

Manuale di Signal Processing Scientifico

Edizione di maggio 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



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? Scrivetemi: toh@umd.edu

Unitevi al gruppo Facebook: “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 Cesarano* 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.

Convoluzione di Fourier	103
Semplici vettori di convoluzione a numeri interi.....	104
Dettagli software per la convoluzione	104
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.....	117
Deconvoluzione sequenziale multipla.....	118
Deconvoluzione segmentata.....	118
Live script del tool per l'Auto-deconvoluzione.....	119
Deconvoluzione interattiva con iSignal	120
Filtro di Fourier	122
Software per il filtraggio di Fourier.....	123
Wavelet e wavelet denoising.....	127
Visualizzazione ed analisi.....	128
Riduzione del rumore con le wavelet.....	130
Integrazione e misura dell'area di un picco	134
La gestione dei picchi sovrapposti.....	135
Misura dell'area di un picco con gli spreadsheet	138
<i>Sharpening per la misura dell'area di picchi sovrapposti.</i>	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	145
<i>Correzione del background/linea di base</i>	146
<i>Picchi asimmetrici e ampliamento del picco: taglio verticale rispetto al curve fitting</i>	149
Approssimazione delle curve A: Quadrati minimi lineare	155
Esempi di approssimazioni polinomiali.....	155
Affidabilità dei risultati col curve fitting	160
<i>Propagazione algebrica degli errori.....</i>	160
<i>Simulazione Monte Carlo</i>	161
<i>Il metodo Bootstrap</i>	162
<i>Confronto dei metodi per la previsione dell'errore</i>	163
<i>Effetto del numero dei punti sulla precisione dell'approssimazione con i minimi quadrati.</i>	164
<i>Trasformare relazioni non-lineari.....</i>	165
<i>Semplice approssimazione di picchi Gaussiani e Lorentziani con trasformazione dei dati.</i>	166
Dettagli matematici e software per i quadrati minimi lineare.....	169
<i>Spreadsheet per i quadrati minimi lineari</i>	170
<i>Applicazioni per la calibrazione analitica e la misura</i>	172
<i>Matlab e Octave.....</i>	173
<i>Approssimazione dei picchi con Gaussiana o Lorentziana singola</i>	178
Approssimazione delle curve B: Spettroscopia Multicomponente.....	180
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
<i>Spreadsheet (Fogli di calcolo).....</i>	183
<i>Matlab e Octave.....</i>	185
<i>I Quadrati Minimi Classici in Python.....</i>	189
Approssimazione delle curve C: Approssimazione iterativa non-lineare.....	191
Spreadsheet e programmi autonomi.....	193
Matlab e Octave.....	195
Approssimazione dei picchi.....	196
<i>Funzione semplificata di approssimazione del picco per scopi generali</i>	197
<i>Tipi di forma variabile</i>	198
Funzioni di Fitting per Matlab e Octave	199
Accuratezza e precisione dei parametri del picco	202
<i>a. Modelli errati</i>	202
<i>b. Correzione del background</i>	210
<i>c. Rumore casuale nel segnale</i>	213
<i>d. Errori nell'approssimazione iterativa</i>	216
<i>Un caso difficile</i>	217
Approssimazione dei segnali soggetti ad ampliamento [broadening] esponenziale	219

<i>Esempi</i>	380
Come trovare gli argomenti di input corretti per <i>peakfit</i> ?	391
Lavorare con la matrice dei risultati dell'approssimazione "FitResults".	391
Script dimostrativo per <i>peakfit.m</i>	392
Approssimazione di picchi in dati multi-colonna	392
Gestire i segnali complessi con molti picchi.	393
Ricerca automatica e Approssimazione dei picchi.	394
Peak Fitter Interattivo Gestito da Tastiera (<i>ipf.m</i>)	395
<i>ipf</i> controlli da tastiera (Versione 13.4): Ottenuti premendo il tasto K.	397
Esempi pratici con dati sperimentali reali:	399
Istruzioni sull'uso di <i>ipf.m</i> (versione 13.4)	401
<i>Demoipf.m</i>	408
Tempo di esecuzione dell'approssimazione ed altri compiti del signal processing	410
Suggerimenti e consigli per il Curve Fitting Iterativo	411
Estrazione delle equazioni per i modelli migliori	413
Come aggiungere un nuovo profilo in <i>peakfit.m</i> , <i>ipf.m</i> , <i>iPeak</i> , o <i>iSignal</i> .	414
Quale usare? <i>peakfit</i> , <i>ipf</i> , <i>findpeaks</i> ..., <i>iPeak</i> o <i>iSignal</i> ?	415
Tool Live Script Peak Fitter	417
Python: un linguaggio alternativo gratuito, open-source	419
Smoothing a media mobile del segnale	419
Trasformazione di Fourier e (de)convoluzione	421
I Quadrati Minimi Classici	421
Rilevamento dei picchi	422
Approssimazione ai Quadrati minimi interattiva	422
Intelligenza Artificiale ed Elaborazione dei Segnali	425
IA come assistente di un programmatore	425
Fogli di calcolo per le Curve di Calibrazione Analitiche	430
Background	430
Compilazione dei fogli di calcolo per diversi metodi di calibrazione	430
Confronto dei metodi di calibrazione	434
Istruzioni per l'utilizzo dei template per la calibrazione	434
Domande frequenti (tratte da email e query sui motori di ricerca)	437
Catalogo delle funzioni, script e spreadsheet per il signal processing	443
Funzioni per il profilo dei picchi (per Matlab e Octave)	443
Aritmetica del Segnale	445
Segnali e Rumore	447
Smoothing	449
Differenziazione e sharpening	451
Analisi delle armoniche	453
Convoluzione e deconvoluzione di Fourier	454
Filtro di Fourier	455
Wavelet ed eliminazione del rumore	456
Misura dell'area del picco	456
I minimi quadrati lineare	458
Ricerca e Misura dei Picchi	460
Spettrosocopia Multicomponente	466
Approssimazione iterativa non lineare e approssimazione dei picchi.	466
Funzioni interattive operanti da tastiera	471
Spettrotometria di Assorbimento Quantitativo Iperlineare	472
File MAT (per Matlab e Octave) e i file Testo (.txt)	472
Spreadsheet (per Excel e OpenOffice Calc)	473
Epilogo	477
Come è nato questo libro	477
Chi ha bisogno di questo software?	477
Organizzazione	478
Metodologia	478
L'influenza di Internet	479
Lo Stile	479
Criterio di selezione della piattaforma software	480
Risultati	481
Impatto	481
Il Futuro	481
Riferimenti	483
Pubblicazioni che citano questo libro, i programmi e/o la documentazione	488

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 (pp. 135, 191, 264). 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. 425).. 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, creando nuovi tipi di dati che prima non avevano. Come hanno scritto Erik Brynjolfsson e Andrew McAfee in *The Second Machine Age* (W. W. Norton, 2014): "...molti tipi di dati grezzi stanno diventando notevolmente più economici, e 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 bi-dimensionali 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 "segna" 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.

mente 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. 425).

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.

I personal computer e i laptop sono oggi così veloci che è possibile lavorare con l'elaborazione di dati e segnali in un modo nuovo, in *tempo reale* (pag. 330) o *interattivamente*, premendo un tasto o cliccando col mouse e visualizzando immediatamente i risultati, ad esempio utilizzando le funzioni guidate da tastiera descritte a pag. 471, i "Live script" descritti a pagina 35 o le "app" Matlab descritte a pagina 350.

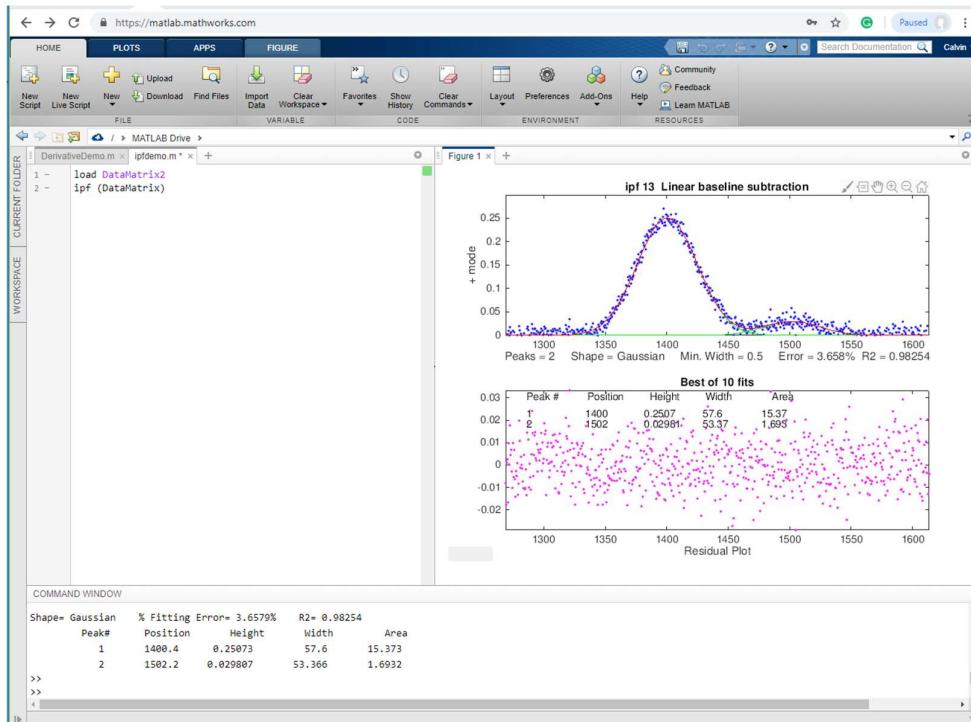
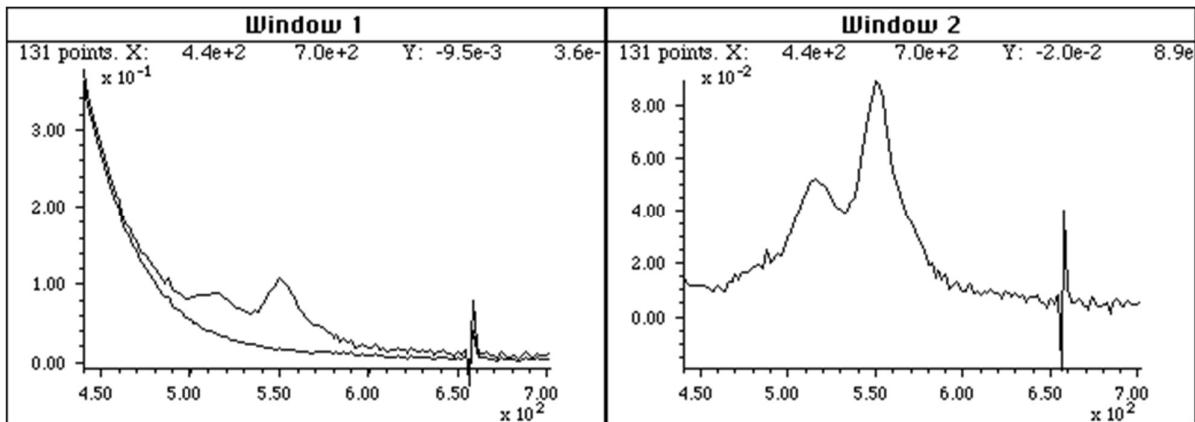


Figura 1. Matlab Online che esegue il Peak Fitter Interattivo (ipf.m) in Chrome su un PC Windows

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>.

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

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.

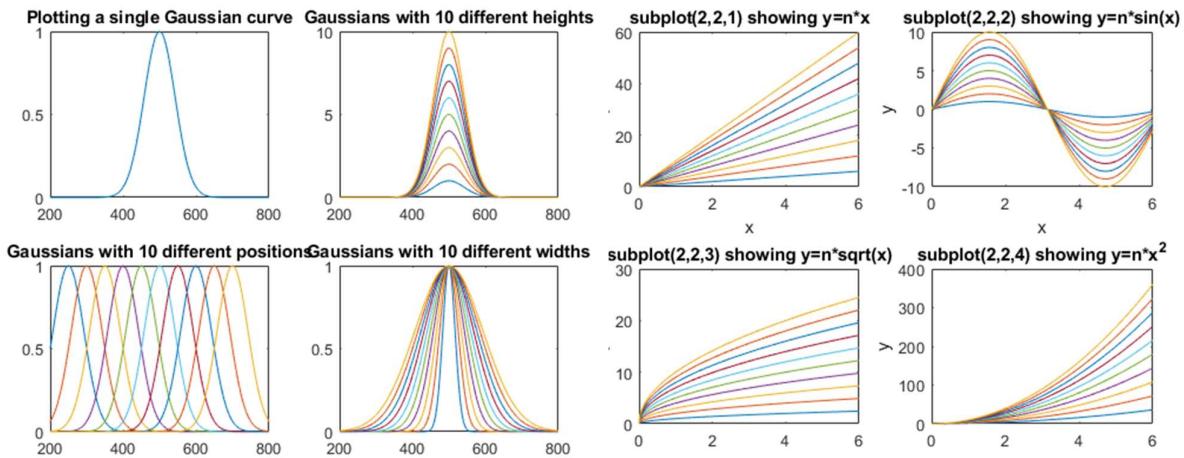
Aritmetica e plottaggio dei segnali in Matlab

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;
```

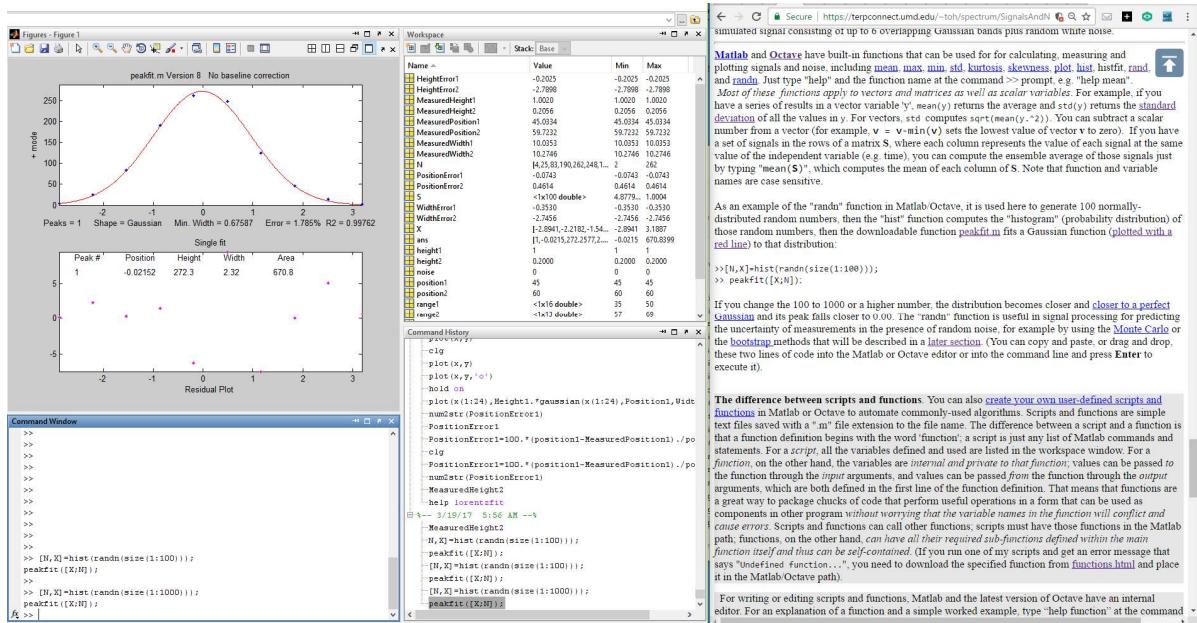
lunghezza d'onda del 50 spettro. Gli script Matlab [plotting.m](#) (a sinistra) e [plotting2.m](#) (a destra) mostrano come plottare più segnali utilizzando una matrice e i "subplot".

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

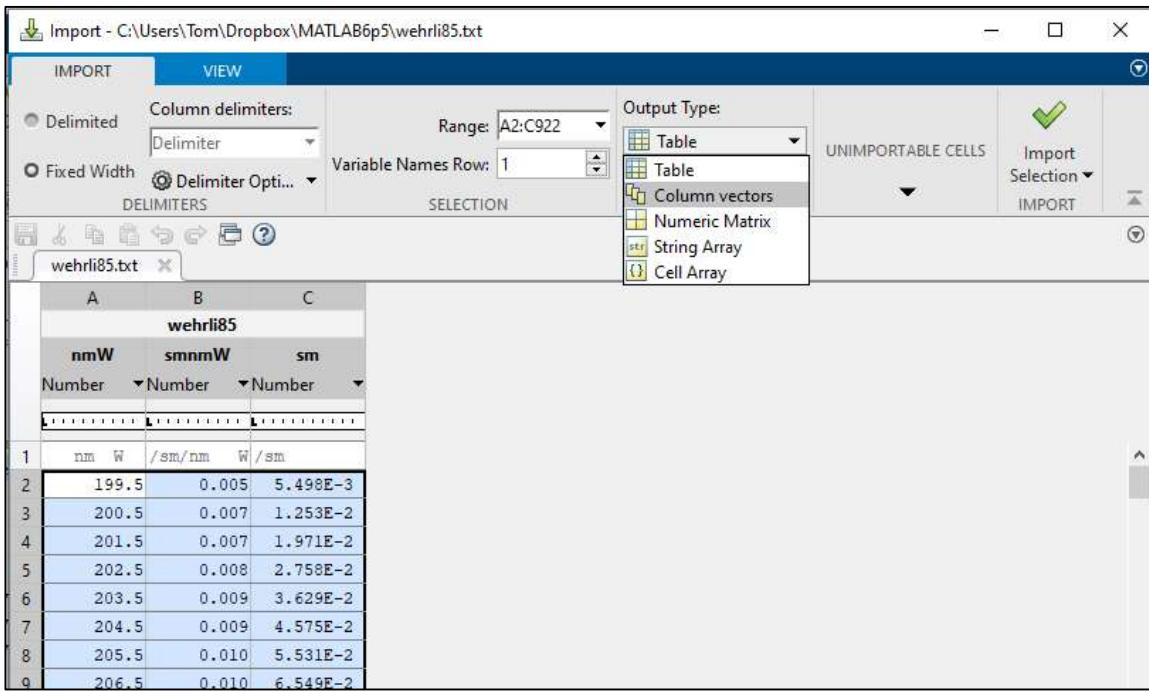


In Matlab/Octave, "/" non è lo stesso di "\\". Digitando "b\|a" verrà calcolata la "divisione matriciale a sinistra", in pratica, il rapporto medio ponderato delle ampiezze dei due vettori (una specie di soluzione per l'approssimazione con i quadrati-minimi). Il punto qui è che *Matlab non richiede di 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, virgolette, 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



[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

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.[_326.](#)

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. 34).

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*.

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.

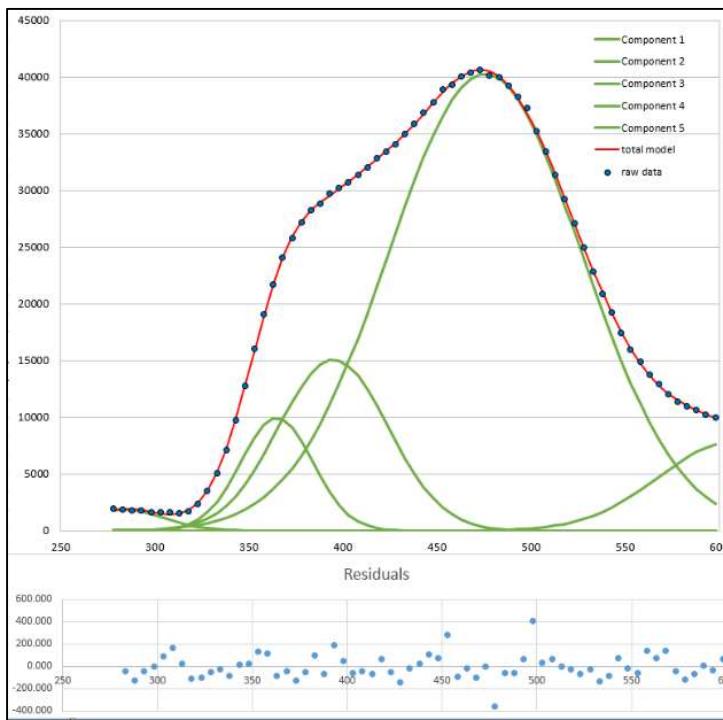
Oltre alla deviazione *standard*, è anche possibile misurare (ma non usuale) la deviazione *mediana assoluta* (mean absolute deviation "mad"). La deviazione standard è maggiore di quella mediana assoluta perché quella standard pesa maggiormente le deviazioni più ampie. Per una variabile casuale normalmente 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 maggiore. 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. 191) 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 *segnale 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 da un modello (la linea rossa) consistente in 5 funzioni di picchi Gaussiani con smoothing sovrapposti (pagina 193). La differenza tra i dati originali e il modello, mostrata in basso (celeste), è una buona misura del rumore casuale nei dati. Tuttavia, è sempre meglio determinare la deviazione standard di misure ripetute della cosa che si vuol misurare (p.es. le altezze o le aree dei picchi o altro), anziché cercare di 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



glioreranno 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.

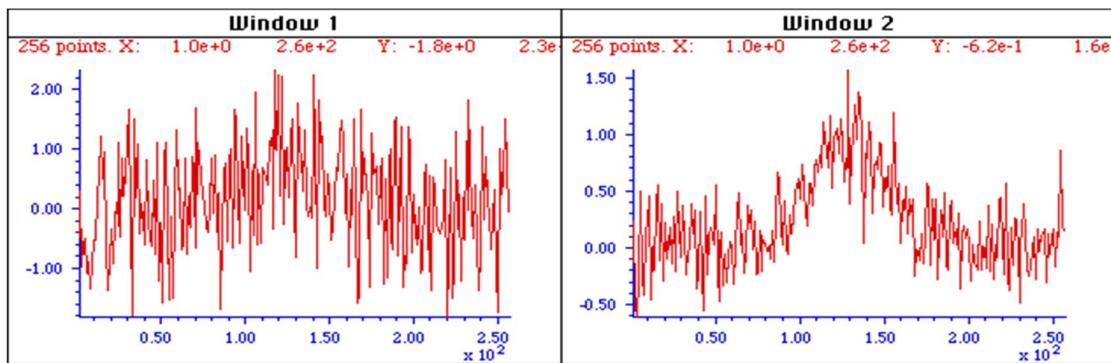
In [SNRdemo.m](#), il rumore è costante e indipendente dall'ampiezza del segnale, che è il caso più comune. Nella variante [SNRdemoHetero.m](#), il rumore nel segnale è direttamente proporzionale al livello del segnale o alla sua radice quadrata, di conseguenza il limite del rilevamento dipende dal rumore 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. 40).

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 *sommendo* 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.



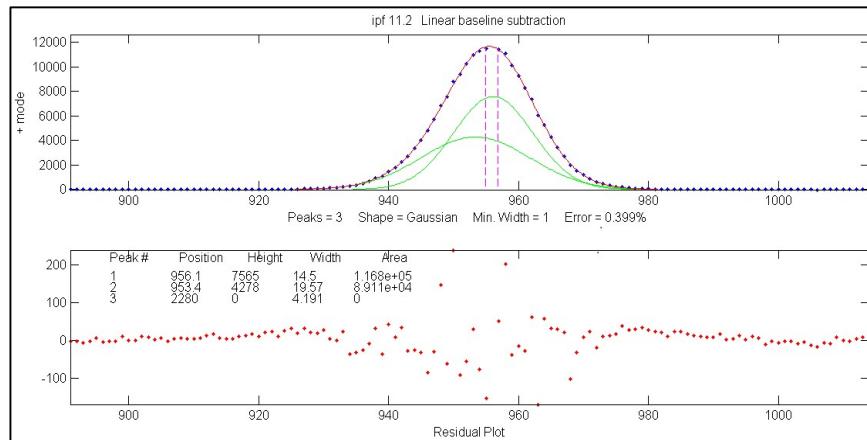
In Window 1 (a sinistra) c'è una singola misura del segnale molto rumoroso. C'è un ampio picco al centro del segnale, ma è difficile misurarne la posizione, la larghezza e l'altezza in modo accurato perché il rapporto S/N è pessimo. In Window 2 (a destra) c'è la media di 9 misure ripetute del segnale, mostra chiaramente il picco emergente dal rumore. Il miglioramento atteso nel rapporto S/N

dioso 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 301: *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. 59) 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



a un modello a picchi multipli (pag. 196), 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. 191) 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 25.

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*,

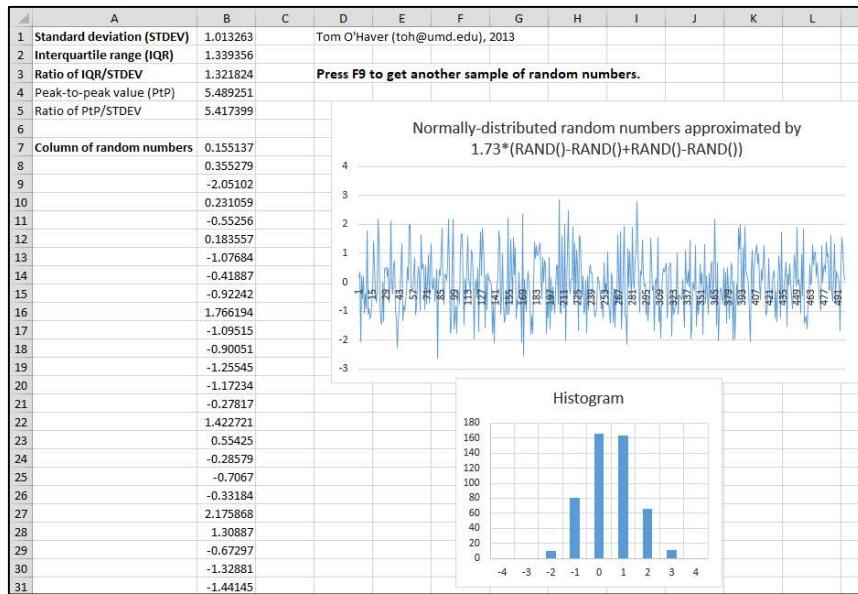
questi due tipi di comportamenti sono rispettivamente *Omoschedastico* e *Eteroschedastico*, rispettivamente. Un modo per osservarlo consiste nel selezionare un segmento di segnale in cui l'ampiezza del segnale varia molto, approssimare il segnale a un polinomio, o

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*.

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 Matlab/Octave). Il grafico in alto a sinistra della figura mostra la distribuzione di probabilità, detta "istogramma", di quella variabile casuale. Successivamente, si combinano due set di tali variabili casuali indipendenti e uniformemente distribuite (cambiando i segni 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 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 approssimazioni 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.

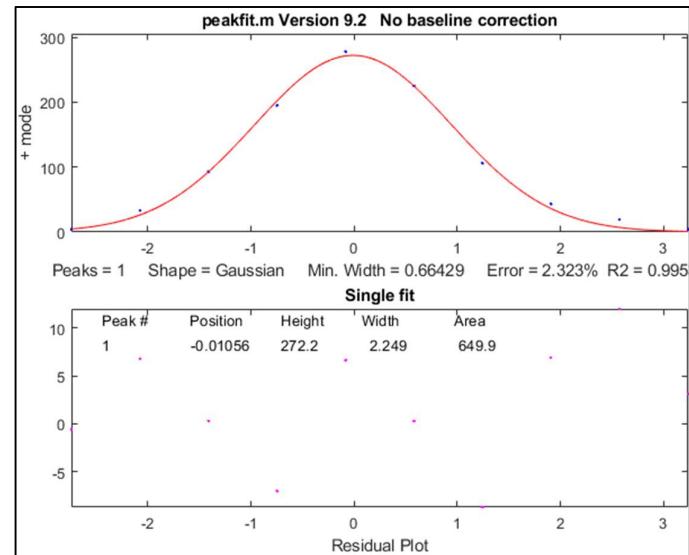


Funzioni random in Matlab e in Python

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 apprenderla 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 375, [link per il download](#)) approssima una funzione Gaussiana (disegnata con una linea rossa) a tale distribuzione.

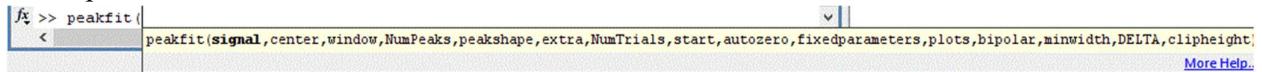
```
[N,X]=hist(randn(size(1:100)));
peakfit([X;N]);
```

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.*

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). Vedere pagina 350. 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 menù 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.

pag. 443 e seguenti per una lista più completa. Lo script [Sech2ShapeComparison.m](#) 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 pag. 401)

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.

[IQRorange.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> [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 443 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

dremo molte applicazioni di questa idea, ad esempio nelle pagine 294 e 320. E la simulazione è applicabile anche nei casi più sofisticati. A pagina 345, 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 sperimentalì, per esplorare i limiti di applicabilità di quel metodo ad altre applicazioni potenzialmente più impegnative.

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, ammorbidente 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 357) ha un'opzione Savitzky-Golay. La funzione *denoise* basata su wavelet è un algoritmo più sofisticato che tenta di distinguere il segnale dal rumore analizzando la struttura della frequenza del segnale. Vedere pagina 130.

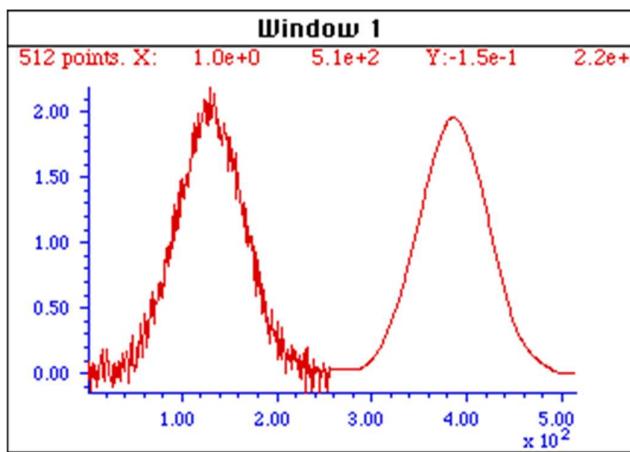
Il profilo di qualsiasi algoritmo di smoothing si può determinare applicando quella operazione a una *funzione delta*, un segnale composto da tutti zeri tranne un punto, come mostrato dal semplice script Matlab/Octave [DeltaTest.m](#). Il risultato è la *funzione di risposta all'impulso*.

Riduzione del rumore

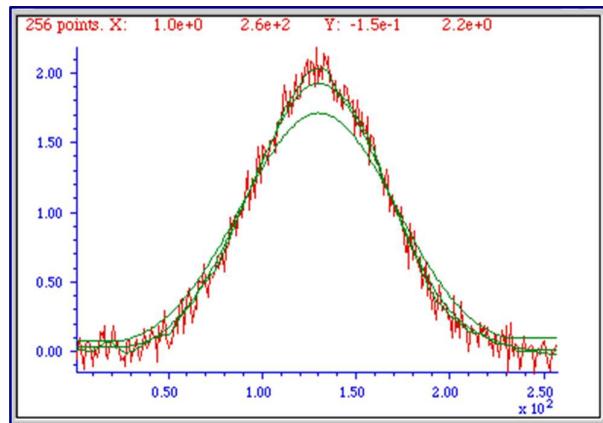
Lo smoothing solitamente riduce il rumore in un segnale. Se il rumore è "bianco" (cioè distribuito uniformemente su tutte le frequenze) e la sua deviazione standard è D , allora la deviazione standard del rumore residuo nel segnale dopo un passaggio dello smoothing rettangolare sarà approssimativamente D/\sqrt{m} , dove m è l'ampiezza dello smoothing. Se, invece si usa quello triangolare, il

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. 449) 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



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).



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

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

Maggiore è l'ampiezza dello smoothing, maggiore è la riduzione del rumore, ma anche maggiore è la possibilità che il segnale venga *distorso* 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

