itefi
Instituto de Tecnologías Físicas y de la Información

Ultrasounds for Biological Applications and Materials Science

CSIC
Consejo Superior de Investigaciones Científicas

THE INVERSE PROBLEM

… and the use of Python-Implemented Gradient Descent Techniques.

Tomás Gómez Álvarez-Arenas

t.gomez@csic.es www.us-biomat.com

OUTLINE

1. The Inverse Problem (IP):

    Definition & examples

2. IP applications.

3. IP main elements.

4. IP optimization algorithms.

    Gradient Descent algorithms (GDA) for IP.

5. A Python implentation of GDA for IP.

    Examples.

Data

KNOWLEDGE

Regression

Inverse Problem

Machine Learning

Bayesian Inference

Statistical Analisys

1. The Inverse Problem: [IP]

Definition & Examples

Direct Problem.

Direct Problem

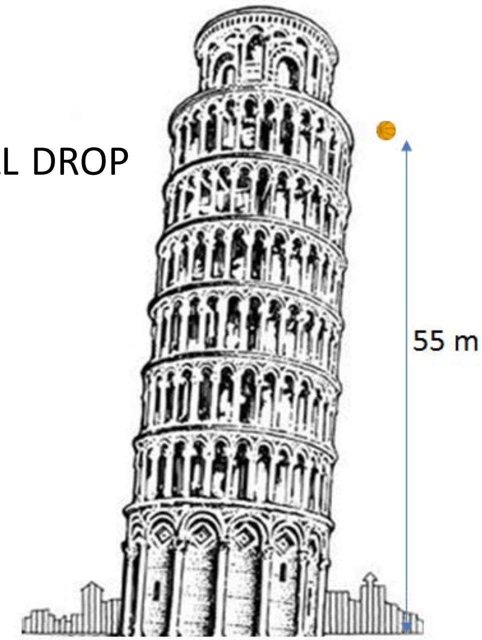
Given:

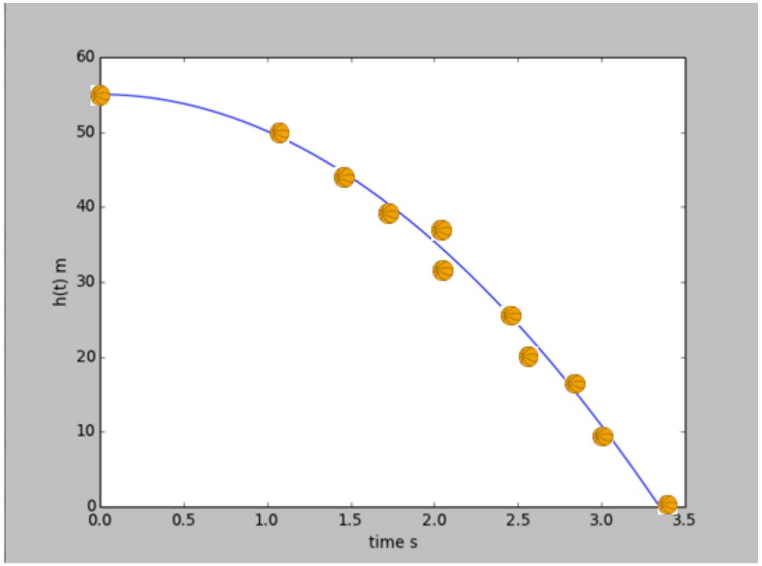
Model parameters: $h_0 = 55$ m, $g = 9.81$ m/s^2 & Model (law): $h(t) = h_0 - 1/2\, g\, t^2$

Predict:

h at any t

Ball drop





IP: Definition.

Inverse Problem

Extract:

