amneh A. "Application of moving average filter for the quantitative analysis of the NIR spectra." *Journal of Analytical Chemistry* 74.7 (**2019**): 686-692.
471. Vitali, L., et al. "Infrared image processing for local convective heat transfer measurements in rib-enhanced channels." *Journal of Physics: Conference Series*. Vol. 1599. No. 1. IOP Publishing, **2020**.
472. Vergel, Ángelo Joseph Soto, Luis Enrique Mendoza, and Byron Medina Delgado. "Analysis of energy and major components in chromatographic signals for the diagnosis of prostate cancer." *Respuestas* 24.1 (**2019**): 76-85.
473. Zhang, Genwei, et al. "Multiscale orthogonal matching pursuit algorithm combined with peak model for interpreting ion mobility spectra and achieving quantitative analysis." *Analytica Chimica Acta* (**2020**).
474. Zhang, Lei, et al. "WiDIGR: Direction-Independent Gait Recognition System Using Commercial Wi-Fi Devices." *IEEE Internet of Things Journal* 7.2 (**2019**): 1178-1191.

475. Wiegand, Patrick. "Raman signal position correction using relative integration parameters." U.S. Patent No. 10,627,289. 21 Apr. **2020**.
476. Botezatu, Irina V., et al. "Asymmetric mutant-enriched polymerase chain reaction and quantitative DNA melting analysis of KRAS mutation in colorectal cancer." *Analytical Biochemistry* 590 (**2020**): 113517.
477. Joubaud, Thomas, and Grégory Pallone. "Electroacoustic method for the calibration of a heterogeneous distributed speaker system." **2020 28th European Signal Processing Conference (EUSIPCO)**. IEEE.
478. Xiang, YuChen, et al. "Background-free fibre optic Brillouin probe for remote mapping of micromechanics." *arXiv preprint arXiv:2005.12266* (**2020**).
479. Huang, Ronggang, et al. "Optical frequency and phase information-based fusion approach for image rotation symmetry detection." *Optics Express* 28.13 (**2020**): 18577-18595.
480. Sosin, M., et al. "Impact of vibrations and reflector movements on the measurement uncertainty of Fourier-based frequency sweeping interferometry." *Photonic Instrumentation Engineering VII*. Vol. 11287. International Society for Optics and Photonics, **2020**.
481. Chakraborty, Saikat, and Anup Nandy. "Automatic Diagnosis of Cerebral Palsy Gait Using Computational Intelligence Techniques: A Low-Cost Multi-Sensor Approach." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 28.11 (**2020**): 2488-2496.
482. Kim, Najin, et al. "Hygroscopicity of urban aerosols and its link to size-resolved chemical composition during spring and summer in Seoul, Korea." *Atmospheric Chemistry and Physics* 20.19 (**2020**): 11245-11262.
483. Ma, Te, et al. "Rapid and nondestructive evaluation of hygroscopic behavior changes of thermally modified softwood and hardwood samples using near-infrared hyperspectral imaging (NIR-HSI)." *Holzforschung* 1.ahead-of-print (**2020**).
484. Laskaris, Nick, et al. "EVIDENCE OF AU-HG GILDING PROCESS IN POST BYZANTINE ECCLESIASTICAL SILVERWARES (CHALICES) OF EASTERN THESSALY BY PXRF." *Mediterranean Archaeology & Archaeometry* 13.1 (**2020**).
485. Obaydo, Reem H., and Amir Alhaj Sakur. "Spectrophotometric strategies for the analysis of binary combinations with minor component based on isoabsorptive point's leveling effect: An application on ciprofloxacin and fluocinolone acetonide in their recently delivered co-formulation." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 219 (**2019**): 186-194.
486. Ma, Liya, and Peter Schegner. "State duration based event detection for domestic power disaggregation." *2019 IEEE Milan PowerTech*. IEEE, **2019**.
487. Коломиец, О. О., and С. В. Глушен. "Суточный ритм роста листьев и пролиферации клеток у перца стручкового (*Capsicum annuum L.*)."*Известия Национальной академии наук Беларусь. Серия биологических наук* 64.4 (**2019**): 448-455.
488. Reynes, Julien, Pierre Lanari, and Jörg Hermann. "A mapping approach for the investigation of Ti–OH relationships in metamorphic garnet." *Contributions to Mineralogy and Petrology* 175 (**2020**): 1-17.
489. Shumeyko, Christopher M., et al. "Tunable mechanical behavior of graphene nanoribbon-metal composites fabricated through an electrocharge-assisted process." *Materials Science and Engineering: A* 800 (**2020**): 140289.
490. Psyrras, N., et al. "Physical Modeling of the Seismic Response of Gas Pipelines in Laterally Inhomogeneous Soil." *Journal of Geotechnical and Geoenvironmental Engineering* 146.5 (**2020**): 04020031.
491. Chen, Hong-Jia, et al. "Self-potential ambient noise and spectral relationship with urbanization, seismicity, and strain rate revealed via the Taiwan Geoelectric Monitoring Network." *Journal of Geophysical Research: Solid Earth* 125.1 (**2020**): e2019JB018196.
492. Mustafa, M. A., et al. "Amplification and structure of streamwise-velocity fluctuations in compression-corner shock-wave/turbulent boundary-layer interactions." *Journal of Fluid Mechanics* 863 (**2019**): 1091-1122.
493. Mekonnen, Alemu, et al. "Improved Biomass Cookstove Use in the Longer Run: Results from a Field Experiment in Rural Ethiopia." *World Bank Policy Research Working Paper* 9272 (**2020**).
494. Bradshaw, Peter R., et al. "Kinetic modelling of acyl glucuronide and glucoside reactivity and development of structure–property relationships." *Organic & Biomolecular Chemistry* 18.7 (**2020**): 1389-1401.