```
end  
plot(x,y)
```

```
end  
plot(x,y)
```

Queste figure mo-

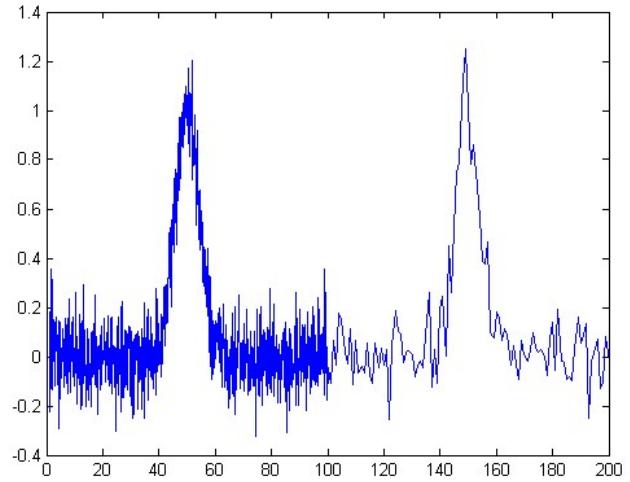
strano dieci grafici sovrapposti con lo stesso picco ma con rumore bianco indipendente, ciascun grafico è disegnato con un colore diverso, senza smoothing a sinistra e con smoothing a destra. Chiaramente, la riduzione del rumore è sostanziale, ma un'attenta ispezione dei diversi segnali con lo smoothing 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. Il fatto che un segnale appaia 'liscio' non significa che non ci sia rumore. Il rumore a bassa frequenza che rimane nei segnali dopo lo smoothing continua ad 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 un numero di punti *dieci volte maggiore*.

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](#).



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 42). 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. 225), 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. 166) di un segmento dei dati originali *senza smoothing* in prossimità del passaggio per lo zero (pag. 227), 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.

Per un altro esempio, vedere pag.281.

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 289).

Media dell'Insieme

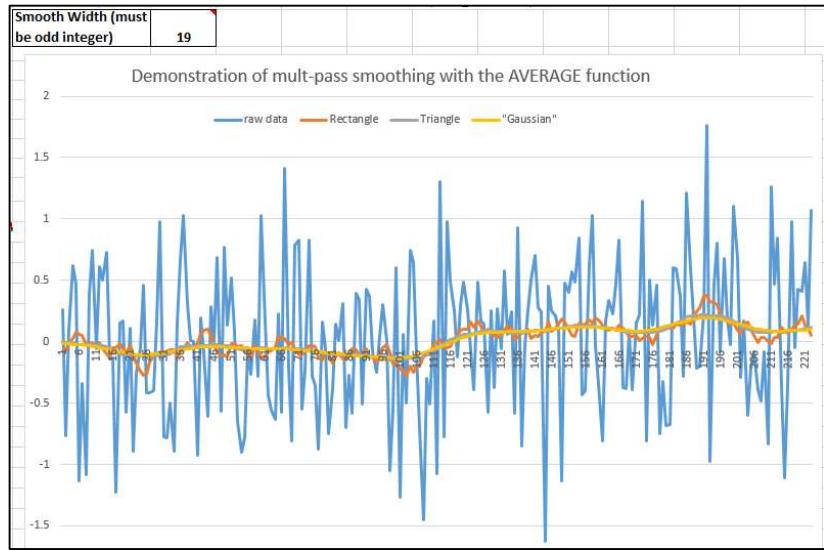
Un altro modo per ridurre il rumore nei segnali ripetibili, come l'insieme di dieci segnali non filtrati a pagina 45, consiste semplicemente nel calcolarne la media, detta *media di insieme [ensemble averaging]*, che si può ottenere in questo caso semplicemente col codice Matlab/Octave **plot(x, mean(y))**; il risultato mostra una riduzione del rumore bianco di circa $\text{sqrt}(10)=3.2$. 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 191) 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 45). 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

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 p-spline. È utilizzato al meglio insieme alla funzione INDIRECT (pagina 335) per controllare un intervallo dinamico di valori. Ciò viene mostrato nello spreadsheet [VariableSmooth.xlsx](#) (a lato) 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 ciascuna larghezza di smoothing debba essere un numero dispari, per evitare uno spostamento sull'asse x nei dati filtrati).

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:

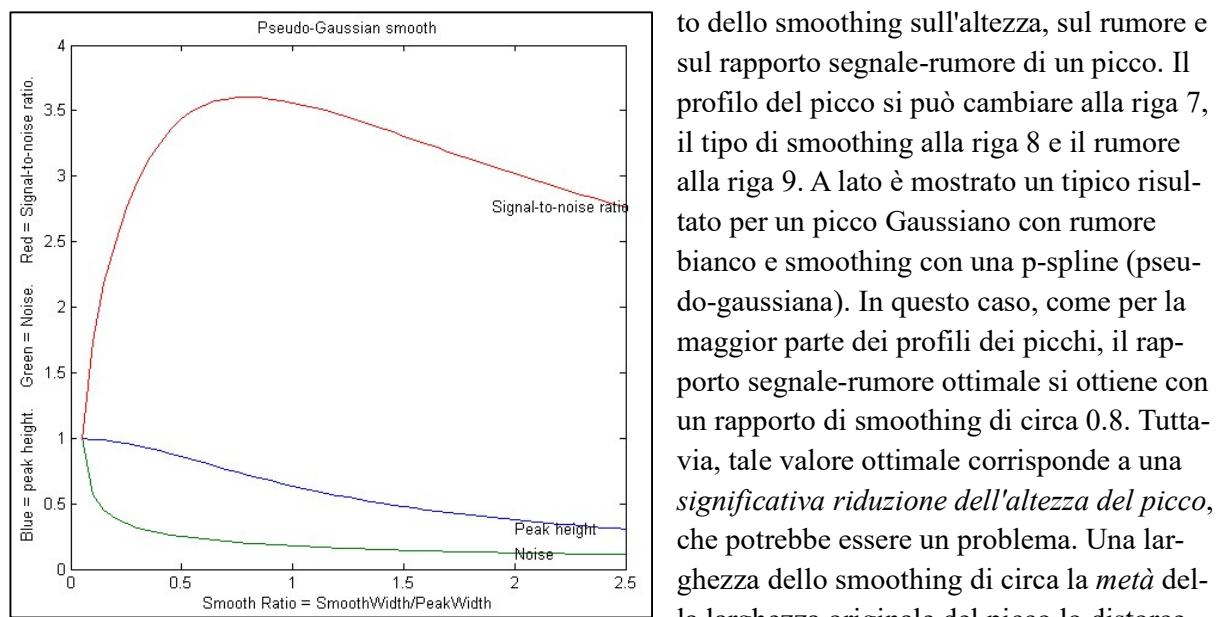
```
wstep=(endw-startw)/NumSegments;
smoothwidths=startw:wstep:endw;
```

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 357).

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



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).

Lo smoothing in tempo reale in Matlab viene trattato a pagina 330. Lo smoothing in *Python* è descritto a pag. 419.

iSignal (pag. 357) 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 scorre 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.

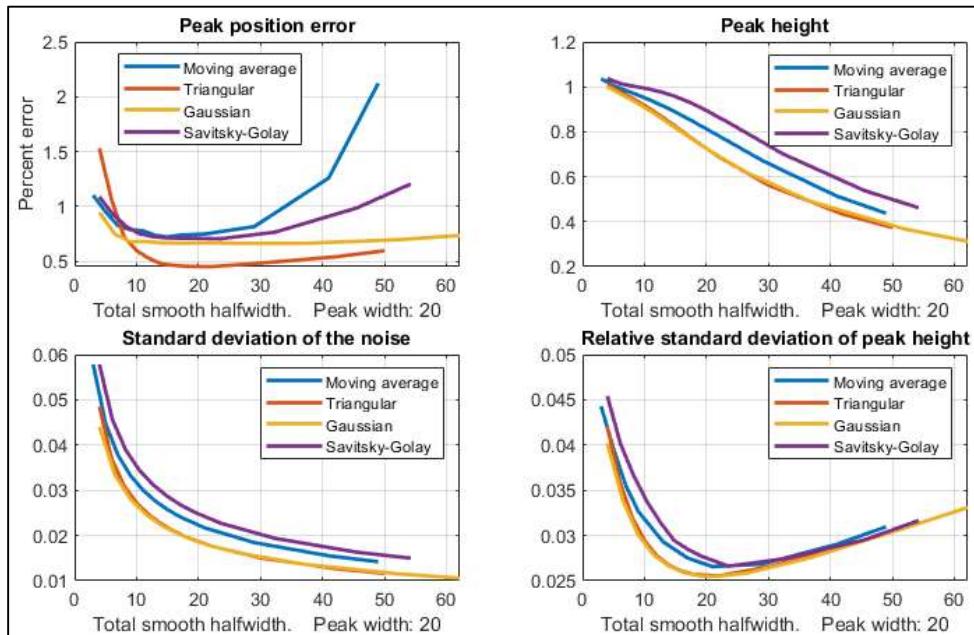
<https://www.mathworks.com/products/matlab-online.html>.

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 350; 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 127, 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".

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 [iSi-](#)



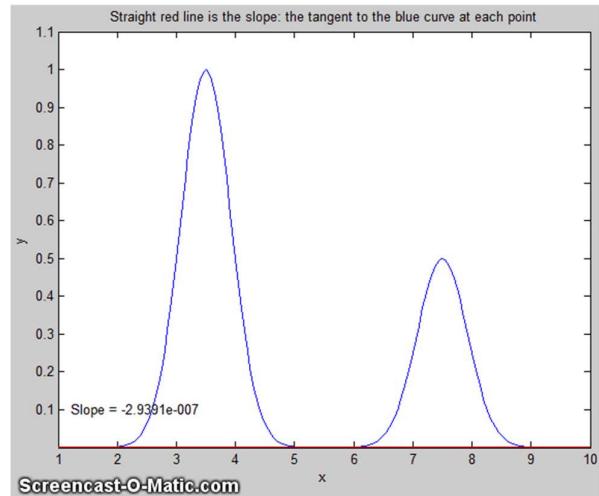
gnal. Queste quattro operazioni di smoothing vengono applicate ad un segnale di 18000 punti consistente in 181 picchi Gaussiani tutti con un'altezza di 1.0 e una FWHM ([full-width at half-maximum \[larghezza intera a metà del massimo\]](#)) di 20 punti ("wid", riga 10), che sono tutti separati da un 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

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:

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

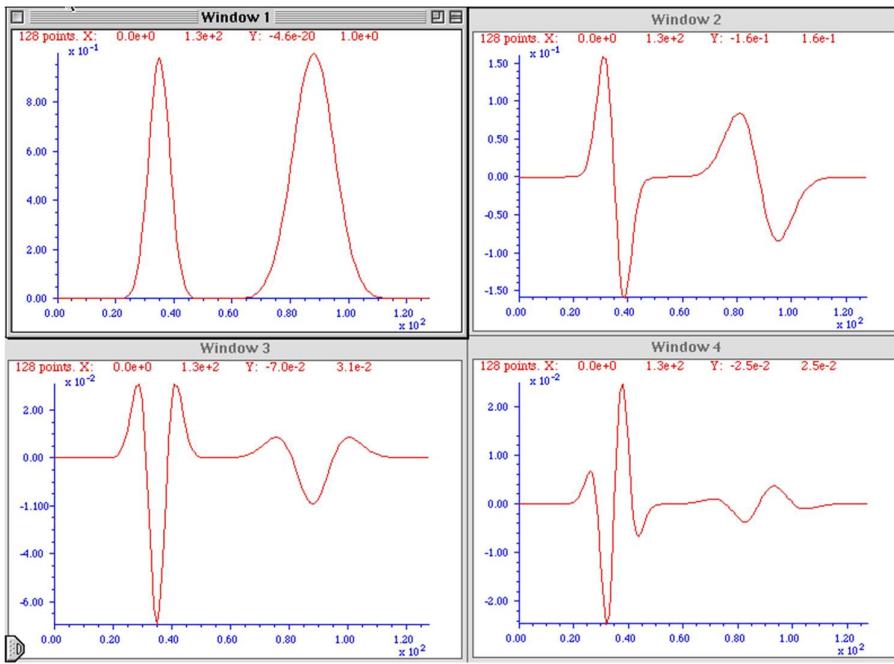
Questo è detto metodo a *differenza centrale*; ha il vantaggio che non provoca uno spostamento della posizione sull'asse x della derivata. È possibile anche calcolare derivate '*gap-segment*' in cui l'intervallo sull'asse x tra i punti nell'espressione precedente è maggiore di uno; per esempio, Y_{j-2} 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.

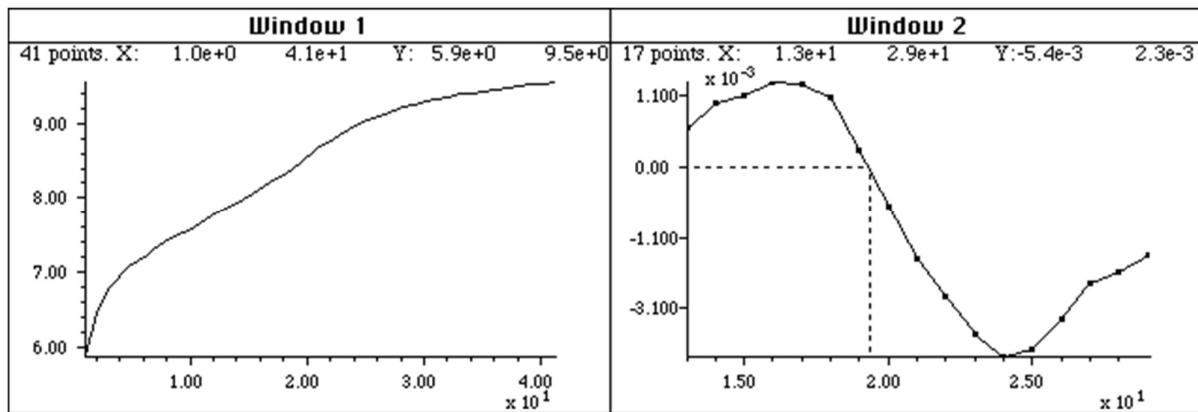
(codice Matlab/Octave). Qui si può anche vedere che l'ampiezza numerica delle derivate (valori sull'asse y) è molto inferiore del segnale originale perché le derivate sono le *differenze* tra i punti adiacenti dei valori y, diviso per l'incremento della variabile indipendente. (È lo stesso motivo per 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 *segnale 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 374 per un esempio.



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.

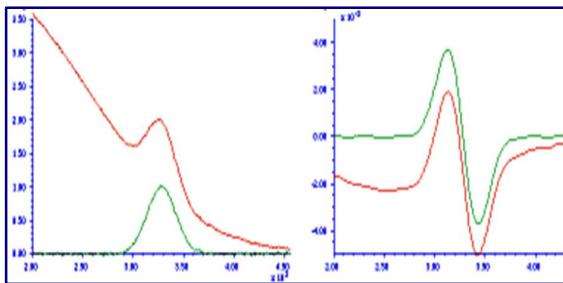
La figura mostra una titolazione della curva del pH di un acido debolissimo con una base forte, col volume in mL sull'asse X e il pH sull'asse Y. Il punto di equivalenza volumetrica (l'endpoint "teorico") è a 20 mL. Il "punto di equivalenza" è quello in cui si ha la maggior pendenza; è anche un punto 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 127), 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 distinguere 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.

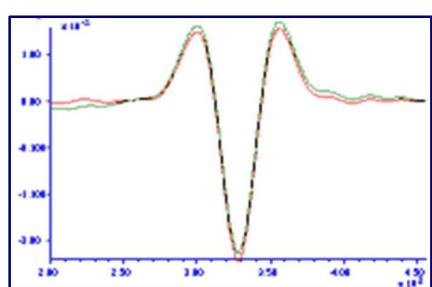
Le derivate possono essere utilizzate anche per *correggere* l'asimmetria del picco, aggiungendo una porzione ponderata della derivata prima al picco originale, come descritto nella sezione sull'asimmetria del picco a pagina 76.

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 430). 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. Spessissimo 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.

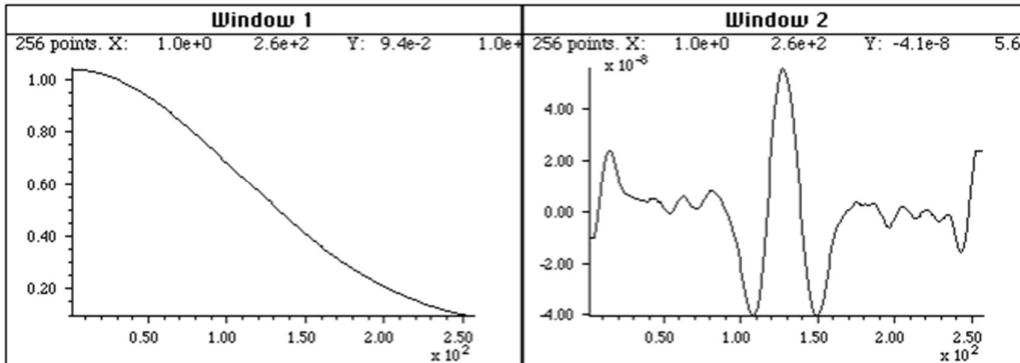


La derivata prima di questi due segnali appare al centro; si può vedere che la differenza tra lo spettro dell'analita puro (verde) e lo spettro della miscela (rosso) è ridotta. Tale effetto è notevolmente più evidenziato nella derivata seconda, mostrata a destra. In questo caso gli spettri dell'analita 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 [Derivative-Demo.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

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.

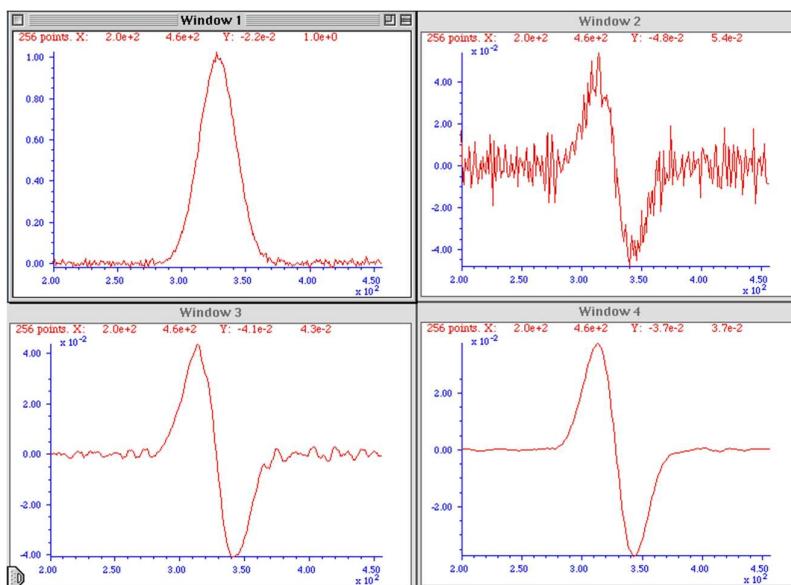


Come la figura precedente, ma in questo caso il picco è 10 volte più basso, tanto da non poter nemmeno essere visto nello spettro a sinistra. La derivata quarta (a destra) mostra che un picco c'è, ma molto ridotto in ampiezza (si noti la scala più piccola dell'asse y) e con un rapporto segnale-rumore 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 283), 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

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 357, 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. 317), 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 430). 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, era troppo presto per la 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.

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 secderivxy, 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**".

iSignal.m (pag. 357), 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](#).

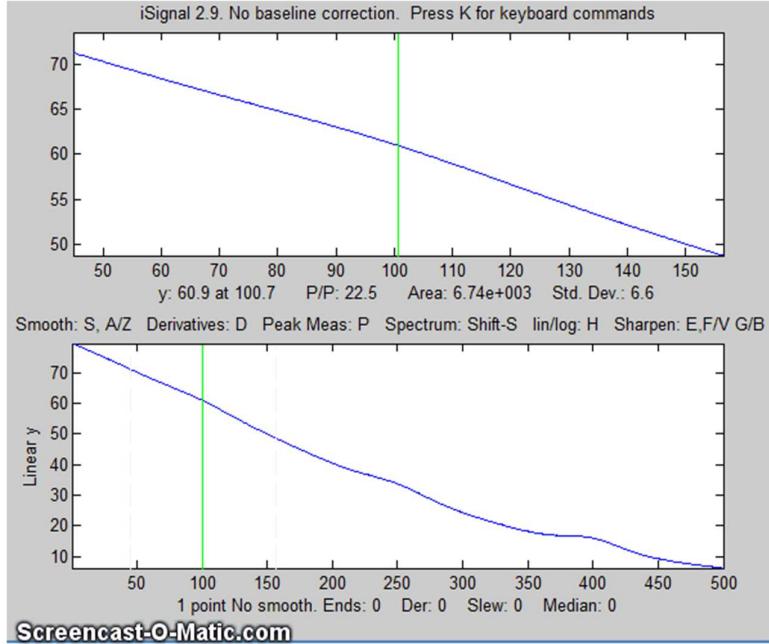
```
>> x=[1:.1:300] ;
>> y=deriv4(100000.*gaussian(x,150,50)+.1*randn(size(x)) );
>> isignal(x,y);
```

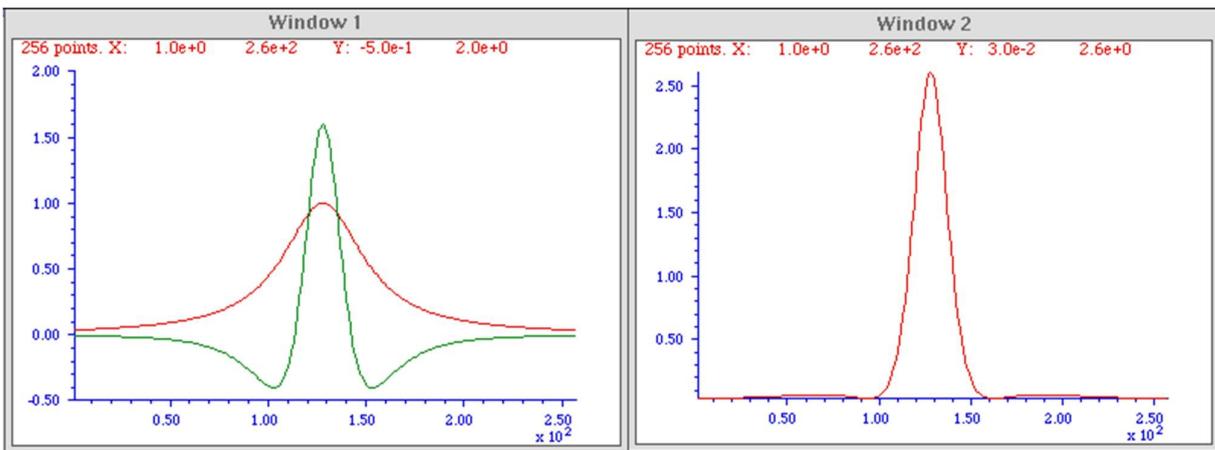
Il segnale è soprattutto rumore blu (a causa della differenziazione del rumore bianco) a meno di non attenuarlo in modo considerevole. Si usano i tasti **A** e **Z** per aumentare e diminuire l'ampiezza dello smoothing e il tasto **S** per scorrere tra i diversi tipi di smoothing. Suggerimento: usare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

La differenziazione in tempo reale in Matlab è trattata a pag. 330.

Live Script per la Differenziazione

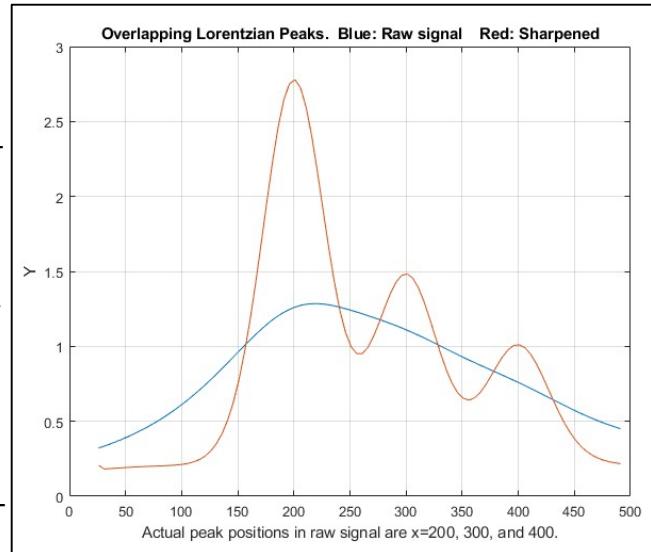
[DataDifferentiation mlx](#) (**grafico**) è un Live Script per la differenziazione e lo smoothing applicato ai dati sperimentali archiviati su disco. È simile a [DataSmoothing mlx](#) discusso a pagina 55, con l'aggiunta di uno slider (riga 9) per selezionare l'ordine di derivazione (fino a 10). Nota: 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).





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 semilarghezze 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.

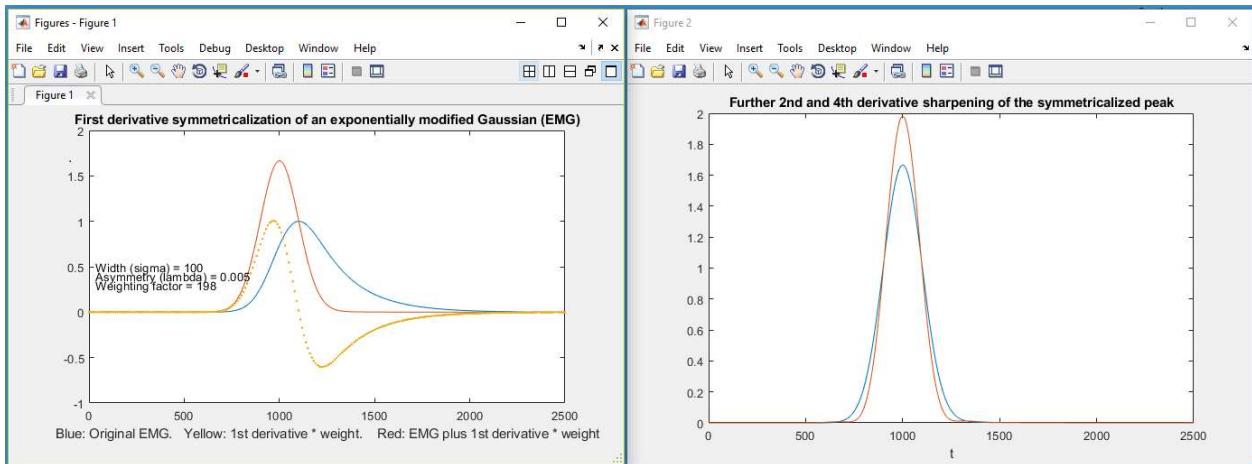


Si noti che la riga di base in entrambi i lati del picco migliorato, non è molto piatto, specie per un picco Lorentziano, perché la cancellazione tra il picco originale e la derivata seconda invertita è solo approssimativa; il fattore di ponderazione k viene regolato per minimizzare questo effetto. Lo 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 alter-

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”.

$$S_j = Y_j + k_1 Y'$$



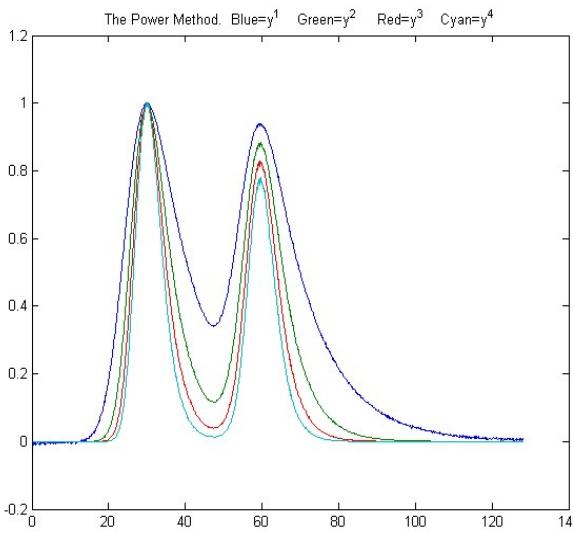
In effetti, è possibile dimostrare che questa semplice tecnica funziona perfettamente per *picchi ampliati esponenzialmente di qualsiasi forma*, come la "[Gaussiana Modificata Esponenzialmente](#)" ([EMG](#)) mostrata qui (riferimento 73).

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 .



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 *sporgenze*; 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

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

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				

5 delle colonne da **K** ad **AA**, proprio come con i fogli di calcolo per lo smoothing (pag. 50). (Nota: Sfortunatamente, questi pulsanti ActiveX non funzionano nella versione iPad di Excel).

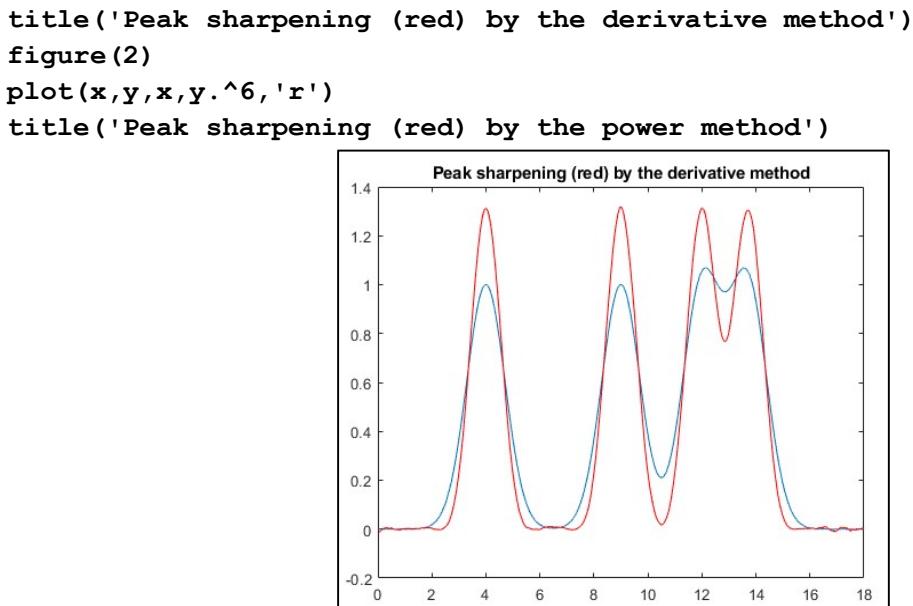
Esiste anche un template “segmentata” del template in cui è possibile specificare le costanti per lo sharpening per ognuno dei 20 segmenti del segnale ([SegmentedPeakSharpeningDeriv.xlsx](#)). Per quelle applicazioni in cui le larghezze dei picchi aumentano (o diminuiscono) gradualmente nel tempo, c’è 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.xlsx](#) (schermata di seguito) esegue la simmetizzazione delle Gaussiane modificate esponenzialmente (EMG) mediante l’aggiunta ponderata della derivata prima. [PeakSymmetricalizationExample.xlsx](#) 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.xlsx \(grafico\)](#). Questi fogli di calcolo consentono anche un ulteriore sharpening con la derivata seconda del picco simmetrico risultante.

[PeakDoubleSymmetrizationExample.xlsx](#) esegue la simmetizzazione 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.

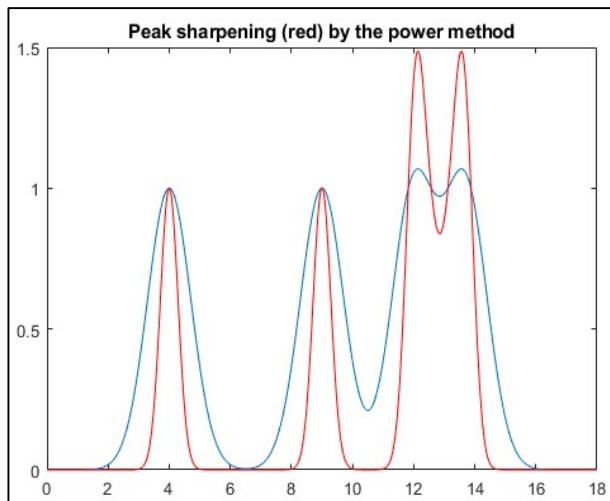


*Quattro picchi Gaussiani sovrapposti di uguale altezza e larghezza.
In blu: l'originale. In rosso: dopo lo sharpening col metodo delle derivate pari.*

[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 halffwidth.m nel path di ricerca di Matlab. Il download si effettua da <https://terpconnect.umd.edu/~toh/spectrum/>).



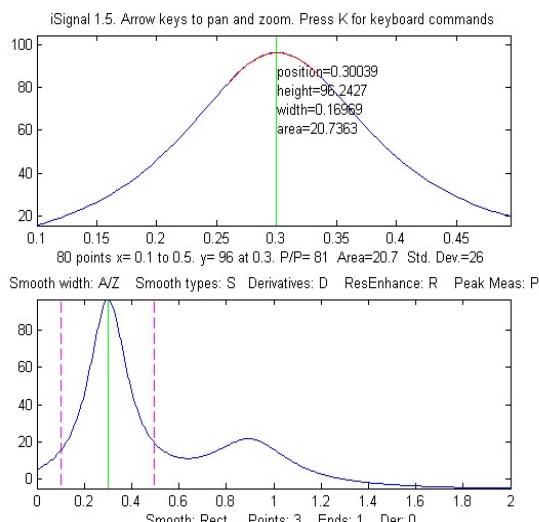
Il metodo dell'elevamento a potenza è efficace se c'è un avvallamento tra i picchi sovrapposti, ma questo introduce una non-linearità che dev'essere corretta in seguito, mentre il metodo delle derivate preserva le aree originali dei picchi e il rapporto tra le altezze dei picchi. [PowerLawCalibrationDemo](#) mostra la linearizzazione delle curve di calibrazione trasformate in potenza per due picchi sovrapposti prendendo la potenza ennesima dei dati, localizzando l'avvallamento tra di essi, misurando le aree col metodo del taglio perpendicolare (pagina 135), e quindi prendendo la potenza 1/n delle aree

zione. 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 possibile 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.

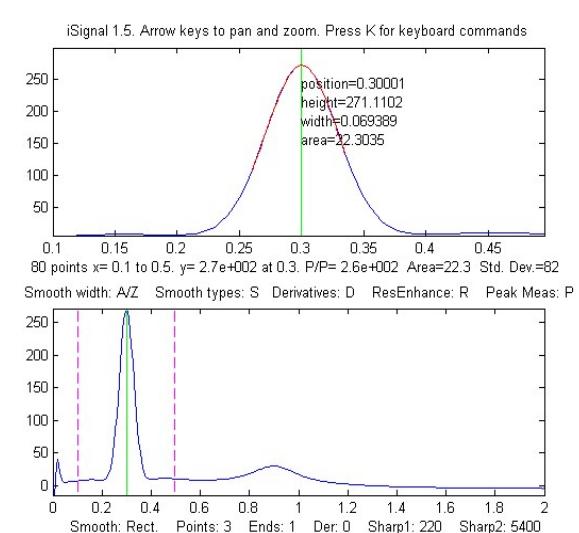
[ProcessSignal](#), è una funzione Matlab/Octave a riga di comando che esegue smoothing, differenziazione e sharpening del picco su una serie temporale x,y (vettori colonne o righe). Digitare "help ProcessSignal". Restituisce il segnale processato come vettore che ha lo stesso profilo di x, indipendentemente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

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

[iSignal](#) (Versione 8.3, pagina 357) 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



Prima dello Sharpening in iSignal



Dopo lo Sharpening in iSignal

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 picchi 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](#)).

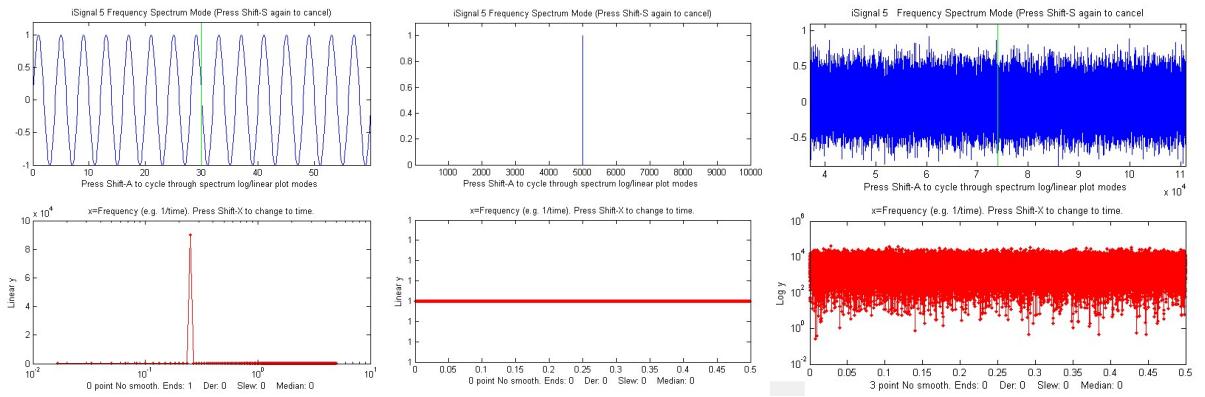
In *iSignal*, e in *iPeak*, il tasto **Shift-Y** attiva la tecnica di simmetrizzazione della derivata prima ed usa i tasti **1**, **Shift-1**, **2** e **Shift-2** per regolare il fattore di ponderazione del 10% o dell'1% ad ogni pressione del tasto. L'idea è quella di aumentare il fattore fino a quando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

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).

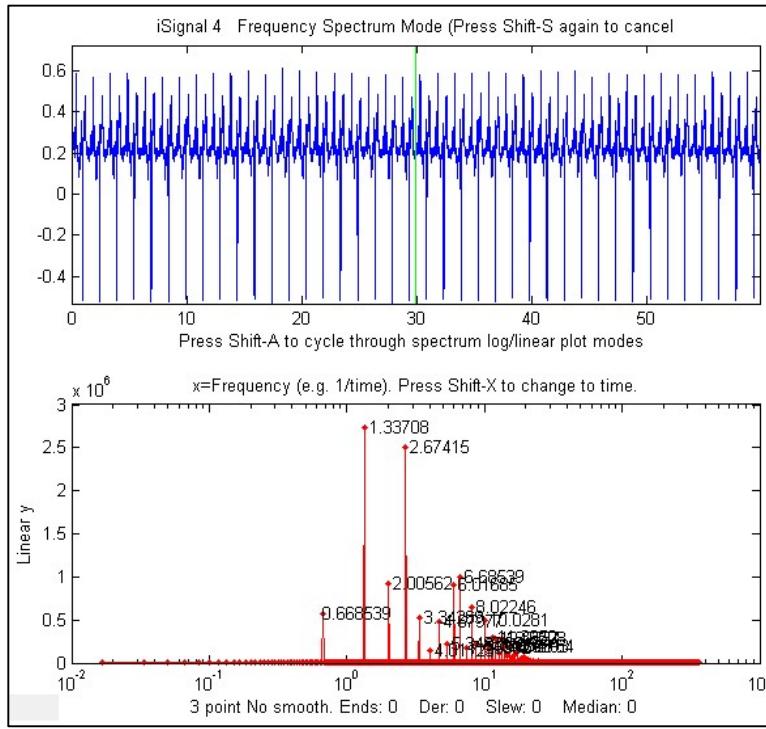
Il concetto di trasformata di Fourier è coinvolto in due moderni metodi strumentali molto importanti nelle analisi chimiche. Nella [Spettrosopia infrarossa in trasformata di Fourier \(FTIR\)](#), la trasformata di Fourier dello spettro viene misurata direttamente dallo strumento, come l'interferogramma 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 segna-



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

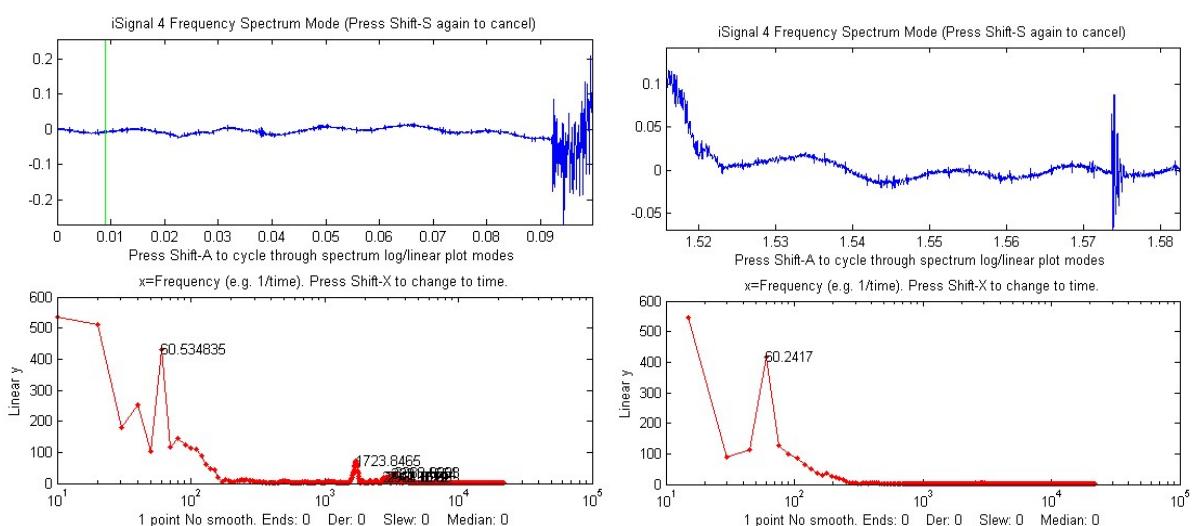
po' lenta per una normale frequenza cardiaca umana in stato di veglia), con molteplici armoniche alle frequenze $\times 2, \times 3, \times 4, \dots$, ecc., volte la frequenza fondamentale. La frequenza più bassa nello spettro è 0.067 Hz (il reciproco della durata della registrazione) e quella più alta 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

```

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 il 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](#).



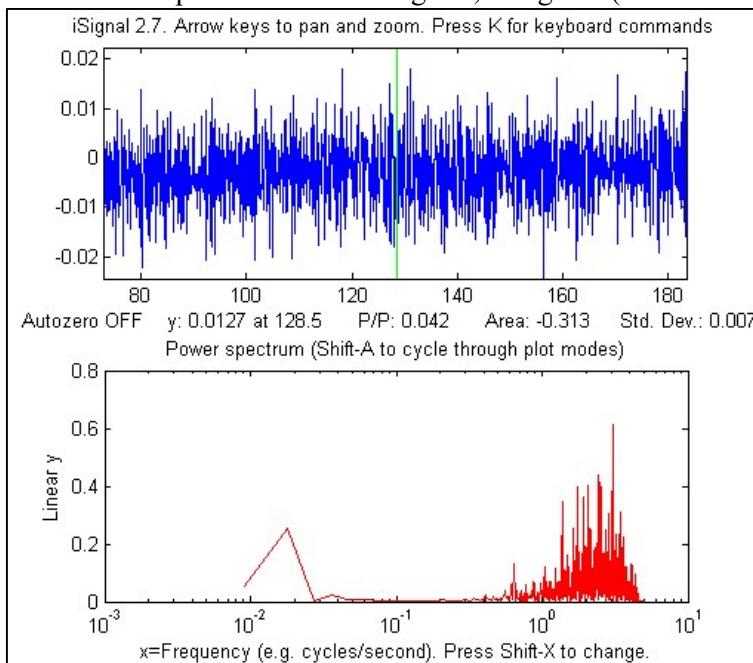
iSignal, che mostra i dati di una registrazione audio, ingrandita nel periodo di "quieta" precedente (a sinistra) e successivo (a destra) del suono effettivo. Si vede che c'è del rumore di fondo con un'oscillazione sinusoidale regolare ($x = \text{tempo in secondi}$). Nel pannello inferiore, lo spettro 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 42).

[sulle reti elettriche.](#)

Un esempio di serie temporale con periodicità multiple complesse sono le [visualizzazioni giornaliere delle pagine](#) (x=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 *segnalet* l'altra componente è [rumore blu](#).

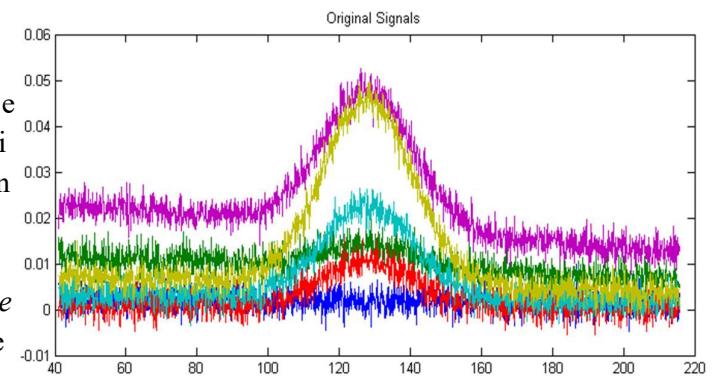
[rumore blu](#), residuo delle precedenti operazioni di signal processing. Le due componenti sono fortunatamente ben separate sull'asse delle frequenze, suggerendo che il filtraggio passa-basso (ad esempio, lo smoothing, pagina 40) sarà in grado di rimuovere il rumore senza distorcere il segnale.

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).

[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. 70). 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. 374) 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 *scarso 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).

mento: 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:

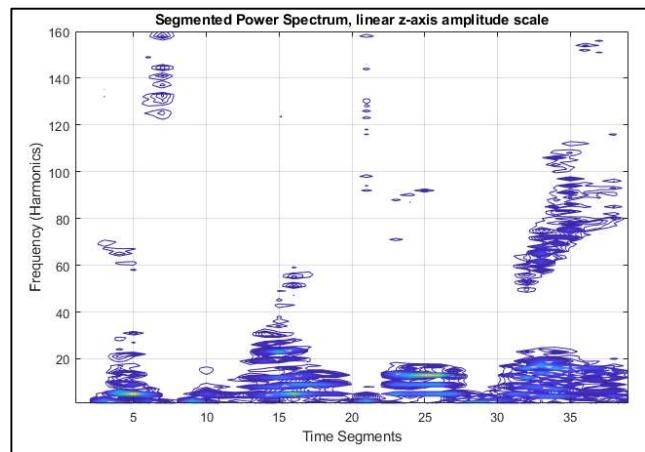
```
x=[0:.01:2*pi]';
f=25; % Frequenza
y=sin(2*pi*f*x)+randn(size(x));
subplot(2,1,1);
plot(x,y);
subplot(2,1,2);
FS=PlotFrequencySpectrum(x,y,1,0,1);
```

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: `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.

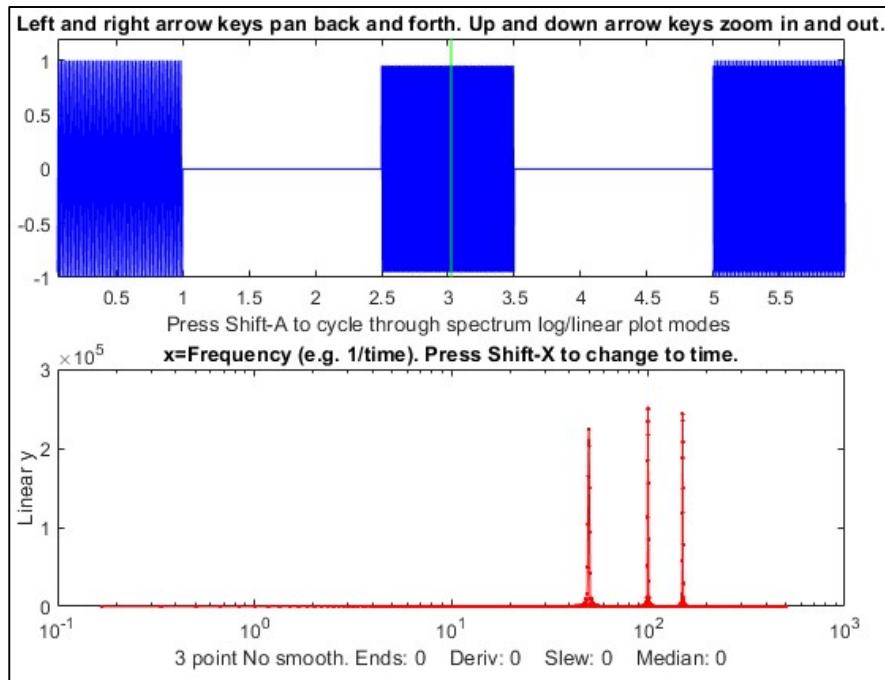
La funzione [PlotSegFreqSpect.m](#), sintassi `PSM = PlotSegFreqSpect(x, y, NumSegments, MaxHarmonic, logmode)`, crea e visualizza un spettro di potenza di Fourier *segmentato nel tempo*. Suddivide il segnale in segmenti di uguale lunghezza "NumSegments", moltiplica ciascuno per una finestra di Hanning 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).



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 357. 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.



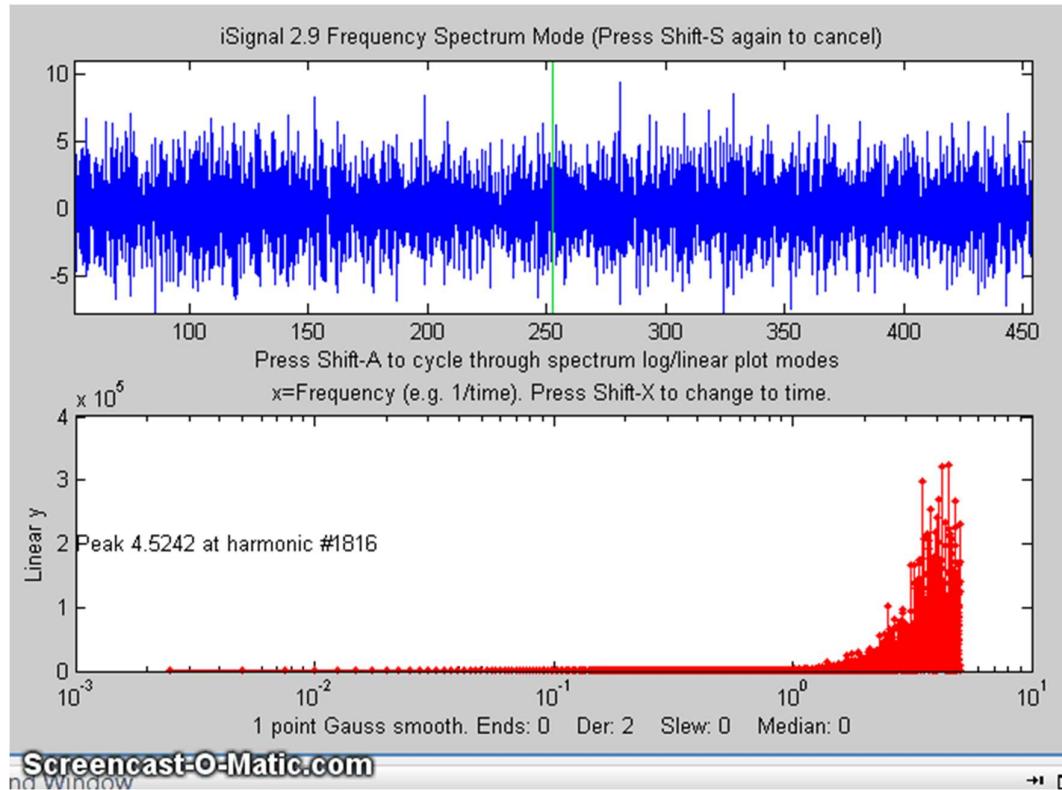
La funzione Matlab iSignal.m mostra un segnale (pannello superiore) e il suo spettro di frequenza (pannello inferiore).

Qui il segnale viene mostrato nel pannello superiore nella funzione [iSignal.m](#) (pagina 357) 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*.

zione 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

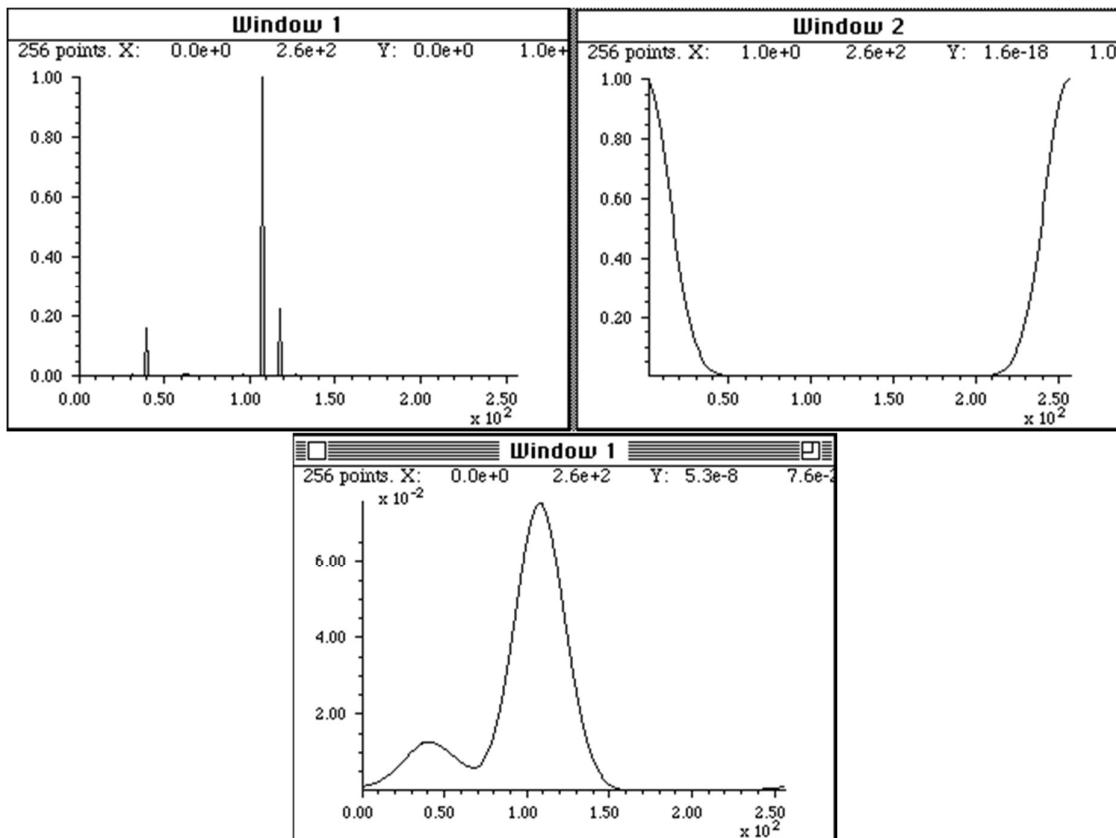
Una cosa speciale (per matematici) riguardo al segnale con profilo *Gaussiano* rispetto a tutte le altre forme è che lo spettro di frequenza di Fourier di una Gaussiana è *anch'esso* una Gaussiana. Lo si può dimostrare numericamente scaricando le funzioni [gaussian.m](#) e [isignal.m](#) ed eseguendo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

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 264).

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 gene-

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 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.

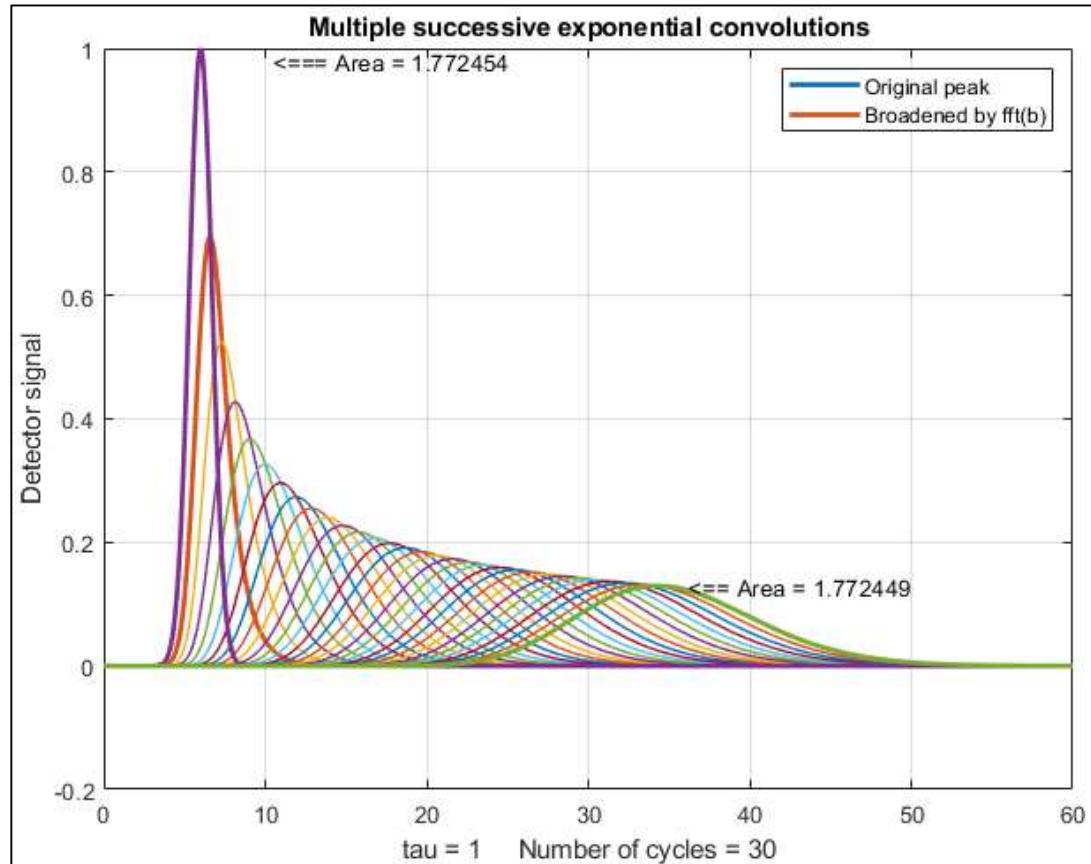
```
yc=ifft(fft(y).*fft(c));
```

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:

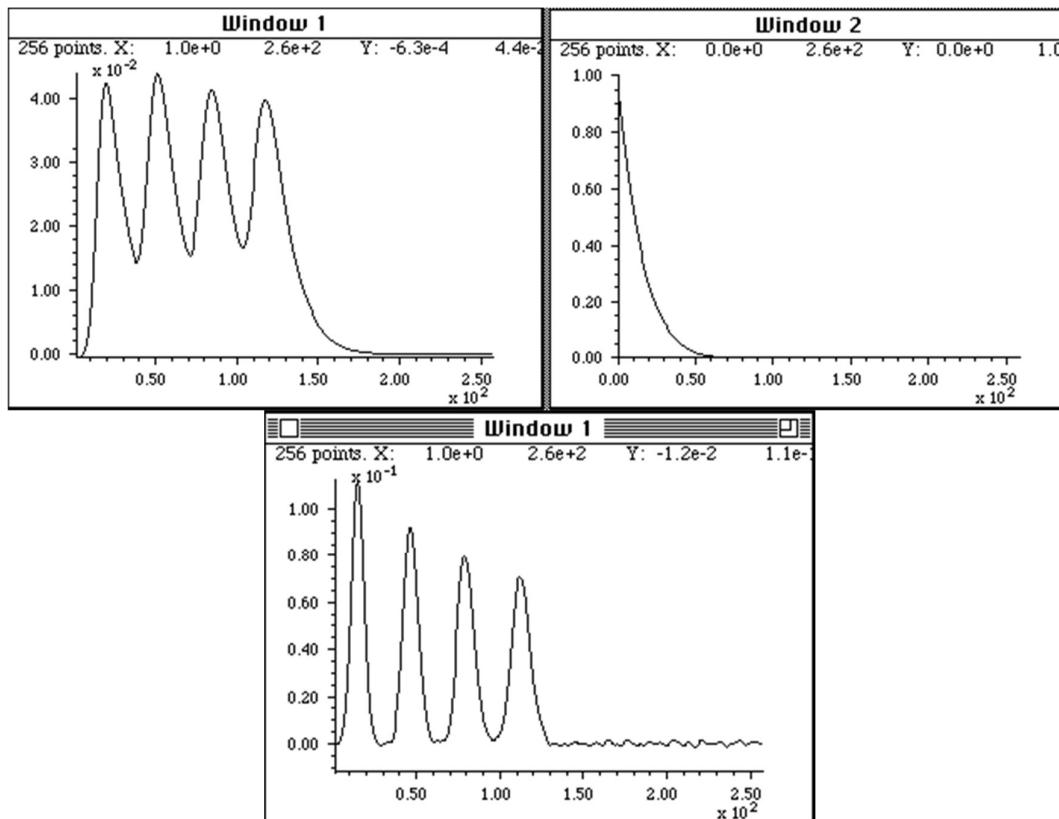
```
yc=ifft(fft(y).*fft(c)) ./sum(c);
```

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

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).



tengono 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. 73) 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 convo-

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.

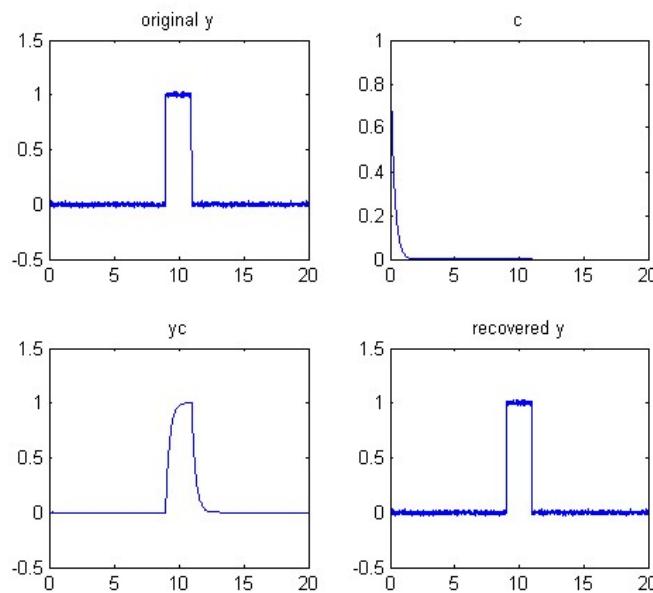
In molti casi, la larghezza della convoluzione fisica non è nota esattamente, quindi la deconvoluzione deve essere regolata empiricamente per ottenere i risultati migliori. Allo stesso modo, anche la larghezza dell'operazione finale di smoothing deve essere regolata per ottenere i migliori risultati. 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

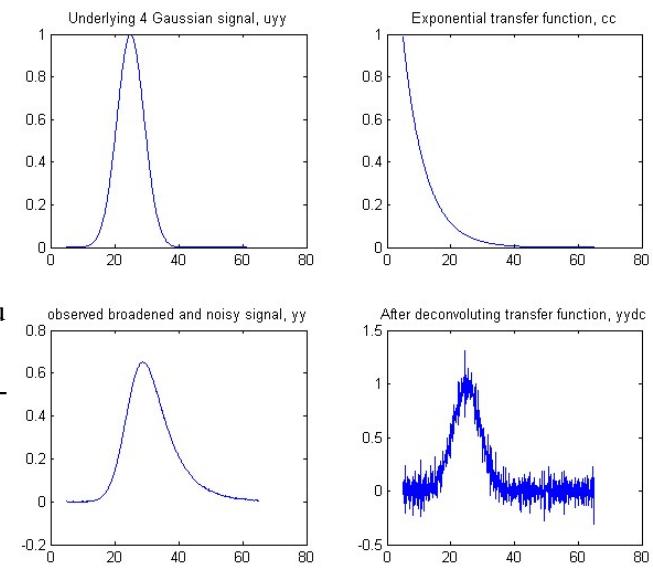


non solo l'effetto della distorsione del segnale da parte della convoluzione con la funzione esponenziale, ma anche l'effetto del passa-basso per il filtraggio del rumore. Come spiegato, c'è una significativa amplificazione di qualsiasi rumore aggiunto *dopo* la convoluzione dalla funzione di trasferimento (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

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 [Loren-tziani](#). 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.