Model parameters:
h_0, g

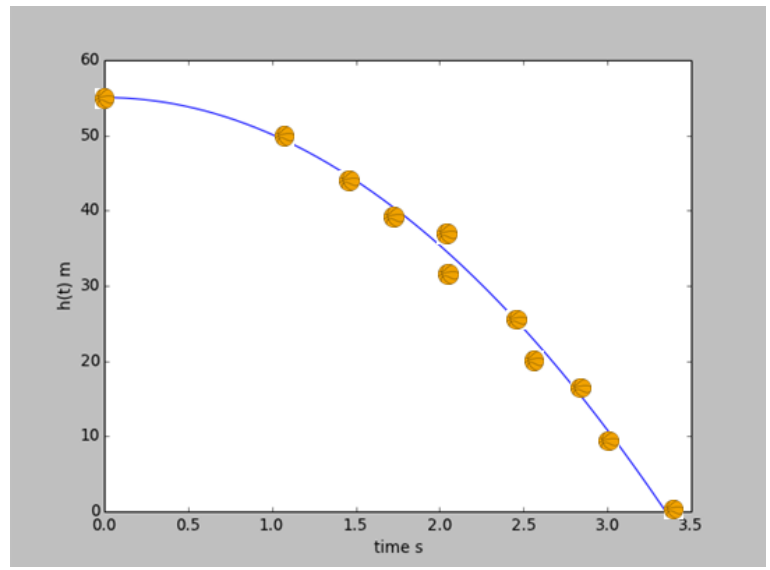
Given

Model (law):
$h(t) = h_0 - 1/2\ g\ t^2$

&

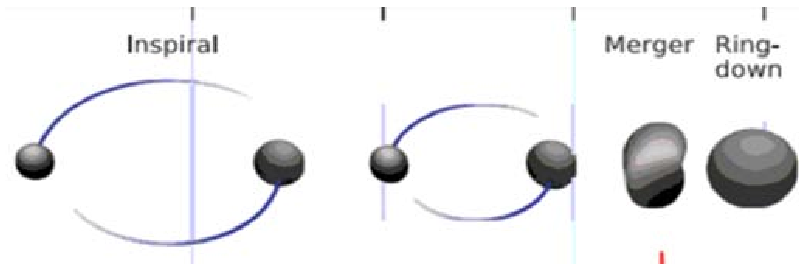
h at some t

Ball drop

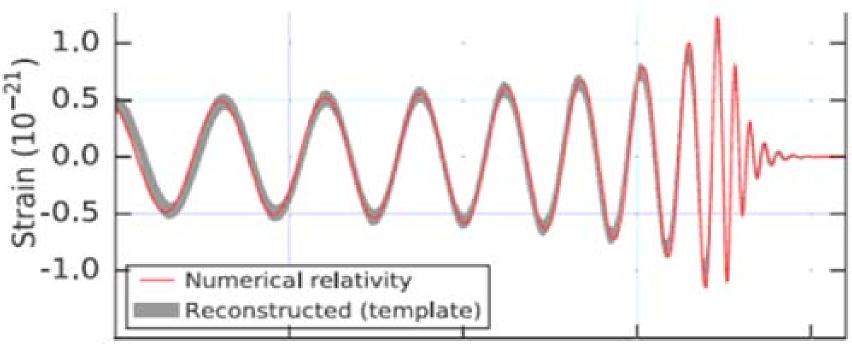


IP Example:
Gravitational waves

DIRECT PROBLEM

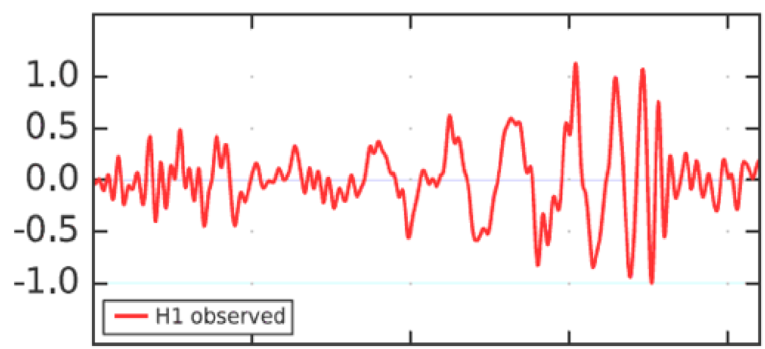


$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} = \frac{c^3}{G}\left[\frac{5}{96}\pi^{-8/3} f^{-11/3} \dot{f}\right]^{3/5},$$