505. Bluffstone, Randall, et al. "Does providing improved biomass cooking stoves free-of-charge reduce regular usage? Do use incentives promote habits?." *LAND ECONOMICS* (2020).
506. Guo, Qimei, et al. "Mediterranean Outflow Water dynamics across the middle Pleistocene transition based on a 1.3 million-year benthic foraminiferal record off the Portuguese margin." *Quaternary Science Reviews* 247 (2020): 106567.
507. Mustafa, Muhammad A., David Shekhtman, and Nick J. Parziale. "Single-Laser Krypton Tagging Velocimetry (KTV) Investigation of Air and N₂ Boundary-Layer Flows Over a Hollow Cylinder in the Stevens Shock Tube." *AIAA Scitech 2019 Forum*. 2019.
508. Wang, M., et al. "Evolution of dislocation and twin densities in a Mg alloy at quasi-static and high strain rates." *Acta Materialia* 201 (2020): 102-113.
509. AlOmar, AbdulAzeez S. "Accurate Chebyshev Approximations for the Width of the Voigt Profile, Differential Peaks, and Deconvolution of the Lorentzian Width." *Optik* 225: 165533, 2021.
510. Hoyer, Jorgen, et al. "Mapping calcium dynamics in a developing tubular structure." *bioRxiv* (2020).
511. Hansen, Lars N., et al. "Low-Temperature Plasticity in Olivine: Grain Size, Strain Hardening, and the Strength of the Lithosphere." *Journal of Geophysical Research. Solid Earth* 124.6 (2019).
512. Du, Siqi, et al. "Complete identification of all 20 relevant epimeric peptides in β-amyloid: a new HPLC-MS based analytical strategy for Alzheimer's research." *Chemical Communications* 56.10 (2020): 1537-1540.
513. Hebden, Jeremy C. "Exploring the feasibility of wavelength modulated near-infrared spectroscopy." *Journal of Biomedical Optics* 25.11 (2020): 110501.
514. Aikin, Timothy J., et al. "MAPK activity dynamics regulate non-cell autonomous effects of oncogene expression." *Elife* 9 (2020): e60541.
515. Pepermans, Vincent, et al. "Column-in-Valve Designs to Minimize Extra-Column Volumes." *Journal of Chromatography A* (2020): 461779.
516. Chua, Emily J., et al. "A mass spectrometer-based pore-water sampling system for sandy sediments." *Limnology and Oceanography: Methods* 19.11 (2021): 769-784.
517. Sanchini, Andrea, and Martin Grosjean. "Quantification of chlorophyll a, chlorophyll b and pheopigments a in lake sediments through deconvolution of bulk UV-VIS absorption spectra." *Journal of paleolimnology* 64 (2020): 243-256.
518. Yuen, Clement, et al. "Towards malaria field diagnosis based on surface-enhanced Raman scattering with on-chip sample preparation and near-analyte nanoparticle synthesis." *Sensors and Actuators B: Chemical* (2021): 130162.
519. Pal, Arpan, et al. "Instant Adaptive Learning: An Adaptive Filter Based Fast Learning Model Construction for Sensor Signal Time Series Classification on Edge Devices." *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.
520. Razzini, Jhonathan, Marcelo Vandresen, and Luciano Amaury dos Santos. *Comparative of the Mathematical Smoothing Model for Inertial Dynamometer Software*. No. 2020-36-0151. SAE Technical Paper, 2021.
521. Wang, Yaheng, et al. "High-speed 600-GHz-Band Terahertz Imaging System Using Polygon Mirror." *2021 International Topical Meeting on Microwave Photonics (MWP)*. IEEE, 2021.
522. Shanmugarajah, Sujeevan, Janani Tharmaseelan, and Luckman Sivagnanam. "AI Approach In Monitoring The Physical And Psychological State Of Car Drivers And Remedial Action For Safe Driving." *2020 2nd International Conference on Advancements in Computing (ICAC)*. Vol. 1. IEEE, 2020.
523. Likhachev, D. V. "On the optimization of knot allocation for B-spline parameterization of the dielectric function in spectroscopic ellipsometry data analysis." *Journal of Applied Physics* 129.3 (2021): 034903.
524. Li, Yuanlu, Kun Li, and Qiyu Lu. "Applying segmentation and classification to improve performance of smoothing." *Digital Signal Processing* 109 (2021): 102913.

525. Dela Cruz, Jennifer, et al. "Deriving Heart Rate and Respiratory Rate from ECG Using Wavelet Transform." *2021 11th International Conference on Biomedical Engineering and Technology*. **2021**.
526. Zhang, Genwei, et al. "Coulombic effects on resolution of ion mobility spectrometry and its application in online qualitative analysis." *Analytica Chimica Acta* 1183 (**2021**): 338969.
527. Ramakrishnan, Saminathan, et al. "Dependence of phase transition uniformity on crystal sizes characterized using birefringence." *Structural Dynamics* 8.3 (**2021**): 034301.
528. Gupta, Smith. "Clustering based method for finding spikes in insect neurons." *arXiv preprint arXiv:2111.11152* (2021).
529. Kocevska, Stefani, et al. "Spectroscopic Quantification of Target Species in a Complex Mixture Using Blind Source Separation and Partial Least-Squares Regression: A Case Study on Hanford Waste." *Industrial & Engineering Chemistry Research* 60.27 (**2021**): 9885-9896.
530. McAvan, Bethan S., et al. "Raman Spectroscopy to Monitor Post-translational Modifications and Degradation in mAb Therapeutics." <http://www.biospec.net/pubs/pdfs/McAvan-AnalChem2020SI.pdf>
531. Torres-Contreras, Ignacio, et al. "Effects of Phase Shift Errors in Recurrence Plot for Rotating Machinery Fault Diagnosis." *Applied Sciences* 11.2 (**2021**): 873.
532. Renney, Harri, Benedict R. Gaster, and Thomas J. Mitchell. "There and Back Again: The Practicality of GPU Accelerated Digital Audio." [PDF link](#)
533. Feukeu, Etienne Alain, and Simon Winberg. "Photoplethysmography Heart Rate Monitoring: State-of-the-Art Design." *International Journal of E-Health and Medical Communications (IJEHMC)* 12.3 (**2021**): 17-37.
534. Munier, Pierre, et al. "Assembly of cellulose nanocrystals and clay nanoplatelets studied by time-resolved X-ray scattering." *Soft Matter* 17.23 (**2021**): 5747-5755.
535. Izima, Obinna, Ruairí de Fréin, and Mark Davis. "Predicting quality of delivery metrics for adaptive video codec sessions." *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*. IEEE, **2020**.
536. Sun, Yuanchang, and Jack Xin. "Lorentzian peak sharpening and sparse blind source separation for NMR spectroscopy." *Signal, Image and Video Processing* (**2021**): 1-9.
537. Tan, Jiajie, et al. "Implicit Multimodal Crowdsourcing for Joint RF and Geomagnetic Fingerprinting." *IEEE Transactions on Mobile Computing* (**2021**).
538. Guo, Zhenyu, et al. "Anthropometric-based clustering of pinnae and its application in personalizing HRTFs." *International Journal of Industrial Ergonomics* 81 (**2021**): 103076.
539. Xu, Susan Shuhong, et al. "Comparison of ISO work of breathing and NIOSH breathing resistance measurements for air-purifying respirators." *Journal of occupational and environmental hygiene* 18.8 (**2021**): 369-377.
540. Hovareshti, Pedram, et al. "VestAid: A Tablet-Based Technology for Objective Exercise Monitoring in Vestibular Rehabilitation." *Sensors* 21.24 (**2021**): 8388.
541. Ramakrishnan, Saminathan, et al. "A combined approach to characterize ligand-induced solid–solid phase transitions in biomacromolecular crystals." *Journal of Applied Crystallography* 54.3 (**2021**).
542. Dioumaev, Andrei K., et al. "Determining material parameters with resonant acoustic spectroscopy." *Applied Optical Metrology IV*. Vol. 11817. International Society for Optics and Photonics, **2021**.
543. Lu, Min, et al. "Accurate construction of 3-D numerical breast models with anatomical information through MRI scans." *Computers in Biology and Medicine* 130 (**2021**): 104205.