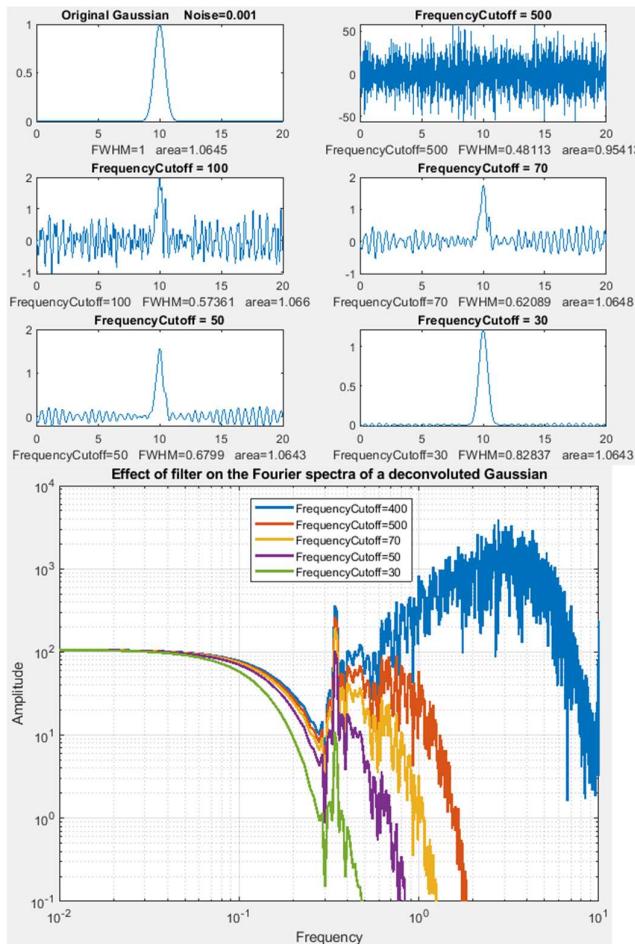
Nell'esempio mostrato a lato. ([Effettuare il download di questo script](#)), il segnale (*uyy*) è una *Gaussian*, ma nel segnale osservato (*yy*) il picco è *ampliato esponenzialmente* risultando un picco *traslato, più basso e più largo*. Supponendo che la costante di tempo dell'ampliamento esponenziale ('*tc*') sia nota, possa essere indovinata o misurata (pag. 76), la deconvoluzione di Fourier di *cc* da *yy* rimuove con successo l'ampliamento(*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. 42) 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:



gli argomenti del centro e della finestra in peakfit.m). Si veda a pagina 291 per un altro esempio con quattro Gaussiane sovrapposte.

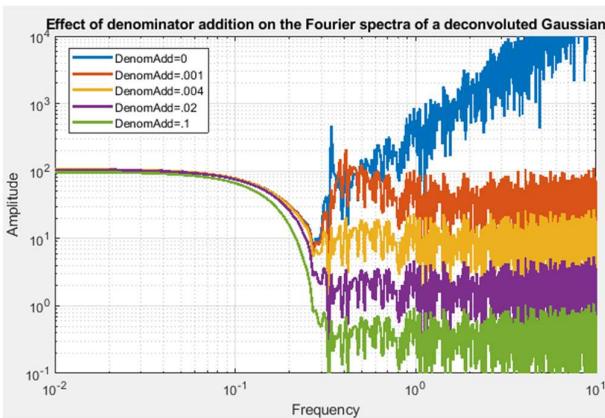
Riduzione del rumore nei segnali deconvoluti

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)



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.



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'aggiunta 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'aggiunta 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'aggiunta del denominatore sono operazioni ortogonali nel dominio della frequenza. Il filtraggio opera "orizzontalmente" lungo l'asse x (frequenza), mentre l'aggiunta 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.

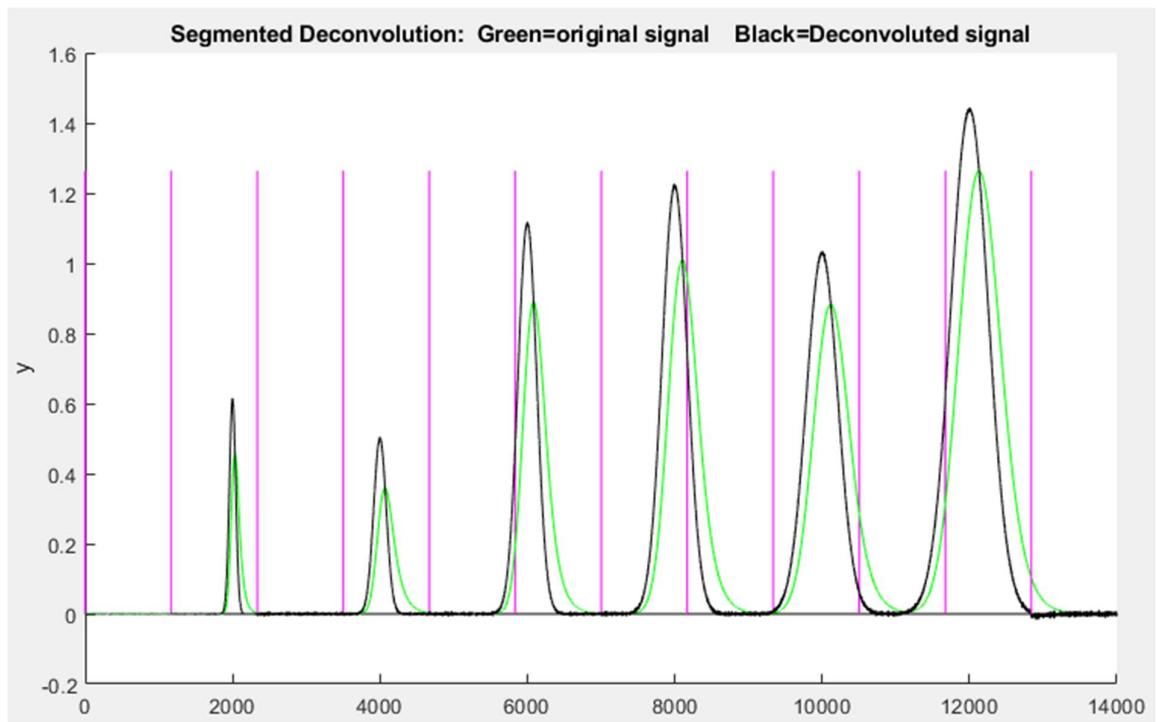
Un buon modo per esplorare l'interazione tra i valori delle numerose variabili è utilizzare Live Script di Matlab [DenomAdditionDemo mlx \(grafico\)](#), che ha *cursori* e un *menu a discesa* per regolare i parametri in modo interattivo. Cfr. pagina 350.

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 119.

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 ca-



[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

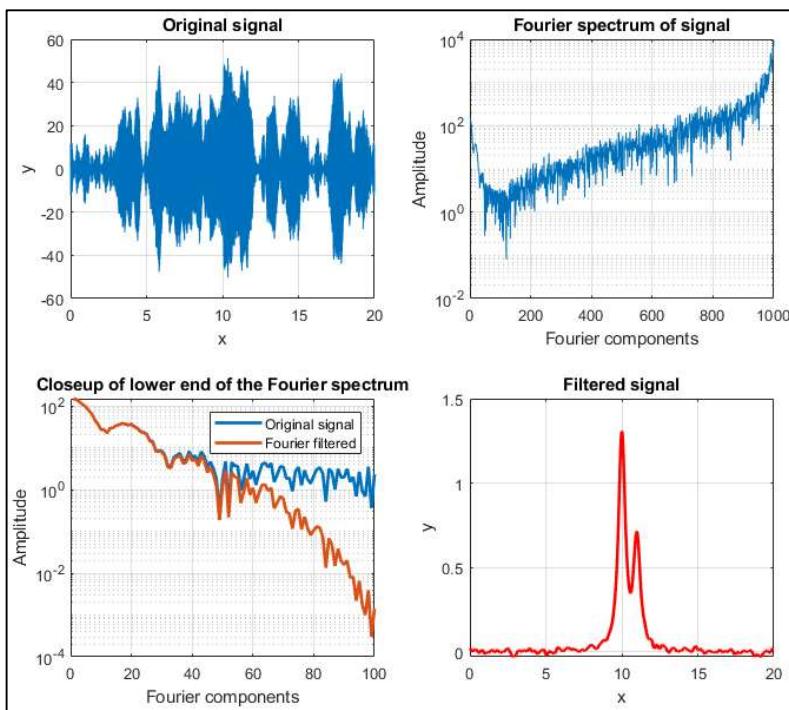
Il Live Script [DeconvoluteData mlx](#) può eseguire l'autodeconvoluzione di Fourier sui dati archiviati nel disco. Cliccando sul pulsante "Open data file" nella riga 1 si apre un browser di file, che consente di navigare fino al file di dati (in formato .csv o .xlsx; lo script presuppone che i dati 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 fre-

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).

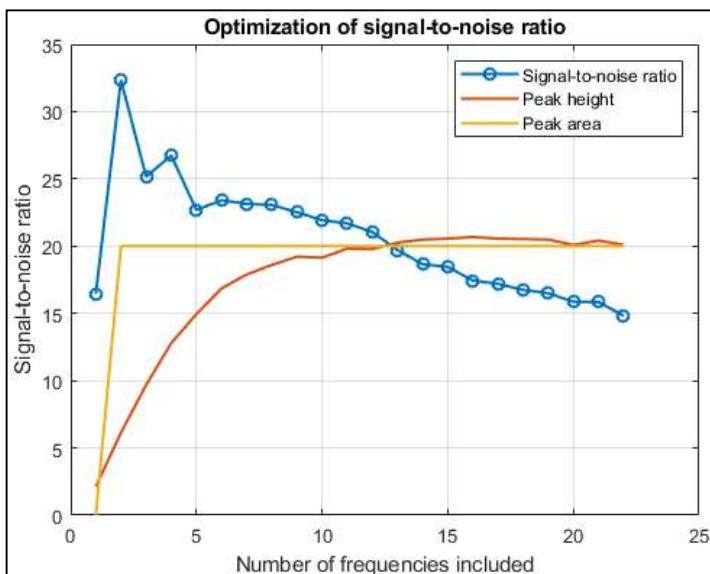
Un esempio più 'drammatico' è mostrato nella figura a lato ([script](#)). In questo caso, il segnale (in alto a sinistra) sembra essere solo rumore casuale ad alta frequenza e il suo spettro di Fourier (in alto a destra, mostrato con una scala logaritmica) mostra che le componenti ad alta frequenza



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



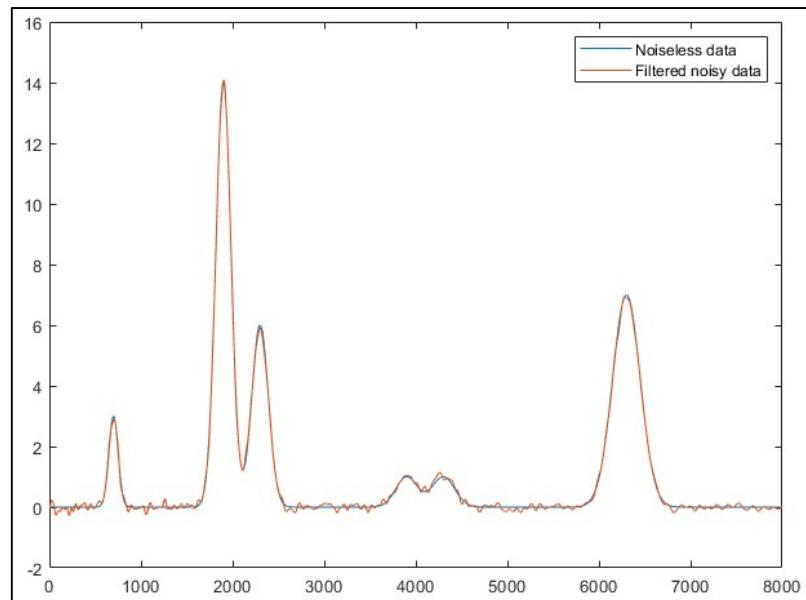
MATLAB. Il codice più semplice possibile per un filtro di Fourier elimina semplicemente tutte le frequenze al di sopra di un certo limite. Per farlo correttamente, è necessario prestare attenzione a utilizzare sia la componente seno che quella coseno (o equivalentemente la frequenza e la fase o la parte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

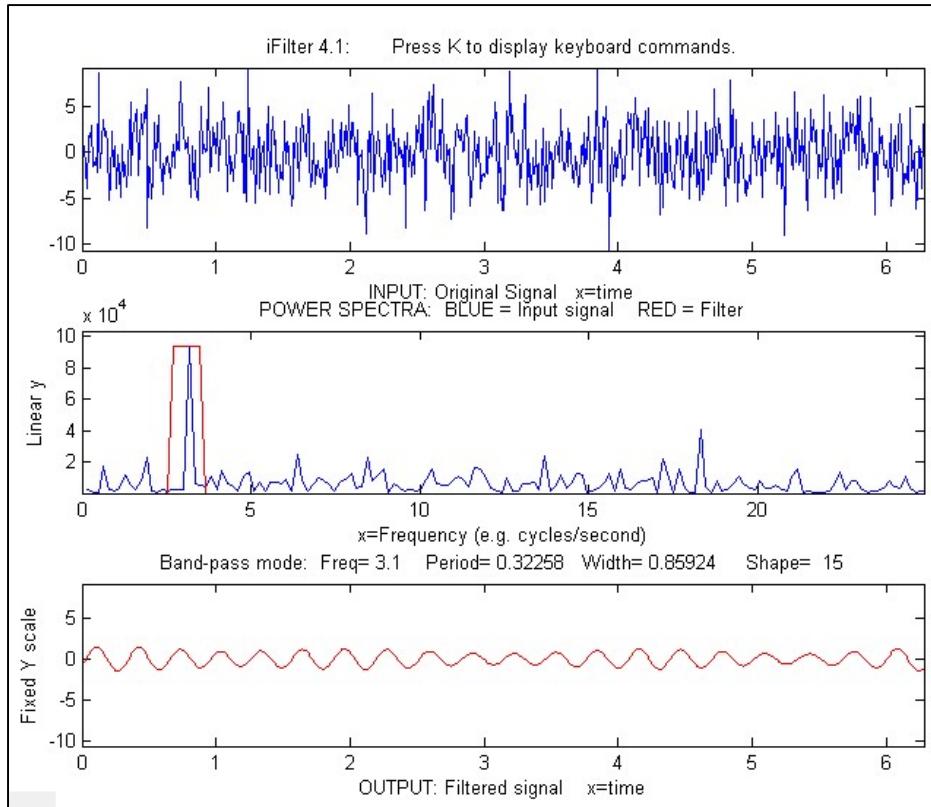
```
ffty=fft(y); % ffty is the fft of y
lfft=length(ffty); % Length of the FFT
```

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 a in seguente a lato 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.

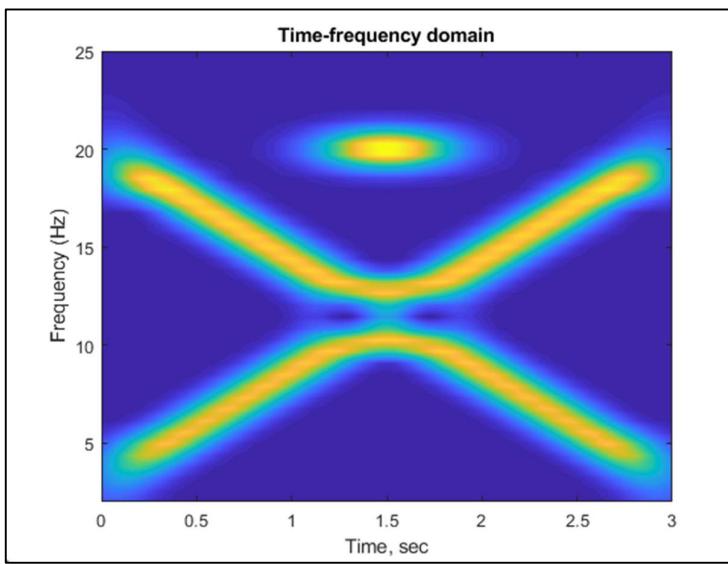
Wavelet e wavelet denoising

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 97 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. Però, 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.

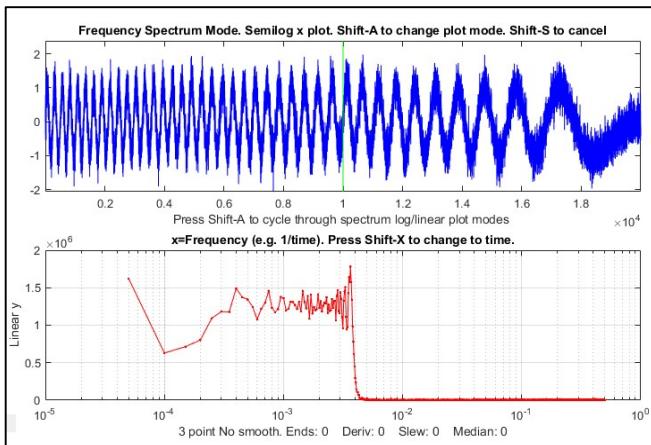
Una [trasformata wavelet](#) (WT) è una decomposizione di un segnale in un insieme di “funzioni base” costituite da contrazioni, espansioni e traslazioni di una funzione wavelet (rif. 85). Può essere calcolata mediante una ripetuta convoluzione del segnale (pag. 103) con la wavelet scelta, mentre viene 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



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'interessante 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).



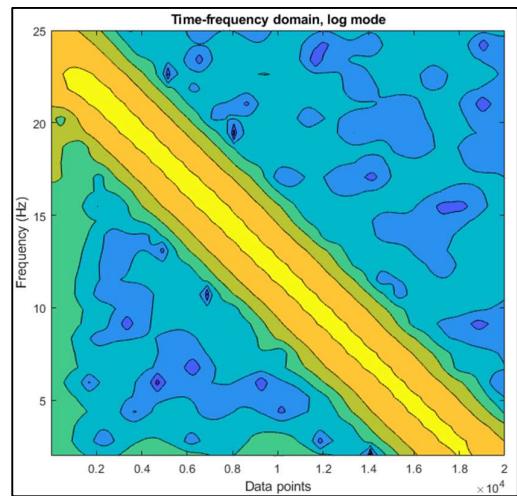
offre alcun indizio evidente dei picchi sottostanti.

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 98. (Si può confrontare questo

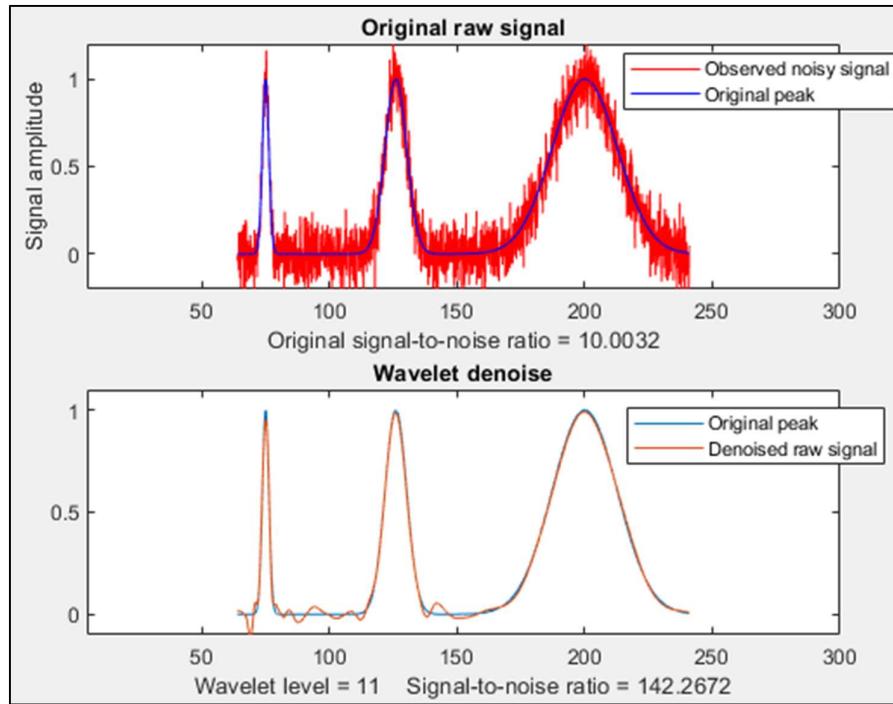
intorno ai 12 Hz. Ma in realtà ci sono *tre* componenti, due delle quali coprono un'ampia gamma di frequenze e la terza è fissa a circa 12 Hz. Lo spettro di Fourier qui non aiuta.

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

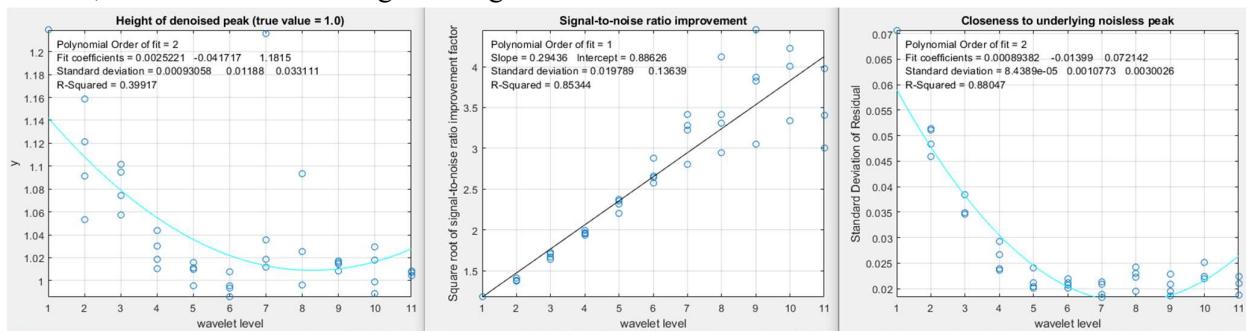
Un altro esempio è più vicino a una tipica applicazione scientifica: estrarre un segnale da un eccesso di rumore e interferenza. Si basa sul segnale dei "picchi sepolti" utilizzato in precedenza, a pagina 98. Il segnale (pannello superiore nella schermata iSignal a lato) ha una coppia di picchi Gaussiani 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



(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 328). 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).

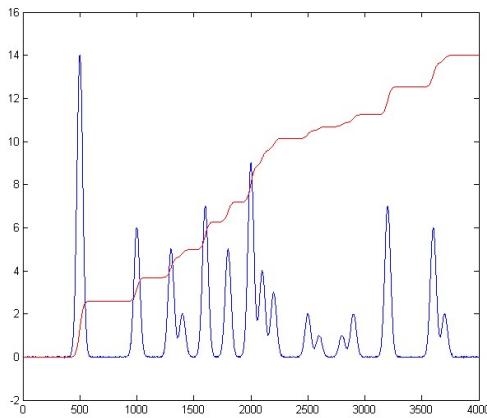
Lo script [WaveletsComparison.m](#) confronta cinque diversi tipi di wavelet con lo stesso segnale: BlockJS, bior5.5, coif2, sym8, e db4, tutte a livello di scala di 12 ([grafico](#)). I risultati sono simili ma il sym8 ha un leggero vantaggio. Per la maggior parte delle forme di picchi con del rumore bianco

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.

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 297: [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 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;
- (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).

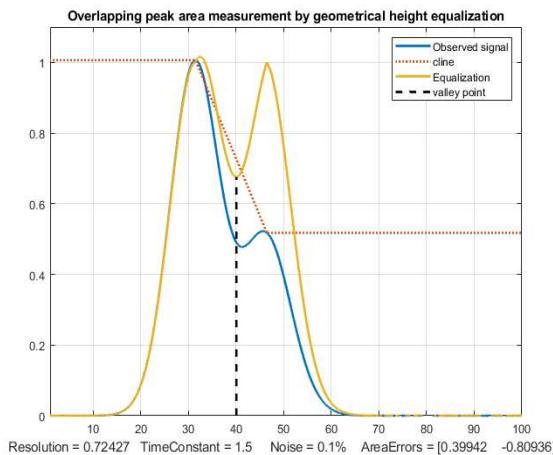
Il metodo migliore per calcolare l'area di un picco dipende dal fatto che il picco sia isolato o sovrapposto ad altri picchi o se sia sovrapposto ad una linea di base diversa da zero. Per un picco isolato, Yuri Kalambet (riferimento 72) ha dimostrato che l'area della [Regola del Trapezio](#), come calcolata 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

(script Matlab [GLSDPerpDropDemo16.m](#)). Consultare anche l'appendice a pagina 354 per un altro esempio.

Altri metodi. Sebbene il metodo del taglio verticale rimanga lo standard, ci sono altri due metodi



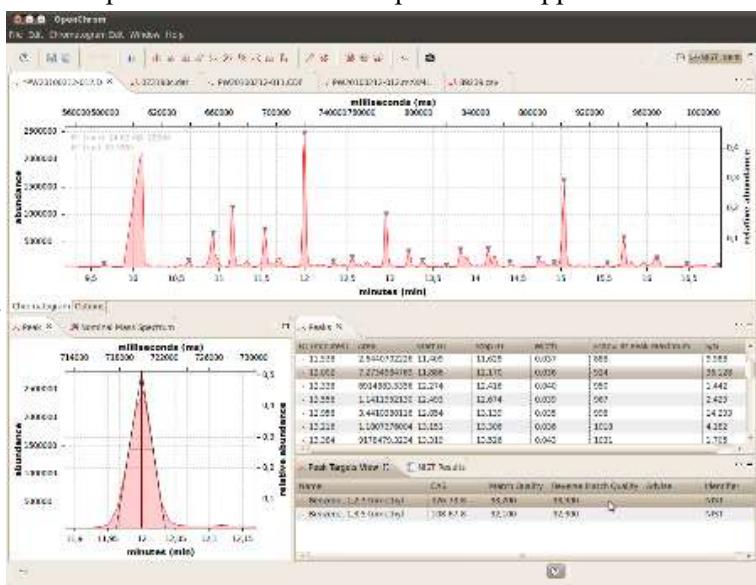
geometrici che possono funzionare meglio in qualche caso. Il metodo della "equalizzazione", illustrato nella figura a lato, localizza il punto del taglio verticale. Viene costruito un insieme di tre segmenti rettilinei che toccano i massimi stimati dei due picchi, indicati dalla linea rossa tratteggiata 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 avallamenti 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'avallamento 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 166. 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



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.xls \(grafico\)](#) esegue la simmetrizzazione dei picchi espansi esponenzialmente mediante l'aggiunta ponderata della derivata prima. Vedere pagina 76. [PeakSymmetrizationExample.xls](#) 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.xls \(grafico\)](#).

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 `yi=sum(y).*dx`. In alternativa, il segnale può essere integrato usando `yi=cumsum(y).*dx`, quindi l'area del picco sarà uguale [all'altezza del gradino risultante](#), `max(yi)-min(yi)=1.7725`.

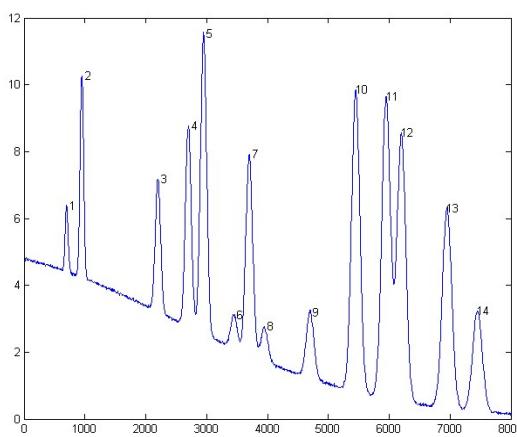
L'area di un picco è proporzionale al prodotto tra la sua altezza e la sua larghezza, ma la costante di proporzionalità dipende dalla forma del picco. Un picco Gaussiano puro con un'altezza *h* e [la larghezza a metà altezza](#) *w* ha un'area totale di $1.064467 * h * w$. Un picco Lorentziano ha un'area 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

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).

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 ciò nonostante 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**

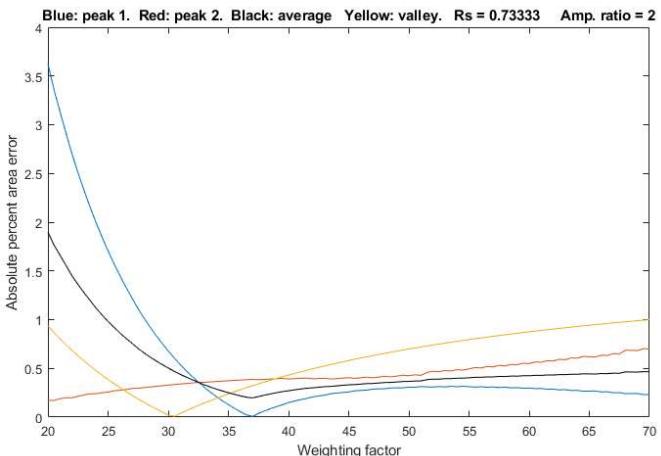
nella finestra di comando, disegnando e numerando i picchi (Window 1) e ciascun singolo picco (Window 2); richiede [gaussian.m](#) e [fastsmooth.m](#) nel path di Matlab. Autopeaks.m e autopeaksplot.m restituiscono una matrice **M**, che elenca ogni picco rilevato nelle righe, nello colonne ha le seguenti misure:

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

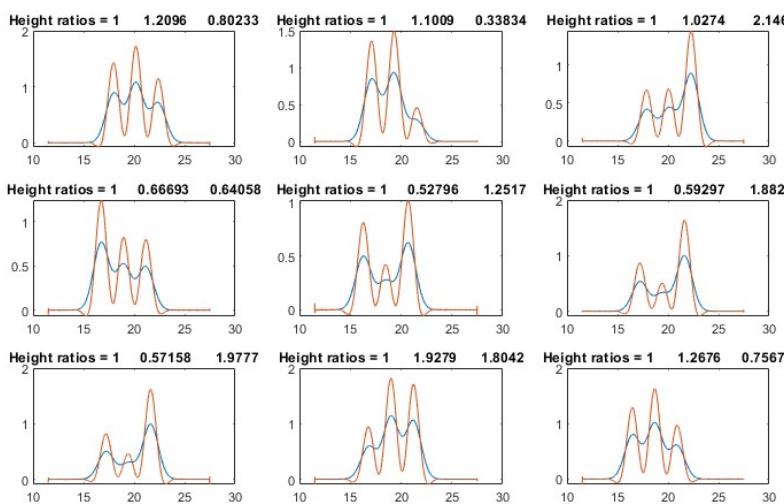
Per determinare l'effetto dello smoothing, dello sharpening, della deconvoluzione o di altri metodi di miglioramento del segnale sulle aree dei picchi sovrapposti misurate col metodo del taglio verticale, la funzione Matlab/Octave [ComparePDAreas.m](#) usa [autopeaks.m](#) per misurare le aree dei picchi 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 73) 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

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 avvallamenti 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'avvallamento 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

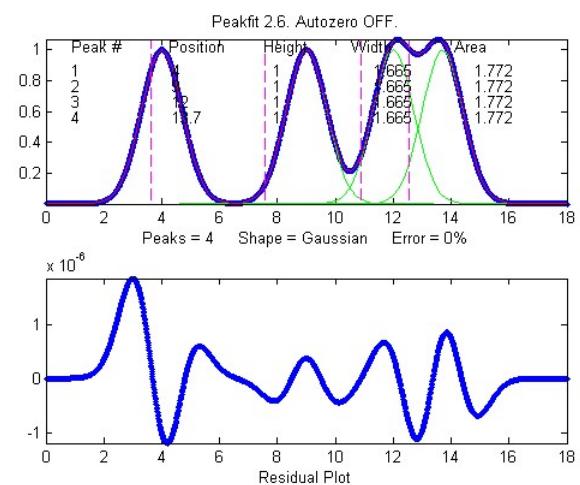
I risultati dell'area sono ragionevolmente accurati in questo esempio solo perché il metodo del taglio verticale compensa approssimativamente la sovrapposizione parziale tra i picchi, ma solo se i picchi sono simmetrici, all'incirca uguali in altezza e senza background.

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).

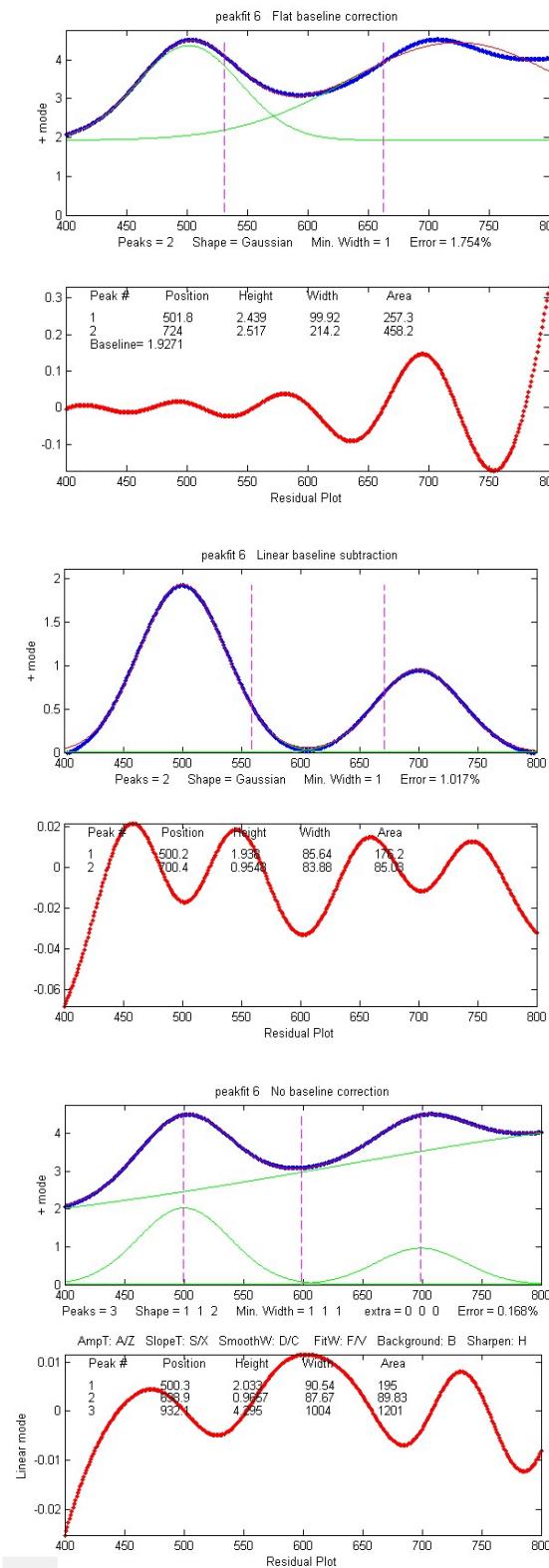
Misura dell'area col curve fitting iterativo

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
```



Una semplice applicazione di iSignal, utilizzando la modalità di base 1 e il metodo del 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.



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:

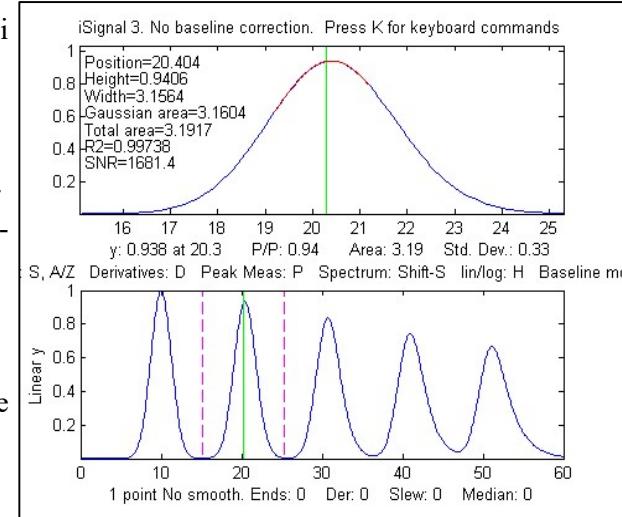
```
>> x=5:.1:65;
>> y=modelpeaks2(x, [1 5 5 5 5], [1 1 1 1 1], [10 20 30 40 50], [3 3 3 3
3], [0 -5 -10 -15 -20]);
>> isignal(x,y);
```

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



```

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 al 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 76.

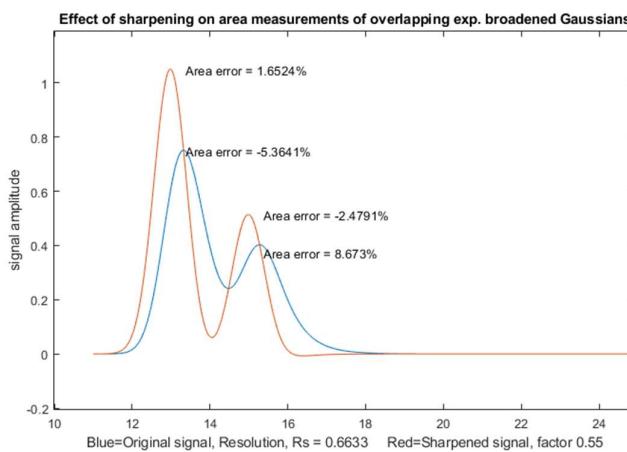
Successivamente, si crea una sfida [ancora più difficile](#) con altezze diverse dei picchi (1, 2, 3, 4 e 5, rispettivamente) e un p' di *rumore casuale addizionale*. Le aree teoriche (Height*Width*1.0645) sono 3.1938, 6.3876, 9.5814, 12.775 e 15.969.

```
>> 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
```

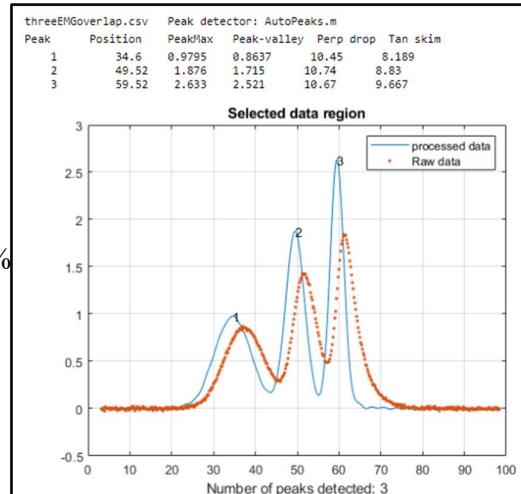
Le aree misurate in questo caso (l'ultima colonna) sono prossime ai valori teorici, mentre tutti gli altri metodi danno una precisione notevolmente inferiore. Maggiore è la sovrapposizione tra i picchi e più disuguali sono le altezze, peggiore è l'accuratezza dei metodi del taglio verticale e della costruzione 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*.



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 esponzialmente. Disegna e confronta i picchi originali (in blu) e quelli con sharpening (in rosso), poi prova i fattori 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

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 76) 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



di

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 191.

"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.

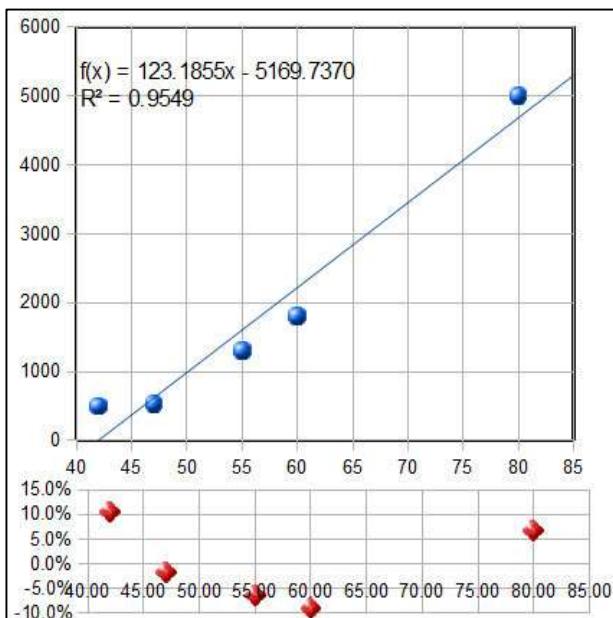
Un'altra cosa: non è corretto dire "approssima i dati a ..." una linea retta o un altro modello; in realtà è il contrario: si sta approssimando un *modello ai dati*. I *dati* non vengono in alcun modo modificati; è il *modello* che viene regolato per approssimare i dati. (In effetti, in qualche caso speciale può risultare utile trasformare i dati prima del 'curve fitting'; vedere pagina 165).

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 pubblicizzato nel numero del 19 febbraio del 2012 del New York Times. (Sì, lo so, 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**

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?



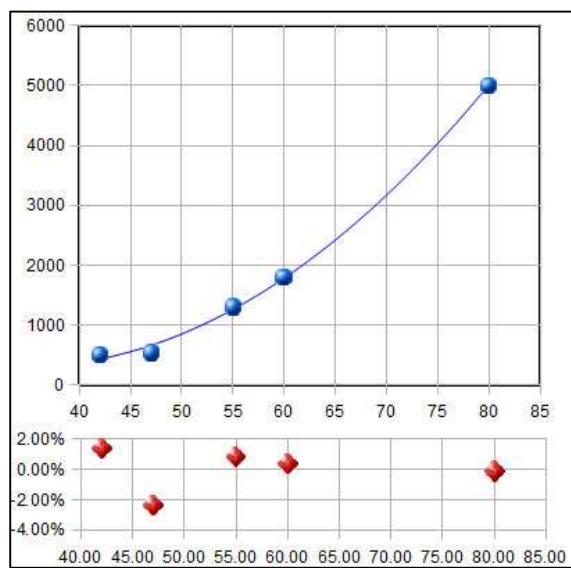
Ecco un altro esempio correlato: i prezzi storici dei televisori LCD a schermo piatto standard ad alta definizione (non UHD) in funzione delle dimensioni dello schermo, come pubblicizzato sul Web nella primavera del 2012 I prezzi dei cinque modelli selezionati, *simili ad eccezione della dimensione dello 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. Pagherebbero 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?)

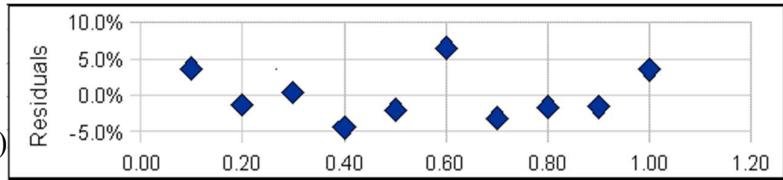
I calcoli ai quadrati minimi possono approssimare i dati non solo a linee rette, ma a *qualsiasi insieme di dati descrivibili da una polinomiale*, per esempio un'equazione del secondo ordine (quadratica)

($Y = a + bX + cX^2$). 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 non 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 a quadrato della misura lineare, quindi l'inclusione di un termine X^2 nel modello è più che ragionevole in questo caso. Con questa approssimazione quadratica, gli apparecchi da 40



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*).

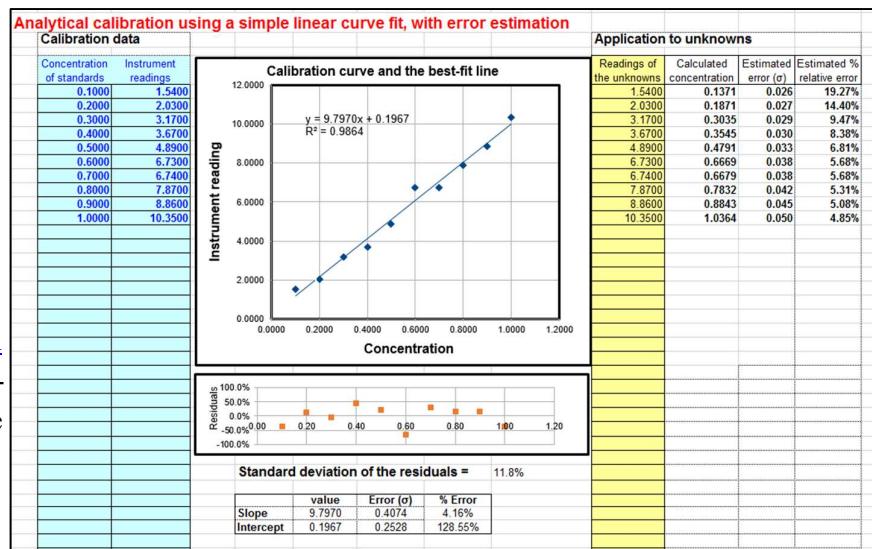
Risolvere l'equazione di calibrazione per la concentrazione. Una volta stabilita la curva di calibrazione, questa può essere utilizzata per determinare le concentrazioni di campioni ignoti misurati sullo stesso strumento, ad esempio *risolvendo l'equazione per la concentrazione come una funzione della lettura dello strumento*. Il risultato per il caso lineare è che la concentrazione del campione Cx è dato da $Cx = (Sx - \text{intercetta})/\text{pendenza}$, dove Sx 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 quadratica, 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).

sente 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.

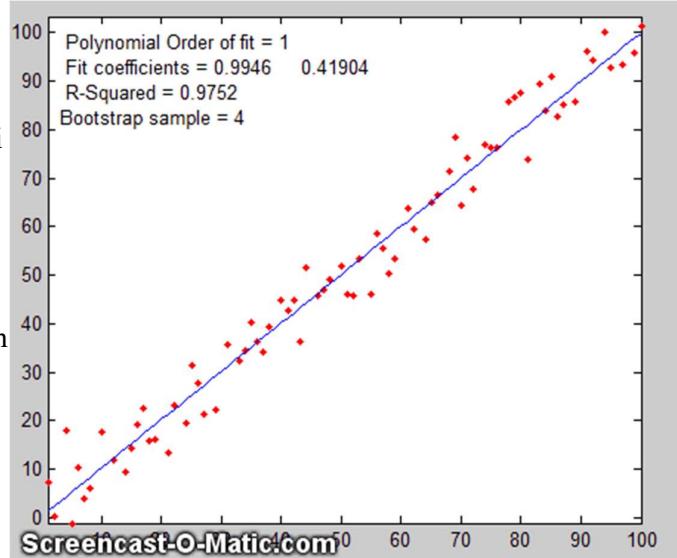
Simulazione Monte Carlo

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

sentato 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 ignote, 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 ricadrebbe *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 sotto-campione ha un sottoinsieme diverso del rumore casuale. Ciò è illustrato dall'animazione a lato (visualizzabile scaricando la versione [Microsoft Word 365](#), altrimenti [clickare 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, maggiore 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.