INVERSE PROBLEM

Parameters of the mass chirp



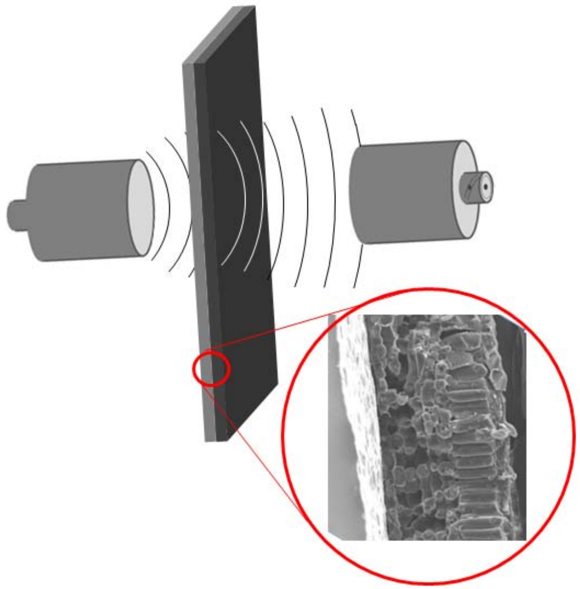
LIGO project

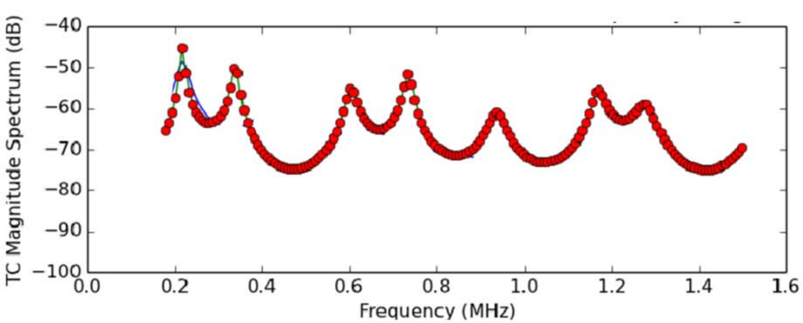
https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html

IP Example:
Echographic Signals

t.gomez@csic.es

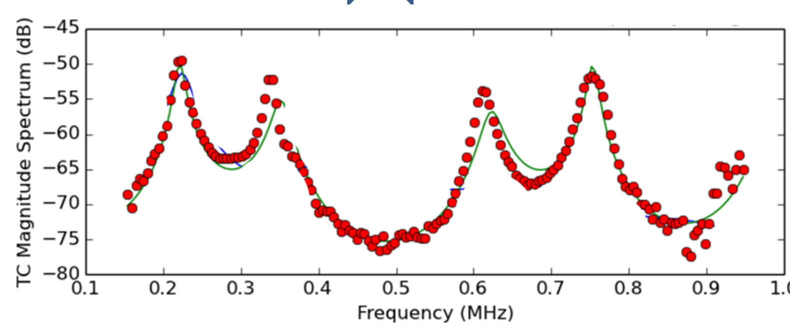
DIRECT PROBLEM





INVERSE PROBLEM

Thicknesses (x2)
Densities (x2)
Elastic constant (x2)
Mechanical damping (x2)
Freq. dep. MD (x2)



t.gomez@csic.es www.us-biomat.com

2. IP Applications

1. Obtention of model parameters .

2. Model confirmation.

3. Model selection.
   Ockham's razor

3. IP Main Elements

INVERSE PROBLEM

MODEL (*f*):

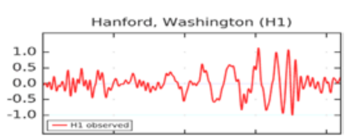
$y^*_i = f(\beta_j, x_i)$

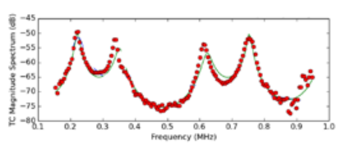
ERROR FUNCTION

ε

DATA

x_i , y_i





OPTIMIZATION ALGORITHM

$\partial\varepsilon/\partial\beta = 0$

ε_{min}

OUTPUT

PREDICT MODEL OUTPUT

MODEL PARAMETERS:

β_j.

IP *vs* ML (Machine Learning)

Machine Learning

Model (f):

$y^*_i = f(\beta_j, x_i)$

Error Function

ε

Data

x_i , y_i

Optimization algorithm

$\partial\varepsilon/\partial\beta = 0$

ε_{min}

Output

Model Parameters:

β_j .

Guess Model Output For New Data

t.gomez@csic.es www.us-biomat.com

Inverse problem

Model (f):
$y^*_i = f(\beta_j, x_i)$

Constrains

Error Function
ε

Initial Guess:
$\beta_j^{\,0}$

Data
x_i , y_i

Optimization algorithm

Output

$\partial\varepsilon/\partial\beta = 0$

ε_{min}

Model Parameters:
β_j.

t.gomez@csic.es www.us-biomat.com

Inverse problem

Well-posed
vs
Ill-posed problems

Hadamard principles (Existence, uniqueness, stability)

"One can find the desired result to any degree of specified precision - provided one is prepared to work hard enough"*

*Working hard enough: making enough measurements with enough accuracy and then doing enough computations

4. IP Optimization Algorithms

t.gomez@csic.es www.us-biomat.com

Optimization Algorithms (just a few of them…)

NA: Newton algorithms.
- GNA: Gauss-Newton algorithm.
- BFGS: Broyden–Fletcher–Goldfarb–Shanno algorithm (1970).