544. Pasquali, Mattia, et al. "Nanomechanical Characterization of Organic Surface Passivation Films on 50 nm Patterns during Area-Selective Deposition." *ACS Applied Electronic Materials* (2021).
545. Kalambet, Yuri. "Data acquisition and integration." *Gas Chromatography*. Elsevier, 2021. 505-524.
546. Ramakrishnan, Saminathan, et al. "Synchronous RNA conformational changes trigger ordered phase transitions in crystals." *Nature communications* 12.1 (2021): 1-10.
547. Ke, Jie, et al. "Self-Optimization of Continuous Flow Electrochemical Synthesis Using Fourier Transform Infrared and Gas Chromatography." *Applied Spectroscopy* (2021): 00037028211059848.
548. Kim, Namgyun, Jinwoo Kim, and Changbum R. Ahn. "Predicting workers' inattentiveness to struck-by hazards by monitoring biosignals during a construction task: A virtual reality experiment." *Advanced Engineering Informatics* 49 (2021): 101359.
549. Jonker, D., et al. "A wafer-scale fabrication method for three-dimensional plasmonic hollow nanopillars." *Nanoscale advances* 3.17 (2021): 4926-4939.
550. Shekhtman, D., et al. "Freestream velocity-profile measurement in a large-scale, high-enthalpy reflected-shock tunnel." *Experiments in Fluids* 62.5 (2021): 1-13.
551. Rezaee, Mohammad, Iulian Iordachita, and John W. Wong. "Ultrahigh dose-rate (FLASH) x-ray irradiator for pre-clinical laboratory research." *Physics in Medicine & Biology* 66.9 (2021): 095006.
552. Chang, Ji Woong, Antonios Armaou, and Robert M. Rioux. "Continuous Injection Isothermal Titration Calorimetry for In Situ Evaluation of Thermodynamic Binding Properties of Ligand-Receptor Binding Models." *The Journal of Physical Chemistry B* 125.29 (2021): 8075-8087.
553. Bärmann, Peer, et al. "Solvent Co-intercalation into Few-layered Ti₃C₂T x MXenes in Lithium Ion Batteries Induced by Acidic or Basic Post-treatment." *ACS nano* 15.2 (2021): 3295-3308.
554. Cuyt, Annie, and Wen-shin Lee. "Parametric spectral analysis: scale and shift." *arXiv preprint arXiv:2008.02125* (2020).
555. Shekhtman, David, Nick J. Parziale, and Muhammad A. Mustafa. "Excitation Line Optimization for Krypton Tagging Velocimetry and Planar Laser-Induced Fluorescence in 200-220 nm Range." *AIAA Scitech 2021 Forum*. 2021.
556. Brasiliense, Vitor, et al. "Nanopipette-based electrochemical SERS platforms: Using electrodeposition to produce versatile and adaptable plasmonic substrates." *Journal of Raman Spectroscopy* 52.2 (2021): 339-347.
557. Qian, Yiwen, et al. "Crystallization of nanoparticles induced by precipitation of trace polymeric additives." *Nature communications* 12.1 (2021): 1-8.
558. Raman, Narayanan, et al. "GaPt Supported Catalytically Active Liquid Metal Solution Catalysis for Propane Dehydrogenation-Support Influence and Coking Studies." *ACS catalysis* 11.21 (2021): 13423-13433.
559. Phounglamcheik, Aekjuthon, et al. "CO₂ Gasification Reactivity of Char from High-Ash Biomass." *ACS Omega* (2021).
560. Leaston, Joshua, et al. "Neurovascular imaging with QUTE-CE MRI in APOE4 rats reveals early vascular abnormalities." *PLoS One* 16.8 (2021): e0256749.
561. Asmala, Eero, Philippe Massicotte, and Jacob Carstensen. "Identification of dissolved organic matter size components in freshwater and marine environments." *Limnology and Oceanography* 66.4 (2021): 1381-1393.
562. Hu, Jennifer F., et al. "Quantitative mapping of the cellular small RNA landscape with AQRNA-seq."

563. Tomlinson, Lauren J., et al. "Exploring the conformational landscape and stability of Aurora A using ion-mobility mass spectrometry and molecular modelling." *bioRxiv* (2021).
564. Zhang, Qin, and Benjamin M. Tutolo. "Geochemical evaluation of glauconite carbonation during sedimentary diagenesis." *Geochimica et Cosmochimica Acta* 306 (2021): 226-244.
565. Parigger, Christian G., Christopher M. Helstern, and Ghaneshwar Gautam. "Hypersonic imaging and emission spectroscopy of hydrogen and cyanide following laser-induced optical breakdown." *Symmetry* 12.12 (2020): 2116.
566. Parigger, Christian G. "Laser-plasma and stellar astrophysics spectroscopy." *Contrib. Astron. Obs. Skalnaté Pleso* 50 (2020): 15-31.
567. Wolf, Moritz, et al. "Coke formation during propane dehydrogenation over Ga– Rh supported catalytically active liquid metal solutions." *ChemCatChem* 12.4 (2020): 1085.
568. Chen, Weiqi, et al. "Experimental data-driven reaction network identification and uncertainty quantification of CO₂-assisted ethane dehydrogenation over Ga₂O₃/Al₂O₃." *Chemical Engineering Science* 237 (2021): 116534.
569. Roy, Sujan Kumar, and Kuldip K. Paliwal. "A noise PSD estimation algorithm using derivative-based high-pass filter in non-stationary noise conditions." *EURASIP Journal on Audio, Speech, and Music Processing* 2021.1 (2021): 1-18.
570. Fannes Claverol, Jean Paul. *Feasibility study of an ADAS using RADAR for In-Cabin pilot health parameters monitoring*. MS thesis. Universitat Politècnica de Catalunya, 2021.
571. Thu, Nguyen Anh. "Quantification of acetaminophen, caffeine and ibuprofen in solid dosage forms by uv spectroscopy coupled with multivariate analysis." *Asian Journal of Pharmaceutical Analysis* 11.2 (2021): 127-132.
572. Mutebi, John-Paul, et al. "Diel Activity Patterns of Two Distinct Populations of Aedes Aegypti in Miami, FL and Brownsville, TX." (2021).
573. Hobson, Eric C., et al. "Resonant acoustic rheometry for non-contact characterization of viscoelastic biomaterials." *Biomaterials* 269 (2021): 120676.
574. Kurtila, Moona, et al. "Site-by-site tracking of signal transduction in an azidophenylalanine-labeled bacteriophytochrome with step-scan FTIR spectroscopy." *Physical Chemistry Chemical Physics* 23.9 (2021): 5615-5628.
575. Wang, Hao, et al. "Rapid SERS quantification of trace fentanyl laced in recreational drugs with a portable Raman module." *Analytical chemistry* 93.27 (2021): 9373-9382.
576. Smith, Alexander J., et al. "Expanded in situ aging indicators for lithium-ion batteries with a blended NMC-LMO electrode cycled at sub-ambient temperature." *Journal of The Electrochemical Society* 168.11 (2021): 110530.
577. Jenkins, Lauren M., et al. "Quantification of Acyl-Acyl Carrier Proteins for Fatty Acid Synthesis Using LC-MS/MS." *Plant Lipids*. Humana, New York, NY, 2021. 219-247.
578. Heck, Anisa, et al. "Volume Fraction Measurement of Soft (Dairy) Microgels by Standard Addition and Static Light Scattering." *Food Biophysics* 16.2 (2021): 237-253.
579. Teixeira, Paulo Sérgio, et Al.. "Avaliação das respostas em frequências naturais de um violão pelo método de excitação por impulso e deconvolução de sinais (Evaluation of the natural frequency responses of a guitar by the method of impulse excitation and signal deconvolution)." *Research, Society and Development* 10.1 (2021)
580. Burg, David, and Jesse H. Ausubel. "Moore's Law revisited through Intel chip density." *PloS one* 16.8