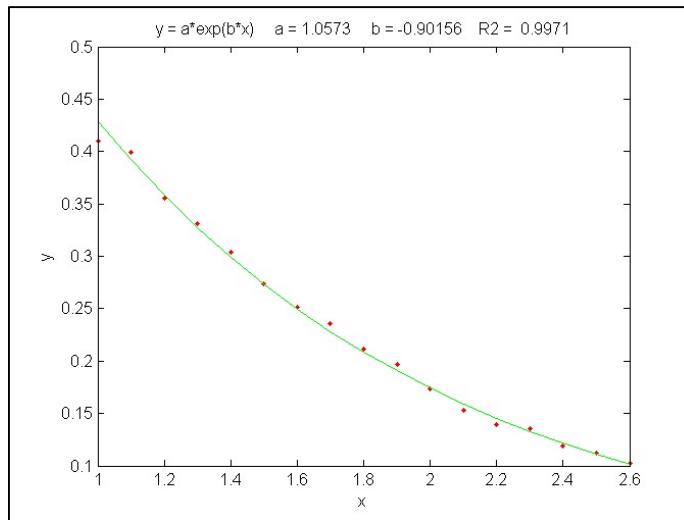
Lo script Matlab/Octave [TestLinearFit.m](#) confronta *tutti e tre* questi metodi (simulazione Monte Carlo, il metodo algebrico e il bootstrap) per un'approssimazione ai quadrati minimi di un set lineare del primo ordine di 100-punti. Ogni metodo viene ripetuto su diversi set di dati con la stessa pendenza 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
```

[renziazione](#) 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).

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 $3*31=93$, rispetto al valore reale di 100. Se si tentasse di calcolare l'*area* del picco da queste misure, si otterrebbe $1.064467*95*93=9404.6$, molto inferiore al valore teorico di $1.064467*\text{height}*\text{width}=10644.67$. 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 (mostrato 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. 191).

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: $\mathbf{h=4*a}/((4*a*c)-b^2)$, $\mathbf{p=-b/(2*a)}$, and $\mathbf{w=sqrt(((4*a*c)-b^2)/a)}/\sqrt{a}$. 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).

punti ([pagina 213](#)). 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.

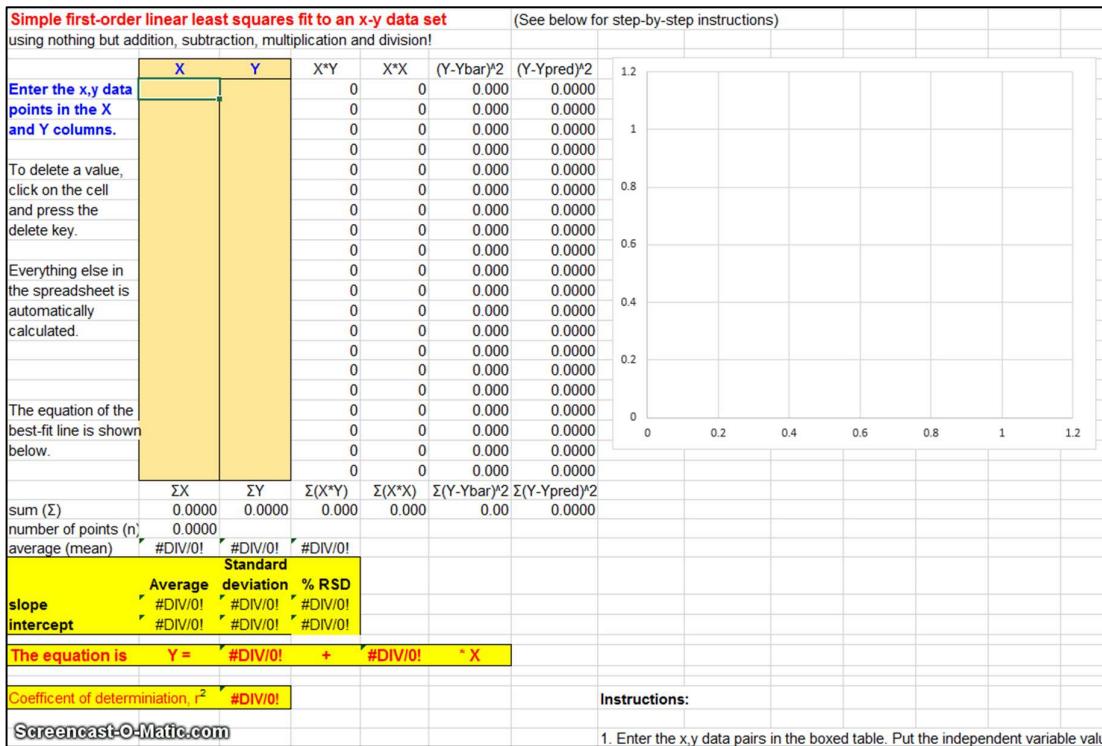
Nota 2: Nelle tecniche di approssimazione della curva descritte qui e nei prossimi due capitoli, non è richiesto che l'intervallo dell'asse x tra i punti sia uniforme, come si presume in molte altre tecniche di elaborazione del segnale descritte in precedenza. Gli algoritmi di curve fitting accettano 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

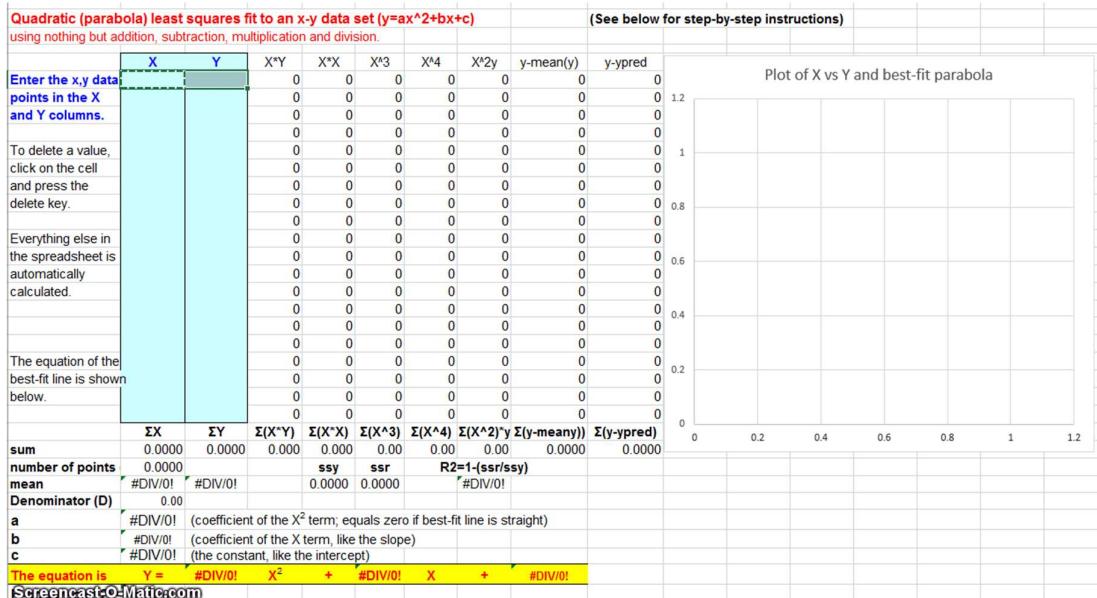
L'approssimazione ai quadrati minimi per un set di dati x,y si può calcolare utilizzando solo l'aritmetica di base. Di seguito sono riportate le equazioni rilevanti per il calcolo della pendenza e dell'intercetta dell'equazione di approssimazione del primo ordine, $y = \text{intercetta} + \text{pendenza} \cdot x$, 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).

$n = \text{numero dei punti } x, y$ $\text{sumx} = \Sigma x$ $\text{sumy} = \Sigma y$

zioni 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

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, `fi-ty=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](#) `sqrt(diag(inv(S.R)*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 (quadratico) da `y/[x.^2;x;ones(size(y))]`. Per polinomi di ordine superiore, basta aggiungere un'altra riga

```

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 13: `[coeff,R2]=plotit(y1,y2,1)`;

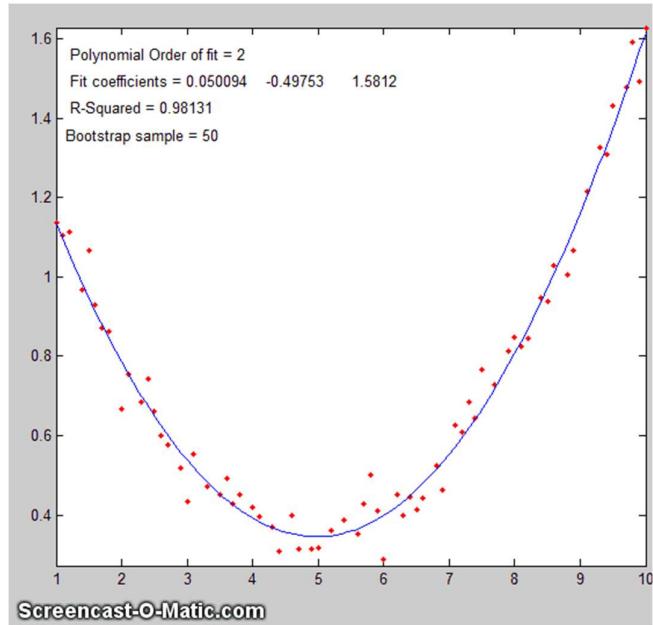
R^2 è una misura della somiglianza. Più R^2 è prossimo a 1.000, più sono simili. Se y_1 e y_2 sono due misurazioni dello stesso segnale con un diverso rumore casuale, il grafico mostrerà una dispersione casuale di punti lungo una linea retta con una pendenza, $\text{coeff}(1)$, di 1.00. Se y_1 e y_2 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 `data-style` e `fitstyle` sono stringhe opzionali che specificano lo stile e il colore della linea e del

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:

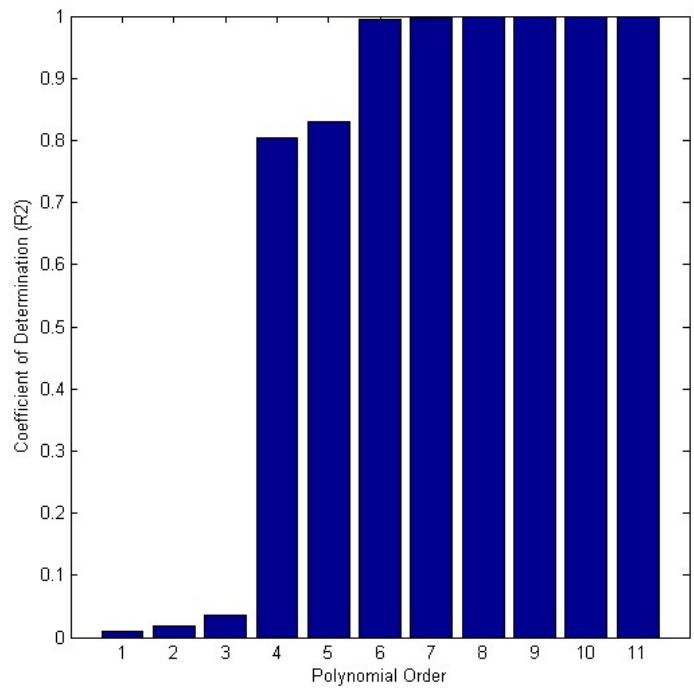


```
[coef, RSquared, BootResults]=plotfita([1 2 3 4 5 6],[1 3 4 3 2 1],2);
```

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 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 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 polyorder=1, è il 5° ad essere il migliore, ovvero x rispetto a ln(y). Un esempio è



```
Bootstrap IQR: 1.7692 0.86874 2.9735
Percent RSD: 1.3346 0.62266 2.1035
Percent IQR: 1.7543 0.85737 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. 191) utilizzando peakfit.m, che è [più accurato ma richiede più tempo per i calcoli](#).

Le funzioni solo-per-Matlab [iSignal.m](#) (pag. 357) e [ipf.m](#) (pag. 394), 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".

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 una *matrice* (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, **Σ** è la matrice rettangolare $n \times w$ dei valori noti di **Σ** 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. **ΣC** significa che **Σ** “pre-moltiplica” **C**; ovvero, *ogni colonna di Σ 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 **Σ**, 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 **Σ** 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 **Σ** 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 **Σ^T** è la *trasposta* di **Σ** (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, **Σ^TΣ** 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 **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 **Σ**. Questo metodo di calcolo è chiamato “Classical Least Squares” (Minimi Quadrati Classici) o semplicemente “CLS”, così

e così via per tutti gli s campioni. In forma matriciale

$$\mathbf{C} = \mathbf{A}\mathbf{M}$$

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:

$$\mathbf{C} = \mathbf{A}\mathbf{M}$$

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:

$$\mathbf{C} = (\mathbf{E}^T \mathbf{E})^{-1} \mathbf{E}^T \mathbf{A}$$

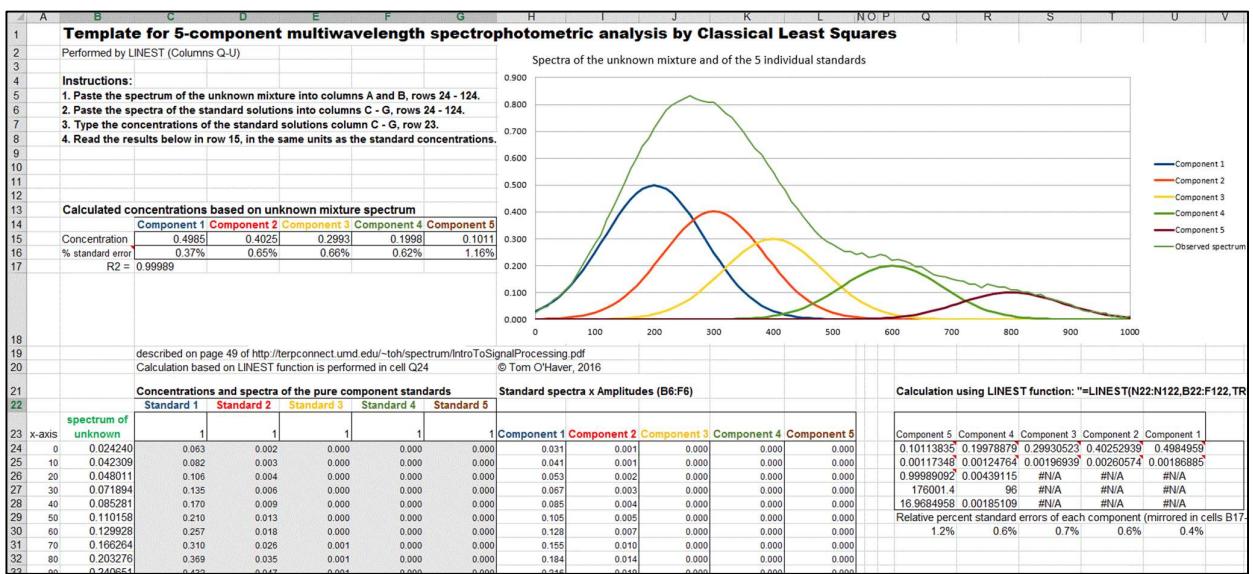
vengono eseguite in questi fogli di calcolo dalle funzioni matriciali TRANSPOSE (trasposta di matrice), MMULT (moltiplicazione di matrice) e MINVERSE (inversa di matrice), disposte passo-passo nelle [righe da 123 a 158 di questo spreadsheet](#). In alternativa, tutte queste operazioni matriciali si possono combinare in un'unica grande equazione in una sola cella, risultando, però, meno leggibile.

$$\mathbf{C} = \text{MMULT}(\text{MINVERSE}(\text{MMULT}(\text{TRANSPOSE}(\mathbf{E}); \mathbf{E})); \text{TRANSPOSE}(\mathbf{E})); \mathbf{A}$$

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 335) 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 335.



Template Excel per la misura di una miscela ignota di 5 componenti a 100 lunghezze d'onda.

Matlab e Octave

Matlab e Octave 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 questi linguaggi, la notazione è molto compatta: la trasposizione della matrice A è A', l'inversa di A è 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

$$C = \text{inv}(E' * E) * E' * A$$

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);
```

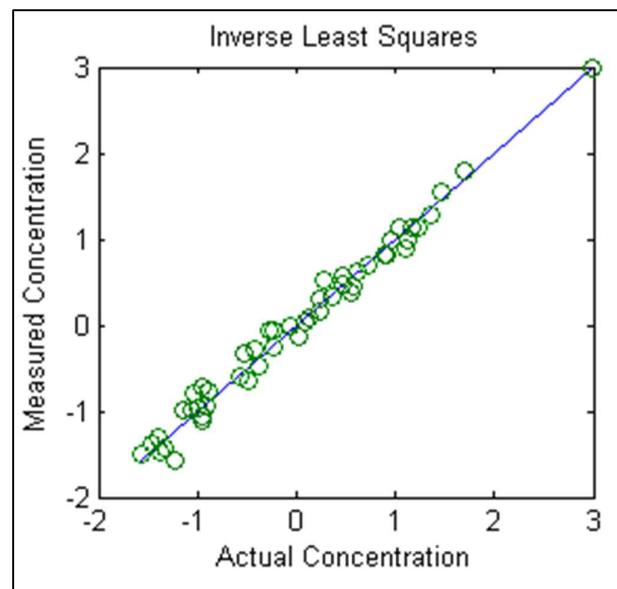
dove T è il vettore dello spettro di trasmissione. Qui, viene usata la divisione matriciale con la barra rovesciata "\\" come scorciatoia per la classica soluzione matriciale dei quadrati minimi (cfr. <http://www.mathworks.com/help/techdoc/ref/mldivide.html>).

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

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, da 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 equivalente 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;
```

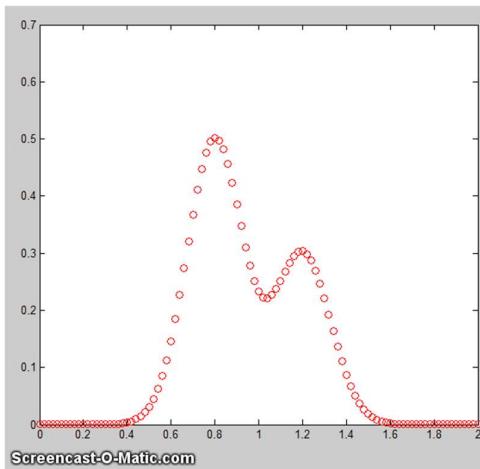
Approssimazione delle curve C: Approssimazione iterativa non-lineare

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 Simplesso 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 mostrata sopra ([script Matlab](#)) mostra il funzionamento del processo iterativo per un'approssimazione di 2 picchi Gaussiani non vincolati a

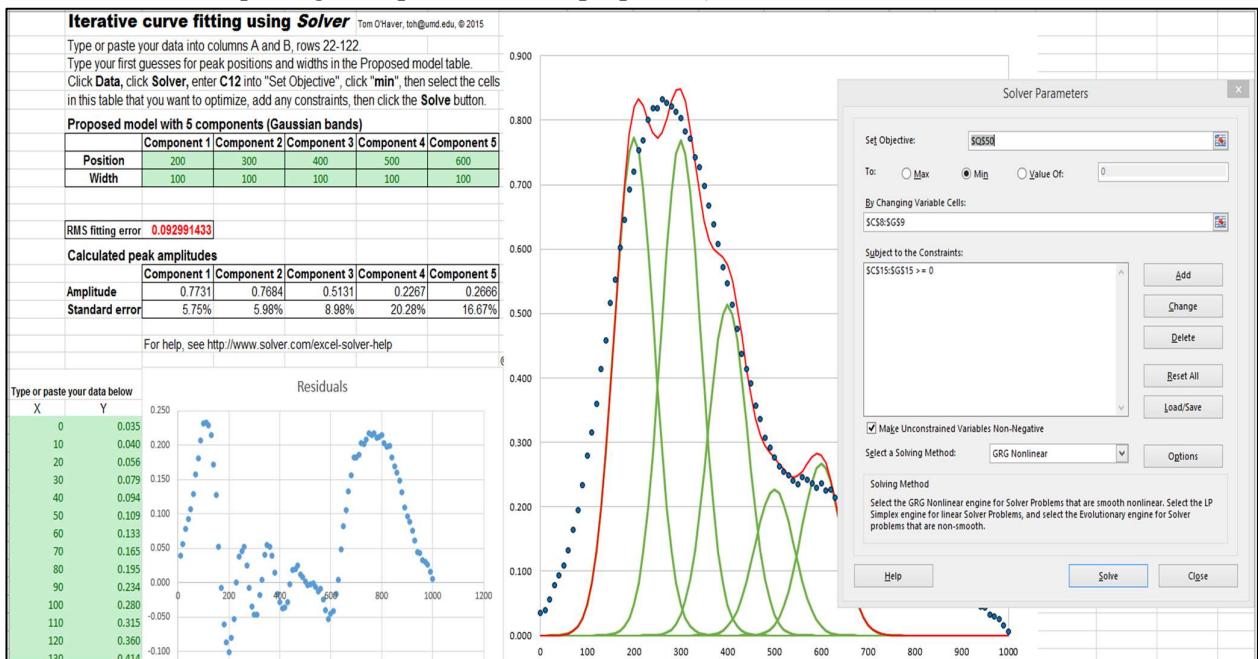
la caratteristica che si vuole misurare. Non è pratico prevedere le deviazioni standard dei parametri del modello misurato utilizzando l'approccio algebrico, ma sono applicabili sia i metodi della [simulazione Monte Carlo che il bootstrap](#) (pag. 162).

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 291.

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

[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 Gaussiana 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 375](#)) *automaticamente* si adatta a qualsiasi numero di punti ed è facilmente impostabile su oltre 40 diverse forme di picchi (grafico a pagina 401) e a qualsiasi numero di picchi, semplicemente cambiando gli argomenti di input. Utilizzando la funzione di *Approssimazione interattiva del Picco*, [ipf.m](#) in Matlab ([pagina 394](#)), si può *premere un solo tasto* per cambiare istantaneamente il profilo, il numero, la modalità della linea di base del picco ([pagina 210](#)), o per ric算olare 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 422 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 fotoelettronica 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

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

una differenza relativa dello 0.3%.

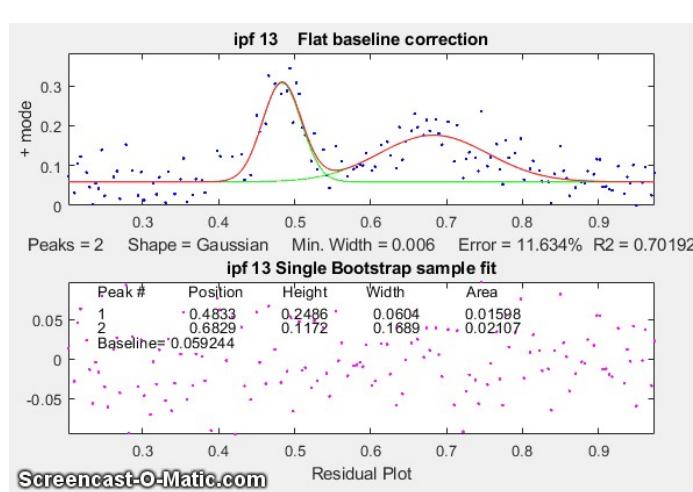
Se *non* si conosce la forma dei picchi, si può usare peakfit.m o ipf.m (pagina 394) per provare profili diversi per vedere se una delle le forme standard incluse in quei programmi approssimano i dati; cercare di trovare un picco nei dati che sia tipico, isolato e che abbia un buon rapporto segnale/rumore. 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. 264), 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

```

>> 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. 40). 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

```

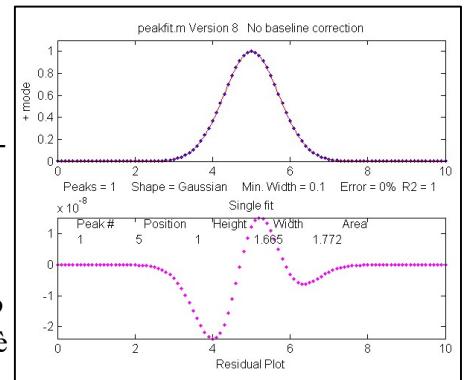
Se viene misurata l'*area* del picco anziché l'*altezza*, non è necessario lo smoothing (a meno che non

```

>> 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
Peak# Position Height Width Area
1 5.0002 0.96652 1.762 1.7419
MeanFitError =1.4095

```

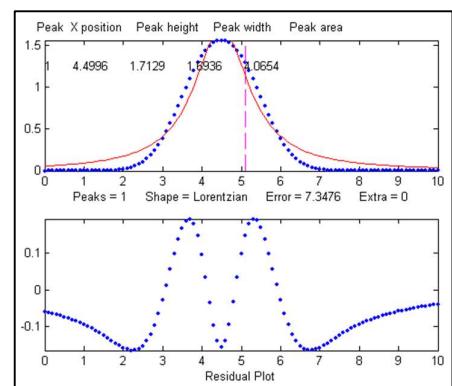
Quando si approssima con un modello *Lorentziano* ancora più sbagliato (profilo numero 2, mostrato a lato, di seguito), questo picco restituisce un errore di approssimazione del 7% e per altezza, larghezza e area, 8%, 20% e 17%, rispettivamente.

```

>> [FitResults,MeanFitError]=peakfit([x' y'],5,10,1,2)
Peak# Position Height Width Area
1 5 1.0876 1.3139 2.0579
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.



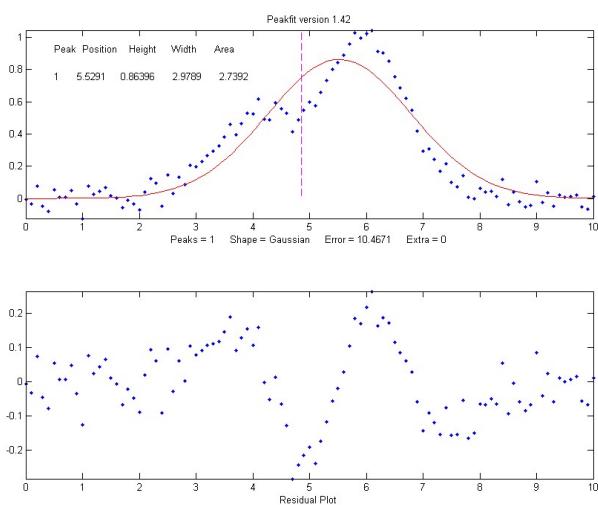
Quindi chiaramente più grandi sono gli errori di approssimazione, maggiori sono gli errori dei parametri, ma gli errori dei parametri ovviamente non sono *uguali* a quello dell'approssimazione (sarebbe *tropppo* facile). Inoltre, l'*altezza* e la *larghezza* sono i parametri

Il numero dei picchi. Un'altra fonte di errore 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):

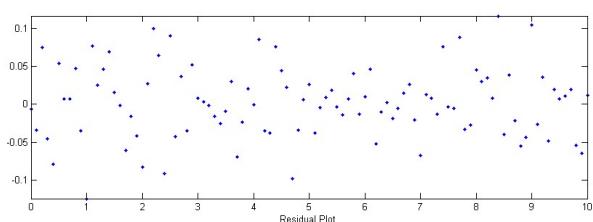
```
>> x=[0:.1:10];
>> y=exp(-(x-6).^2)+.5*exp(-(x-4).^2)+.05*randn(size(x));
```

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
```



```
>> [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
```



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).

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

```

>> 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 4.1115 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)

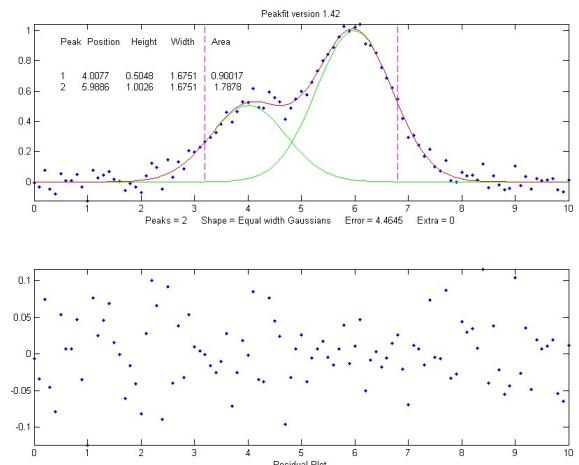
del previsto con questo livello di rumore casuale nel segnale (5%). Se si dovesse eseguire un'approssimazione a 2 picchi sugli stessi nuovi dati, otterremmo misure molto migliori delle altezze.

```

>> [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 rumorosi generati dal computer contenenti 3, 4, 5 o 6 picchi selezionati dall'utente. Con dati molto rumo-



```

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 389) 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 180) 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 errori nelle altezze dei picchi misurati sono spesso *minori*, come in questo esempio tratto da [peakfit9demo.m](#), dove le altezze reali dei *tre picchi Gaussiani sovrapposti* sono 10, 30 e 20.

Risultati della regressione multilineare (posizione e larghezza note) :

```

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

```

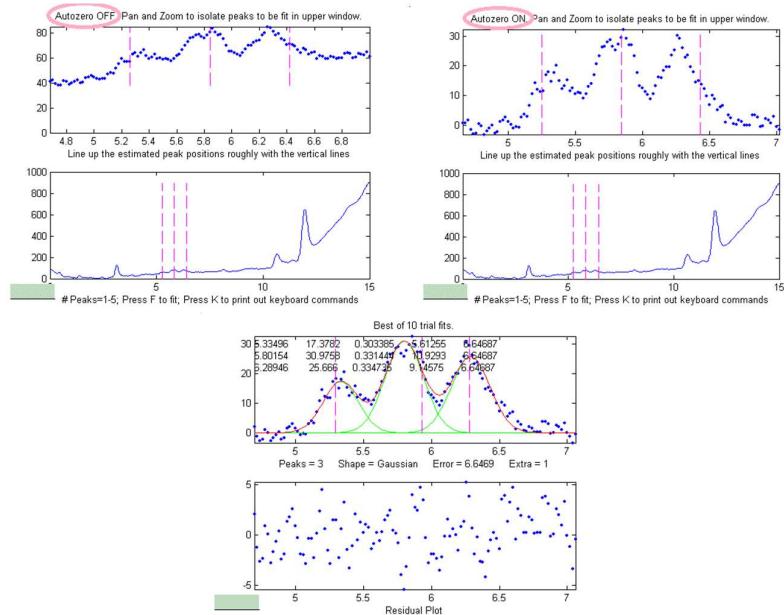
Risultati dei quadrati minimi iterativi non lineari e non vincolati:

```

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

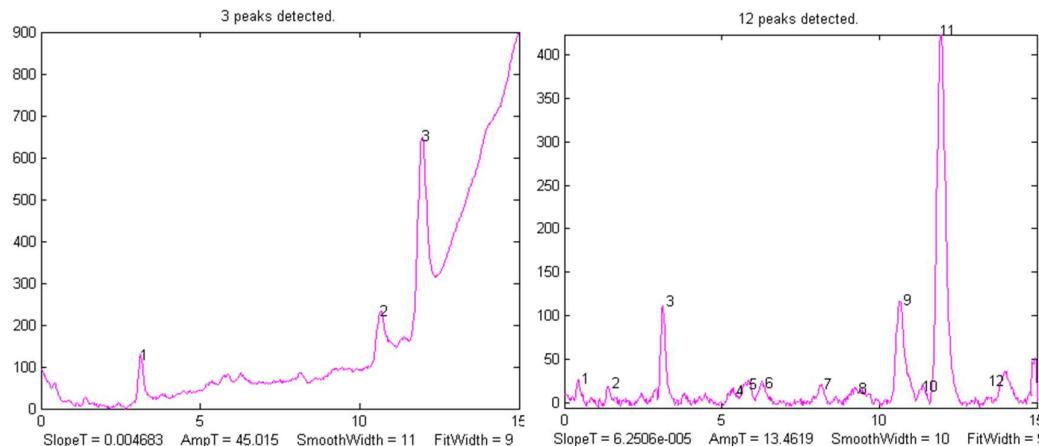
```

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).

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 [CurveFit-ter2GaussianBaseline.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 disperdonano 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 *sovrastimando* 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 parame-

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 $x=150$. En-

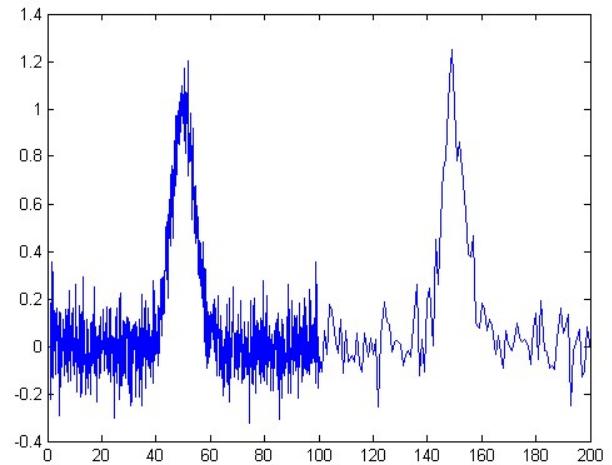
trambi i picchi hanno un'altezza di 1.0 e una semi-larghezza di 10. All'intero segnale è stato aggiunto un rumore bianco casuale normalmente distribuito con una deviazione standard di 0.1. L'*intervallo di campionamento sull'asse x*, tuttavia, è diverso per i due picchi; è 0.1 per il primo e 1.0 per 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 223) 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 rumo-

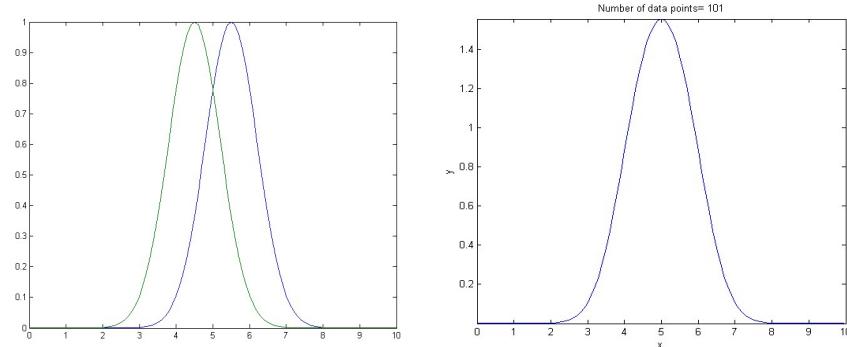


controlli di pan e zoom. Questa è semplicemente un'indicazione dell'inevitabile incertezza nei parametri misurati.

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.

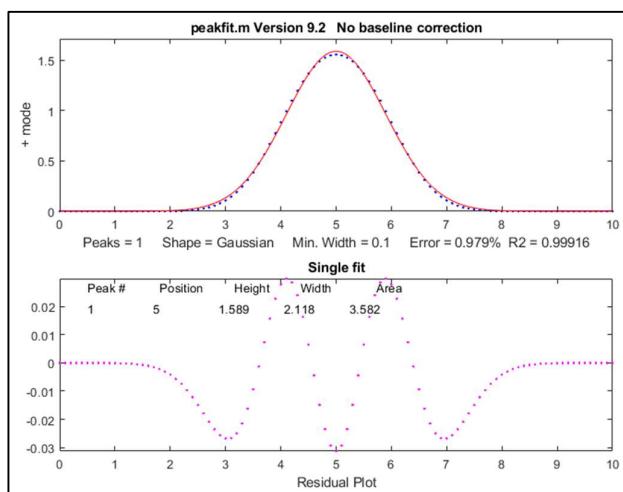
```
>> x=[0:.1:10]';
>> y=exp(-(x-5.5).^2)+exp(-(x-4.5).^2);
```



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:

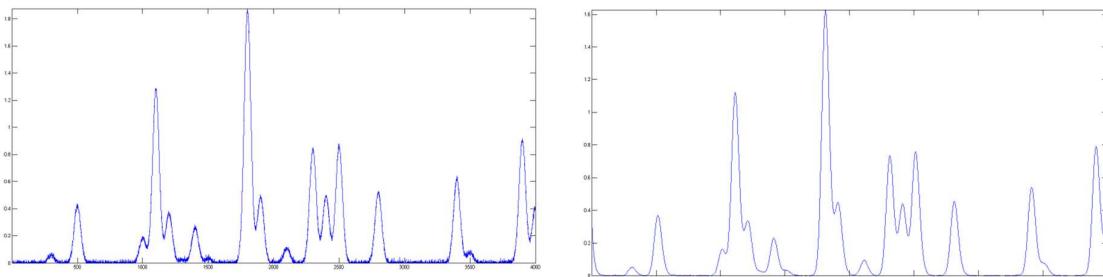
```
>> y=exp(-(x-5.5).^2)+exp(-(x-4.5).^2)+.01*randn(size(x))
>> peakfit([x y],5,19,2,1)
Peak# Position Height Width Area
```

(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 348 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



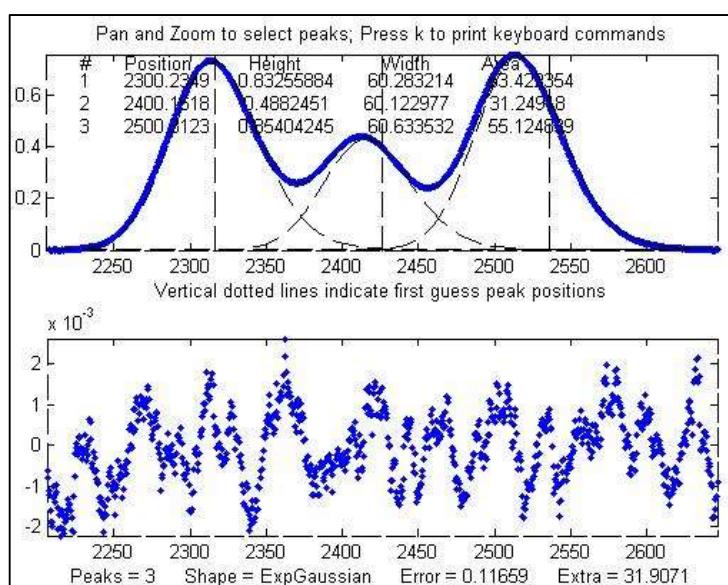
parametri dei picchi. Cliccare con il tasto destro e selezionare "Save" per scaricare questo segnale, inserirlo nel path di ricerca di Matlab, poi digitare "load [DataMatrix2](#)" al prompt dei comandi per caricarlo nello spazio di lavoro di Matlab. [DataMatrix3](#) (in alto a destra) è una versione [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 394), 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 segna-

colata, 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 esponenziale e il *numero dei punti*, il profilo 39 riporta il *reciproco della costante di decadimento* in unità di tempo. (Vedere lo script [DemoExp-gaussian.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

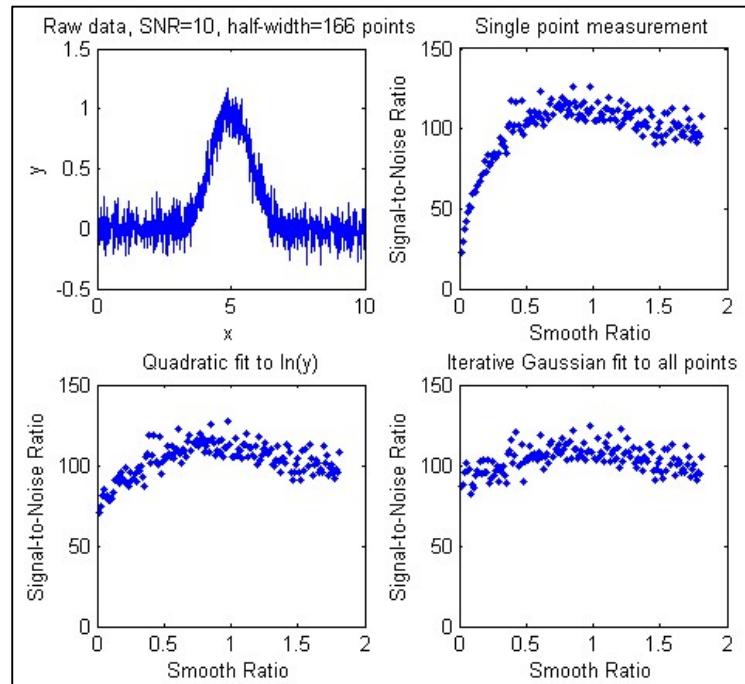


Altri picchi nello stesso segnale, sotto l'influenza di ampliamento della stessa costante di decadimento, possono essere approssimati con impostazioni simili, ad esempio, l'insieme dei tre picchi sovrapposti vicino a $x = 2400$. Come prima, le posizioni dei picchi vengono recuperate quasi esattamente 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 gradualmente all'aumentare di x , l'impostazione della co-

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. 394), 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 76) 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:

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).

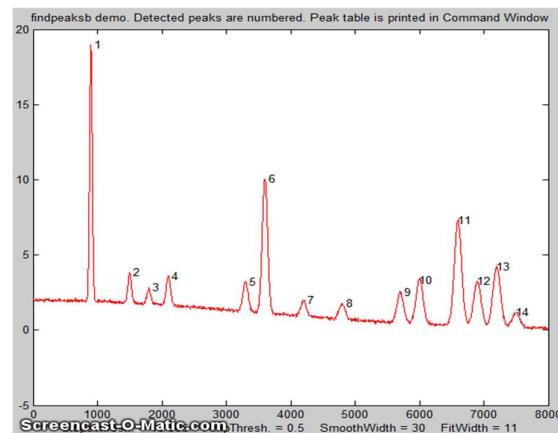
Tuttavia, la presenza di rumore casuale nel segnale sperimentale reale causerà molti falsi passaggi per lo zero semplicemente a causa del rumore. Per evitare questo problema, la tecnica descritta esegue lo [smoothing](#) della derivata prima del segnale, poi cerca i passaggi per lo zero discendenti, 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.m](#), [findpeaksxw.m](#), [findpeaksG.m](#), [findpeaksGw.m](#), (riduzione del rumore basato sulle Wavelet, richiede la funzione "wdenoise" che si trova nel Wavelet Toolbox), [findvalleys.m](#), [findpeaksL.m](#), [measurepeaks.m](#), [findpeaksGd.m](#), [findpeaksb.m](#), [findpeaksb3.m](#), [findpeakspplot.m](#), [findpeakspplotL.m](#), [peakstats.m](#), [findpeaksE.m](#), [findpeaksGSS.m](#), [findpeaksLSS.m](#), [findpeaksT.m](#), [findpeaksfit.m](#), [autofindpeaks.m](#), e [autopeaks.m](#). 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 242, per regolare i criteri di rilevamento del picco in modo interattivo ottimizzato 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*.



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. 40)

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 della velocità, confrontando le funzioni di rilevamento 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 impiegato è 0.11 sec. su un desktop Dell XPS i7 3.5Ghz, ovvero *110,000 picchi al secondo*.

Misura del picco Gaussiano

```
P=findpeaksG(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth,
smoothtype)
```

```
P=findpeaksL(x, y, SlopeThreshold, AmpThreshold, SmoothWidth, FitWidth,
smoothtype)
```

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)
```

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 delle 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:

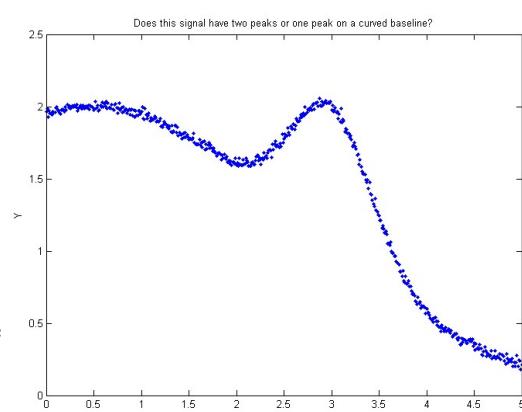
```
x=[0:.1:50];
y=10+10.*sin(x).^2+randn(size(x));
[P,A]=autofindpeaks(x,y,20);
```

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 "sintonizzato" o ottimizzato per i picchi desiderati. Per esempio, lo script dimostrativo Matlab/Octave [OnePeakOrTwo.m](#) crea un segnale (mostrato a lato) che



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, si crea 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

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).

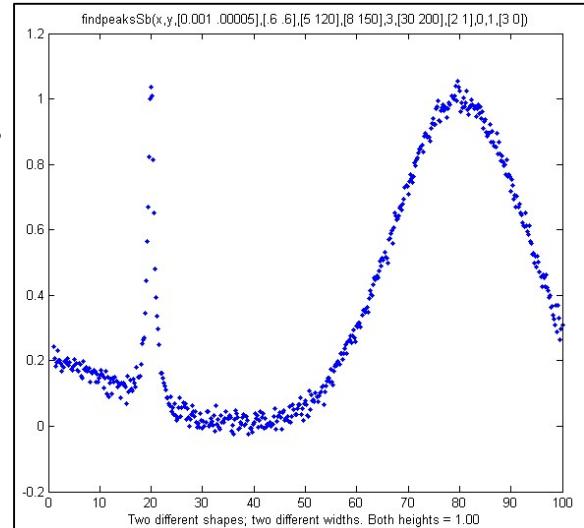
```
x=1:.2:100;Heights=[1 2 3];Positions=[20 50 80];Widths=[3 3 3];
y=2-(x./50)+modelpeaks(x,3,1,Heights,Positions,Widths)+.02*randn(size(x));
plot(x,y);
disp(' Peak Position Height Width Area % error')
PlainFindpeaks=findpeaksG(x,y,.00005,.5,30,20,3)
NoBackgroundSubtraction=findpeaksb(x,y,.00005,.5,30,20,3,150,1,0,0)
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 la 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*.



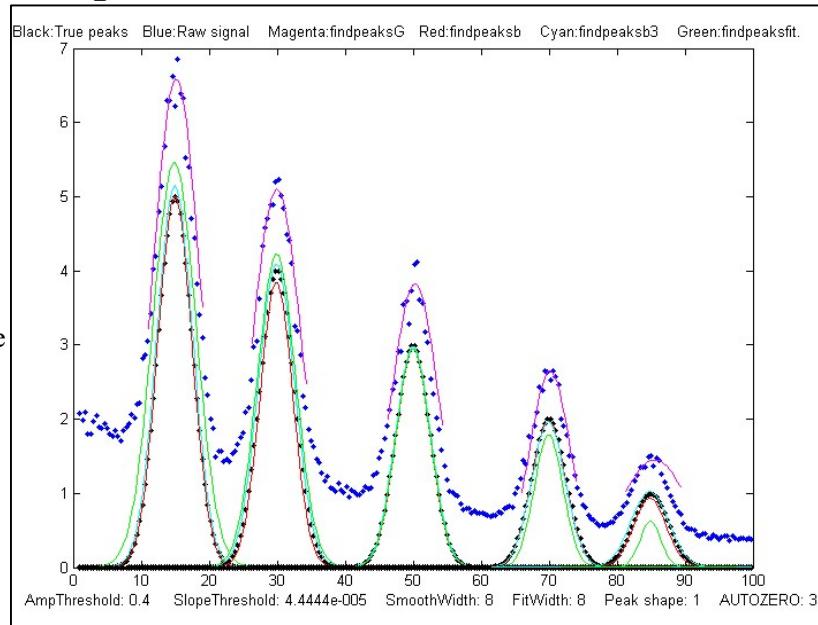
funzione [peakfit](#); se è stata utilizzata peakfit.m o [ipf.m](#) (pagina 394) per approssimare i picchi nei segnali, allora si possono utilizzare gli stessi valori per gli argomenti di input di findpeaksfit.m. Lo script dimostrativo [findpeaksfitdemo.m](#) è stato utilizzato per generare [l'animazione 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
findpeaksG  0.35955% 38.573%  25.797%  0.005768
findpeaksb  0.38828% 8.5024%  14.329%  0.069061
findpeaksb3 0.27187% 3.7445%    3.0474%  0.49538  findpeaksfit  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,

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);
```

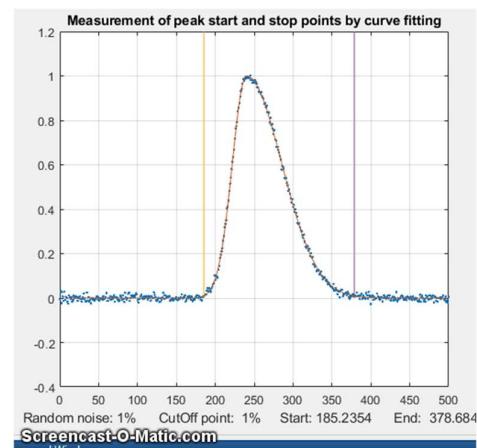
FindpeaksE.m è una variante di findpeaksG.m che stima inoltre l'errore di approssimazione relativo percentuale di ciascun picco (assumendo una forma Gaussiana del picco) e lo restituisce nella 6a colonna della tabella dei picchi.

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...
```

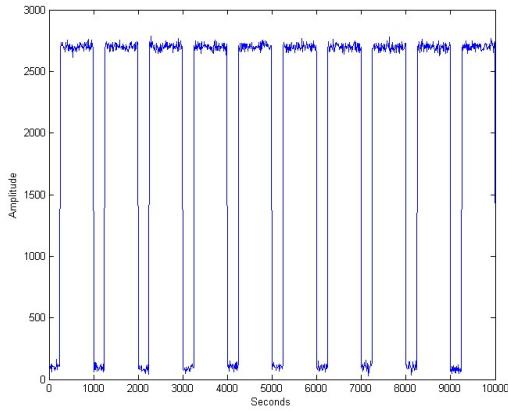
Inizio e fine del picco

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,



[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,`



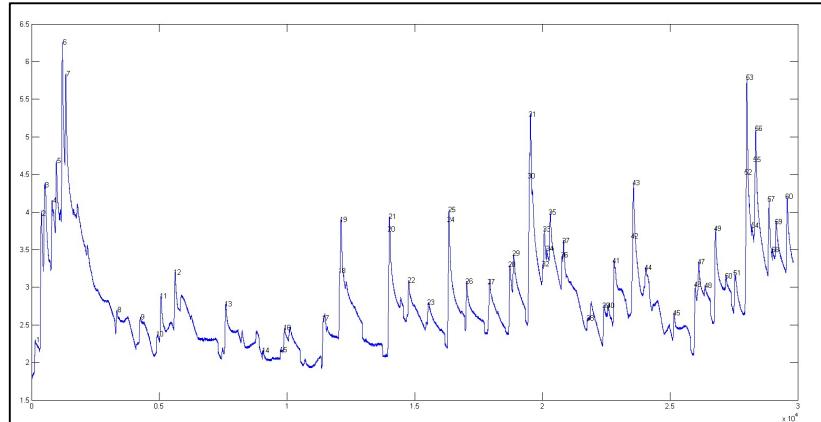
`AmpThreshold, SmoothWidth, peakgroup)`, localizza i transitori positivi in serie temporali rumorose x-y, calcolando la derivata prima di y che supera "SlopeThreshold", calcola l'altezza del transitorio [step] come la differenza tra il valore massimo e il minimo di y su un numero di punti pari a "Peakgroup" 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

Script dimostrativi

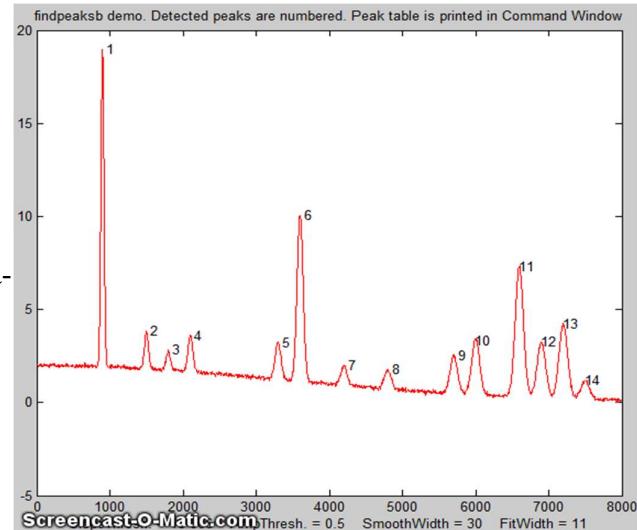
[DemoFindPeak.m](#) è un semplice script dimostrativo che utilizza la funzione [findpeaksG](#) su dati sintetici rumorosi. La funzione numera i picchi e stampa la tabella dei picchi nella finestra di comando Matlab:

```
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 sovrapposti 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

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 losharpening 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.

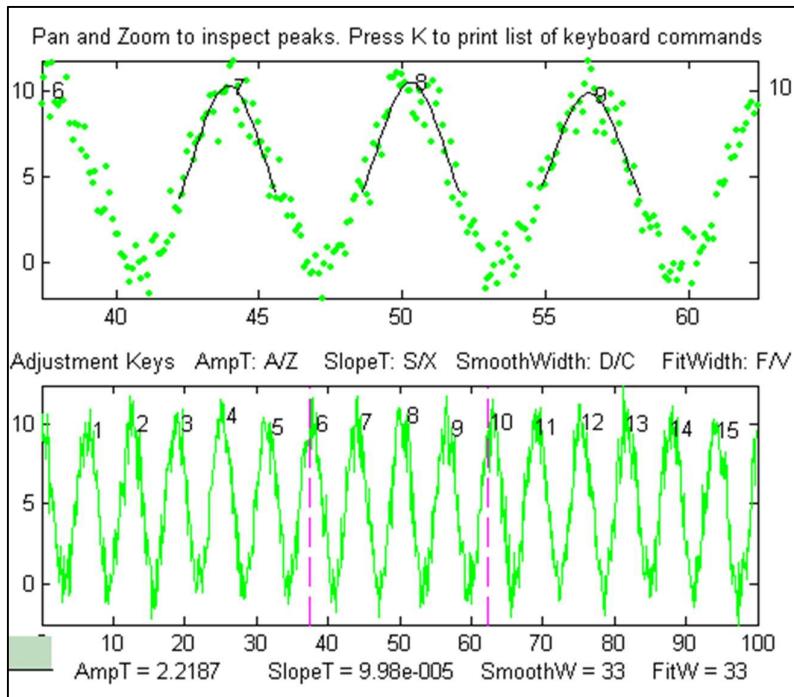
```
WidthPoints=length(y)/20;
SlopeThreshold=WidthPoints^-2;
AmpThreshold=abs(min(y)+0.1*(max(y)-min(y)));
SmoothWidth=round(WidthPoints/3);
FitWidth=round(WidthPoints/3);
```

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)
```



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).

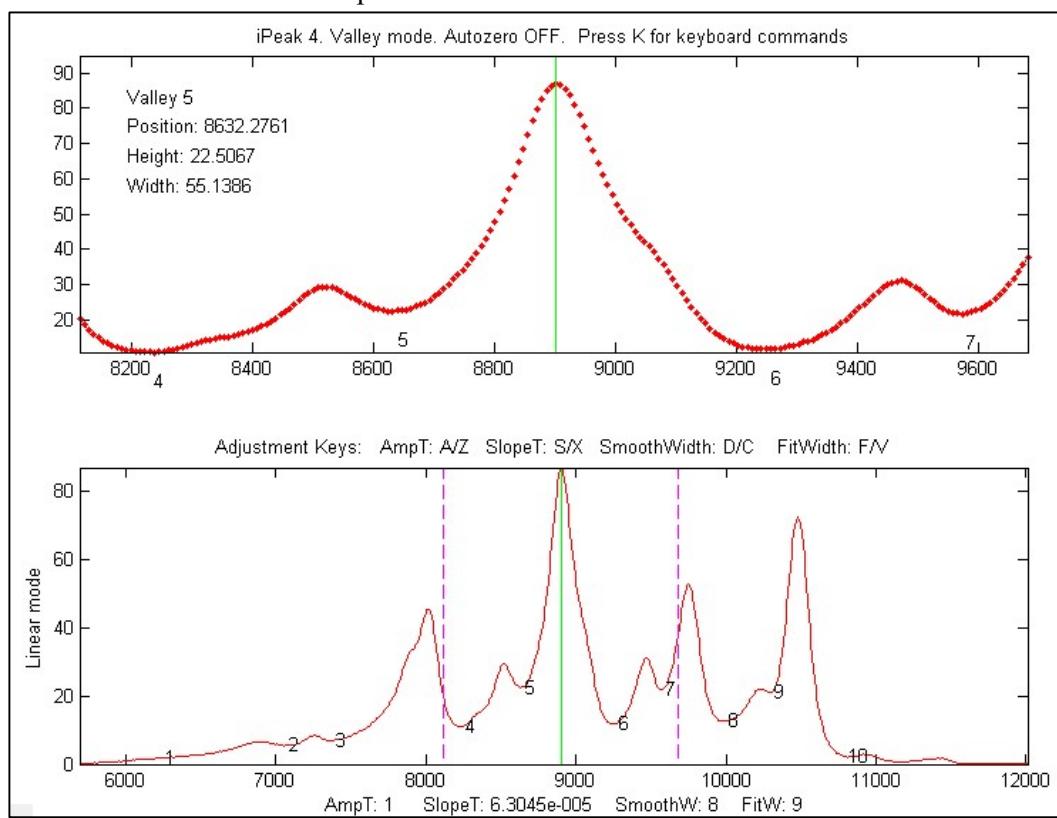
```
>> load sunspots
>> ipeak(year,number,20)
```

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
.....
```

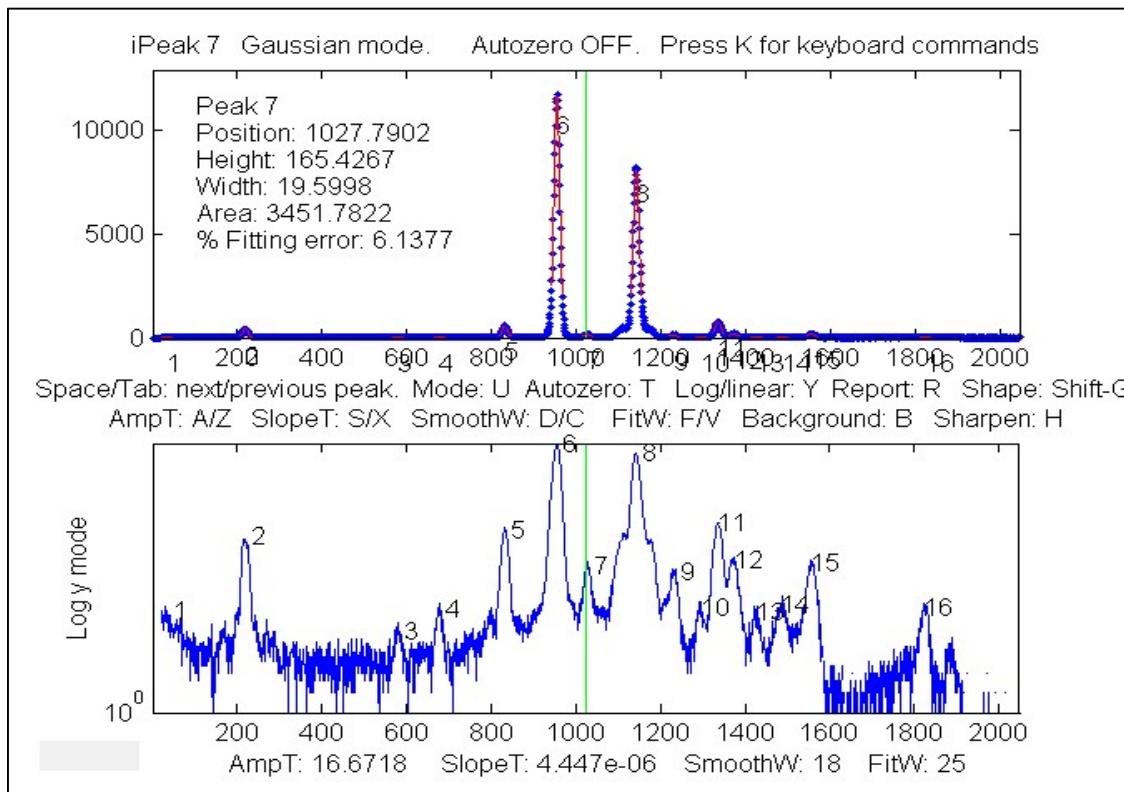
Premere **Shift-G** per scorrere tra le modalità Gaussiana, Lorentziana, e "flat-top" (picco piatto). Premere **Shift-P** per salvare la tabella dei picchi in un file su disco. Premere **U** per passare dalla modalità picco a quella avvallamento. Da non dimenticare che solo gli avvallamenti *superiori 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 *troppo 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

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 ippeakdata.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

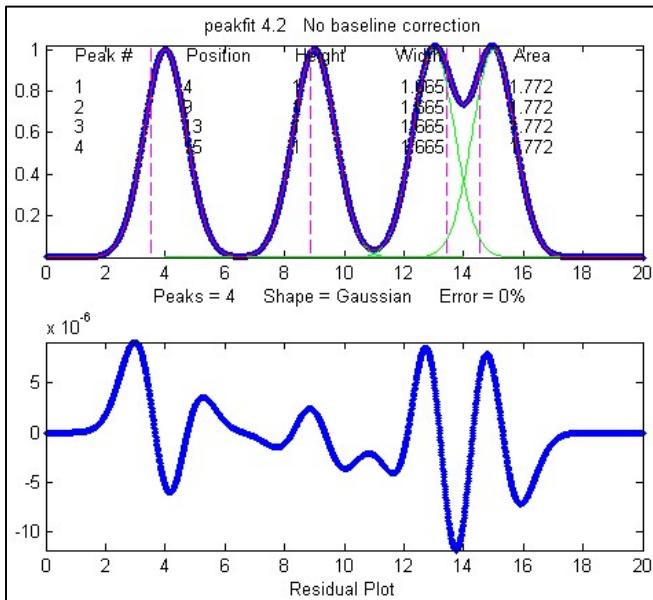
Approssimazione Normale e Multipla dei picchi in iPeak: Il tasto N applica [l'approssimazione iterativa della curva](#) ai *picchi rilevati e visualizzati nella finestra superiore* (qui denominato come approssimazione "Normale"). L'uso della funzione iterativa dei minimi quadrati può produrre misure 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 210). 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. 401):

Gaussians: $y=\exp(-((x-pos) / (0.6005615.*width)) .^2)$	
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
Lorentziane: $y=ones(size(x)) ./(1+((x-pos) / (0.5.*width)) .^2)$	
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:

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

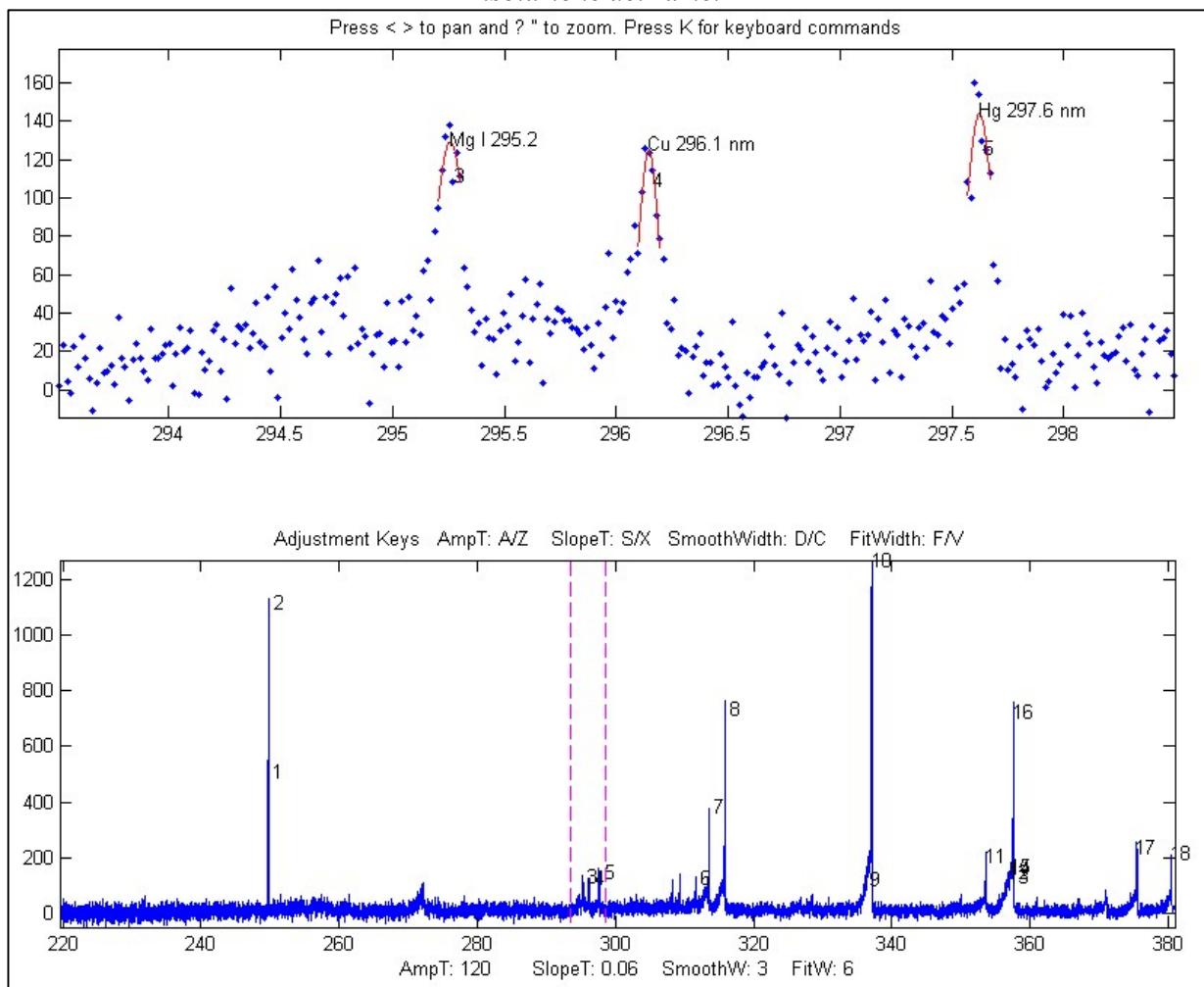
In questo caso, l'approssimazione della curva (usando i tasti **N** o **M**) fa un lavoro molto migliore, anche se la sovrapposizione è ancora maggiore, ma *solo se il profilo del picco è noto*:

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

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]

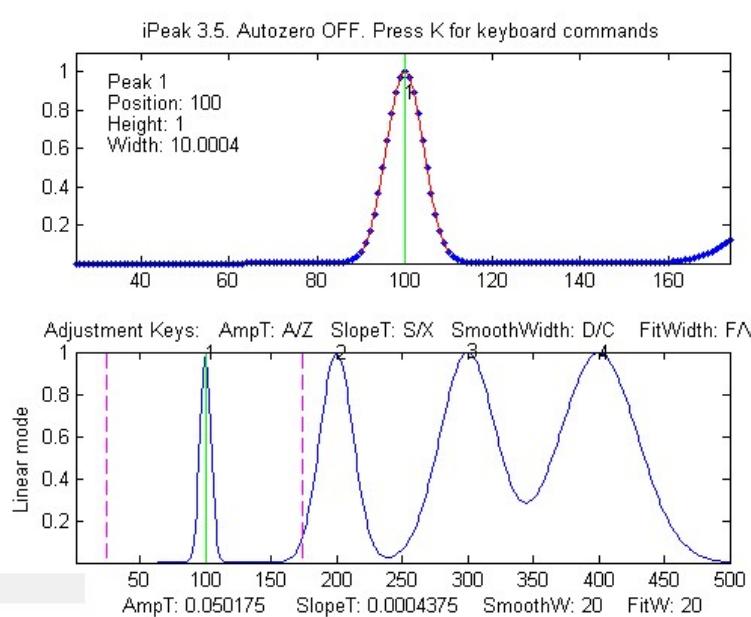
Ecco un altro esempio, da un ampio spettro di emissione atomica con oltre 10.000 punti dati e molte centinaia di picchi. La tabella di riferimento dei picchi noti in questo caso è presa dalla Tabella 1 di [ASTM C1301 - 95\(2009\)e1](#). Con le impostazioni che si stavano usando, sono stati identificati 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

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

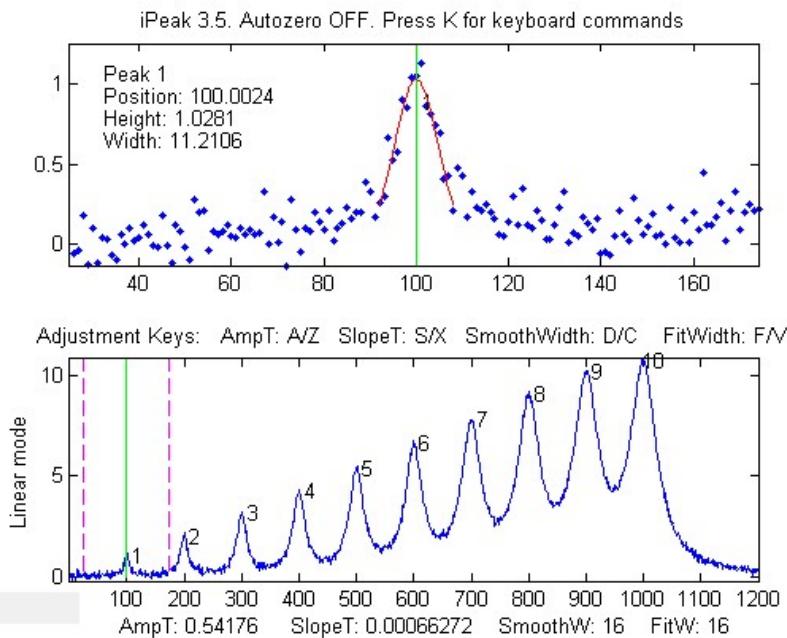
[demoipeak.m](#) (o [demoipeakoctave](#)) è una funzione dimostrativa che genera un segnale rumoroso con picchi, chiama *iPeak* e poi stampa una tabella dei parametri effettivi e un elenco dei picchi rilevati e misurati da *iPeak* per un confronto. Prima di eseguire questa demo, è necessario scaricare [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.

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

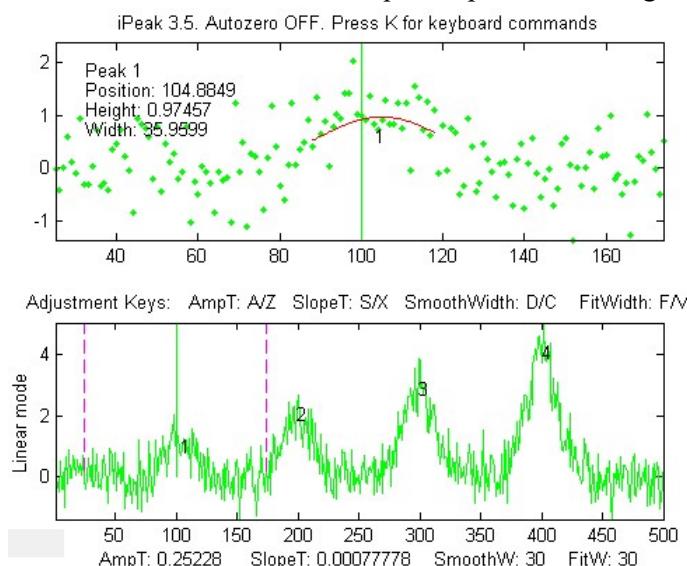


Dimostrazione di picchi Lorentziani sovrapposti, senza un background aggiunto. Nella finestra di comando viene stampata una tabella delle posizioni, delle altezze, delle larghezze e delle aree effettive dei picchi; in questo esempio, le vere altezze dei picchi sono 1,2,3,...,10. La sovrapposizione dei 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 segna-



le/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.)

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 335. I dati originali possono essere optionalmente 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 226. 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 entrambi 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 [PeakDetectionDemo2b.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](#)). [PeakDetectionDemo2.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

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 422) 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 328). Vedere anche [Excel VS Matlab](#).

(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.

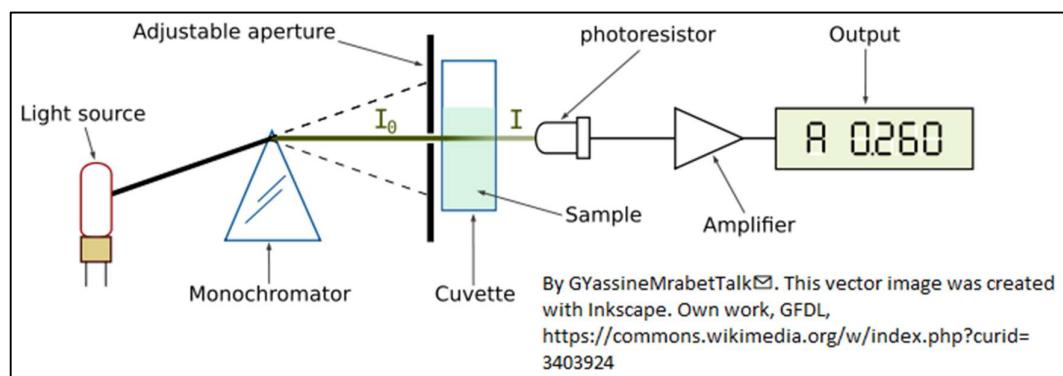
(c) È un metodo iterativo che, in circostanze sfavorevoli, può convergere su un errato ottimo locale, ma questo viene gestito mediante una corretta selezione del valore iniziale calcolato con il metodo logaritmico tradizionale $\log(I_0/I)$, e

(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 "I₀" (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, I₀ e α sono tutte in funzione della lunghezza d'onda; L e c sono scalari.

Tradizionalmente, i valori misurati di I e I₀ vengono utilizzati per calcolare una quantità chiamata ["assorbanza"](#), definita come

$$A = \log_{10}(I_0/I)$$

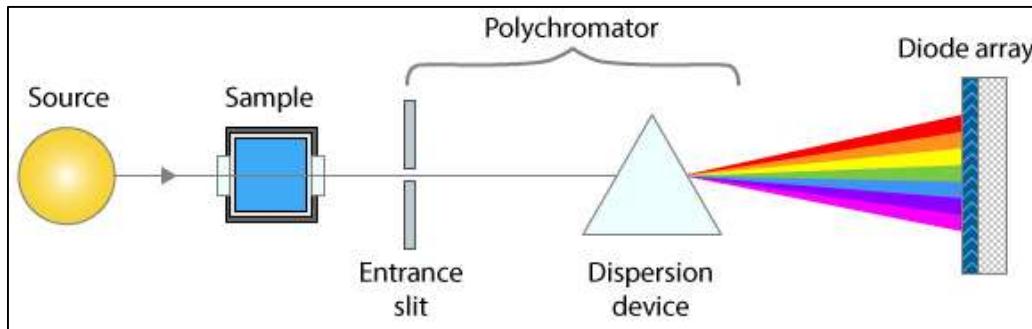
L'assorbanza è definita così in modo che, quando si combina questa definizione con la legge di Beer-Lambert, si ottiene:

$$A = \alpha \cdot L \cdot c$$

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

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 "policromicità". 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 rivelatori*, 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 rivelatori. 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 assorbane 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 li-

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 imprecisa della larghezza di banda strumentale e/o della luce diffusa è migliore della semplice supposizione che siano zero.

Implementazione dello spreadsheet

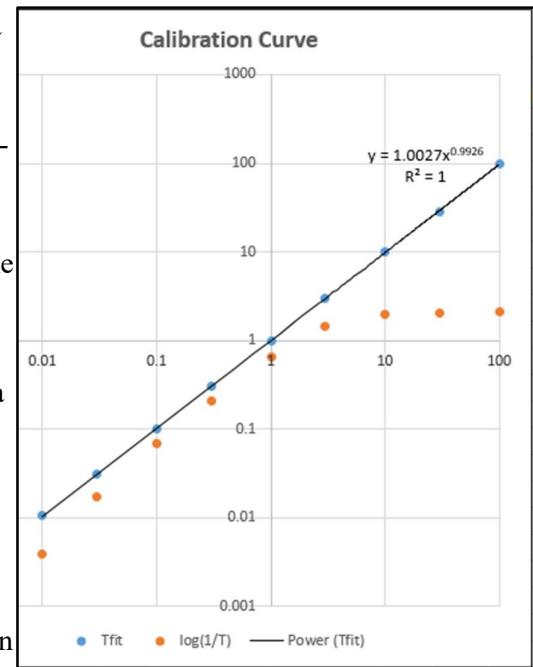
Il metodo Tfit può anche essere implementato in un foglio di calcolo *Excel* o *Calc*; è un po' più macchinoso dell'[implementazione in Matlab/Octave](#), ma funziona. Il metodo trasla-e-moltiplica viene utilizzato per la convoluzione dello spettro di riferimento con la funzione di fenditura e il l'add-in "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-AC6**). ([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 fotonic, 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 299). 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



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. 326) 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
```

KEYBOARD COMMANDS

```
Peak shape....Q Toggles between Gaussian and Lorentzian
absorption peak shape
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 influisce 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 rivelatore 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.

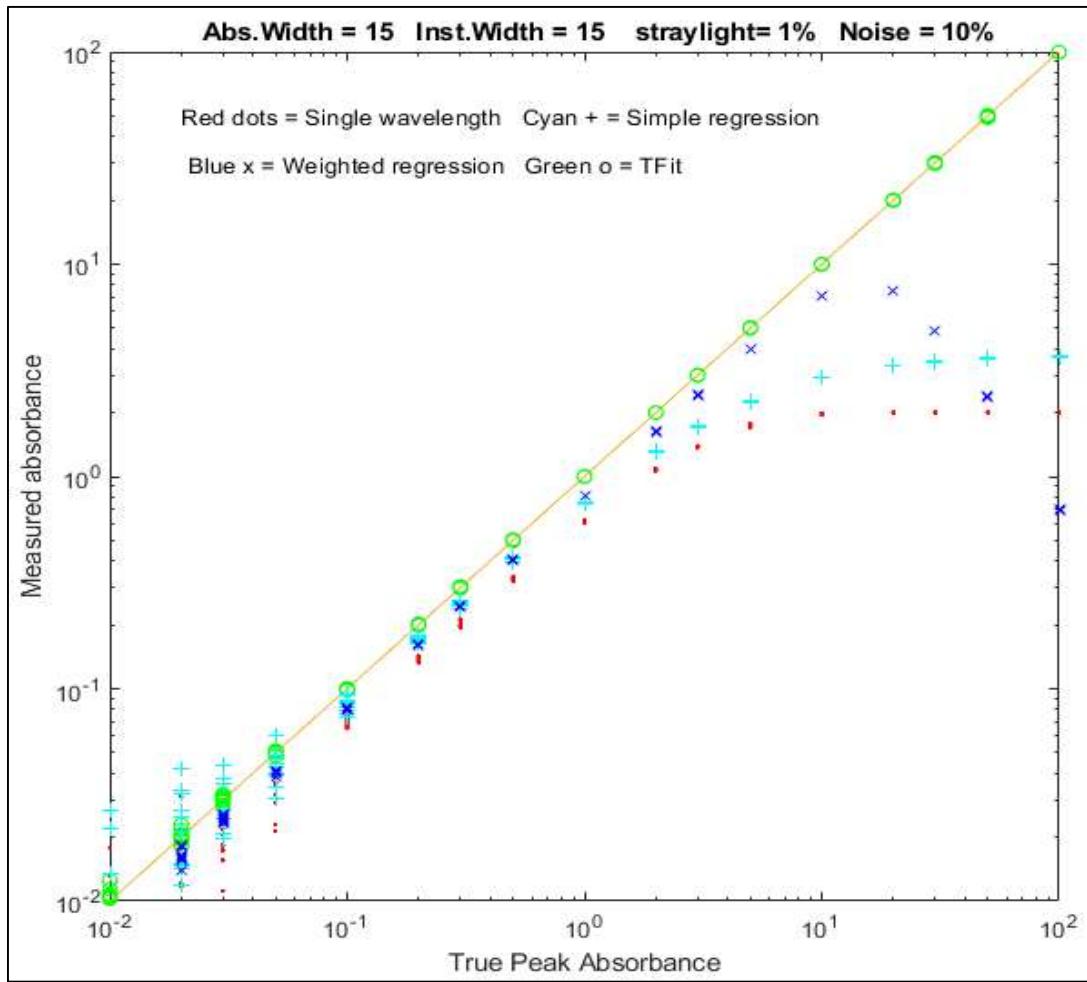
Nell'output di esempio mostrato di seguito, il programma ha calcolato i risultati per le assorbanze reali di 0,001 e 100, dimostrando che l'accuratezza e la precisione del metodo TFit sono superiori agli altri metodi su un intervallo di 10.000 volte.

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
```



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. 180) che al metodo T-Fit.

(spectral bandpass)

Noise **H/N** Increase/decrease random noise level when

InstWidth = 1

Peak shape **Q** Toggles between Gaussian and Lorentzian absorption peak shape

Table **Tab** Print table of results

K Print this list of keyboard commands

Esempio di tabella dei risultati tipici (visualizzata premendo il tasto **Tab**):

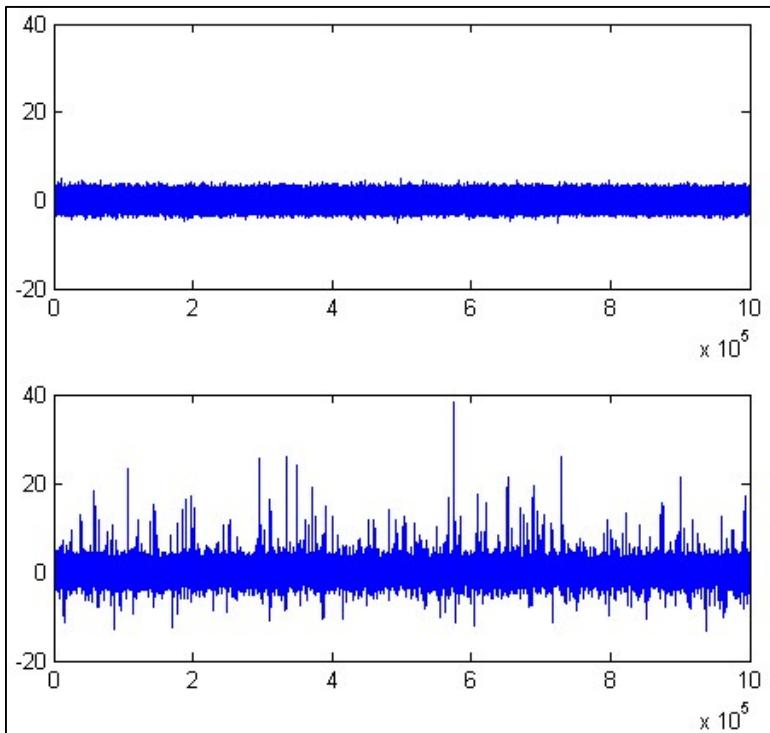
True Weighted TFit

Absorbance Regression method

Component 1 3 **2.06** **3.001**

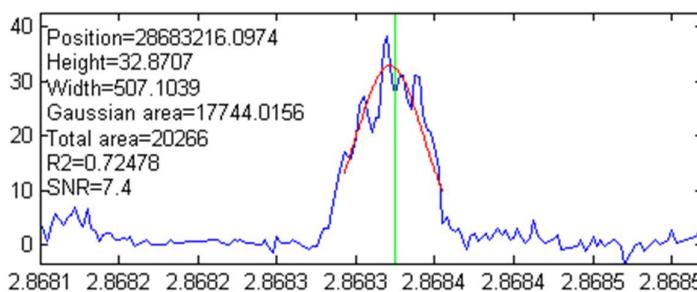
Component 2 0.1 **0.4316** **0.09829**

Component 3 5 **2.464** **4.998**



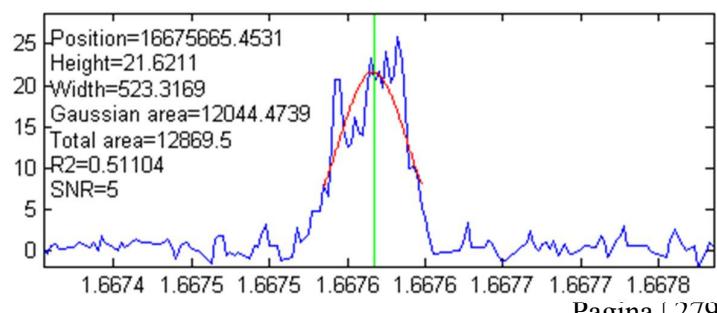
Come si vede, la differenza principale è che il segnale ha "picchi" più grandi, soprattutto in direzione positiva. Questa differenza è evidente quando si guardano le [statistiche descrittive](#) del segnale e della funzione randn:

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

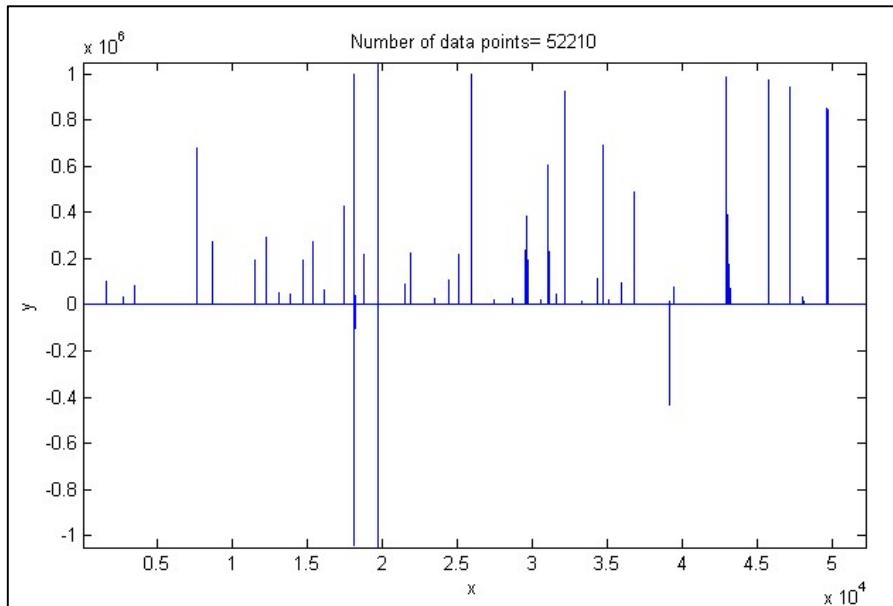
segnale; sembrano spike solo perché la lunghezza del segnale (più di 1,000,000 di punti) fa sì che i picchi vengano compressi in un pixel dello schermo o meno quando l'intero segnale viene disegnato sullo schermo. Nelle figure a lato, [ISignal](#) (pagina 357) viene usato per "in-



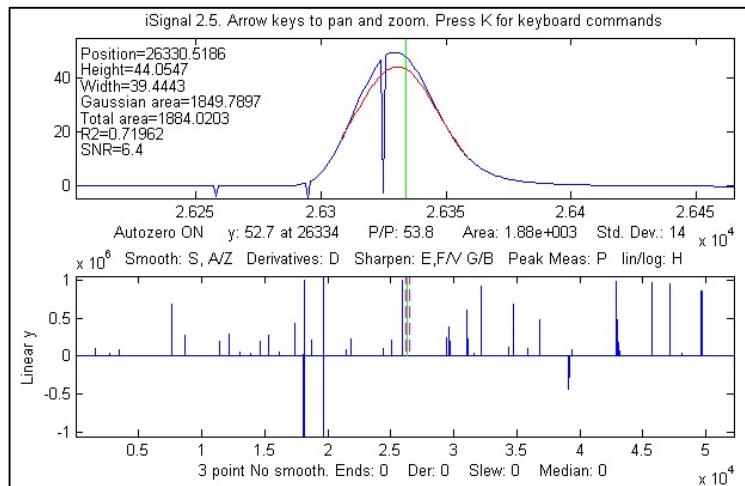
vi come [iPeak](#), [iSignal](#) e [ipf](#) potrebbe risultare lenti nel funzionamento, specialmente se il computer non è veloce dal punto di vista computazionale o grafico. Se questo è un problema serio, potrebbe essere meglio suddividere il segnale in due o più segmenti e trattare ogni segmento separatamente, 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

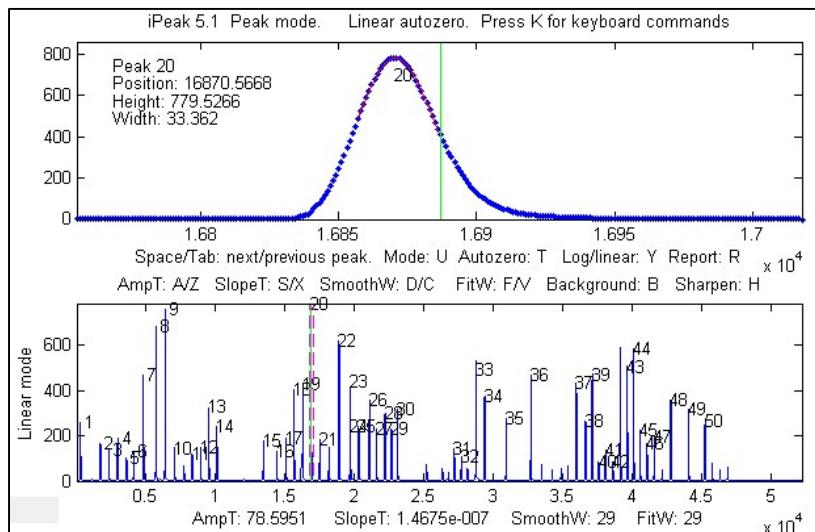
Il segnale sperimentale in questo caso di studio presentava diversi spike stretti che superavano una *linea di base apparentemente piatta*.



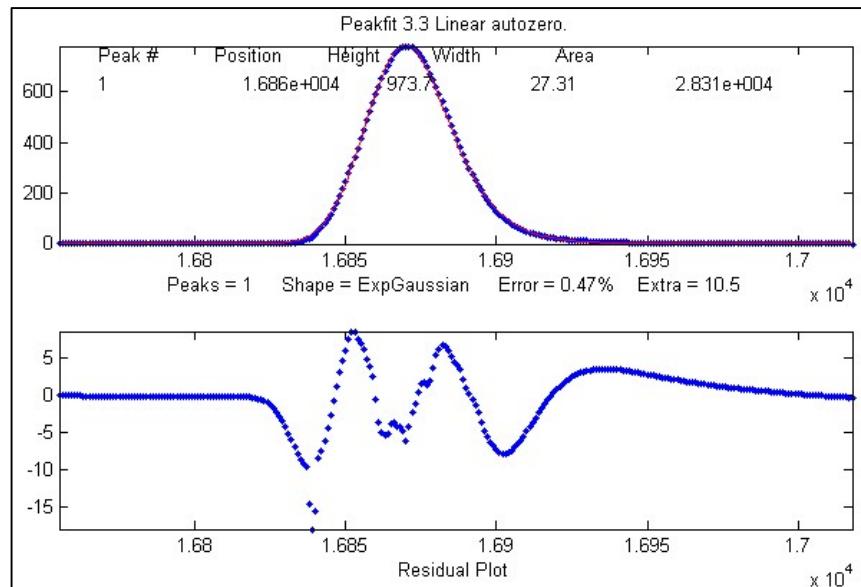
Utilizzando [iSignal](#) (pagina 357) per indagare sul segnale, si è scoperto che i picchi positivi visibili erano *punti singoli* di ampiezza molto grande, mentre le regioni *tra* gli spike non erano realmente piatte ma contenevano molti picchi a forma di campana che erano tanto più piccoli (di un fattore 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 so-



Se necessario, i singoli picchi possono essere misurati in modo più accurato approssimando l'intero picco in *iPeak* con il tasto “N” (pagina 394) o con *peakfit.m* o con *ipf.m* (pagina 394). I picchi sono tutti leggermente asimmetrici; si approssimano a un **modello Gaussiano esponenzialmente allargato** (pag. 219) 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).

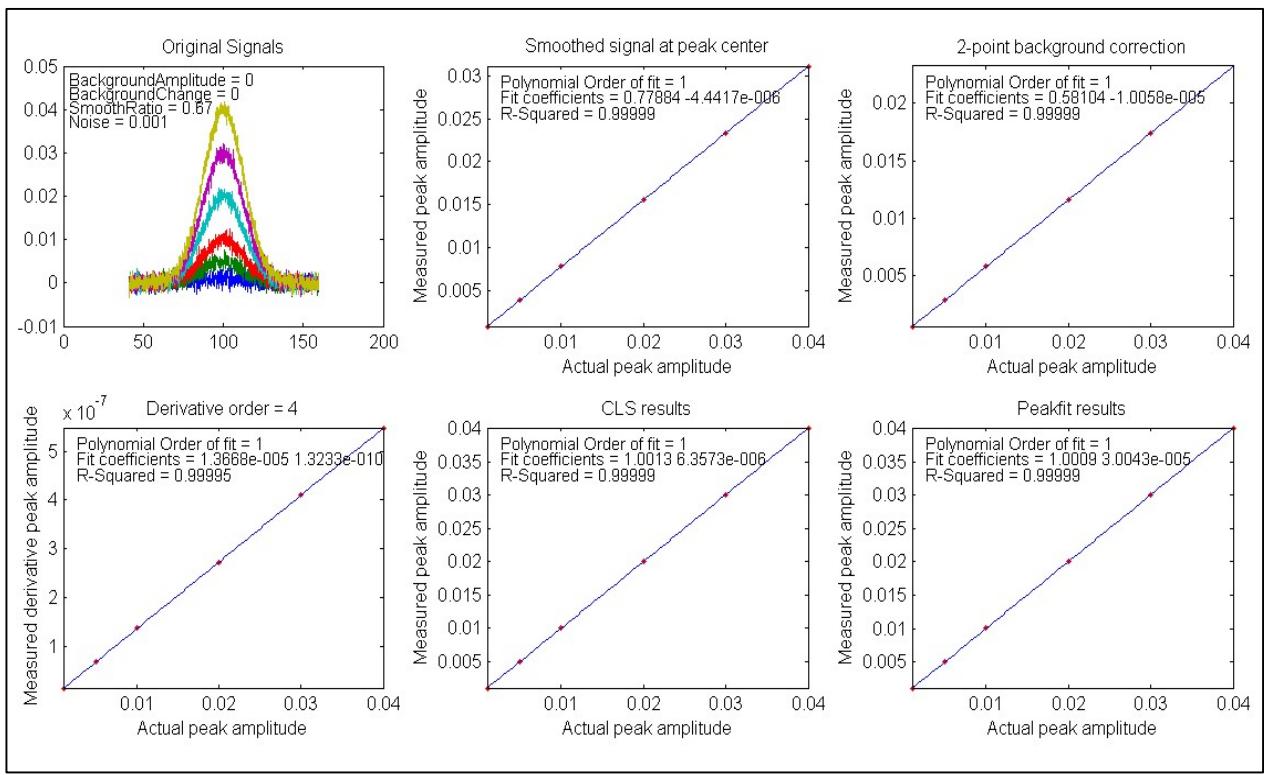


Si noti che l'approssimazione con un modello Gaussiano espanso esponenzialmente fornisce i **parametri del picco Gaussiano prima dell'espansione**. *iSignal* (pagina 357) e *iPeak* (pagina 394) stimano i parametri del picco espanso. Come in precedenza, l'effetto dell'ampliamento è quello di spostare la posizione del picco su valori maggiori, ridurne l'altezza e aumentarne l'ampiezza.