581. Poorna, S. S., et al. "A transfer learning approach for drowsiness detection from EEG signals." *Innovations in Computational Intelligence and Computer Vision*. Springer, Singapore, 2021. 369-375.
582. Yang, Guofeng, et al. "Injection profile surveillance using impulse oxygen activation logging based on optimization theory." *Journal of Petroleum Science and Engineering* 196 (2021): 107701.
583. Navarro-Huerta, Jose Antonio, et al. "Ultra-short ion-exchange columns for fast charge variants analysis of therapeutic proteins." *Journal of Chromatography A* 1657 (2021): 462568.
584. Wahab, M. Farooq, Daipayan Roy, and Daniel W. Armstrong. "The theory and practice of ultrafast liquid chromatography: A tutorial." *Analytica Chimica Acta* 1151 (2021): 238170.
585. Samokhvalov, Alexander. "Understanding the structure, bonding and reactions of nanocrystalline semiconductors: a novel high-resolution instrumental method of solid-state synchronous luminescence spectroscopy." *Physical Chemistry Chemical Physics* 23.12 (2021): 7022-7036.
586. Ghadimloozaadeh, Shaghayegh, Mahmoud Reza Sohrabi, and Hassan Kabiri Fard. "Development of rapid and simple spectrophotometric method for the simultaneous determination of anti-parkinson drugs in combined dosage form using continuous wavelet transform and radial basis function neural network." *Optik* 242 (2021): 167088.
587. de Falco, Giacomo, et al. "Proposing an unbiased oxygen reduction reaction onset potential determination by using a Savitzky-Golay differentiation procedure." *Journal of Colloid and Interface Science* 586 (2021): 597-600.
588. Readel, Elizabeth R., Michael Wey, and Daniel W. Armstrong. "Rapid and selective separation of amyloid beta from its stereoisomeric point mutations implicated in neurodegenerative Alzheimer's disease." *Analytica Chimica Acta* 1163 (2021): 338506.
589. Tatarinov, Danila A., Sofia R. Sokolnikova, and Natalia A. Myslitskaya. "Applying of Chitosan-TiO₂ Nanocomposites for Photocatalytic Degradation of Anthracene and Pyrene." *Journal of Biomedical Photonics & Engineering* 7.1 (2021): 010301.
590. Kim, Min-Yeong, et al. "Highly stable potentiometric sensor with reduced graphene oxide aerogel as a solid contact for detection of nitrate and calcium ions." *Journal of Electroanalytical Chemistry* 897 (2021): 115553.
591. Karongo, Ryan, et al. "Rapid enantioselective amino acid analysis by ultra-high performance liquid chromatography-mass spectrometry combining 6-aminoquinolyl-N-hydroxysuccinimidyl carbamate derivatization with core-shell quinine carbamate anion exchanger separation." *Journal of Chromatography Open* 1 (2021): 100004.
592. Gritti, Fabrice, and Farooq Wahab. "Extraction of intrinsic column peak profiles of narrow-bore and microbore columns by peak deconvolution methods." *Analytica Chimica Acta* 1180 (2021): 338851.
593. Russell, B., et al. "Investigating the potential of tandem inductively coupled plasma mass spectrometry (ICP-MS/MS) for Ca determination in concrete." *Journal of Analytical Atomic Spectrometry* 36.4 (2021): 845-855.
594. Tanács, Dániel, et al. "Enantioseparation of β2-amino acids by liquid chromatography using core-shell chiral stationary phases based on teicoplanin and teicoplanin aglycone." *Journal of Chromatography A* 1653 (2021): 462383.
595. Ewusi-Annan, Ebo, and Noureddine Melikechi. "Unsupervised fitting of emission lines generated from laser-induced breakdown spectroscopy." *Spectrochimica Acta Part B: Atomic Spectroscopy* 177 (2021): 106109.
596. Niezen, Leon E., Peter J. Schoenmakers, and Bob WJ Pirok. "Critical comparison of background

correction algorithms used in chromatography." *Analytica Chimica Acta* 1201 (2022): 339605.

597. Kendir, Gülsen, Ayşegül Köroğlu, and Erdal Dinç. "SIMULTANEOUS SPECTROPHOTOMETRIC QUANTITATION OF RUTIN AND CHLOROGENIC ACID IN LEAVES OF *Ribes uva-crispa* L. BY ONE-DIMENSIONAL CONTINUOUS WAVELET TRANSFORMS." *Journal of the Chilean Chemical Society* 66.1 (2021): 5041-5046.

598. Nosal, Daniel G., Douglas L. Feinstein, and Richard B. van Breemen. "Chiral liquid chromatography-tandem mass spectrometry analysis of superwarfarin rodenticide stereoisomers—Bromadiolone, difenacoum and brodifacoum—in human plasma." *Journal of Chromatography B* 1165 (2021): 122529.

599. Wang, Zhengshuo, et al. "Recent progress in organic color-tunable phosphorescent materials." *Journal of Materials Science & Technology* 101 (2022): 264-284.

600. Ksert, Alexander, et al. "Deep convolutional autoencoder for the simultaneous removal of baseline noise and baseline drift in chromatograms." *Journal of Chromatography A* 1646 (2021): 462093.

601. Zaynidinov, H. N., et al. "Algorithms and Service for Digital Processing of Two-Dimensional Geophysical Fields Using Octave Method." *International Conference on Intelligent Human Computer Interaction*. Springer, Cham, 2021.

602. Singh, Ritu, Navin Rajpal, and Rajesh Mehta. "Non-invasive Single Channel integration model for fetal ECG extraction and sustainable fetal healthcare using wavelet framework." *Multimedia Tools and Applications* (2022): 1-27.

603. Luo, Canhuang, et al. "Nudging the N170 forward with prior stimulation—Bridging the gap between N170 and recognition potential." *Human brain mapping* 43.4 (2022): 1214-1230.

604. Kurapati, Hemasai. "How does the period of oscillation of a cantilever relate with its mass?." (2022). [PDF file](#)

605. Edelman, Joel. "Overtone Invariants are the Framework for Musical Consonance; An Illustrated Overview." Humanities Commons, [PDF file](#). (2022).

606. Hsu, W. Y., Y. W. Cheng, and C. B. Tsai. "An Effective Algorithm to Analyze the Optokinetic Nystagmus Waveforms from a Low-Cost Eye Tracker. *Healthcare* 2022, 10, 1281." (2022).

607. Ghosh, Koushik. "Smart filter and smoothing: A new approach of data denoising." Noise Filtering for Big Data Analytics 12 (2022): 139.

608. Maggiore, F., et al. "Process optimization by real time analysis of liquids' composition in Metal & Mining." *TOS forum*. No. 1. IM Publications Open, 2022.

609. Mehic, Amela. "Development of a computational method for determining gamma energy escape from calorimeters at CLAB." (2022).

610. Xue, Qingsheng, et al. "Detection of microplastics based on spatial heterodyne Raman spectroscopy." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 283 (2022): 121712.

611. Zhao, Nie, et al. "SGTools: a suite of tools for processing and analyzing large data sets from in situ X-ray scattering experiments." *Journal of Applied Crystallography* 55.1 (2022).

612. Jeong, Mok Kun, Min Joo Choi, and Sung Jae Kwon. "High-spatial-resolution, instantaneous passive cavitation imaging with temporal resolution in histotripsy: a simulation study." *Ultrasonography* 41.3 (2022): 566.

613. Li, X., et al. "Process-Oriented Estimation of Chlorophyll-a Vertical Profile in the Mediterranean Sea Using MODIS and Oceanographic Float Products. Front." *Mar. Sci* 9 (2022): 933680.

614. Riddick, Stuart N., et al. "Estimating Regional Methane Emission Factors from Energy and Agricultural Sector Sources Using a Portable Measurement System: Case Study of the Denver–Julesburg Basin." *Sensors* 22.19 (2022): 7410.