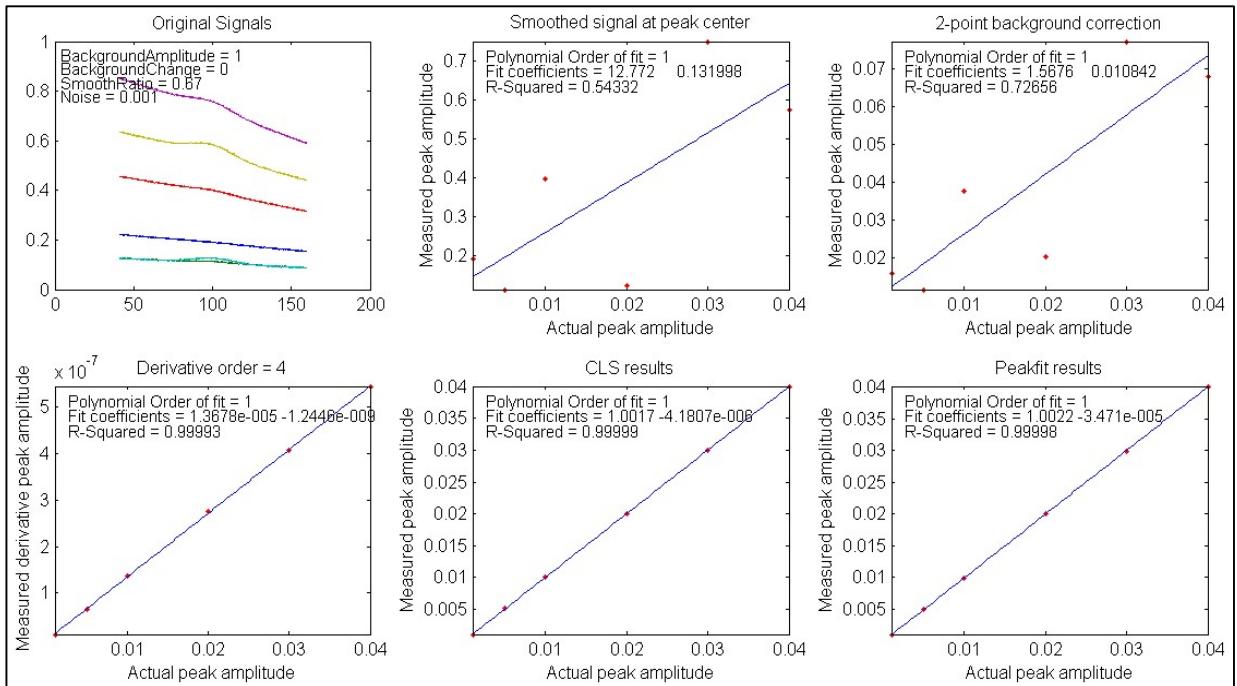
```
Position  Height  Width  Area  error
isignal  16871  788.88  32.881  27612      S/N Ratio = 172
ipeak   16871  785.34  33.525  28029
peakfit  16871  777.9   33.488  27729  1.68% Gaussian model
peakfit  16863  973.72  27.312  28308  0.47% Exponentially-broadened Gaussian
```

Il Torneo: un confronto dei metodi

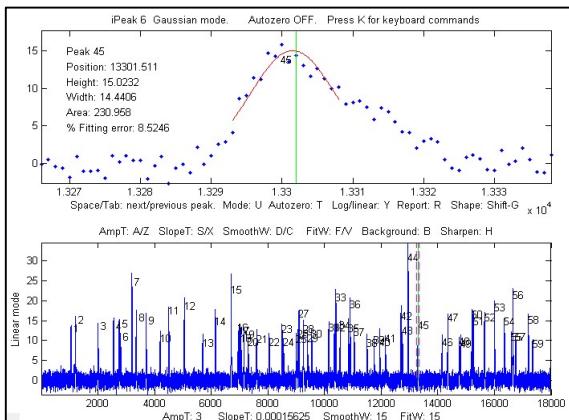
Questa simulazione mostra l'applicazione di diverse tecniche descritte in questo documento per la misura quantitativa di un picco sepolto in un background instabile, una situazione che si verificarsi spesso nelle applicazioni di analisi quantitativa di varie forme di spettroscopia, monitoraggio di



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.



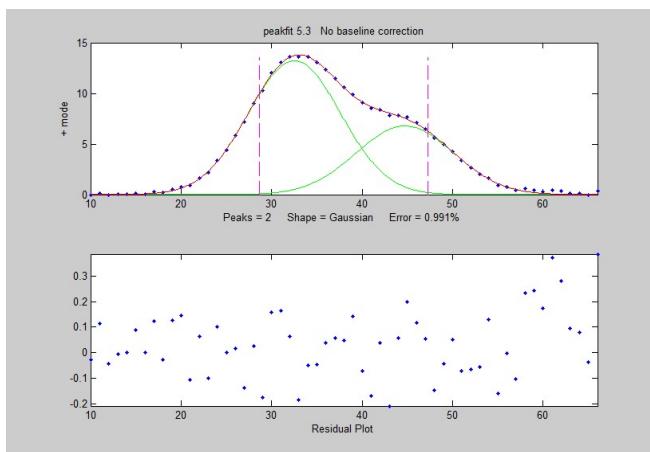
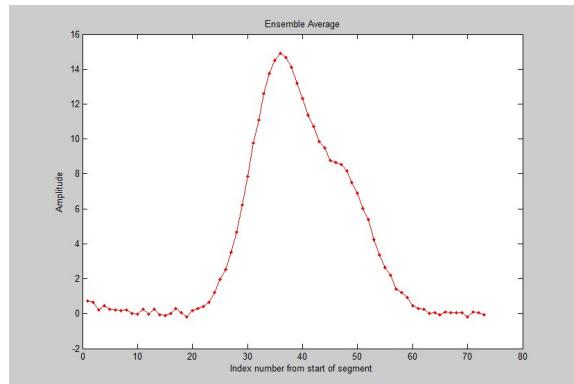
Per il secondo test (mostrato nella figura immediatamente sopra), "BackgroundAmplitude"=1 e "BackgroundChange"=0, quindi il background ha una variazione di ampiezza significativa ma forma, posizione e larghezza fisse. In questo caso, i primi due metodi falliscono, ma i metodi derivativi, CLS e INLS funzionano bene.



iPeak 6 (pag. 394) 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

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



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)
```

za 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. 394) 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

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 394):

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

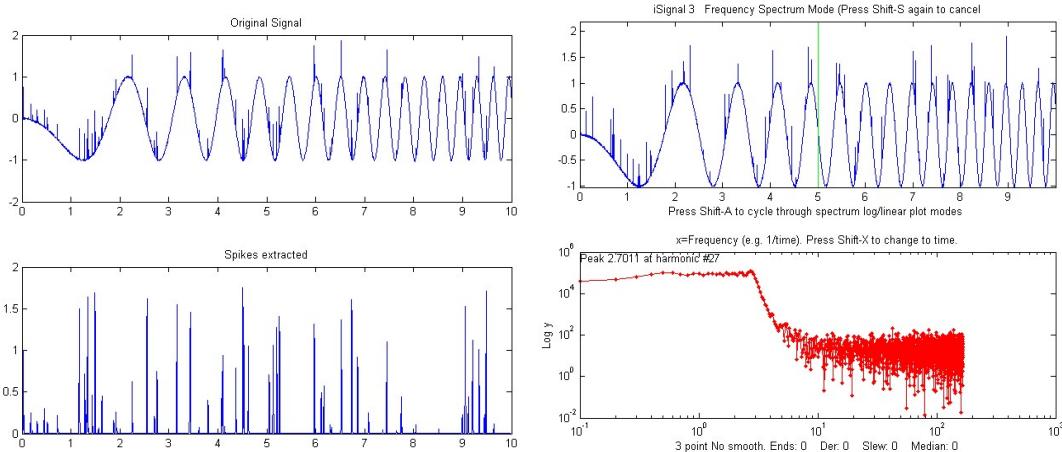
La precisione stimata delle misure della posizione (cioè della frequenza) è di circa lo 0.2% relativo, in base al [metodo bootstrap](#), abbastanza buona da consentire un calcolo accurato dello spostamento della frequenza (circa il 4.2%) e della [velocità del veicolo](#) e per dimostrare che il rapporto 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 contare 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. 394) 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 interes-

```
y1=fastsmsooth(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. 97) potrebbe essere utilizzato per [visualizzare questo segnale](#) e un [filtro di Fourier](#) (pag. 121) 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:

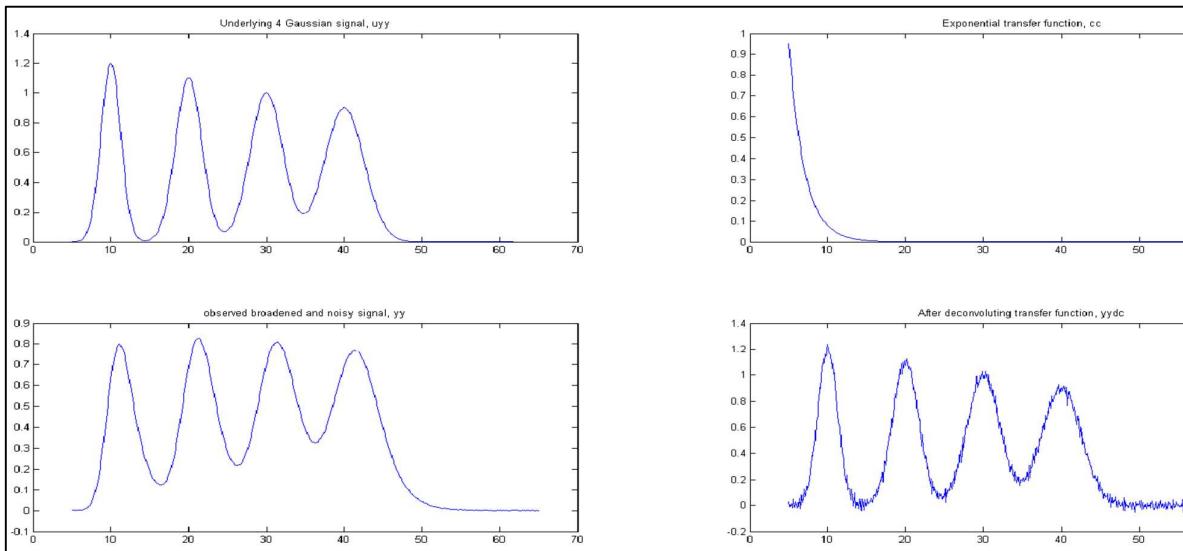
```
P=findpeaksG(x,y1,2e-005,0.01,2,5,3);
0
P=findpeaksplot(x,y1,2e-005,0.01,2,5,3);
0
PS=peakstats(x,y1,2e-005,0.01,2,5,3,1);
```

Il semplice script “[iSignalDeltaTest](#)” mostra lo spettro di potenza delle funzioni di smoothing e differenziazione di iSignal (pagina 357) 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.

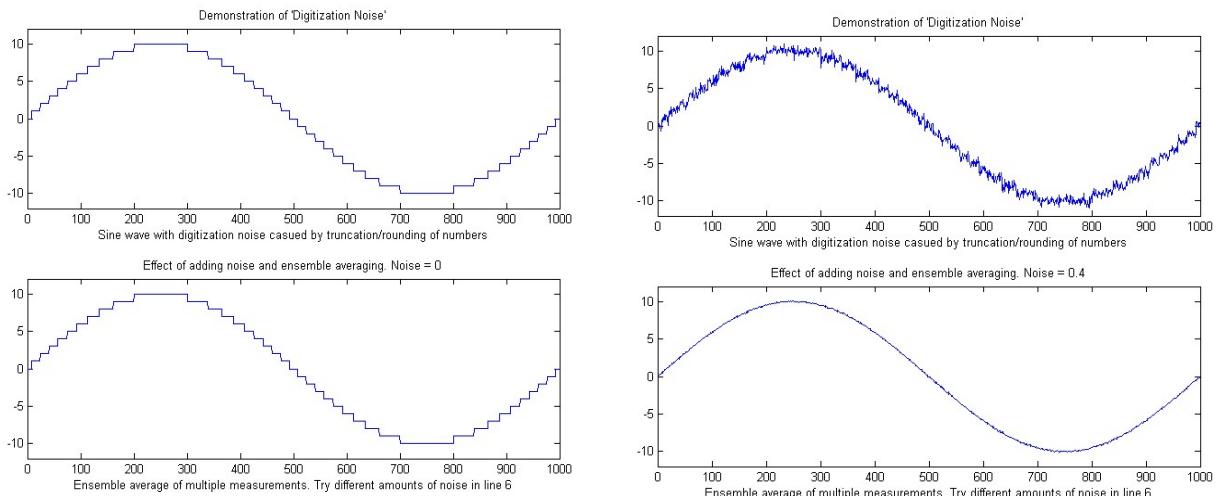


Nello script mostrato sotto e il grafico risultante mostrato a sopra ([Scaricare questo script](#)), il segnale sottostante (uyy) è un insieme di quattro Gaussiane con altezze di picco di 1.2, 1.1, 1, 0.9 situate

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.

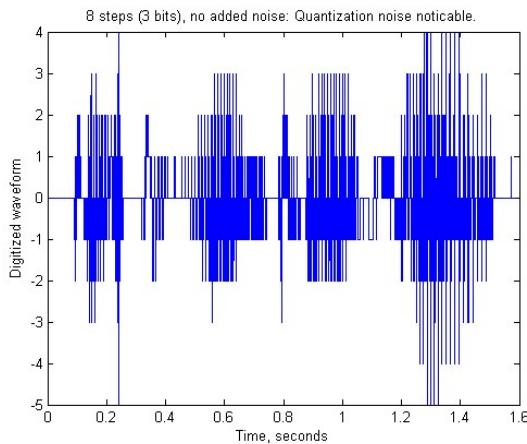
Rumore di digitalizzazione - l'aggiunta di rumore può davvero aiutare?

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

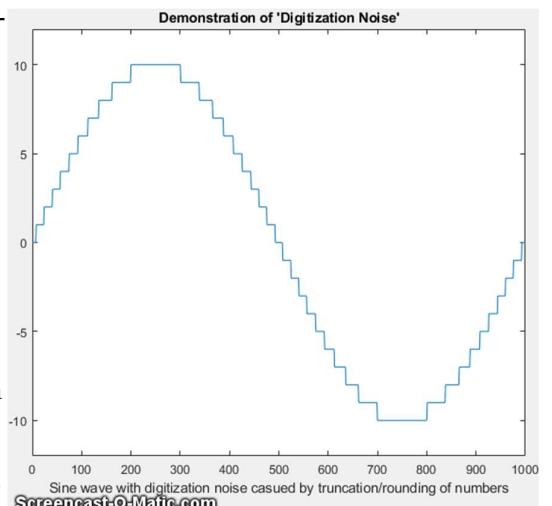


software coinvolti nella trasmissione del segnale digitale a un computer, o anche nel processo di trasferimento dei dati da un programma a un altro, come nel copiare e incollare dati in e da un foglio di calcolo. Il risultato è una serie di salti non casuali di uguale altezza. La distribuzione delle 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'ampliezza di +/- 10. La media dell'insieme, che di solito è la più efficace delle tecniche di riduzione del rumore, non rideuce questo tipo di rumore (pannello inferiore) perché non è casuale.

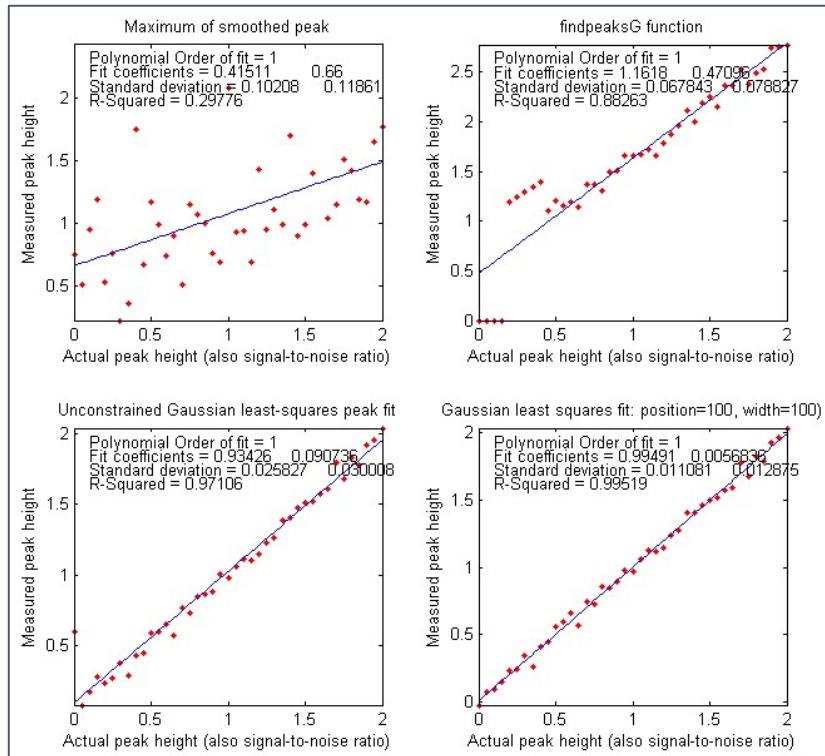


È interessante notare che, se nel segnale è presente ulteriore rumore casuale, la media dell'insieme diventa efficace nel ridurre sia il rumore casuale, che il rumore della digitalizzazione. In sostanza, il rumore aggiunto *randomizza la digitalizzazione*, consentendone la riduzione mediante la media dell'insieme. Inoltre, se il rumore già presente nel segnale è troppo basso, può essere utile *aggiungere artificialmente ulteriore rumore!* Sul serio. Lo



Screencast-O-Matic.com

trato ha un valore da picco a picco diverso da zero). La funzione `findpeaks` (in alto a destra) funziona 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-quadro. Ma se si osserva attentamente l'estremità inferiore dell'intervallo di altezza del picco, vicino a $x = 0$, 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%
<code>findpeaksG.m</code>	32.3709%	33.363%
<code>peakfit.m</code>	2.7542%	4.6466%
<code>cls2.m</code>	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

sante 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.

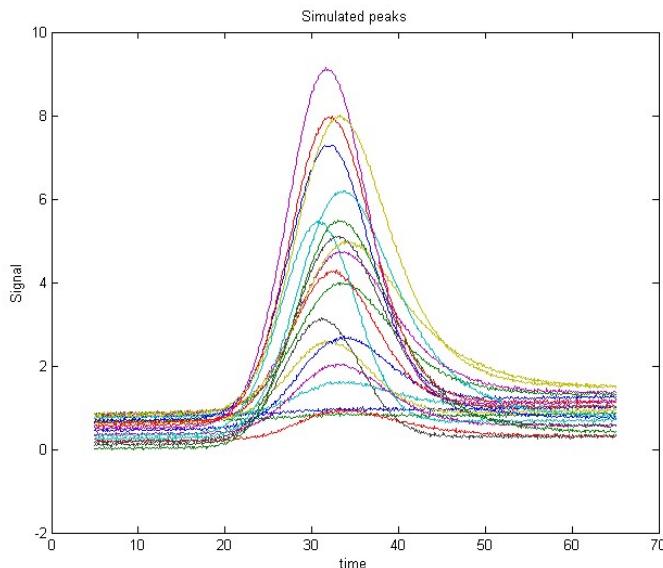
Una cosa che sappiamo per certo è che la terra ruota attorno al suo asse una volta al giorno e che ruota attorno al sole una volta all'anno. Quindi, se guardiamo in una direzione fissa fuori dalla Terra, le stelle lontane sembreranno muoversi secondo uno schema prevedibile, mentre le sorgenti terrestri 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.

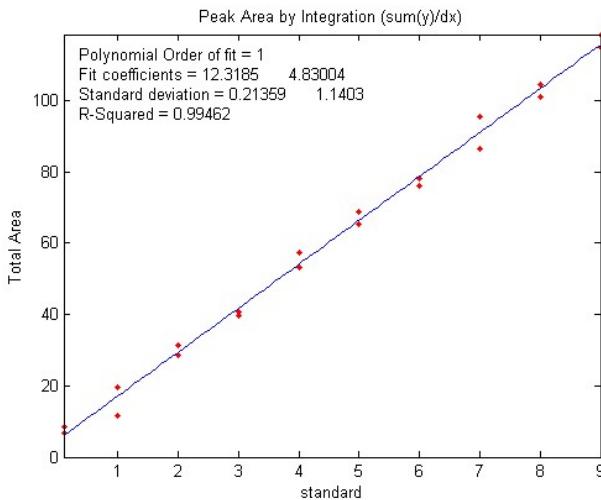
Sappiamo anche che ci sarà uno spostamento Doppler nelle frequenze osservate se la sorgente si muove rispetto al ricevitore; questo si osserva con le onde sonore così come con le onde elettromagnetiche come la radio o la luce. Poiché la Terra ruota e gira a una velocità nota e costante, possiamo prevedere 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 'stan-



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 la scorciatoia Ctrl-tasto verrà eseguita la macro e tutte le operazioni sul foglio di calcolo registrate.

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 193) si descrive l'uso della funzione "Solver" applicata all'ap-

```

Selection.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.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.

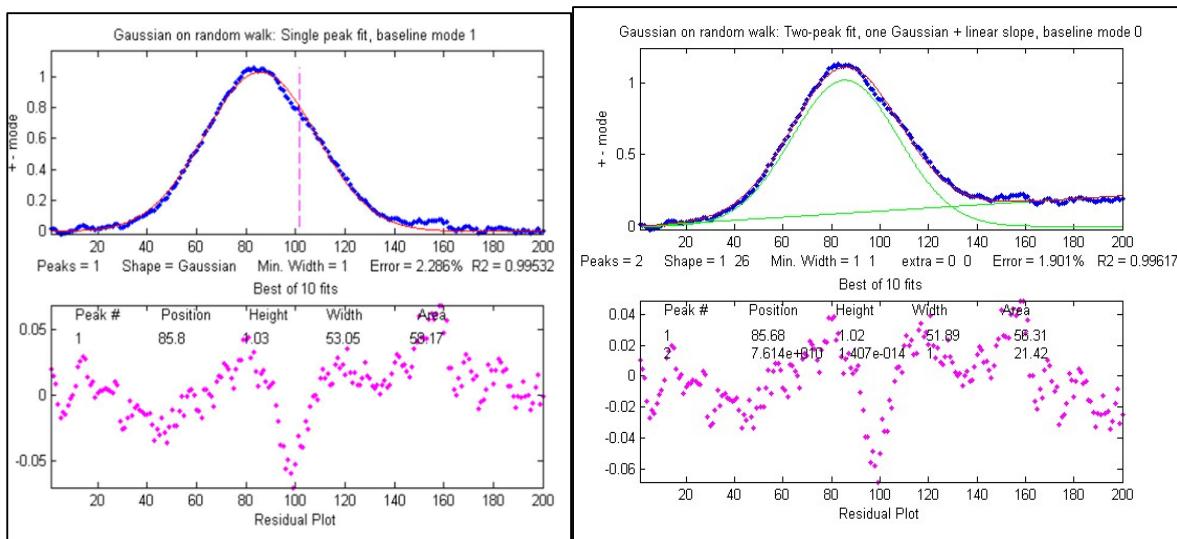
Vale la pena notare che da dicembre 2022 è disponibile un ulteriore modo per creare funzioni e macro di Excel: utilizzando il chatbot conversazionale di intelligenza artificiale ChatGPT (pagina 425). Una volta ottenuto un account gratuito su <https://chat.openai.com/>, si può semplicemente chiedere "Scrivi 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. 23) 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

```
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.

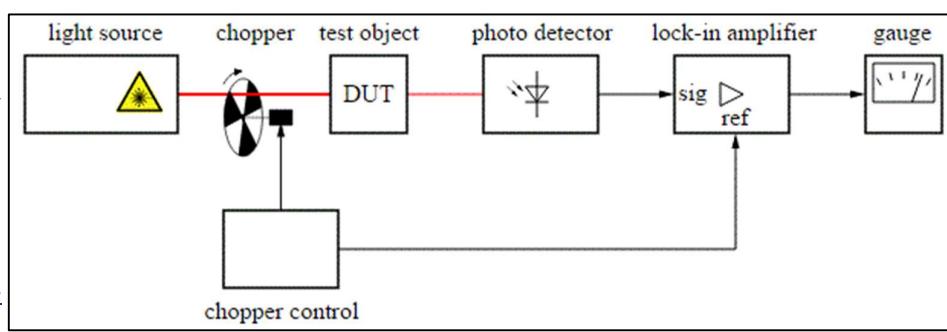
Position	Error	Height	Error	Width	Error
Method a:	0.2772	3.0306		0.0125	
Method b:	0.4938	2.3085		1.5418	

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.

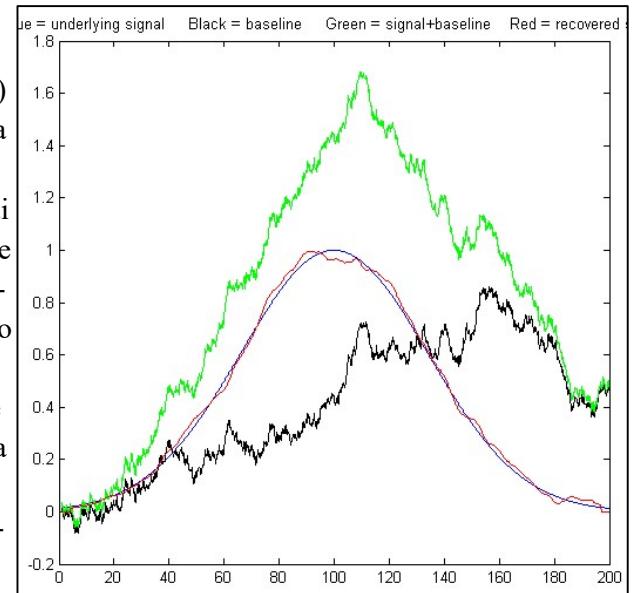
Modulazione e rilevamento sincrono.

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](#).



care 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](#)).

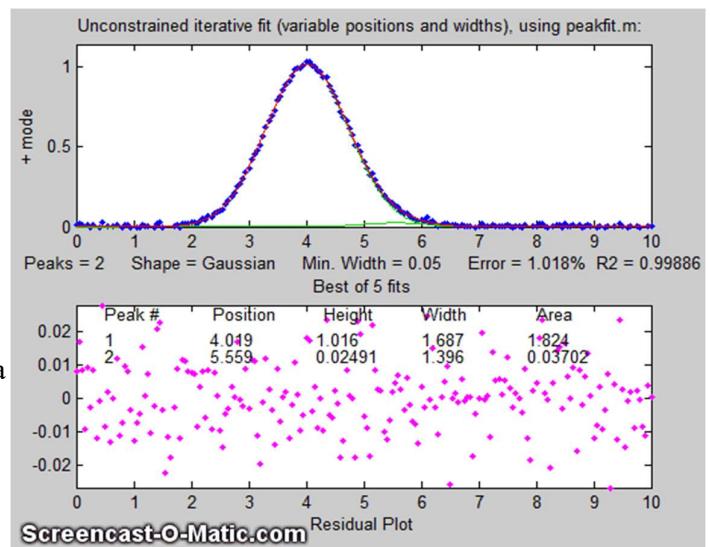
picco misurato è sufficientemente piccolo e abbastanza vicino al picco sovrapposto più forte (separato da un valore inferiore alla larghezza dei picchi) in modo da *non formare mai un massimo* nel segnale totale. Quindi *sembra* che ci sia *un solo* picco, come mostrato nella figura a lato. Per questo 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 calcolasse 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 casuale, 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



Peaks.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

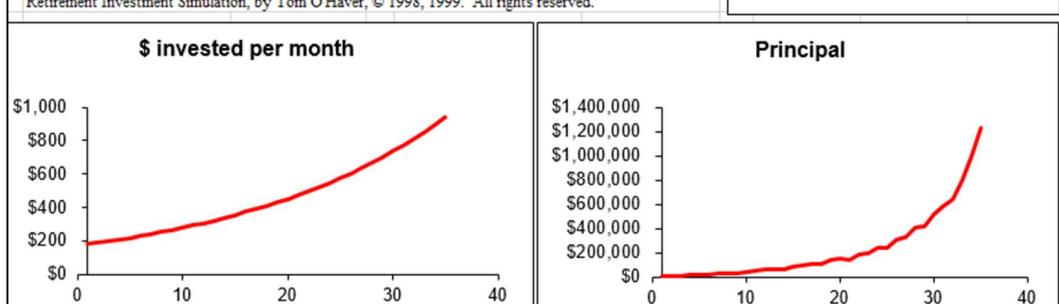
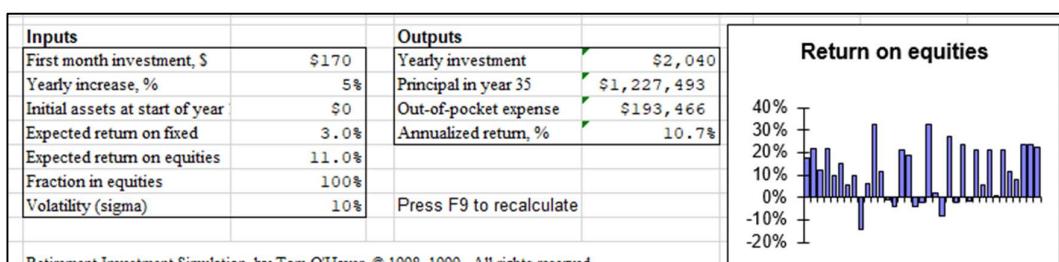
Dal punto di vista del rapporto segnale/rumore, il mercato azionario è un esempio interessante. Un mercato azionario nazionale o globale è un'aggregazione di un gran numero di acquirenti e venditori di azioni di società quotate in borsa. Sono descritti dagli *indici* del mercato azionario, calcolati 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. Altri indici azionari, come il Russel 2000, includono un numero ancora maggiore di società più piccole. 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 giornaliero, V, dell'indice S&P 500 rispetto al tempo, T, dal 1950 a settembre 2016 è mostrato nei grafici seguenti.

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 crescendo in modo esponenziale all'incirca alla stessa velocità di prima, ma con fluttuazioni imprevedibili simili a ciò che è accaduto in passato.

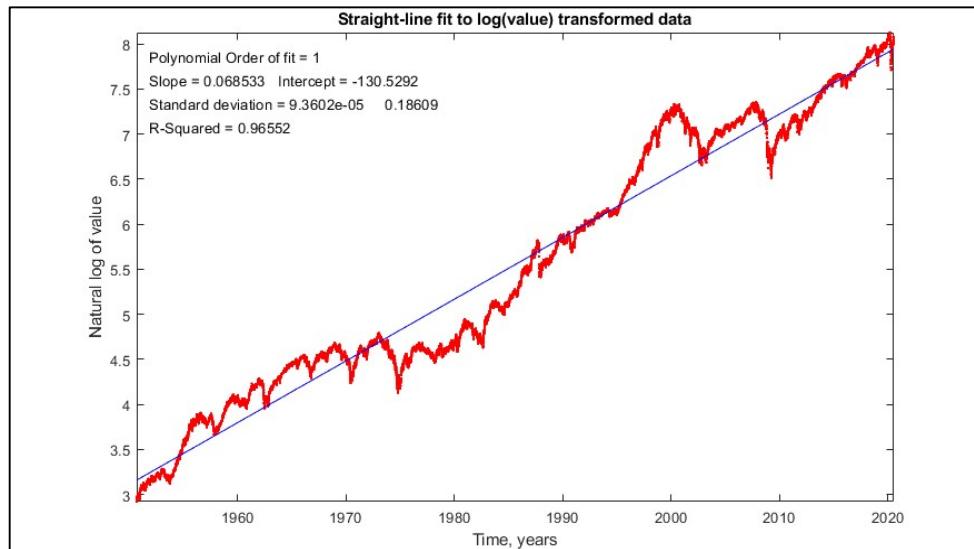
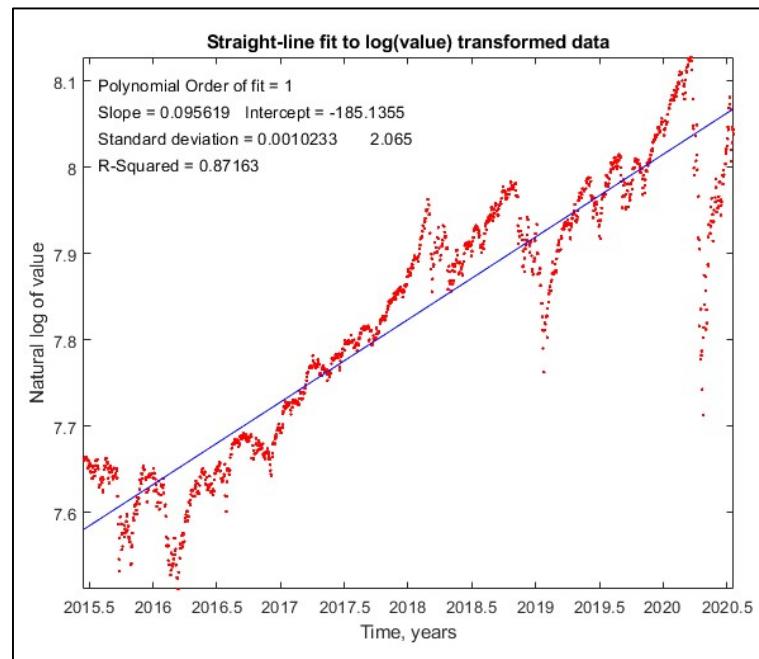
Possiamo dare un'occhiata più da vicino a queste fluttuazioni ispezionando i *residui* – cioè sottraendo la curva approssimata dai dati originali, come mostrato nel [grafico iSignal](#) (pag. 357) a sinistra sopra. Ci sono molte caratteristiche notevoli di questo "rumore". In primo luogo, le deviazioni sono approssimativamente proporzionali a *V* e quindi relativamente uguali quando tracciate su una scala logaritmica. In secondo luogo, il rumore ha un carattere chiaramente a *bassa frequenza*[https://terpconnect.umd.edu/~toh/spectrum/SignalsAndNoise.html - Frequency](https://terpconnect.umd.edu/~toh/spectrum/SignalsAndNoise.html) (pagina 28); il *periodogramma* (pannello inferiore, in rosso) mostra picchi a 33, 16, 8 e 4 anni. Vi sono anche, in particolare, numerosi casi nel corso degli anni in cui si è verificato un forte calo seguito da una ripresa più lenta prossima al 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



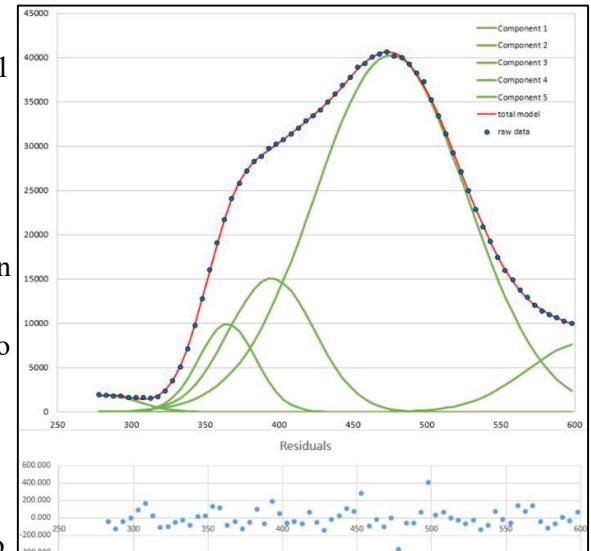
Nota aggiunta a giugno 2020. I dati del mercato azionario utilizzati sopra hanno ormai diversi anni. Ci si starà chiedendo quanto siano stati buoni quei dati nel prevedere le tendenze del mercato azionario dal 2016. Da allora, ci sono state molte perturbazioni del mercato, in particolare le guerre commerciali del 2019 e la pandemia di Coronavirus del 2020. Se estendiamo i dati tracciati sopra per includere i risultati S&P per le date fino a giugno 2020, si può vedere che ciò ha un effetto notevolmente ridotto, come si vede nel grafico del log nella pagina successiva. I dati aggiunti sono solo

nell'angolo in alto a destra e *le fluttuazioni sono piccole rispetto agli eventi storici precedenti*. Il rendimento medio complessivo è ancora intorno al 7% (senza reinvestimento dei dividendi). In altre parole, *i dati del 1950-2016 erano ottimi predittori delle performance di mercato più recenti, nonostante le allarmanti interruzioni*.



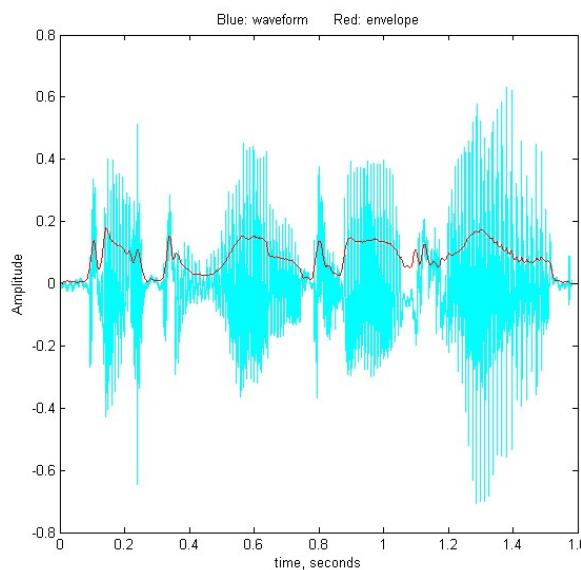
A guardare più da vicino, i cambiamenti recenti sono più evidenti se si guarda al solo periodo dal 2016 al 2020, di seguito, per il quale il rendimento in quel breve periodo è stato effettivamente maggiore, circa il 9,5%. I cali del 2019 e del 2020, sebbene all'epoca fossero piuttosto bruschi e causassero molta ansia, si sono ripresi rapidamente e, di conseguenza, *hanno avuto scarsi effetti sulle prestazioni complessive a lungo termine*. Quando le azioni scendono, anche per ragioni note e valide, alcuni investitori acquistano azioni a prezzi ridotti, e quando le azioni salgono, soprattutto quando raggiungono i massimi storici, alcuni investitori vendono azioni per "bloccare i loro guadagni". Questo comportamento è stato costante nel corso degli anni e funge da freno naturale alle fluttuazioni del mercato.

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 mostrato a lato. In questo esempio, 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 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. 40) del valore assoluto della forma d'onda:

```
envelope = fastsmooth(abs(waveform), SmoothWidth, SmoothType);
```



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 357), *iPeak* (pagina 243) e *ipf.m* (pagina 395) o con la funzione Matlab/Octave [segplot.m](#) (pagina 447). 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 [fastspeed.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 *fastspeed.m*, tranne per il fatto che il secondo argomento di input "smoothwidths" può essere un vettore: [SmoothedSignal, SmoothMatrix] =

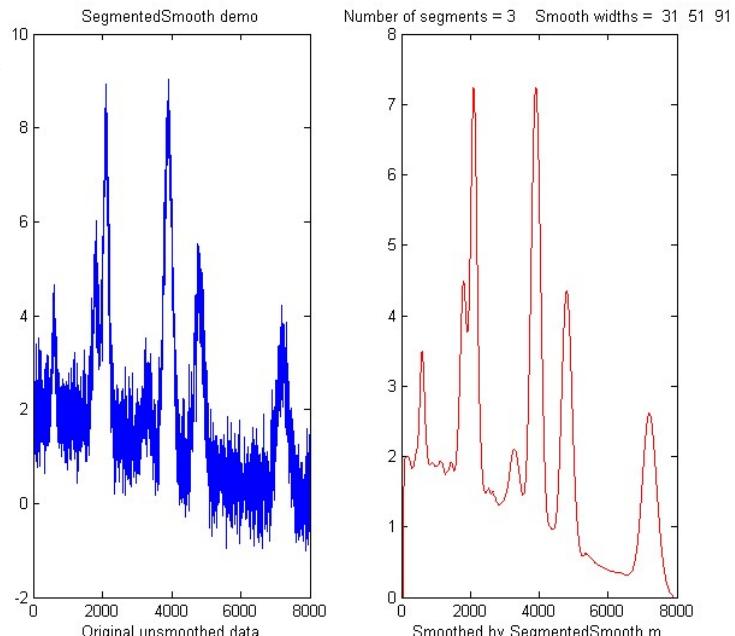
SegmentedSmooth (Y,

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

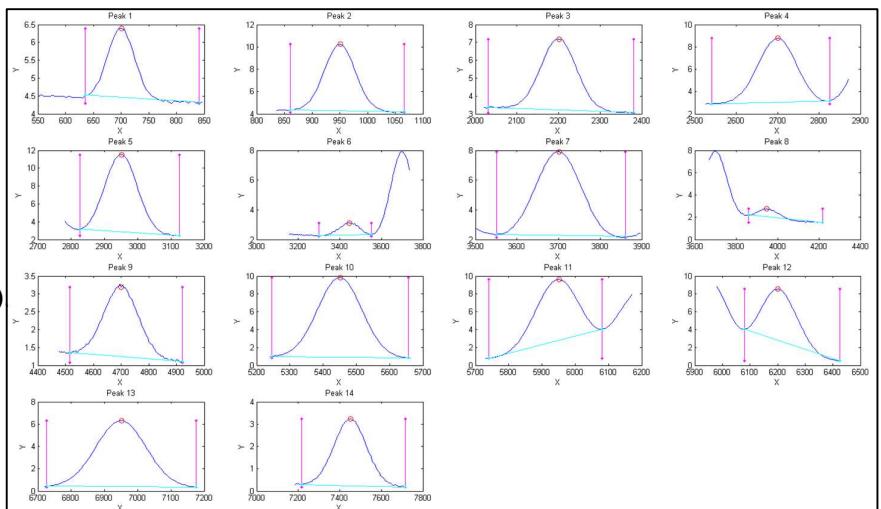
fronta la precisione e l'accuratezza per la posizione del picco e la misurazione dell'altezza sia per entrambe le funzioni [findpeaksSG.m](#) e [findpeaksSGw.m](#).

[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 [autopeaksplot.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.

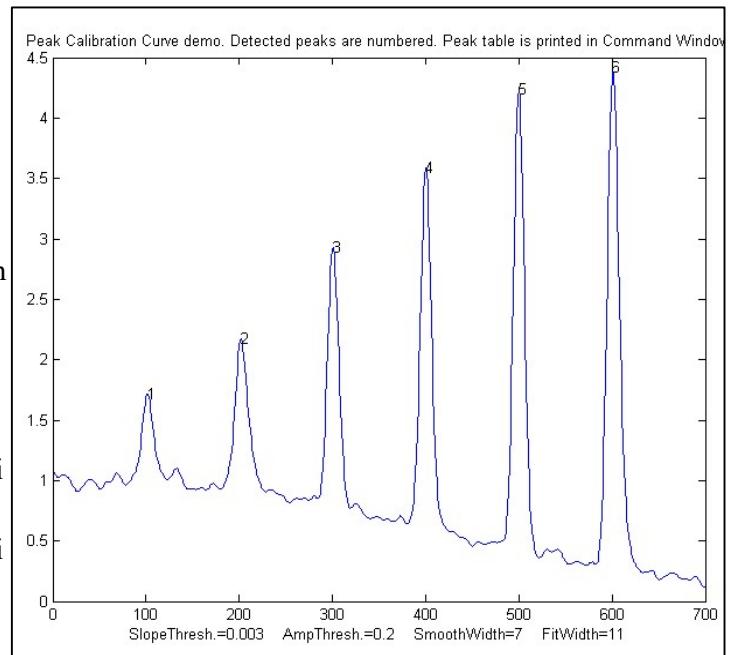


Lo script [HeightAndArea.m](#) verifica l'accuratezza della misura dell'altezza e dell'area del picco con i segnali che hanno più picchi con larghezza, rumore, sfondo e sovrapposizione variabili. In generale, i valori per l'altezza assoluta e l'area di taglio verticale (pagina 140) sono migliori per 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

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](#)).

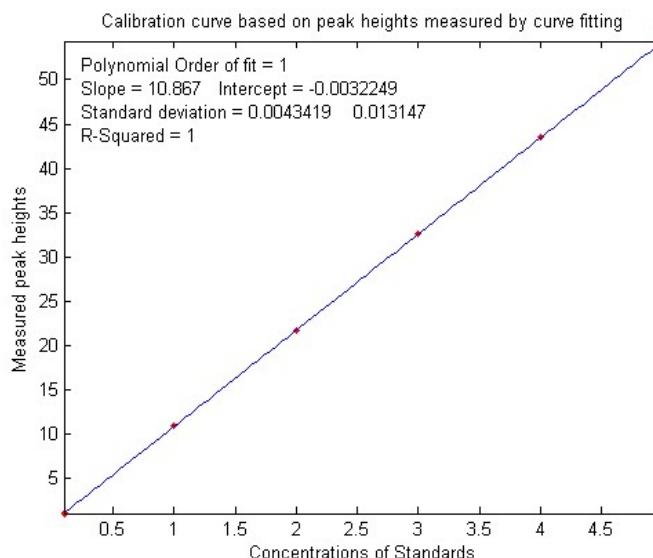
[PeakCalibrationCurve.m](#) ne è un esempio in Matlab/Octave. Questo script simula la calibrazione di un sistema di [iniezione di flusso](#) che produce picchi correlati a una concentrazione e un'ampiezza in esame ('amp'). In questo esempio, sei standard noti vengono misurati in sequenza, ne risultano 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:



- un *casuale* [rumore bianco](#) aggiunto a tutti i punti del segnale, controllato dalla variabile "Noise";
- background* - ampio background curvo di *ampiezza casuale*, inclinazione e curvatura, controllato da "background";
- ampliamento* - [ampliamento esponenziale del picco](#) che *varia casualmente* da picco a picco, controllato da "broadening";
- 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-avvallamento, 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, funzione-

dei minimi quadrati classico (CLS) e in quello iterativo non-lineare dei minimi quadrati, la selezione 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*).