615. Yang, Fanlin, et al. "An airborne LiDAR bathymetric waveform decomposition method in very shallow water: A case study around Yuanzhi Island in the South China Sea." *International Journal of Applied Earth Observation and Geoinformation* 109 (2022): 102788.
616. Utt, Kainen L., et al. "Spatially-Resolved Mid-Infrared Spectral Evidence of Space Weathering." *Authorea Preprints* (2022).
617. Termsuk, C., S. J. Sweeney, and C. Shenton-Taylor. "Thermoluminescence glow curve study of beta irradiated germanium doped core fibre with different dopant concentrations." *Radiation Physics and Chemistry* 193 (2022): 109974.
618. Alomar, Abdulazeez S. "Impact of Faddeeva–Voigt broadening on line-shape analysis at critical points of dielectric functions." *AIP Advances* 12.6 (2022): 065127.
619. Likhachev, D. V. "A method of optimal dielectric function modeling by B-splines for spectroscopic ellipsometry analysis.", researchgate.net, 07/31/2022
620. Li, Dongmei, Zhiwei Zhu, and Da-Wen Sun. "Visualization and quantification of content and hydrogen bonding state of water in apple and potato cells by confocal Raman microscopy: A comparison study." *Food Chemistry* 385 (2022): 132679.
621. Zhong, Yu, et al. "Summary Report of CALPHAD GLOBAL, 2021." Available at SSRN 4229474.
622. Xuan, Doan Thanh, and Vu Dang Hoang. "Application of Fourier transform-based algorithms to resolve spectral overlapping for UV spectrophotometric co-assay of spiramycin and metronidazole in tablets." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 277 (2022): 121253.
624. Lâm, Vũ Tùng, et al. "Simultaneous quantification and solubility test of tenofovir disoproxil fumarate and emtricitabine in tablets by ultraviolet spectrum derivation using fast Fourier transform algorithm." Link: hup.edu.vn
625. Pasquali, Mattia, et al. "Understanding Selectivity Loss Mechanisms in Selective Material Deposition by Area Deactivation on 10 nm Cu/SiO₂ Patterns." *ACS Applied Electronic Materials* 4.4 (2022): 1703-1714.
625. Zlabinger, Johannes. *Development of a peak deconvolution software for X-ray fluorescence spectra.* Diss. Wien, 2022.
626. García-Figueroa, Jesús M. *Depiction of the Ambipolar-State of the Gas-Substrate Interphase of the Electron Cyclotron Resonance–Microwave–Chemical Vapor Deposition (Ecr-Mw-Cvd) Method and Its Influence over the Properties of Vapor-Deposited Hydrocarbon Films.* Diss. University of Rochester, 2022.
627. Schulze, H. Georg, et al. "Critical Evaluation of Spectral Resolution Enhancement Methods for Raman Hyperspectra." *Applied spectroscopy* 76.1 (2022): 61-80.
628. Danilenko, S. D. "Research of methods of digital-analog conversion for the implementation of mechanical co-living for the support of the software and hardware complex." (2022).
629. Vasquez, Krystal TonyBeth. *Isomer separation of multifunctional atmospheric compounds using gas chromatography and chemical ionization mass spectrometry.* Diss. California Institute of Technology, 2022.
630. Hesgrove, Cherie S., et al. "Tardigrade CAHS Proteins Act as Molecular Swiss Army Knives to Mediate Desiccation Tolerance through Multiple Mechanisms." *bioRxiv* (2021): 2021-08.
631. Shareef, Abdulwahhab F., and Riyadh Z. Mahmoud. "Study on Turning Arabic Text into Spoken Words." *AL-Rafidain Journal of Computer Sciences and Mathematics* 15.1 (2021): 197-209.
632. Teixeira, Paulo Sérgio, Alexandre Furtado Ferreira, and José Flávio Silveira Feiteira. "Simulação matemática dos sinais sonoros do violão através da convolução." *Cadernos UniFOA* 16.46 (2021).

633. Moss, Frank R., et al. "Brominated Lipid Probes Expose Structural Asymmetries in Constricted Membranes." *bioRxiv* (2021).
634. Barmann, Peer, et al. "Solvent co-intercalation into few-layered Ti₃C₂T x MXenes in lithium ion batteries induced by acidic or basic post-treatment." *ACS nano* 15.2 (2021): 3295-3308.
635. Guo, Shanzeng, Salman Akhtar, and Anthony Mella. "A method for radar model identification using time-domain transient signals." *IEEE Transactions on Aerospace and Electronic Systems* 57.5 (2021): 3132-3149.
636. Dasappa, Shruthi. *Fundamental Studies of Gas-to-Particle Conversion for Nanoparticle Synthesis in Flames*. Diss. University of California, San Diego, 2021.
637. Raab, Meaghan T., Alexandra K. Prymek, and Andrea N. Giordano. "ESTIMATION OF THE GROUND AND EXCITED STATE DIPOLE MOMENTS FOR IBUPROFEN AND NAPROXEN SODIUM USING THE SOLVATOCHROMIC SHIFT METHOD." *Journal of Undergraduate Chemistry Research* 20.4 (2021): 68.
638. Hu, Jennifer F., et al. "Quantitative mapping of the cellular small RNA landscape with AQRNA-seq." *Nature biotechnology* 39.8 (2021): 978-988.
639. Lee, Sung Hoon, et al. "A molecular clock controls periodically driven cell migration in confined spaces." *Cell Systems* (2022).
640. Balewski, Zuzanna Z., Eric B. Knudsen, and Joni D. Wallis. "Fast and slow contributions to decision-making in corticostriatal circuits." *Neuron* (2022).
641. Firrone, Christian Maria, Antonio Giuseppe D'Ettore, and Matteo Nicita. "Extraction of relevant data from experimental test and matching with FEA." [PDF file link](#). 2022
642. Baroudi, Ahmad Jamal. "A Tool for Biometric Interpretation of Forensic STR DNA Profiles." (2022). [PDF file link](#).
643. Senecaute, Nicolas. Nouvelles approches de quantification des variations du protéome au niveau des protéines intactes: analyses expérimentales et computationnelles. Diss. Universite Paris Cité, 2022. [PDF file link](#).
644. Zhang, Linshan, et al. "Colorimetric detection for uranyl ions in water using vinylphosphonic acid functionalized gold nanoparticles based on smartphone." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 269 (2022): 120748.
645. Ji, Songbai, Shaoju Wu, and Wei Zhao. "Dynamic characteristics of impact-induced brain strain in the corpus callosum." *Brain Multiphysics* 3 (2022): 100046.
646. Sabr, Muhammad W., and Diyar S. Ali. "H-point standard addition method for simultaneous determination of phenylephrine hydrochloride, chlorpheniramine maleate, and paracetamol as a ternary mixture in pharmaceutical formulations." *Journal of the Indian Chemical Society* (2022): 100526.
647. Takihata, Yasuhiro, et al. "In vivo diffuse reflectance spectroscopic analysis of fatty liver with inflammation in mice." *Surgery Open Science* 6 (2021): 21-28.
648. Liang, Chen, et al. "Meter scale and sub-second resolution coherent Doppler wind lidar and hyperfine." *Optics Letters Vol. 47, 10* (2022)
649. Shang, Qiu Feng, et al. "Pink noise removal and spectral distortion correction based fiber Bragg grating demodulation algorithm." *Optics Express* 30.2 (2022): 1066-1080.
650. Liu, Luzheng, et al. "Selective Detection of Mixtures via a Single Nonselective Sensor—Making the Unworkable Sensor Workable by Machine Learning." *Advanced Intelligent Systems* (2022): 2200136.
651. Li, Ping, et al. "Discrimination of raw and sulfur-fumigated ginseng based on Fourier transform infrared spectroscopy coupled with chemometrics." *Microchemical Journal* 181 (2022): 107767.

652. Lum, Jordan S., et al. "In Situ Optical Detection for Ultrasonic Characterization of Materials in a Mach 10 Hypersonic Wind Tunnel." *Physical Review Applied* 18.4 (2022): 044062.
653. de CASTRO, PEDRO AA, et al. "Assessment of bone dose response using ATR-FTIR spectroscopy." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* (2022).
654. Zanotto, S., et al. "Optomechanical Modulation Spectroscopy of Bound States in the Continuum in a Dielectric Metasurface." *Physical Review Applied* 17.4 (2022): 044033.
655. Wang, Kristen. *Quantification of Resveratrol in Red Wine using Liquid Chromatography—Surface-Enhanced Raman Spectroscopy (LC-SERS)*. Diss. The Ohio State University, 2022.
656. Belal, Fathalla, et al. "Multi-spectroscopic, thermodynamic and molecular docking studies to investigate the interaction of eplerenone with human serum albumin." *Luminescence* 37.7 (2022): 1162-1173.
657. Kim, Myung-Hoon. "Advances in Derivative Voltammetry-A Search for Diagnostic Criteria of Several Electrochemical Reaction Mechanisms." *Analytical Chemistry-Advancement, Perspectives and Applications*. IntechOpen, 2021.
658. Zhou, Yongjie, et al. "An improved algorithm for peak detection based on weighted continuous wavelet transform." *IEEE Access* (2022).
659. Al-Samarrai, Mumin F. Hamad, Emad T. Hanon, and Ali I. Khaleel Khaleel. "Development of derivative of subtracting spectra method for the simultaneous determination of some decongestant drugs." *Samarra Journal of Pure and Applied Science* 3.3 (2021): 18-30.
660. Bishop, Logan DC, Anastasiia Misiura, and Christy F. Landes. "A new metric for relating macroscopic chromatograms to microscopic surface dynamics: the distribution function ratio (DFR)." *Analyst* 146.13 (2021): 4268-4279.
661. de Castro, Pedro Arthur Augusto, et al. "Assessment of bone dose response using ATR-FTIR spectroscopy: A potential method for biodosimetry." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 273 (2022): 120900.
662. Luo, Hao, et al. "In Situ Nanofluid Dispersion Monitoring by Liquid–Solid Triboelectric Nanogenerator Based on Tuning the Structure of the Electric Double Layer." *Advanced Functional Materials* (2022): 2200862.
663. Leal, Ana L., et al. "Data driven models exploring the combination of NIR and ¹H NMR spectroscopies in the determination of gasoline properties." *Microchemical Journal* 175 (2022): 107217.
664. Liang, Haibo, and Gang Liu. "Research on quantitative analysis method of PLS hydrocarbon gas infrared spectroscopy based on net signal analysis and density peak clustering." *Measurement* 188 (2022): 110392.
665. Hassanzadeh, Amirhossein. *On the Use of Imaging Spectroscopy from Unmanned Aerial Systems (UAS) to Model Yield and Assess Growth Stages of a Broadacre Crop*. Diss. Rochester Institute of Technology, 2022.
666. Rojas, Myriam, et al. "Kinetic Studies on Cocoa Roasting Including Volatile Characterization." *ACS Food Science & Technology* (2022).
667. Chen, Yi-Hsuan, et al. "Green Polymer Electrolytes Based on Polycaprolactones for Solid-State High-Voltage Lithium Metal Batteries." *Macromolecular Rapid Communications* 43.20 (2022): 2200335.
668. Manasi, Iva, et al. "Surfactant effects on the synthesis of porous cerium oxide from a type IV deep eutectic solvent." *Journal of Materials Chemistry A* 10.35 (2022): 18422-18430.
669. Jander, Julius H., et al. "Determination of hydrogen loading in the carrier system diphenylmethane/dicyclohexylmethane by depolarized Raman spectroscopy." *International Journal of*

670. Dunham-Cheatham, Sarrah M., Seth Lyman, and Mae Sexauer Gustin. "Comparison and calibration of methods for ambient reactive mercury quantification." *Science of The Total Environment* 856 (2023): 159219.
671. Martin, Jeremy E., et al. "The stability of dinosaur communities before the K–Pg boundary: A perspective from southern Alberta using calcium isotopes as a dietary proxy." *GSA Bulletin* (2022).
672. Scarpitti, Brian T., et al. "In Vitro Imaging of Lycopene Delivery to Prostate Cancer Cells." *Analytical Chemistry* 94.12 (2022): 5106-5112.
673. Schuster, Miriam. "Investigation of the semicrystalline structure of EVA and ionoplastic interlayers." *Characterization of Laminated Safety Glass Interlayers*. Springer Vieweg, Wiesbaden, 2023. 117-162.
674. Cuss, Chad W., and Celine Gueguen. "The contribution of endmembers to mixtures of leaf leachates and riverine DOM can be determined by measuring their size and fluorescence properties." *Frontiers in Environmental Chemistry* (2022): 14.
675. Liu, Donghong, and Chuanjiang He. "Peak-aware guided filtering for spectrum signal denoising." *Chemometrics and Intelligent Laboratory Systems* 222 (2022): 104508.
676. Gupta, S., Singh, A., Sharma, A., & Tripathy, R. K. (2022). Higher Order Derivative-Based Integrated Model for Cuff-Less Blood Pressure Estimation and Stratification Using PPG Signals. *IEEE Sensors Journal*, 22(22), 22030-22039.
677. Hua, Zhen-Ming, et al. "Large hardening response mediated by room-temperature dynamic solute clustering behavior in a dilute Mg-Zn-Ca-Sn-Mn alloy." *Acta Materialia* 240 (2022): 118308.
678. Bhalode, Pooja, et al. "Statistical Data Pre-Treatment for Residence Time Distribution Studies in Pharmaceutical Manufacturing." Available at SSRN 4249747 (2022).
679. Kukk, Anatoly Fedorov, Elias Blumenröther, and Bernhard Roth. "Self-made transparent optoacoustic detector for measurement of skin lesion thickness in vivo." *Biomedical Physics & Engineering Express* 8.3 (2022): 035029.
680. Wang, Yaheng, et al. "High-Speed 600 GHz-Band Terahertz Imaging Scanner System with Enhanced Focal Depth." *Photonics*. Vol. 9. No. 12. MDPI, 2022.
681. Likhachev, D. V. "Optimization of the dielectric-function modeling by B-splines in spectroscopic ellipsometry analysis: A hybrid approach." *Thin Solid Films* 762 (2022): 139545.
682. Ai, Xupeng, et al. "Phase Segmentation and Percentage Prediction of Trunk Movement Cycle." 2022 9th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob). IEEE, 2022.
683. Fan, Han, Erik Schaffernicht, and Achim J. Lilienthal. "Ensemble Learning-Based Approach for Gas Detection Using an Electronic Nose in Robotic Applications." *Frontiers in chemistry* 10 (2022).
684. Chuang, Chun-Hsiang, et al. "IC-U-Net: a U-Net-based denoising autoencoder using mixtures of independent components for automatic EEG artifact removal." *NeuroImage* 263 (2022): 119586.
685. Du, Mao, et al. "Combined application of online FIGAERO-CIMS and offline LC-Orbitrap mass spectrometry (MS) to characterize the chemical composition of secondary organic aerosol (SOA) in smog chamber studies." *Atmospheric Measurement Techniques* (2022): 4385-4406.
686. Schito, Andrea, et al. "Calibrating Carbonization Temperatures of Wood Fragments Embedded within Pyroclastic Density Currents through Raman Spectroscopy." *Minerals* 12.2 (2022): 203.
687. Sosa, Jonathan. Advanced Optical Diagnostics in Hypersonic Flows. NAVAL RESEARCH LAB WASHINGTON DC, 2022.
688. Li, Chunyan. "Global shockwaves of the Hunga Tonga-Hunga Ha'apai volcano eruption measured at