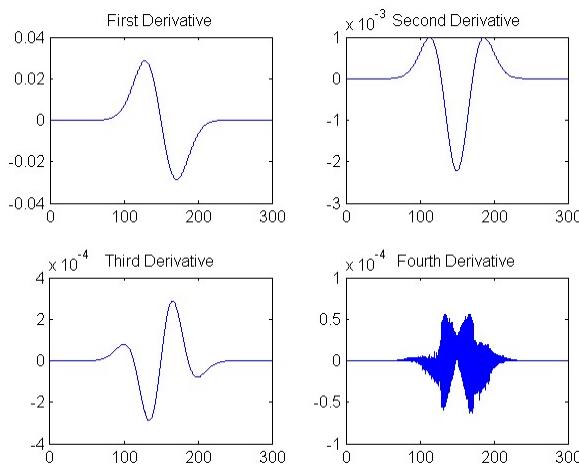
Nonostante tutto ciò, è comunque meglio utilizzare un profilo quanto più accurato possibile per il segnale, poiché è possibile utilizzare l'errore di approssimazione percentuale o l' R^2 del segnale approssimato possono essere usati per evidenziare che c'è qualcosa che non va, come un aumento del rumore 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.

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 sovrapposte 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 293.

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 *segnale* 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 scorrimento della media ne rimuove la maggior parte ([click per visualizzare](#)).

4.2. Lo si può avere con tantissimo di codice installato, compresa una versione del sistema operativo Linux, una semplice ma efficace GUI grafica sul modello di Windows, un browser Web, la [suite LibreOffice](#) completa, [Mathematica](#) di Wolfram ([schermata](#)), diversi linguaggi di programmazione, alcuni giochi e varie utility. [Tutti questi sono installati di default dal programma di installazione per](#) Raspberry Pi del [sistema operativo](#). (Ci sono modelli ancora più piccoli ed economici chiamati microcontrollori [Zero](#) e [Pico](#), che costano 4-5 dollari. A causa del loro basso costo e delle dimensioni ridotte, questi modelli sono ideali in situazioni in cui potrebbero essere danneggiati o persi, come negli esperimenti con missili o palloni-sonda). Il nuovo [Raspberry Pi 4](#), una versione da 70 dollari con una tastiera completa e che esegue Windows 10, necessita solo di un monitor e di un mouse.

Per costruire un computer completo dalla versione 3, c'è bisogno di un alimentatore da 5 volt, 2 ampere, un TV/monitor con input HDMI, una tastiera e un mouse USB (acquistabili probabilmente in un negozio 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 o usarne una vuota in cui è possibile scaricare il software gratuito autonomamente). Infatti, se dispone già di una rete Wi-Fi e di un computer, tablet o smartphone connesso a Internet, non c'è bisogno di monitor, tastiera e mouse: una volta configurato, si può accedere al Raspberry Pi tramite la rete Wi-Fi o da Internet, utilizzando [Putty](#) (per un accesso a riga di comando in stile UNIX) o utilizzando un sistema di condivisione grafico 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. Può anche [condividere file con Windows](#). 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 (il word processor *Writer*, lo spreadsheet *Calc*, ecc.), e per la programmazione (Python, pag. 417, C, C++, Java, Scratch e Ruby). È ideale anche per applicazioni “[headless](#)” a cui, dopo averlo configurato, si *accede solo da remoto* via WiFi o Bluetooth, per esempio un [file server di rete](#), [stazione meteo](#), [media center](#) o come [telecamera di sorveglianza](#) collegata in rete.

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. 417). 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](#) dell'autore (pagina 473) funzionano perfettamente sulla versione di *Calc* del Pi, così come quelli di [Calibrazione](#) (pagina 430) e i [modelli di strumenti analitici](#) (pagina 338).

Per le applicazioni scolastiche, Element14 commercializza [Learn to Program Pack Starter Kit](#) (\$177) che comprende una

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

- (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).

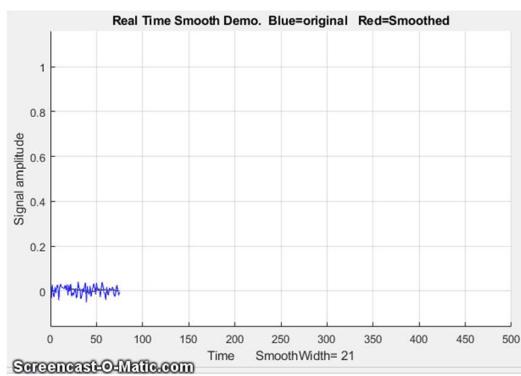
Il secondo metodo è illustrato dallo script [realtimeplotautoscale2.m](#), che simula i dati in tempo reale utilizzando [dati pre-calcolati](#) (caricati dal disco rigido nella riga 13) a cui si accede punto per punto nelle righe 25 e 26 (Se l'animazione non è visibile, cliccare sulla figura per aprirla 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](#).

Screencast-O-Matic.com

Smoothing. Lo script [RealTimeSmoothTest.m](#) mostra lo [smoothing \(pag. 40\)](#) 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



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](#)

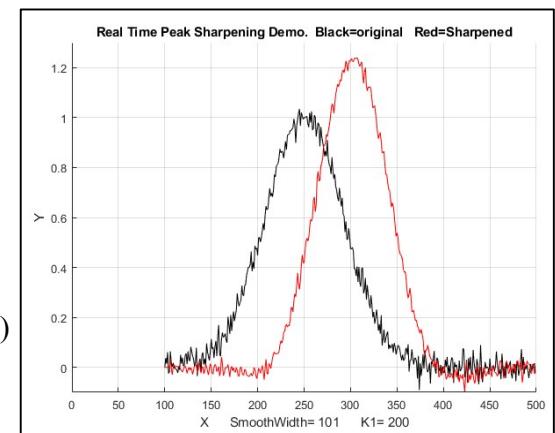
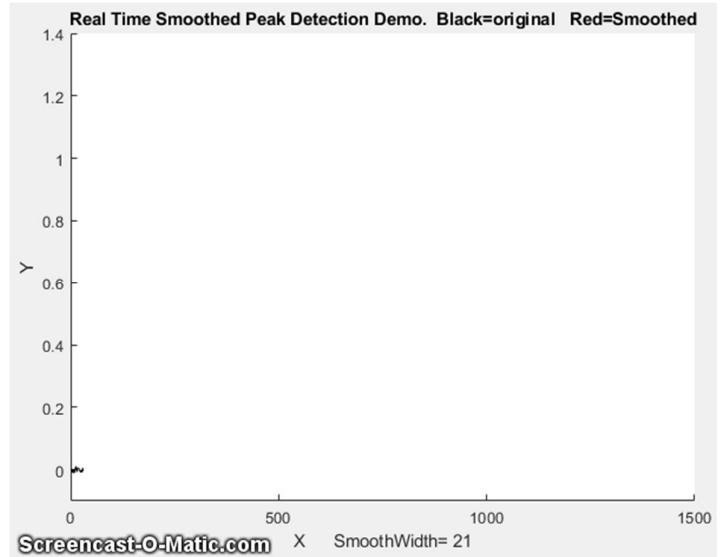
227; individua i picchi positivi in un insieme di dati rumoroso che salgono al di sopra di una soglia di ampiezza impostata ("AmpThreshold" nella riga 55), esegue un'[approssimazione della curva dei quadrati minimi a una funzione Gaussiana](#) alla *parte superiore del picco dei dati originali* (riga 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 239 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. 73) 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-



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 180) 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 345.

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.).

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 ($x=\text{lunghezza d'onda}$, $y=\text{assorbanza o intensità}$) o cromatogrammi ($x=\text{tempo}$, $y=\text{risposta del rilevatore}$). 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

	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

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 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

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		

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).

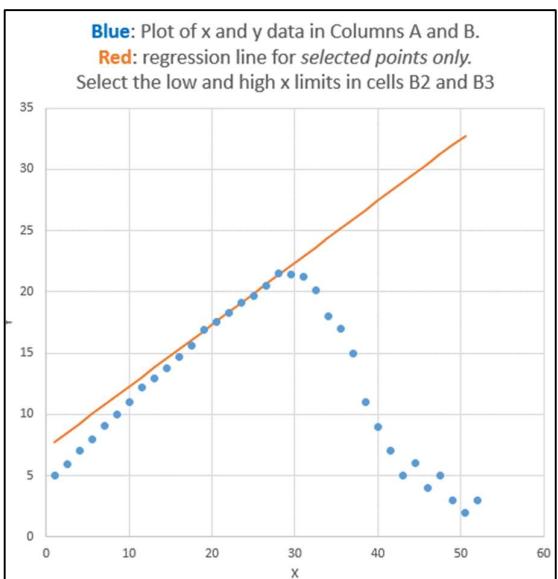
Le colonne J e K di questo foglio mostrano anche come utilizzare le funzioni "IF" e "AND" per copiare i dati dalle colonne A e B nelle colonne J e K solo quei punti che rientrano tra i due limiti di x specifici.

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).

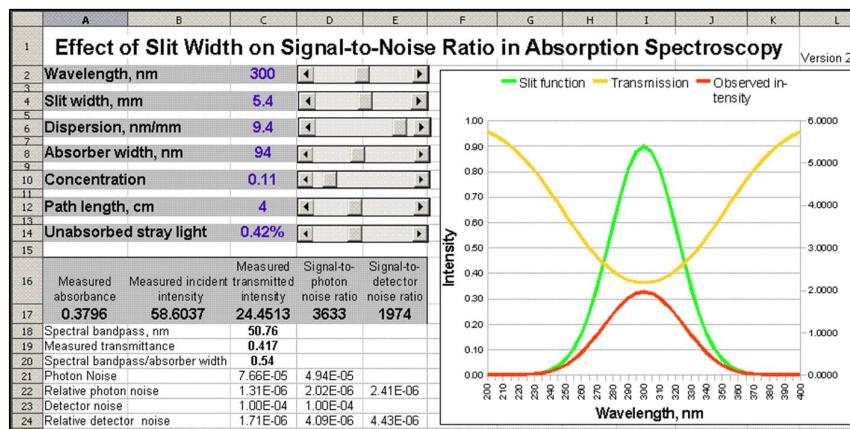
Misura della posizione di un picco. Una comune operazione di elaborazione del segnale è trovare il valore dell'asse x dove il valore dell'asse y è massimo. Questo può essere suddiviso in tre fasi: (1) determinare il valore y massimo nell'intervallo selezionato con la funzione MAX; (2) determinare il 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 260 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'a-

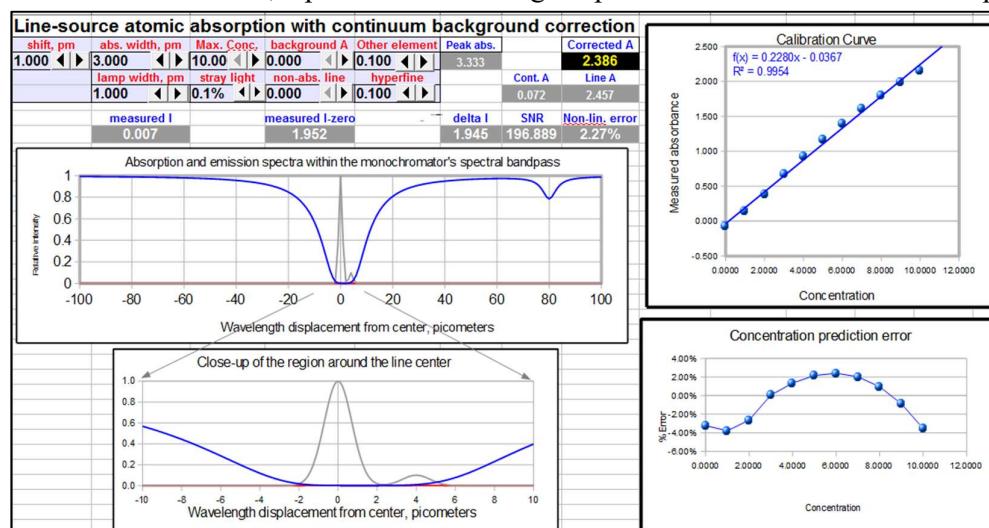


comportamento reale del segnale che potrebbe essere necessario misurare. Ad esempio, nella sezione “Il Torneo: un confronto dei metodi” a pagina 283, il segnale da misurare è un picco Gaussiano situato vicino al centro del segnale registrato, con una forma e larghezza fisse. La linea di base, però, è 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 297, 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 306, 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 [lar-](#)

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 432) 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

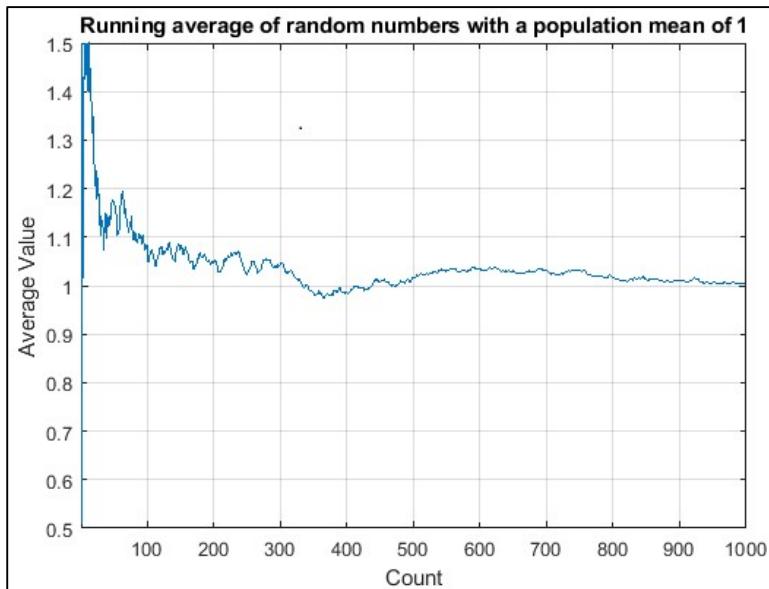
- il numero di *persone che parlano inglese* in ciascun paese,
- il numero di *utenti Internet* in ciascun paese,
- il numero di *università* in ciascun paese, e
- il *budget totale di ricerca e sviluppo* di ciascun paese.

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

ca 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 486).

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*.

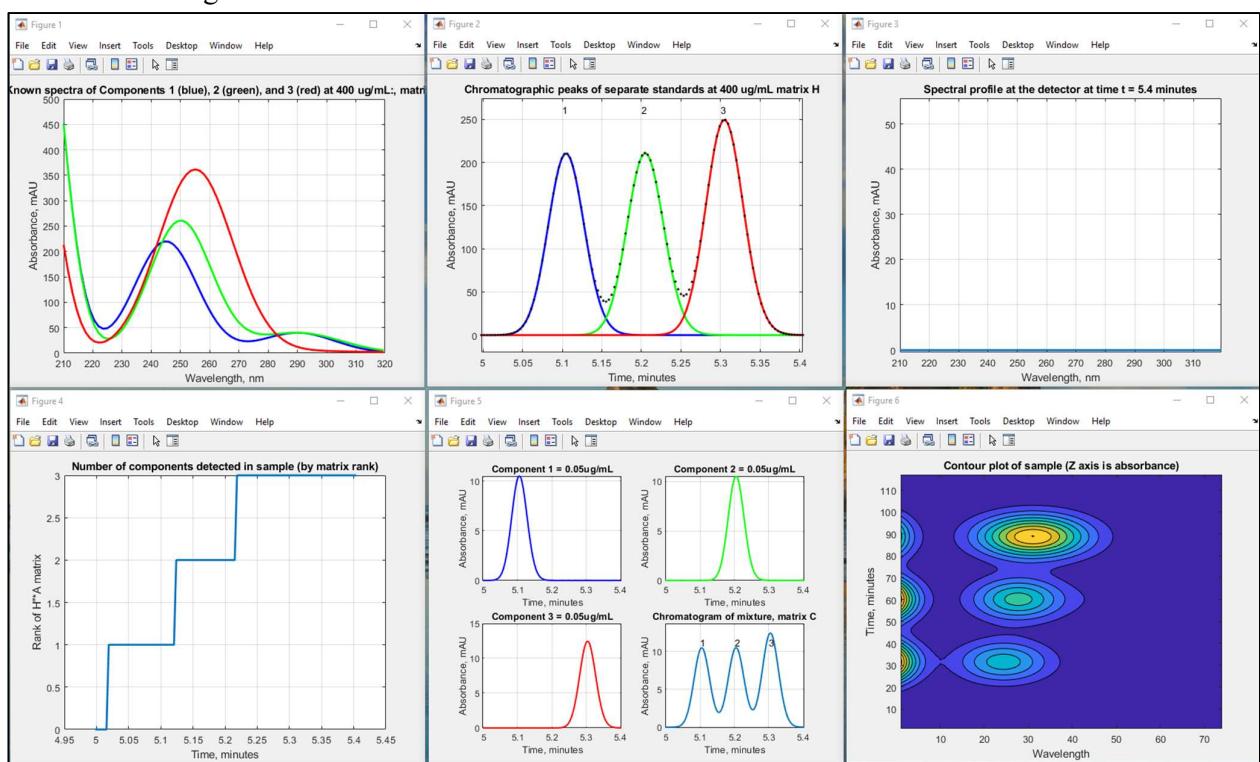


Ma l'idea che “La media tenderà ad avvicinarsi man mano che vengono eseguite più prove” *non significa* che la media diventa *costantemente e irreversibilmente* più vicina. In effetti, la media può variare un po' man mano che vengono inclusi più dati. Prendiamo l'esempio a lato, che mostra la media corrente di un insieme di numeri casuali indipendenti normalmente distribuiti con una popolazione media di 1.000 e una deviazione standard di 1.000, man mano che 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.

Combinazione di spettroscopia e cromatografia: I Minimi Quadrati Classico risolto nel tempo

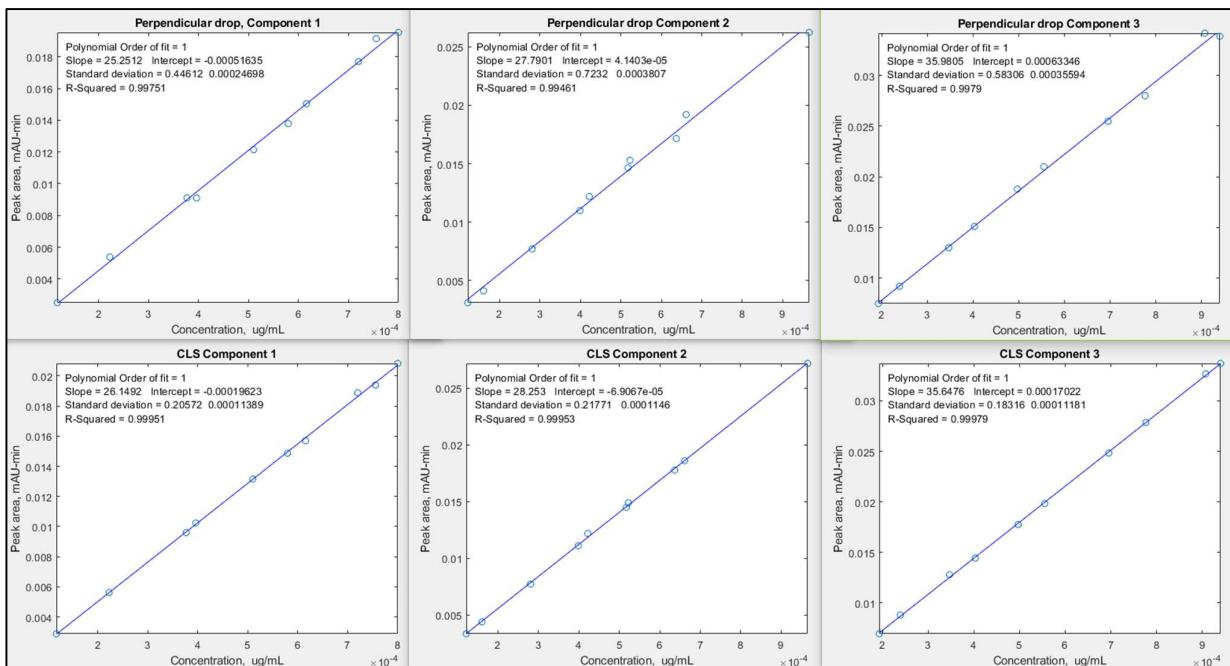
L'introduzione dei veloci array di rilevatori UV-Visibile negli strumenti di cromatografia liquida ad alte prestazioni (HPLC) ha notevolmente aumentato la potenza di questo metodo. La velocità di tali rivelatori consente loro di acquisire uno spettro completo UV-Visibile più volte al secondo sull'intero cromatogramma. A titolo di esempio ci si riferirà a un rapporto tecnico della Shimadzu Scientific Instruments (<https://solutions.shimadzu.co.jp/an/n/en/hplc/jpl217011.pdf>) che considera la separazione di tre isomeri posizionali del metile acetofenone: o-metile (o-MAP), m-metile (m-MAP) e p-metile (p-MAP) mediante cromatografia liquida con un rivelatore UV a 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 separazione cromatografica, utilizzando la colonna e le condizioni specificate nel loro rapporto; i picchi sono solo parzialmente risolti. Il rapporto Shimadzu 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. 180), 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

3. Miscelato	Normale	Grande: tau=40	.05 .05 .05	-.0004% -.013% -.066%	Grafico mfile
4. Non risolto	Molto prossimo	Leggero: tau=10	.01 .01 .01	.054% .049% .04%	Grafico mfile
5. Non risolto	Molto prossimo	Leggero: tau=10	.01 .0001 .01	0.026% 2.4% 0.019%	Grafico mfile
6. Non risolto	Molto prossimo	Grande: tau=40	.01 .0001 .01	-0.04% -3.8% -0.03%	Grafico mfile

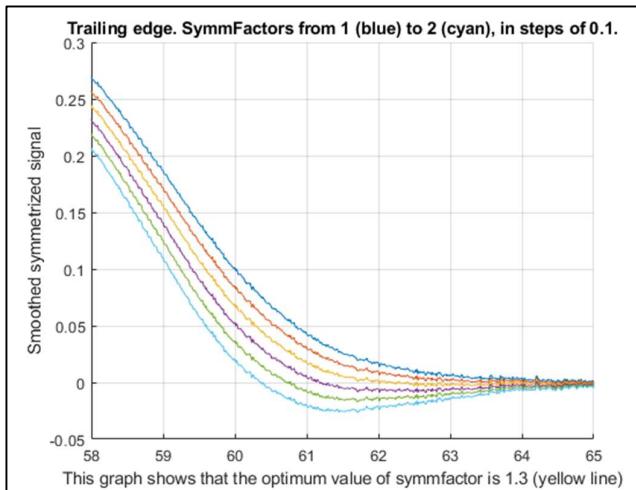
Anche quando i picchi sono abbastanza distinti da far funzionare il metodo del taglio verticale, può soffrire di un'interazione tra le altezze dei picchi adiacenti; cioè una variazione dell'altezza di un picco può influire sulla misura dell'area dei picchi sovrapposti adiacenti, a causa delle variazioni 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.

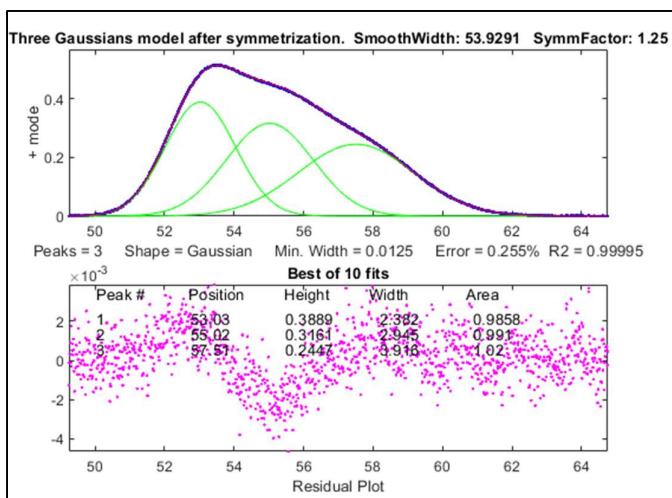
In altre applicazioni, alcuni o tutti i componenti potrebbero essere semplicemente sconosciuti e si potrebbe voler ottenere i loro spettri. Questo può essere fatto *in situ* se la separazione dei picchi è buona almeno quanto quella illustrata a pagina 345, perché in ogni massimo di picco non c'è vir-

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 217). Un altro approccio al problema dei picchi asimmetrici consiste nell'utilizzare la tecnica di *simmetrizzazione della derivata prima* descritta a pagina 76. 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 $1/10^{\circ}$ 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-](#)

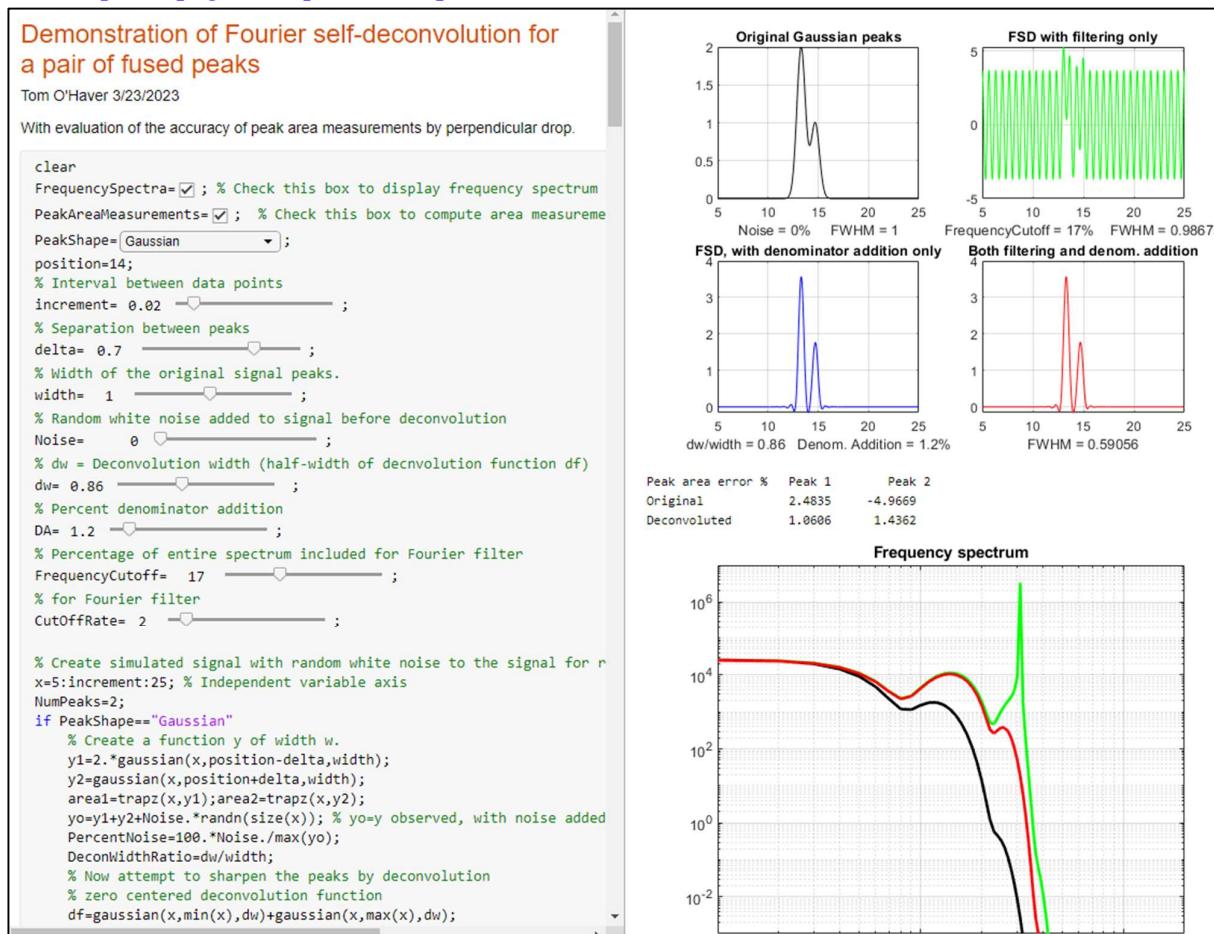


fwidht(1:length(y),y), quindi una larghezza di smoothing prossima a 53 non distorce il picco del segnale, ma elimina la maggior parte il rumore dalla derivata (sopra a destra). Inoltre, possiamo vedere che la derivata, in unità *y/x*, è paragonabile come grandezza numerica al segnale originale, quindi 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 effetti è costituito da tre EMG, con massimi di picco a



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.

Interactive Tutorial

Respond to Numerical Input

Respond to User Selections

Embed HTML Content

Lay Out Controls in a Grid

▼ Examples: Programming Tasks

Link Data to a Tree

Analyze an Image

Configure a Timer

Display Specialized Axes

Create a Table

Query Website Data

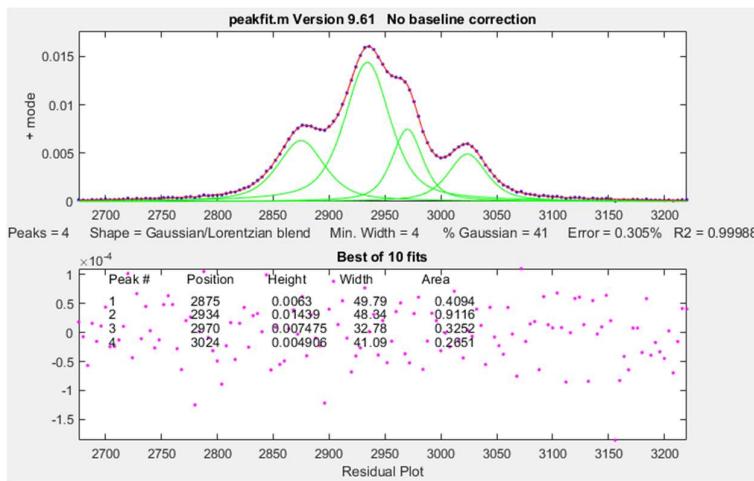
Lo svantaggio delle app è che sono più complesse per il programmatore. In effetti, la quantità di codice, il tempo e lo sforzo di codifica necessari alla progettazione e all'interattività dell'interfaccia utente di solito supera di gran lunga il codice richiesto per gli effettivi calcoli matematici.

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.mplappinstall](#).

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 191). 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 comandando 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 59) per individuare le posizioni originali dei picchi, misura dell'area dei picchi mediante taglio verticale (pagina 134), 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)

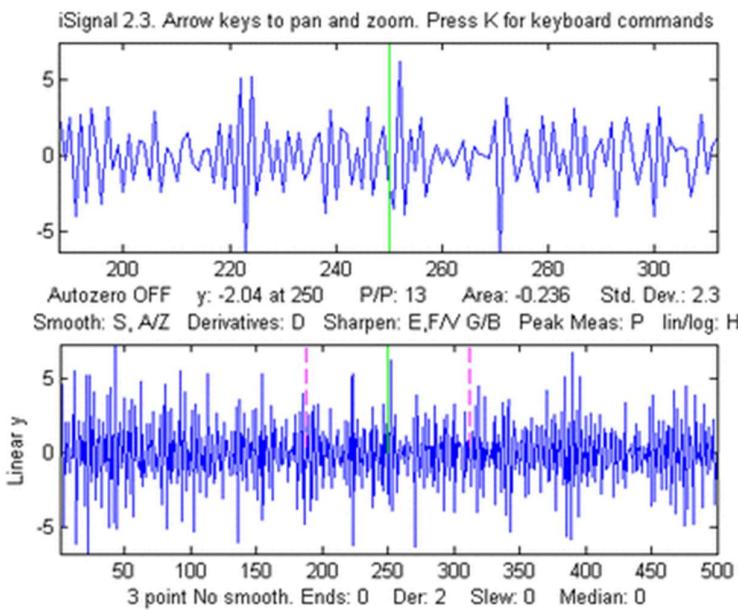
Come mostrato dalla **prima figura** nella pagina seguente, lo sharpening dell'auto-deconvoluzione può infatti migliorare sostanzialmente la precisione dell'area del taglio verticale, da un errore medio del 29% a solo il 3,1% dopo la deconvoluzione. Ma poiché i picchi hanno larghezze diverse, non esiste 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

Dettagli software del signal processing.

Smoothing, differenziazione e analisi del segnale interattivamente (iSignal)

[iSignal](#) (o la versione Octave [isignaloctave.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);
```

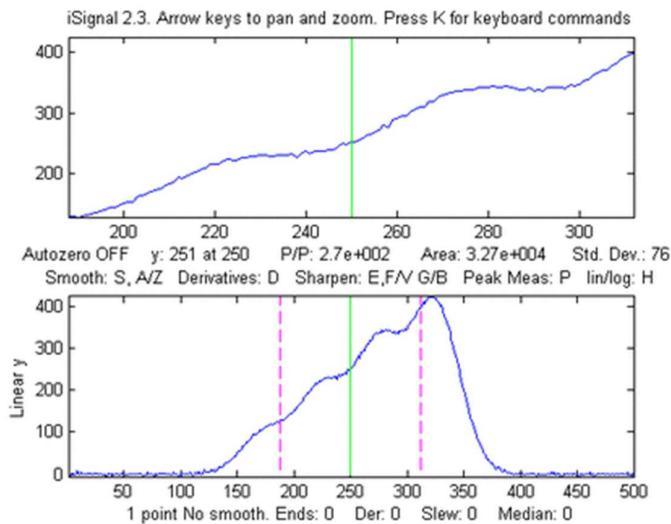
"Data" può essere una matrice con 2 colonne con la variabile indipendente (valori x) nella prima colonna e quella dipendente (valori y) nella seconda colonna, o vettori x e y separati, o un singolo vettore y (in questo caso i punti vengono disegnati rispetto ai loro numeri di indice sull'asse x). È richiesto 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

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](#).

Sharpening del picco

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 317). 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.

Menù per la convoluzione/deconvoluzione di Fourier

- 1. Convolution
- 2. Deconvolution

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.

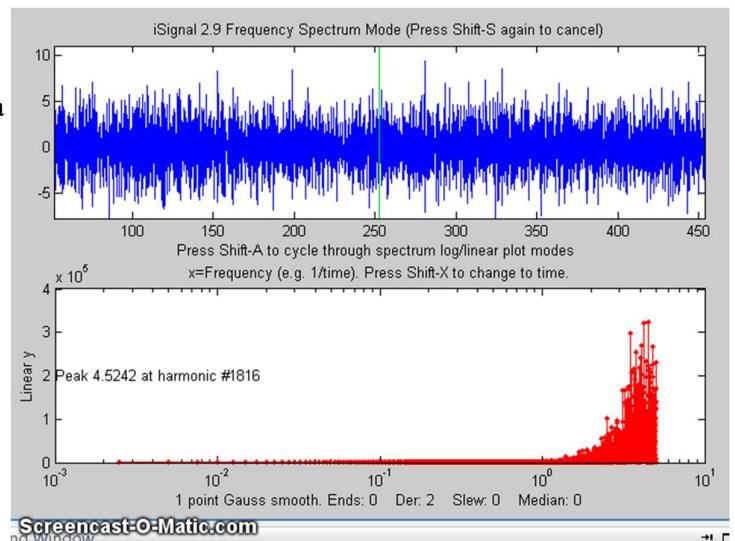
Modalità Spettro delle Frequenze

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);

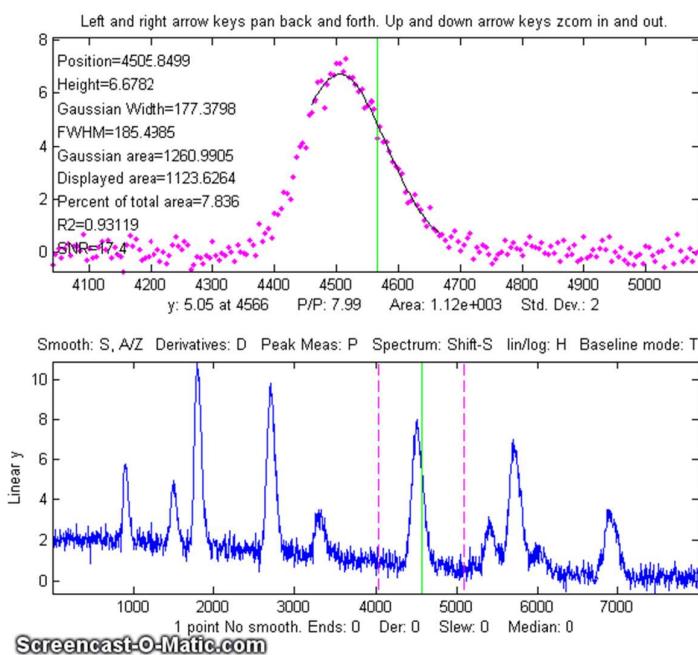
>> plot(Spectrum(:,1),Spectrum(:,2)) o plotit(Spectrum) o
isignal(Spectrum); o ipf(Spectrum); o ipeak(Spectrum)
```



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.).

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.

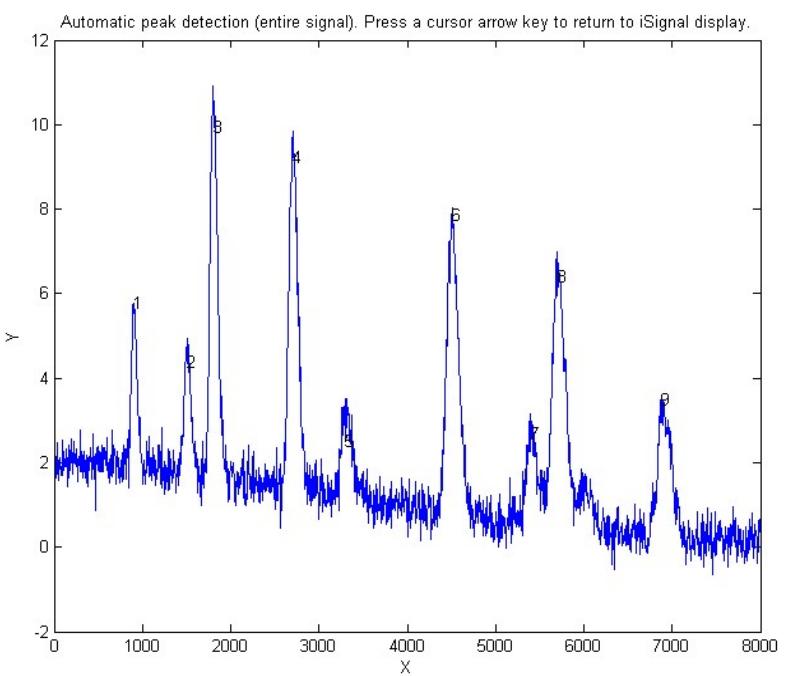
Un esempio è mostrato nella figura a lato. Se i picchi sono sovrapposti su un background diverso da



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 approssima meglio la porzione centrale del segnale visualizzato nella finestra superiore, contrassegnata in rosso. "L'area totale" è



Gaussiane in posizioni fisse.....	16
Gaussiane asimmetriche con diverse semi-larghezze su entrambi i lati.....	14
Lorentziane: $y=ones(size(x))./(1+((x-pos)./(0.5.*width)).^2)$	
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/Lorentziane (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
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
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 Alpha: $y=(x-spoint)./pos.*exp(1-(x-spoint)./pos)$;.....	19
Sigmoide in salita (funzione logistica): $y=.5+.5*erf((x-tau1)/sqrt(2*tau2))$...10	
Sigmoide in discesa $y=.5-.5*erf((x-tau1)/sqrt(2*tau2))$	23
Triangolare.....	21

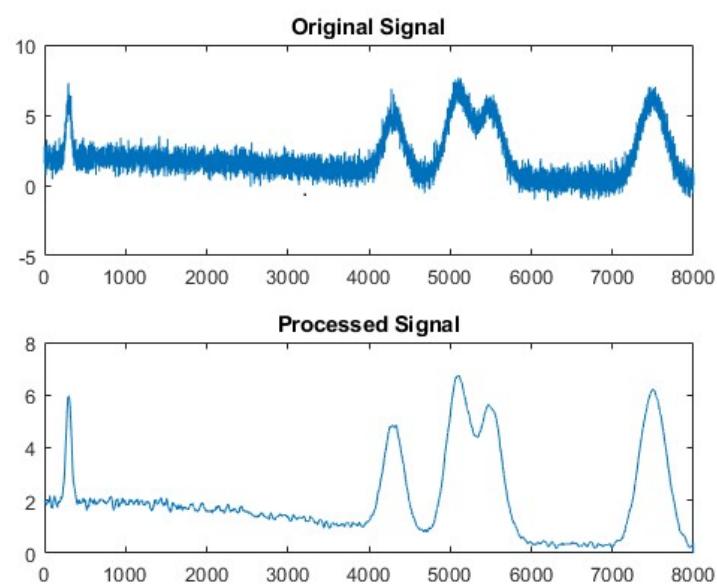
Nota: se si ha un picco Gaussiano o Lorentziano espanso esponenzialmente, si può misurare sia l'altezza, la posizione e la larghezza "dopo l'ampliamento" utilizzando il tasto **P**, sia l'altezza, la posizione e la larghezza "pre-ampliamento", e approssimare la larghezza con l'approssimazione del picco 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



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.

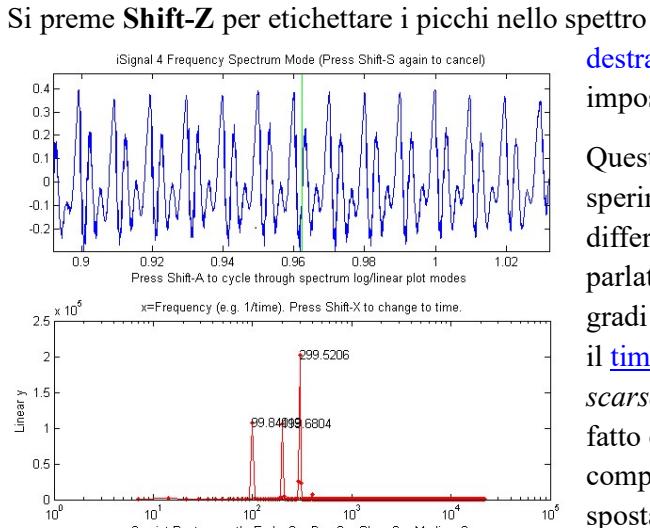
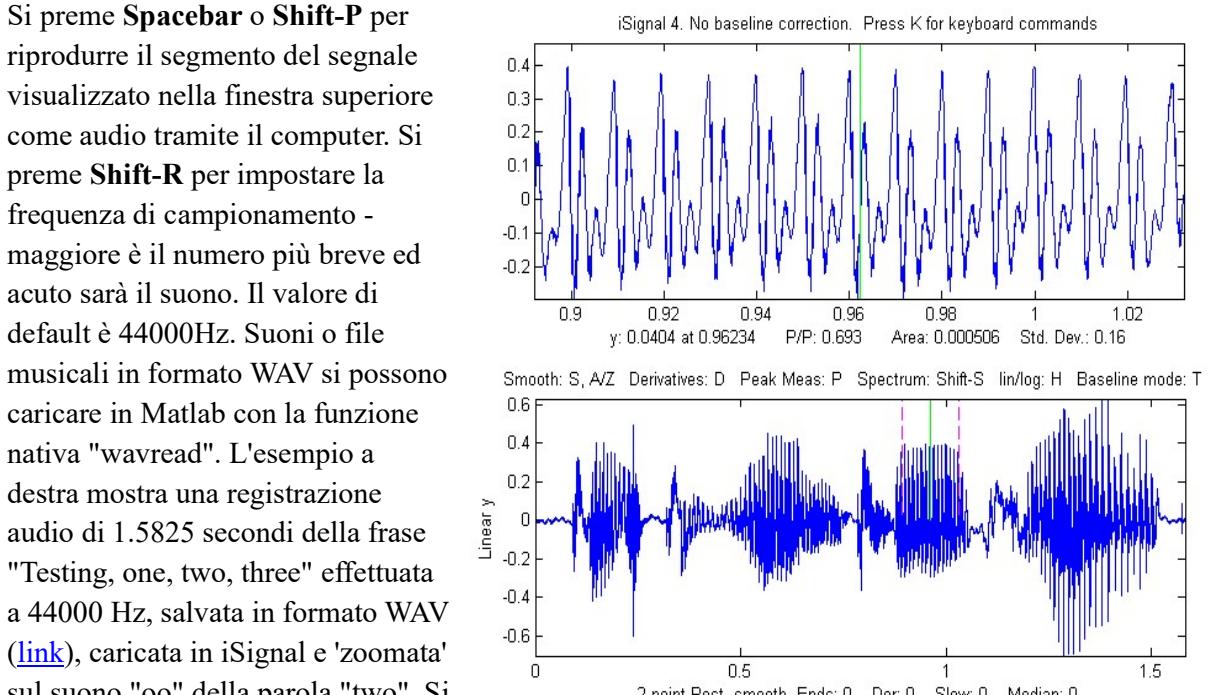
Si preme **Spacebar** o **Shift-P** per riprodurre il segmento del segnale visualizzato nella finestra superiore come audio tramite il computer. Si preme **Shift-R** per impostare la frequenza di campionamento - maggiore è il numero più breve ed acuto sarà il suono. Il valore di default è 44000Hz. Suoni 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.

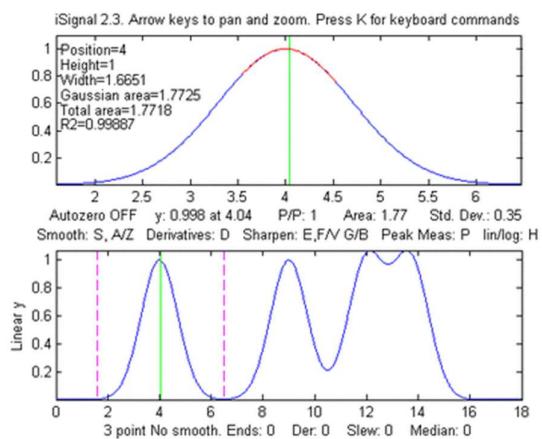
```
>> v=wavread('TestingOneTwoThree.wav');
>> t=0:1/44001:1.5825;
>> isignal(t,v(:,2));
```

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





nella misura dell'area utilizzando iSignal può essere 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 \sim) 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.

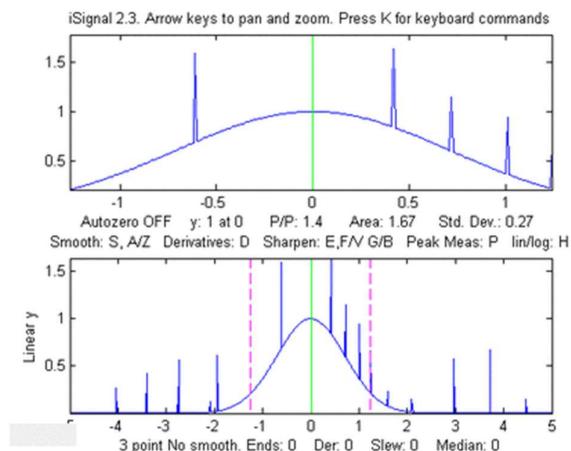
Lo script demo [isignaldemo2](#) (mostrato a lato) crea un segnale di prova contenente quattro picchi con altezze 4, 3, 2, 1, con larghezze uguali, sovrapposti, su una linea di base curva molto forte, con in più del rumore bianco casuale.