- ground stations." *Iscience* 25.11 (2022): 105356.
689. Mekonnen, Alemu, et al. "Do improved biomass cookstoves reduce fuelwood consumption and carbon emissions? Evidence from a field experiment in rural Ethiopia." *Ecological Economics* 198 (2022): 107467.
690. da Silva, Igor JG, Ivo M. Raimundo, and Boris Mizaikoff. "Analysis of sugars and sweeteners via terahertz time-domain spectroscopy." *Analytical Methods* 14.27 (2022): 2657-2664.
691. Lee, Sea On. RNA-TECHNOLOGIES TO FACILITATE THE SYNTHESIS AND PROCESSING OF BIOMATERIALS WITH REPETITIVE AMINO ACID SEQUENCES. Diss. Johns Hopkins University, 2022.
692. Pištek, Peter, Simon Harvan, and Michal Valicek. "Automated People Counting in Public Transport." *Industry 4.0 Challenges in Smart Cities* (2022): 75.
693. Ke, Jie, et al. "Self-Optimization of Continuous Flow Electrochemical Synthesis Using Fourier Transform Infrared Spectroscopy and Gas Chromatography." *Applied Spectroscopy* 76.1 (2022): 38-50.
694. Schultz, Jeremy F., et al. "Chemically imaging nanostructures formed by the covalent assembly of molecular building blocks on a surface with ultrahigh vacuum tip-enhanced Raman spectroscopy." *Journal of Physics: Condensed Matter* 34.20 (2022): 204008.
695. Valero, Maria, et al. "Vibration sensing-based human and infrastructure safety/health monitoring: A survey." *Digital Signal Processing* 125 (2022): 103572.
696. Oussalah, Abderrahim, David-Alexandre Trégouët, and Jean-Louis Guéant. "The Smoothing Method for DNA Methylome Analysis Identifies Highly Accurate Epigenomic Signatures in Epigenome-Wide Association Studies." (2022).
697. Chunyan Li, "Analysis of globally propagating spherical shockwaves from great volcano eruptions using barometric pressure data". STAR Protocols, Volume 4, Issue 2, 2023, ISSN 2666 1667, <https://doi.org/10.1016/j.xpro.2023.102207>. (<https://www.sciencedirect.com/science/article/pii/S266616672300165X>)
698. Zhao, Ziwen, et al. "Automated Processing of Nano-Impact Electrochemistry Signals Using Data-Driven Template Matching." *Electrochemical Society Meeting Abstracts* 244. No. 57. The Electrochemical Society, Inc., 2023.
699. Singh, Ritu, Navin Rajpal, and Rajesh Mehta. "Non-invasive Single Channel integration model for fetal ECG extraction and sustainable fetal healthcare using wavelet framework." *Multimedia Tools and Applications* 82.25 (2023): 39669-39695.
700. Kurapati, Hemasai. "How does the period of oscillation of a cantilever relate with its mass?." *Authorea Preprints* (2023).
701. Ma, Jielin, et al. "Effect of High Relative Humidity on the Thermal Aging of Composite Epoxy Insulation Materials." *2023 International Symposium on Electrical Insulating Materials (ISEIM)*. IEEE, 2023.
702. Moss III, Frank R., et al. "Brominated lipid probes expose structural asymmetries in constricted membranes." *Nature Structural & Molecular Biology* 30.2 (2023): 167-175.
703. Mouton, J. W. A., et al. "Development and validation of a bioanalytical assay for the measurement of total and unbound teicoplanin in human serum." *Journal of Antimicrobial Chemotherapy* 78.11 (2023): 2723-2730.
704. Durán, Antonio Andrés Figueroa, and Efren Fernandez Grande. "Navigable Reconstruction of Reverberant Sound Fields Using Distributed Microphone Arrays." *2nd International Conference on Immersive and 3D Audio*. IEEE, 2023.

705. Pace, DM Diaz. "τ-algorithm for gathering spectroscopic information by modeling emission line shapes: application to laser-induced plasmas." *JOSA B* 40.4 (2023): C1-C7.
706. Dummitt, Richard. *Chemical Effects in Protein Analysis: A Systematic Investigation of Amino Acid Spontaneous Raman and SERS Responses*. Diss. The Ohio State University, 2023.
707. Zhong, Yu, et al. "Summary report of CALPHAD GLOBAL, 2021." *Calphad* 81 (2023): 102527.
708. Lueder, Mona, Renée Tamblyn, and Jörg Hermann. "A framework for quantitative in situ evaluation of coupled substitutions between H⁺ and trace elements in natural rutile." *European Journal of Mineralogy* 35.2 (2023): 243-265.
709. Yang, Jerry A., et al. "Biaxial Tensile Strain Enhances Electron Mobility of Monolayer Transition Metal Dichalcogenides." *arXiv e-prints* (2023): arXiv-2309.
710. Mahajan, Vishal, et al. "Treating Noise and Anomalies in Vehicle Trajectories From an Experiment With a Swarm of Drones." *IEEE Transactions on Intelligent Transportation Systems* (2023).
711. Figueroa-Duran, Antonio, and Efren Fernandez-Grande. "Navigable Reconstruction of Reverberant Sound Fields Using Distributed Microphone Arrays." *2023 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*. IEEE, 2023.
712. Duarte, Gabriela, et al. "Raman microspectroscopy as a tool for identifying biosignatures in speleothem microbialites." *Annals XVI Latin American Congress on Organic Geochemistry*. 2023.
713. "表面增強拉曼散射探針及微流道系統 用於癌症細胞及外泌體表面生物標記物的多重檢測" (Surface-enhanced Raman scattering probes and microfluidics systems are used for multiplex detection of biomarkers on the surface of cancer cells and exosomes), PhD diss., National Central University, 2023.
714. Handlovic, Troy T., et al. "Automated Regularized Deconvolution for Eliminating Extra-Column Effects in Fast High-Efficiency Separations." *Analytical Chemistry* 95.29 (2023): 11028-11036.
715. Akinci, Tahir Cetin, et al. "High Order Spectral Analysis of Ferroresonance Phenomena in Electric Power Systems." *IEEE Access* (2023).
716. de Araújo Gomes, Adriano, et al. "Pattern recognition techniques in food quality and authenticity: A guide on how to process multivariate data in food analysis." *TrAC Trends in Analytical Chemistry* (2023): 117105.
717. Fekete, Szabolcs, and Davy Guillarme. "Ultra-short columns for the chromatographic analysis of large molecules." *Journal of Chromatography A* 1706 (2023): 464285.
418. Hollands, Patrick Mark. *Estimating the Effect of Pore Water and Ice on Martian Rock Analogues Using Ultrasonic and Resonant Ultrasound Spectroscopy Methods*. Diss. ResearchSpace@ Auckland, 2023.
- Sanchez-Martinez, Silvia, et al. "Labile assembly of a tardigrade protein induces biostasis." *bioRxiv* (2023): 2023-06.
719. Liu, Chongshan, et al. "Microbiome-induced Increases and Decreases in Bone Tissue Strength can be Initiated After Skeletal Maturity." *bioRxiv* (2024): 2024-01.
720. Cuyt, Annie, and Wen-shin Lee. "Multiscale matrix pencils for separable reconstruction problems." *Numerical Algorithms* 95.1 (2024): 31-72.
721. Allan, Matthew C., et al. "Baked sweetpotato textures and sweetness: An investigation into relationships between physicochemical and cooked attributes." *Food Chemistry: X* 21 (2024): 101072.
722. Yang, Guofeng, et al. "An improved radioactive tracer response analysis and injection profile quantification method based on intelligent algorithms." *Journal of Radioanalytical and Nuclear Chemistry* (2024): 1-15.