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)

```
>> 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



```

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

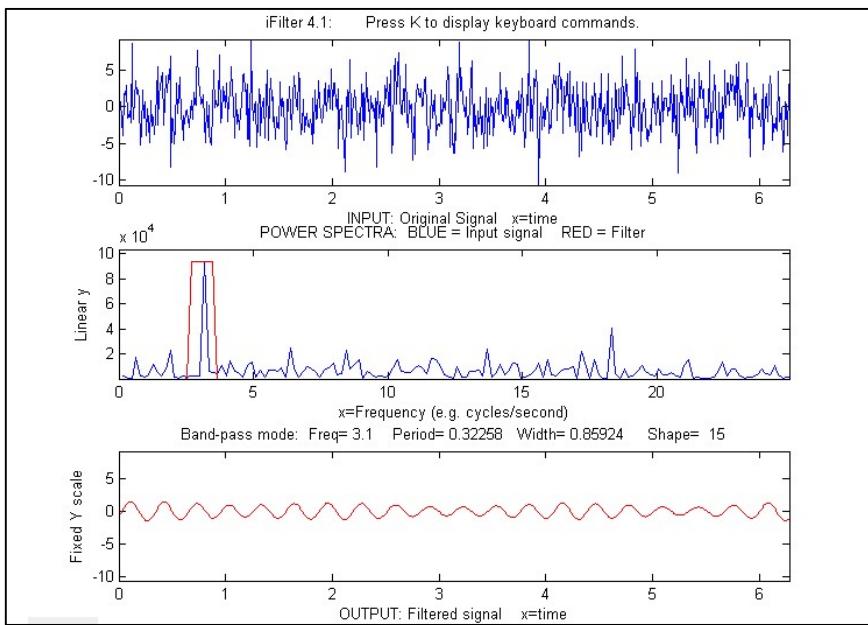
```

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,Symize,Symfactor,SlewRate,MedianWidth)
```

Filtro di Fourier gestito da tastiera

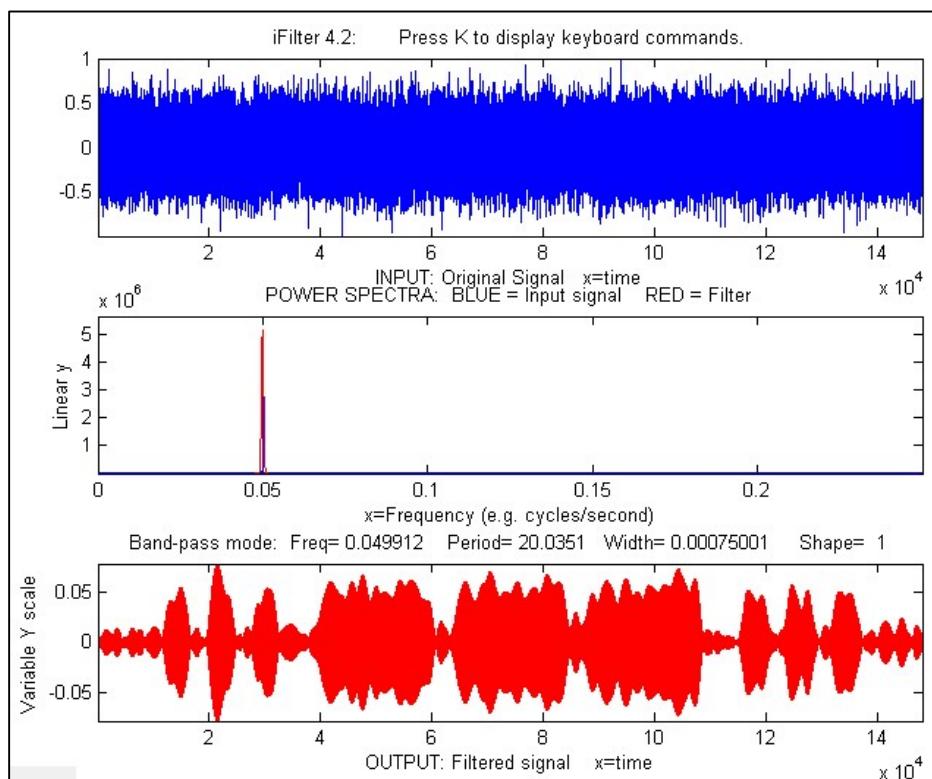
iFilter.m è una funzione Matlab per un filtro di Fourier interattiva da tastiera per un segnale temporale (x,y), con controlli da tastiera che consentono di regolare continuamente i parametri del filtro mentre, dinamicamente, se ne osserva l'effetto sul segnale. Gli argomenti opzionali di input impostano 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



Esempio 4: Onda quadra con filtro passa banda rispetto al filtro a pettine passante

```
t = 0.:0.0001:0.0625;
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](#)



```

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 191) 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**,

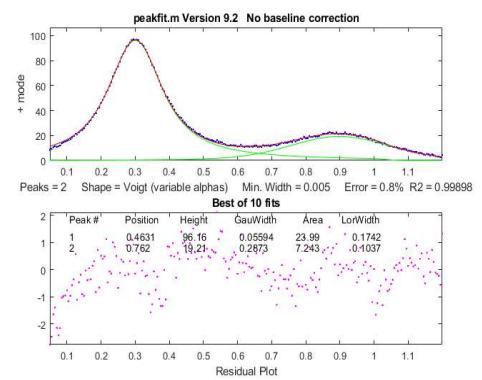
(1) una [versione a riga di comando](#) (**peakfit.m**) per Matlab o Octave, "[Scelta della Settimana](#)" in Matlab File Exchange. La versione corrente è la **9.61**.

(2) una [versione interattiva da tastiera](#) (**ipf.m** o **ipfoctave.m**), pagina 394. 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 328); 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 401; [si possono aggiungere](#) dei profili personalizzati (vedere pagina 414). Vedere [le pagine](#) 410 e 415 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-**
ne, coeff, residuals, xi, yi, BootRe-
sults]=**peakfit**(**signal, center, window,**
NumPeaks, peakshape, extra, NumTrials, start, BASELINEMODE, fixedpara-
eters, plots, bipolar, minwidth, DELTA, clipheight). Solo il primo argomento di



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: $Bo * \exp(-\exp((Kh * \exp(1) / Bo) * (L - t) + 1))$); **44**= $1 - \exp(-k * x)$; **45**: Quattro parametri logistici $y = maxy * (1 + (miny - 1) / (1 + (x / ip)^{slope}))$; **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 basiliari dei picchi (grafico a pagina 401) 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,
```

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

```

> y=[0 1 2 4 6 7 6 4 2 1 0 ]; x=1:length(y);
> peakfit([x;y],length(y)/2,length(y),0,0,0,0,0,0)
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 più 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

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 quadratica 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],5,5.5,0,0,0,0,0,2)
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.

prossimazione 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)
```

ficare 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. (Il default è 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.

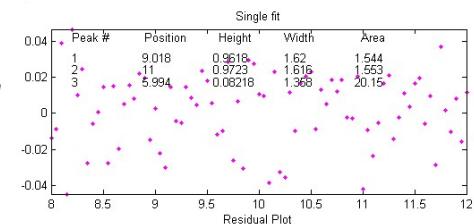
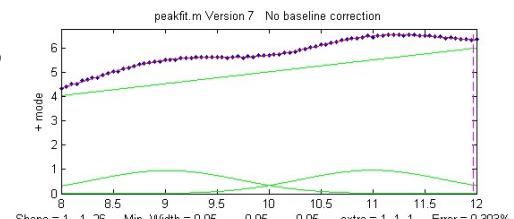
```
>> x=[-10:.05:10];y=exp(-
```

```
(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);
```



```

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.

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 R^2 resta prossimo a 1.00.

Come trovare gli argomenti di input corretti per peakfit?

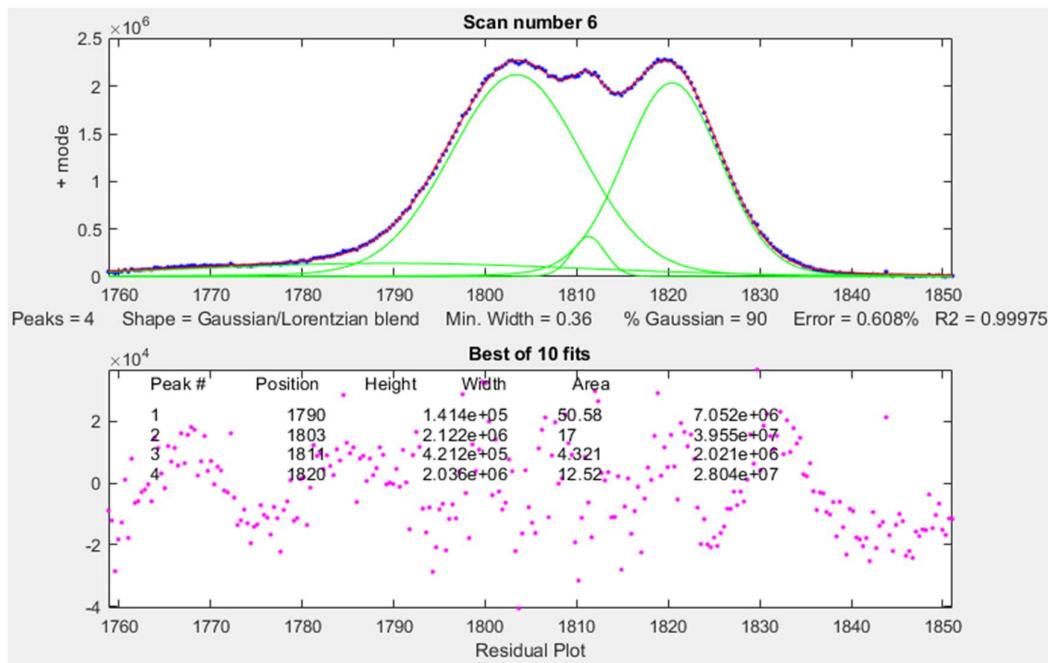
Se non si sa da dove iniziare, si può usare l'[Interattivo Peak Fitter \(ipf.m\)](#) per provare rapidamente diverse regioni di approssimazione, profili, numero di picchi, modalità di correzione della linea di base, numero di prove, ecc. Ottenuta una buona approssimazione, si preme il tasto "W" per stampare 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](#).



Gestire i segnali complessi con molti picchi

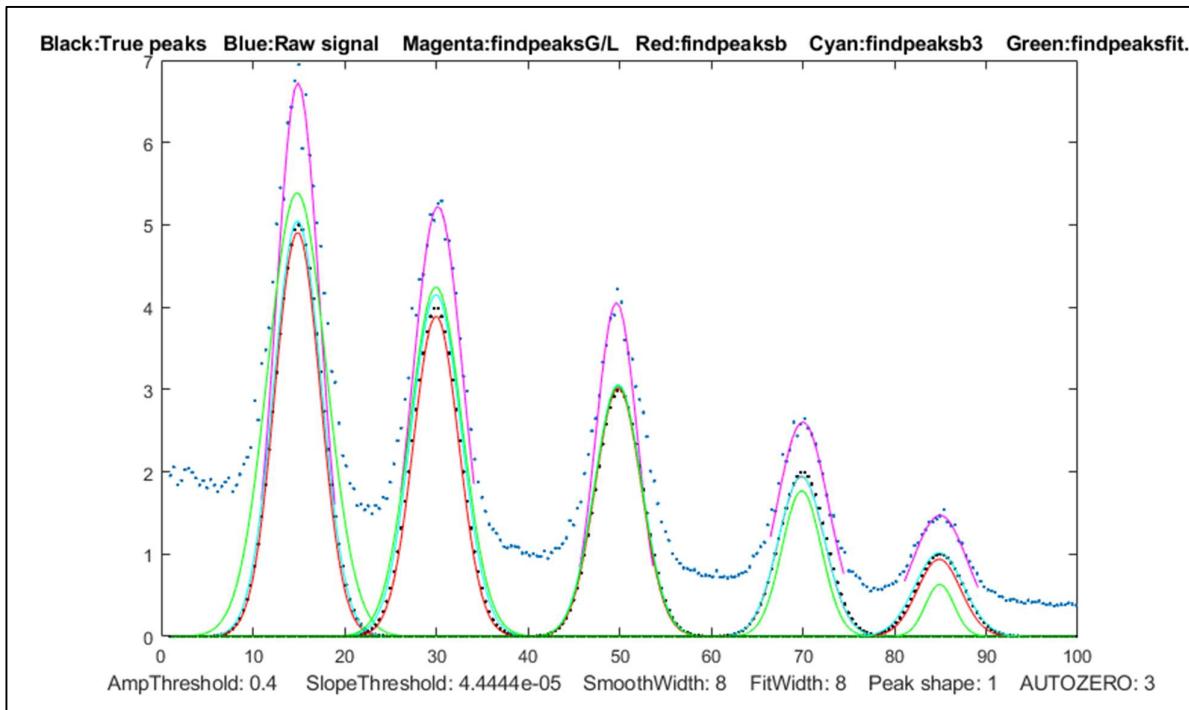
Quando un segnale è costituito da molti picchi su un background altamente variabile, l'approccio migliore è spesso quello di utilizzare `peakfit` con gli argomenti "center" e "window" per suddividere il segnale in segmenti contenenti gruppi più piccoli di picchi sovrapposti con i loro segmenti di background, 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. 401) 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`.



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:

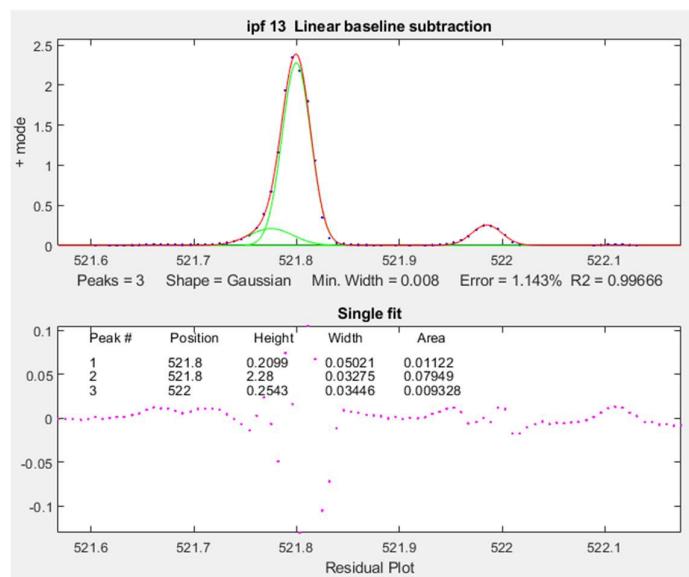
`ipf(y) or ipf([x;y]) or ipf([x;y]');`

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) oppure`



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.

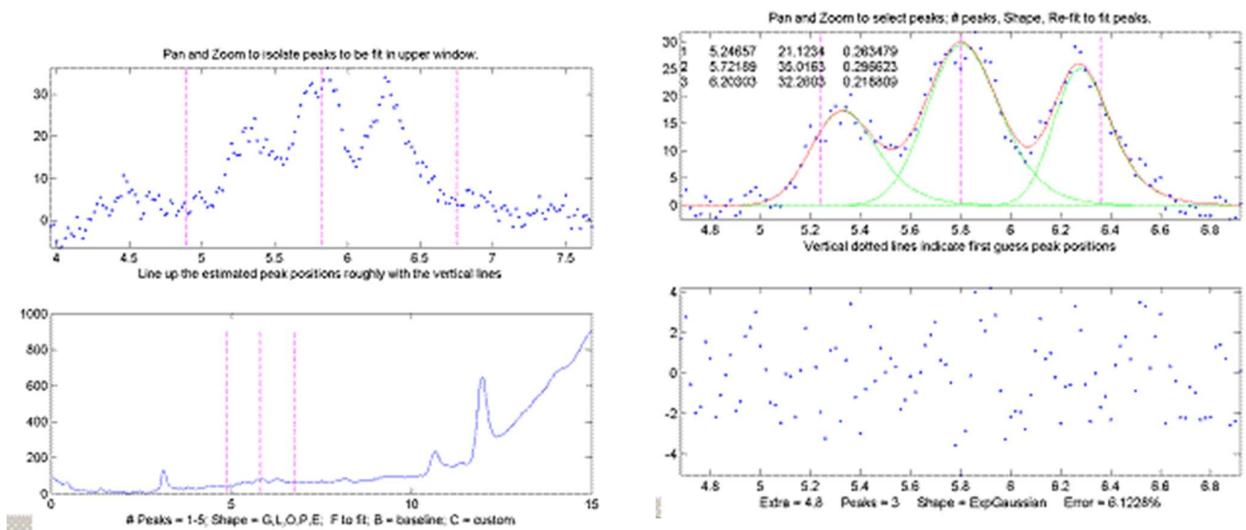
```
Height=[1 2 2 3 3 3 4 4 4 4];
Position=[10 30 35 50 55 60 80 85 90 95]; Width=[2 3 3 4 4 4 5 5 5 5];
x=[0:.01:100];y=modelpeaks(x,10,1,Height,Position,Width,0);
ipf(x,y);
```

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.....Crtl-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
```

Esempi pratici con dati sperimentali reali:

1. Approssimazione di picchi cromatografici deboli e rumorosi con Gaussiane modificate esponenzialmente.



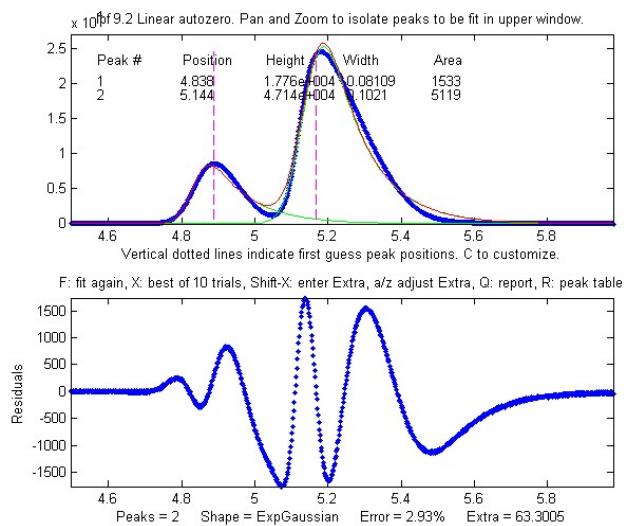
a. In questo esempio di dati reali, i controlli di pan e zoom vengono utilizzati per isolare un segmento di un cromatogramma da 4 a 7,5 minuti che contiene tre picchi molto deboli vicini a 5,8 minuti, su una linea di base sfalsata e leggermente inclinata. Il grafico inferiore mostra l'intero cromatogramma 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,

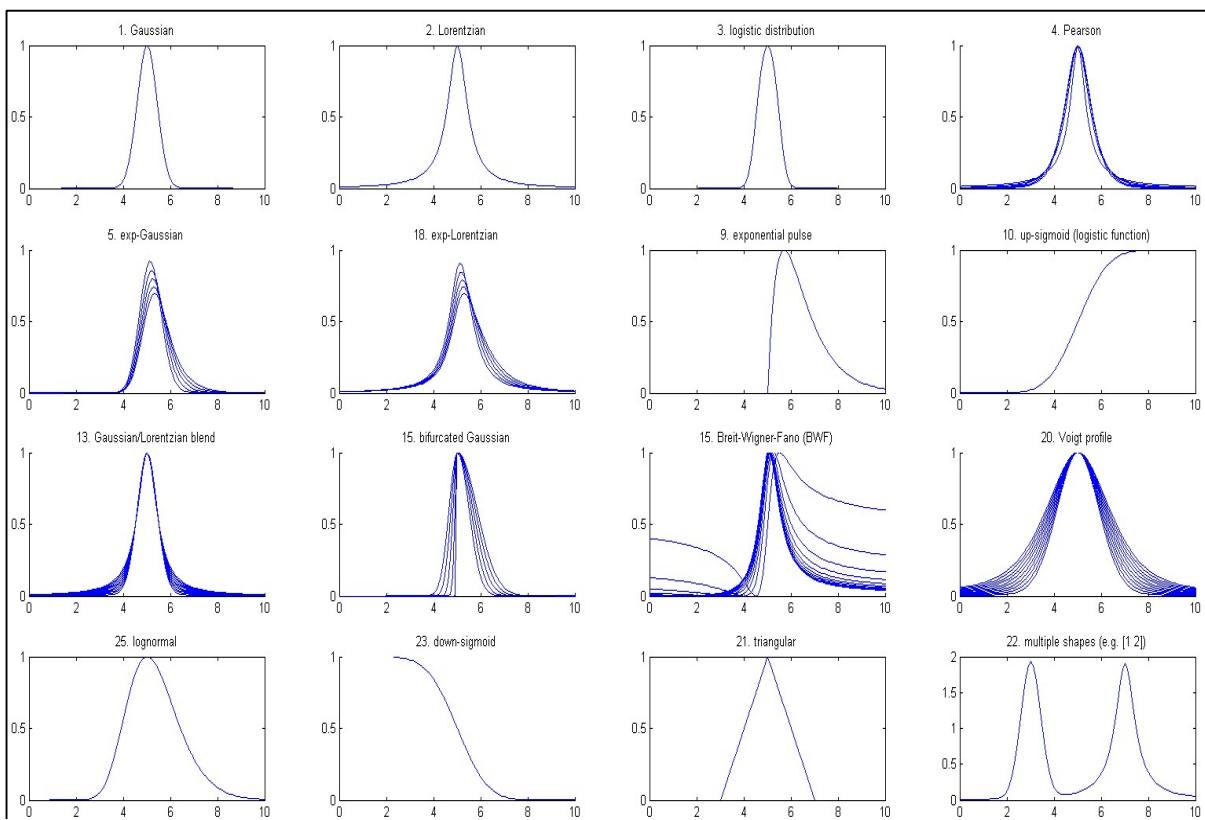


così chiaramente indicato. In un esempio descritto in precedenza a pagina 214, l'errore di approssimazione continua a diminuire man mano che vengono aggiunti più picchi al modello, tuttavia i residui rimangono "ondeggianti" senza mai diventare casuali. Senza un'ulteriore conoscenza dell'esperimento, 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. (*Usare le coppie di tasti < e > nonché ? e " per pan e zoom grossolani e i tasti delle parentesi quadre [e] per spostare di un punto a sinistra o a destra*). *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

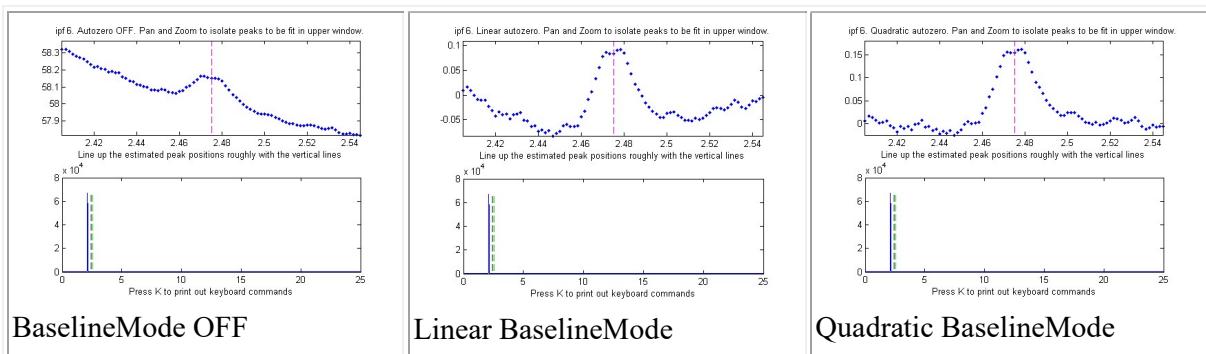


zata dopo la larghezza). Premendo **Q** viene stampato un report delle impostazioni e i risultati nella finestra di comando, in questo modo:

```

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

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(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-

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

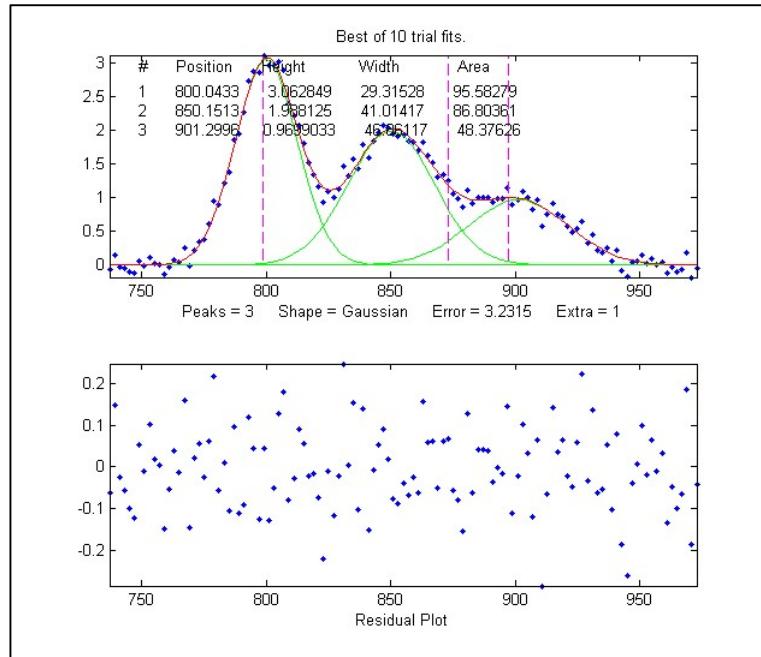
Elapsed time is 10.655257 seconds.

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 va-

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

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,

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 matematiche dei profili:

```
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 Alfa: y =(x-spoint)/pos*exp(1-(x-spoint)/pos)
Sigmoida in salita: y =.5+.5*erf((x-tau1)/sqrt(2*tau2))
Sigmoida 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:

$$y = 87.67 * \exp(-((x-0.30284) / (0.6005612 * 0.23732)) .^2)$$

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:

1. 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, $g = 1 - (1 ./ wid) .* abs(x-pos);$). 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 $\max(y)$).
2. 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*?

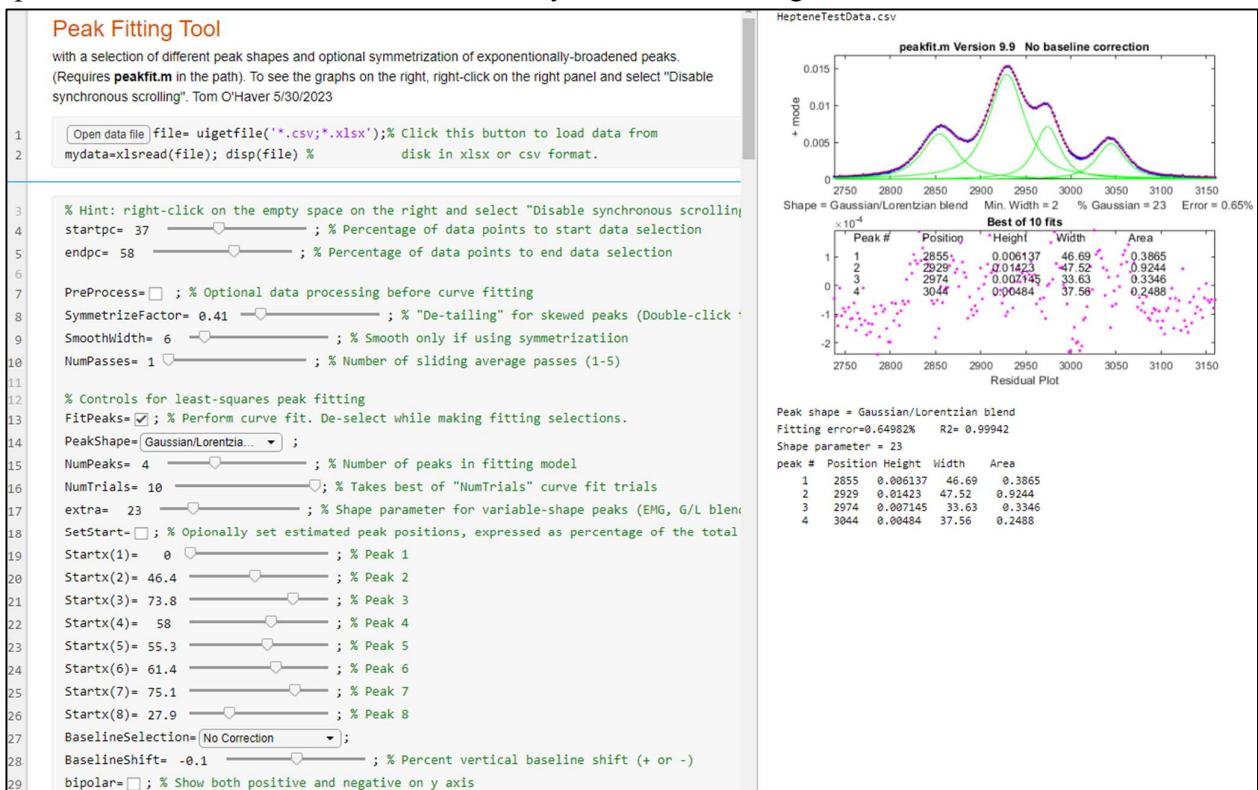
iPeak (pagina 394), *iSignal* (pagina 357), *peakfit* (pagina 375) e *ipf* (pagina 394), o le loro versioni Octave, sono state progettate ciascuna con un'enfasi diversa, sebbene vi siano alcune sovrapposizioni nelle loro funzioni. In breve, iSignal combina diverse funzioni di base, tra cui livellamento, 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.

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, si potrebbe non essere in grado di utilizzare il [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

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](#)

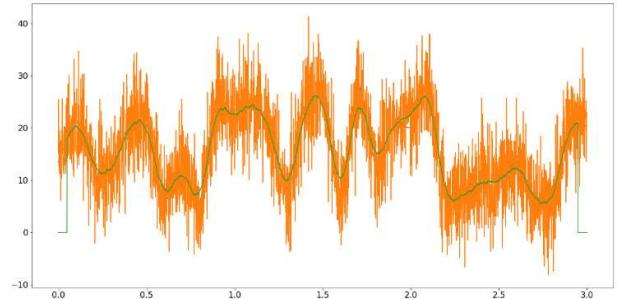
Uno smoothing a “media mobile” è semplice da implementare utilizzando la funzione “mean” nativa in entrambi i linguaggi.

Python

```
import numpy as np
    import matplotlib.pyplot as plt
from pytictoc 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=',')
```



Matlab: 7 lines: 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 pytictoc 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
```

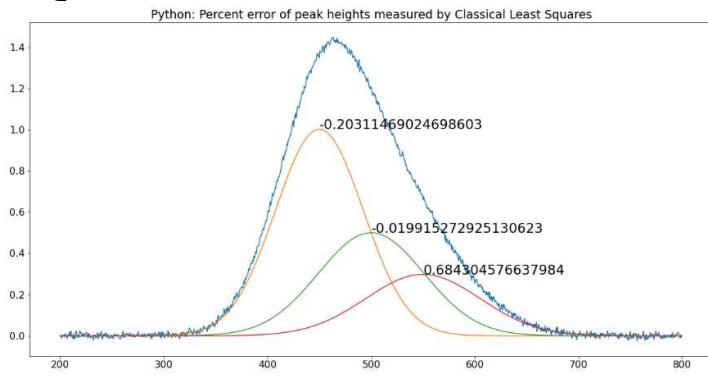
Matlab:

```
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. 180). 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.

```

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. 196, 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(m,:)*PEAKHEIGHTS (m));
end
hold off

```

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, all'inizio del 2024, è ChatGPT-4. Ora ci sono anche chatbot concorrenti come *Gemini* di Google e *CoPilot* di Microsoft. La forza di questi chatbot sta nell'interpretazione e nella scrittura del linguaggio. Ad esempio, sono abbastanza bravi nel suggerire possibili titoli per articoli, conferenze o proposte su cui si sta lavorando. Basta inserire tutto o una parte di ciò che si è scritto. Possono anche parafrasare, il che può essere utile per scrivere riassunti o "abstract" ridotti. Ma poiché vengono formati da un gruppo di scienziati e ingegneri informatici, non sorprende che sappiano qualcosa sui computer e sulle loro applicazioni tipiche. I chatbot sono spesso migliori di Google nel rispondere a domande specifiche sui programmi software che altrimenti ti costringerebbero a sfogliare pagine e pagine di documentazione.

Ma cosa sanno i chatbot sul signal processing? Ecco alcuni esempi di domande relative all'elaborazione del segnale che chiede al ChatGPT originale, eseguiti nel dicembre del 2022, circa due settimane dopo la sua disponibilità pubblica. È stato chiesto un consiglio su come (1) ridurre il rumore in un vettore di segnale campionato digitalmente; (2) rilevare i picchi in un vettore di segnale campionato digitalmente e (3) diminuire l'ampiezza dei picchi sovrapposti in un segnale campionato digitalmente. In tutti questi casi, ChatGPT ha dato risposte molto ragionevoli, fornite in un inglese perfetto. È interessante notare che se si pone di nuovo la stessa domanda, si otterrà una risposta leggermente diversa, proprio come se si ponesse la stessa domanda a un'altra persona. Questo è molto diverso da Google, che visualizza semplicemente una serie di link a siti Web pertinenti e che fornisce la stessa risposta se si ripete la stessa ricerca.

Ma ChatGPT può andare ben oltre; può scrivere codice in diversi linguaggi comunemente usati dagli scienziati, come Matlab, Python, Wolfram Mathematica, C/C++, R, Pascal, Fortran, HTML, JavaScript, Macro di Excel e probabilmente altri. Sulla base di test limitati dell'autore, ChatGPT può generare codice funzionante per *semplici* applicazioni, *se si descrive adeguatamente l'attività*.

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 restituiscia 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

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 40). 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 ripetute 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 .

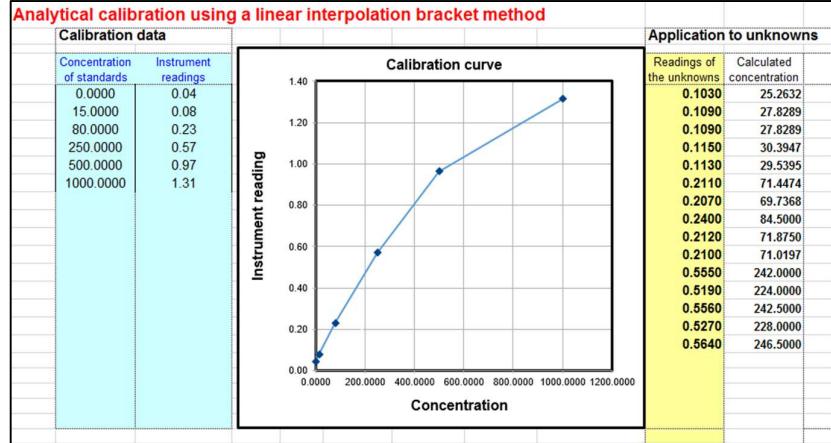
In terzo luogo, potrebbero esserci dettagli non specificati o effetti collaterali che potrebbero richiedere una gestione, come l'aspettativa che il numero di dati nel segnale *dopo* l'elaborazione debba essere lo stesso di *prima* dell'elaborazione. Nel caso dello smoothing, ad esempio, c'è anche la 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 42). 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 xo. Se xo 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 tra i punti dei dati quelli di mezza intensità rientrano *tra* i punti dati e quindi è imprecisa quando la frequenza di campionamento è bassa. "[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

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 $C_x = \frac{C_1 + (C_2 - C_1) \cdot (S_x - S_1)}{(S_2 - S_1)}$, 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, C₁ e C₂ sono le concentrazioni di queste due soluzioni standard e S_x è il segnale fornito dalla soluzione campione.

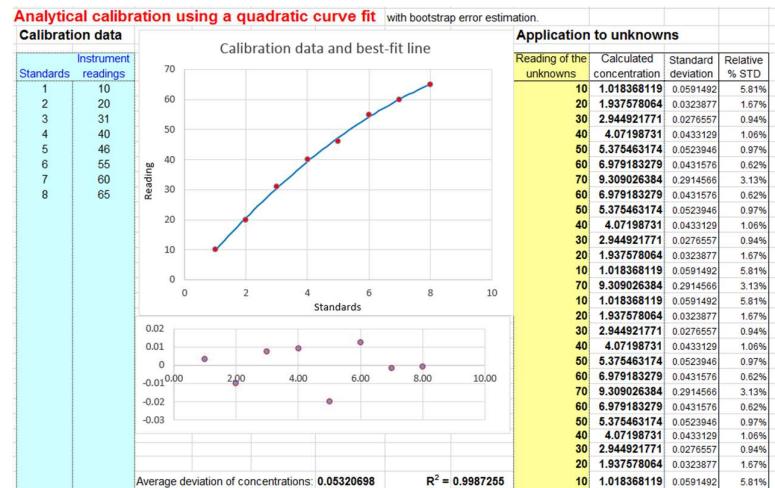


Questo metodo può essere utile se nessuno dei metodi dei minimi quadrati può approssimare adeguatamente l'intero intervallo di calibrazione (ad esempio, se contiene due segmenti lineari con pendenze diverse). Funziona abbastanza bene se gli standard sono sufficientemente ravvicinati in modo che la risposta 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 $A = aC^2 + bC + c$. 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 $C = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, 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).

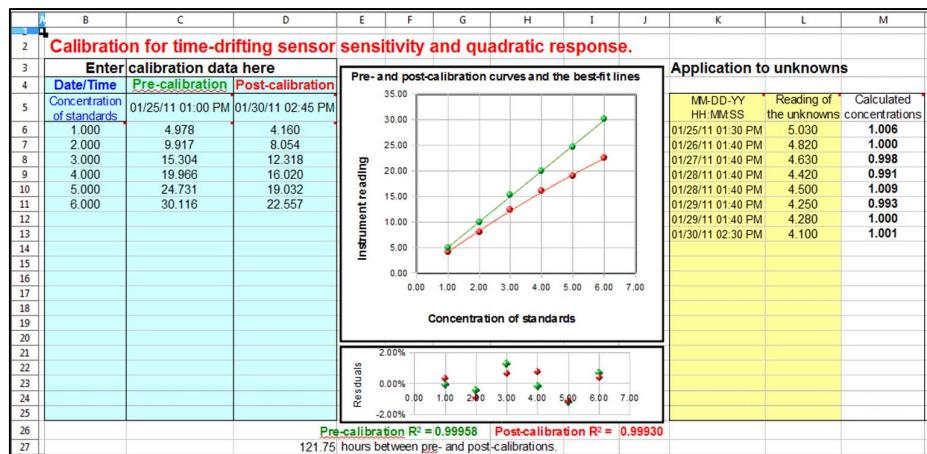


(ad esempio le [relazioni della Legge di Potenza](#)) una relazione non lineare tra segnale e concentrazione può essere completamente linearizzata da una trasformazione log-log. (Alcuni laboratori ufficiali regolamentati dal governo operano secondo regole che [non consentono l'uso di approssimazioni 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 calibrazione 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'approssima-

zione 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. 435)

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

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 160, "[Affidabilità dei risultati dell'approssimazione della curva](#)". Lo spreadsheet della calibrazione lineare (scaricabile in formato [Excel](#) o

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 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

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. Domande: 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

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)

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-distro 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 ordinare 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 esponenzialmente allargata](#) (forma variabile)

[Triangle-broadened Gaussian](#) (forma variabile)

[bifurcated Gaussian](#) (forma variabile)

[Flattened Gaussian](#) (forma variabile)

[ShapeDemo](#) mostra 16 forme di picco elementari graficamente, mostrando i picchi di forma variabile come linee multiple. (Grafico a pagina 401)

[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, $100 \cdot * (b-a) ./ a$, 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, pendenza1 e pendenza2 e le semi-larghezze del bordo iniziale e finale a metà del massimo, hwhm1 e hwhm2, rispettivamente. Se x_0 viene omesso, determina la metà della larghezza del picco più grande. Esempio: `xo=500; width=100; x=1:1000; y=exp(-1.*((x-xo)/(0.60056120439323.*width)).^2); halfwidth(x, y, xo)`. La funzione analoga `[twidht, 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.

[IQRrange.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 funziona meglio per calcolare la dispersione (diffusione) di un set di dati contenente valori anomali. Essenzialmente è la deviazione standard con i valori anomali rimossi. La sintassi è $b = IQRrange(a)$.

[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 ver-

plotit" al prompt di Matlab/Octave per degli esempi.

[plotxrange](#) estraе 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 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 23](#). 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 $\sqrt{(\text{std}(m1-m2)^2)/2}$.

[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 23.

[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 nel segnale. Questo è un raro esempio in cui l'aggiunta di rumore è vantaggiosa. Vedere pagina 293.

[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

[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 303.

[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 323.

[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 (linea 15) o instabilità del segnale produce un errore molto maggiore. Usato a pagina 323.

[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 323.

[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

[fastsimooth](#), una versatile funzione per un rapido smoothing di dati. La sintassi è **SmoothY=fastsimooth(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-](#)

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 70](#).

[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 82. 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 370) 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 348).

stato 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 98 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 a 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 *Gaussianiana*, il cui spettro di potenza è una funzione Gaussianiana, ma che è la stessa ai due estremi della durata dell'impulso ([grafica animata](#)).

[isignal.m](#) o [isignaloctave.m](#), (pag. 370) 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 segnale 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 inse-

SegExpDeconv(x,y,tc) è una versione segmentata di deconvexp.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 370) 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 119.

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. 371). [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

[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 370) 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 ([cliccare 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 320.

che seguono una [power law relationship](#) o che coprono un intervallo numerico molto ampio.

[RSquared.m](#) Calcola la R² (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 (R²) 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 R² 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 160.

[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 160.

[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 166.

[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 166.

[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 [309](#).

[gaussfit.m](#) [**H**eight, **P**osition, **W**idth]=**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](#) [**H**eight, **P**osition, **W**idth]=**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.

posizione e dell'altezza del picco.

[autofindpeaks.m](#) (e [autofindpeaksplot.m](#)) sono simili a findpeaksSG.m tranne per il fatto si può *tralasciare 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 avallamento (magenta) e le linee tangenti (ciano). Lo script [testautopeaks.m](#) esegue tutti gli esempi nel file help `autopeaks`, con una pausa di 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 370) ha una 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

`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 242) 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 394) 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 225. [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 225. 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

te 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 225.

Quale usare: iPeak o Peakfit? Provare queste funzioni demo di Matlab che confrontano `iPeak.m` (pag. 242) con `peakfit.m` (pag. 225) 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 289.

[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 [RealTimeSmoo-](#)

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 165. 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>"). **NEW** 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 225) 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 375. 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 375, per una breve descrizione delle nuove funzionalità in ciascuna versione

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 espanso, 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 196. 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 196.

[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 225), 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 392.

[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.

[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.

[peakfitVfindpeaks.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 348).

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, menu 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 *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 fun-

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. 23). 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 23. 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 40). 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 335). Permette di cambiare la larghezza dello smoothing semplicemente cambiando il valore di una singola cella. Vedere pagina 50 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

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. 127). [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'effetto 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.xlsxm](#) (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. 180). [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 335) 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 166). 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. 225). 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 456). Lo spreadsheet [PeakDetection.xls](#) implementa un metodo di rilevamento dei picchi con l'attraversamento dello zero della derivata più selettivo descritto a pagina 227. 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

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

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 "incampandoci" 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 matematica 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 universitarie: 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, rilego 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 somma-

lab/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](#), 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 pro-

Riferimenti

1. Douglas A. Skoog, *Principles of Instrumental Analysis*, 3^a Edizione, Saunders, Philadelphia, 1984. Pages 73-76.
2. Gary D. Christian and James E. O'Reilly, *Instrumental Analysis*, Second Edition, Allyn and Bacon, Boston, 1986. Pages 846-851.
3. Howard V. Malmstadt, Christie G. Enke, and Gary Horlick, *Electronic Measurements for Scientists*, W. A. Benjamin, Menlo Park, 1974. Pages 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. Efstathiou, 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/syb/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>

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.

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 \(springer.com\)](#)

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 NOx 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.engin.swarthmore.edu/e90/2008/reports/Thomas%20Kelleher.pdf>](http://www.engin.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, Juliusz Lech Kulikowski, Springer (Sep 30, **2009**), page 191.
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.

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-Tokhy, M.S. Mahmoud, I.I. ; Konber, H.A. Informatics and Systems (INFOS), 2012 8th International Conference on 14-16, May **2012**
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 Wiegaebea, 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 Schirrmeyer, 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.

- 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. Žáčik, 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. "Corticolumbar 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. Mccombe, 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 Emilius 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.

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 Espíñola, 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 E Darvin, Laser Phys. 24 10560, 2014. (<http://iopscience.iop.org/1555-6611/24/10/105601>)

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.

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 -octylglucoside and phosphate salt solutions on phase behavior of monoolein mesophases, [PDF] from researchgate.net
181. Mahmoud, Imbaby 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.

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. Ilewickz, 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

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 and amino acid sample dimensionality in two dimensional nLC \times μ FFE 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

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.

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 ^{36}S 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.

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. Oeltzschnner, 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, and Ю. В. Першин. "Вольт–амперные характеристики коммерческих сегнетоэлектрических конденсаторов: отклонения от модели Преизаха." <<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 N 2 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 of Nuclear Materials* 519:157-165, March 2019, DOI: 10.1016/j.jnucmat.2019.03.049
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/ijspes.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. Demetrou, 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 phosphomonesters 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](#).

"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 Cr³⁺ and Al³⁺ 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, Bezhan. "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

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 ^{226}Ra 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.

486. Ma, Liya, and Peter Schegner. "State duration based event detection for domestic power disaggregation." *2019 IEEE Milan PowerTech*. IEEE, **2019**.
497. 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.
498. 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.
499. 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.
500. 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.
501. 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.
502. 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.
503. 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).
504. 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.

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.

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." *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 **(2021)**: e0256245.
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. Ghadimloozaheh, 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.

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." *TOSforum*. 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.

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.

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>.
[\(https://www.sciencedirect.com/science/article/pii/S266616672300165X\)](https://www.sciencedirect.com/science/article/pii/S266616672300165X)

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
724. Feng, Vincent. "Spectroscopy of Chamber Plasma and Plume of a 2.4-GHz Microwave Electrothermal Thruster." (2023).
725. Zhang, Le, et al. "Absolute thermometry of human brown adipose tissue by magnetic resonance with laser polarized ¹²⁹Xe." *Communications Medicine* 3.1 (2023): 147.

745. Nieto Benito, Guillermo. "Predicción de series temporales mediante técnicas de aprendizaje automático. Automatización del proceso." (2023). [PDF link](#).
746. 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).
747. Mikheenkova, Anastasiia, et al. "Ageing of High Energy Density Automotive Li-ion Batteries: The Effect of Temperature and State-of-Charge." (2023).
748. Chaiamarit, Tai, et al. "Mutant Prion Protein Endoggresomes are Hubs for Local Axonal Organelle-Cytoskeletal Remodeling." *bioRxiv* (2023): 2023-03.