723. Dong, Wen, et al. "Study of UV–Vis molar absorptivity variation and quantitation of anthocyanins using molar relative response factor." *Food Chemistry* (2024): 138653.
724. Chen, Huo, et al. "Adaptive variational simulation for open quantum systems." *Quantum* 8 (2024): 1252.
725. Yu, Lihuan, et. al., Derivative Spectroscopy and its Application at Detecting the Weak Emission/Absorption Lines, *Research in Astronomy and Astrophysics* (2024).
726. R Senthilkuma, S.Shek Dhavud, RealTime AC Speech Denoising Analog Digital Filters, *National Conference on Emerging Trends and Technology*, ISBN: 978-81959812-3-6, June 2023
727. Matteo BruschiFederico GallinaFederico GallinaBarbara FreschBarbara Fresch, A Quantum Algorithm from Response Theory: Digital Quantum Simulation of Two-Dimensional Electronic Spectroscopy, *The Journal of Physical Chemistry Letters* 15(5):1484-1492. DOI: 10.1021/acs.jpclett.3c03499, January 2024
728. Fatma R.M. Abda Rehmann lati1, Salah I.S. Tnatin. Effect the window type in design of FIR Filter to reduce the noise in the signal, *ICRSE 2021: The 1st International Conference on Renewable and Sustainable Energy*, VOLUME 5 No(1) October 10-13, 2021
729. Krzysztof Sozanski, Overview of Signal Processing Problems in Power Electronic Control Circuits, *Energies* 16(12):4774, DOI: 10.3390/en16124774, June 2023
730. Dhanarasi Gowtham B.T.Krishna, Design and Applications of Digital Differentiators Using Model Order Reduction Techniques, *Journal of Propulsion Technology*, October 2023
731. Feng, Vincent. "Spectroscopy of Chamber Plasma and Plume of a 2.4-GHz Microwave Electrothermal Thruster." (2023).
732. Zhang, Le, et al. "Absolute thermometry of human brown adipose tissue by magnetic resonance with laser polarized ^{129}Xe ." *Communications Medicine* 3.1 (2023): 147.
733. Andreani, Muriel, et al. "The rocky road to organics needs drying." *Nature Communications* 14.1 (2023): 347.
734. Schorr, Hannah C., and Zachary D. Schultz. "Chemical conjugation to differentiate monosaccharides by Raman and surface enhanced Raman spectroscopy." *Analyst* 148.9 (2023): 2035-2044.
735. Chowdhury, Nildari Roy, et al. "Influence of state of charge window on the degradation of Tesla lithium-ion battery cells." *Journal of Energy Storage* 76 (2024): 110001.
736. Sanchez-Martinez, Silvia, et al. "Labile assembly of a tardigrade protein induces biostasis." *bioRxiv* (2023): 2023-06.
737. Lokini, Parneeth, et al. "Plasma Parameters of Laser Irradiated Hydrocarbon Droplets in Air." *AIAA SCITECH 2024 Forum*. 2024.
738. Zheng, Shuailin, et al. "Rapid detection of phosphorus in water using silicon attenuated total reflectance infrared spectroscopy coupled with the algorithms of deconvolution and partial least squares regression." *Sensors and Actuators B: Chemical* 380 (2023): 133372.
739. Lee, Se Hun, et al. "Synthesis of conducting polymer intercalated sodium vanadate nanofiber composites as active materials for aqueous zinc-ion batteries and NH₃ gas sensors at room temperature." *Composites Part B: Engineering* (2024): 111305.
740. Morder, Courtney, and Zachary D. Schultz. "A 3D printed sheath flow interface for surface enhanced Raman spectroscopy (SERS) detection in flow." *Analyst* (2024).
741. Deal, Alexandra M., et al. "Infrared Reflection–Absorption Spectroscopy of α -Keto Acids at the Air–Water Interface: Effects of Chain Length and Headgroup on Environmentally Relevant Surfactant Films." *The Journal of Physical Chemistry A* 127.18 (2023): 4137-4151.

742. Rehmann, Kelsi MS, John Klier, and Jessica D. Schiffman. "Anionic polymerization and transport of diethyl methyldene malonate on polyolefin copolymer surfaces." *Polymer Chemistry* 14.32 (2023): 3695-3706.
743. Sargent, Alexander T., et al. "Reclamation and reuse of graphite from electric vehicle lithium-ion battery anodes via water delamination." *Journal of Materials Chemistry A* 11.17 (2023): 9579-9596.
744. Guilherme da Fonseca, Bruno. *Application of Raman Spectroscopy for the characterization of carbon materials*. Diss. 2023.
745. Lima, Sofía, et al. "An allosteric switch ensures efficient unidirectional information transmission by the histidine kinase DesK from *Bacillus subtilis*." *Science Signaling* 16.769 (2023): eabo7588.
746. GODINHO, Madame Maria Helena, et al. "Bioinspired photonic cellulose films", 2021. ([PDF file link](#))
747. Chen, Huo, et al. "Adaptive variational simulation for open quantum systems." *Quantum* 8 (2024): 1252.
748. Piskors, Nico, et al. "High Protein—Low Viscosity? How to Tailor Rheological Properties of Fermented Concentrated Milk Products." *Dairy* 4.4 (2023): 594-605.
749. Boukra, Amine, et al. "Sampling terrigenous diffuse sources in watercourse: Influence of land use and hydrological conditions on dissolved organic matter characteristics." *Science of The Total Environment* 872 (2023): 162104.
750. Farmer, Jessica, and Adam J. Jackson. "A fast approximate method for variable-width broadening of spectra." *arXiv preprint arXiv:2309.12135* (2023).
751. Tao, Linmi, et al. "Applications of Tao General Difference in Discrete Domain." *arXiv preprint arXiv:2401.15287* (2024). [2401.15287.pdf \(arxiv.org\)](#)
752. Nieto Benito, Guillermo. "Predicción de series temporales mediante técnicas de aprendizaje automático. Automatización del proceso." (2023). ([PDF link](#))
753. Le, Duc, et al. "sCL-ST: “Supervised Contrastive Learning with Semantic Transformations for Multiple Lead ECG Arrhythmia Classification.” *IEEE journal of biomedical and health informatics* (2023).
754. Mikheenkova, Anastasiia, et al. “Ageing of High Energy Density Automotive Li-ion Batteries: The Effect of Temperature and State-of-Charge.” (2023).
755. Chaiamarit, Tai, et al. "Mutant Prion Protein Endoggresomes are Hubs for Local Axonal Organelle-Cytoskeletal Remodeling." *bioRxiv* (2023): 2023-03.
756. José María Martínez-Blanes Galo Romero-García Daniel Sánchez-Gómez, “Crafting illusions: Human-made composite coating used to simulate amber beads in prehistoric Iberia”, *Journal of Archaeological Science*, June 2024, DOI: 10.1016/j.jas.2024.106011