NMA: Nelder-Mead algorithm (1965).
PA: Powell's algorithm (1964).
LMA: Levenberg–Marquardt algorithm (1944-1963).
GDA: Gradient Descent algorithms.

PYTHON IMPLEMENTATION OF OPTIMIZATION ALGORITHMS

(just a few of them…)

scipy.optimize
http://docs.scipy.org/doc/scipy/reference/optimize.html
Optimization and root finding

Sherpa
http://cxc.cfa.harvard.edu/contrib/sherpa47b/
Levenberg-Marquardt, Nelder-Mead Simplex or Monte Carlo/Differential Evolution.

PyQt-Fit
https://anaconda.org/pypi/pyqt-fit
Regression toolbox and GUI

LMFIT
Non-Linear Least-Square Minimization and Curve-Fitting for Python
https://lmfit.github.io/lmfit-py/
High-level interface for scipy.optimize, extends LMA.

Python Implementation of Optimization Algorithms

(just a few of them…)

scipy.optimize
http://docs.scipy.org/doc/scipy/reference/optimize.html
Optimization and root finding

The minimize function supports the following methods:

- minimize(method='Nelder-Mead')
- minimize(method='Powell')
- minimize(method='CG')
- minimize(method='BFGS')
- minimize(method='Newton-CG')
- minimize(method='L-BFGS-B')
- minimize(method='TNC')
- minimize(method='COBYLA')
- minimize(method='SLSQP')
- minimize(method='dogleg')
- minimize(method='trust-ncg')

GRADIENT DESCENT ALGORITHMS FOR IPS.

t.gomez@csic.es www.us-biomat.com

Python Implementation of Gradient Descent Algorithms

https://github.com/mattnedrich/GradientDescentExample

http://tillbergmann.com/blog/articles/python-gradient-descent.html

http://scikit-learn.org/stable/modules/sgd.html

http://www.r-bloggers.com/r-and-python-gradient-descent/

Key Elements of GDA For IP.

IP:

Data set: y_i; x_i

Model: $y = f([\beta_j],x)$

Error function

Optimization algorithm (GDA)

GDA:

Initial guess: β_j

Step size (constant / variable)

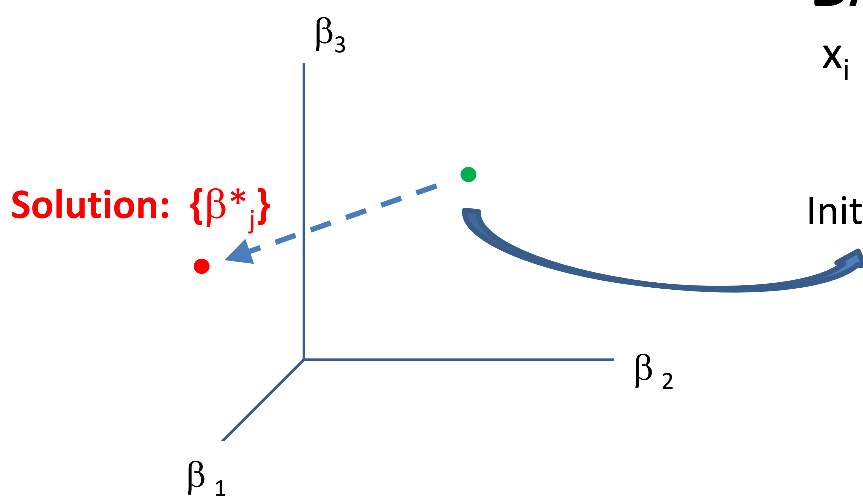
Constrains: $C_k(\beta^*) = 0$

The Gradient Descent approach for Inverse Problems.

1. Set initial guess

DATA x_i , y_i

MODEL (f) $y^*_i = f(\beta^0_j, x_i)$



Initial guess: β^0_1, β^0_2, β^0_3

$f(\{\beta^0_j, x_i\})$

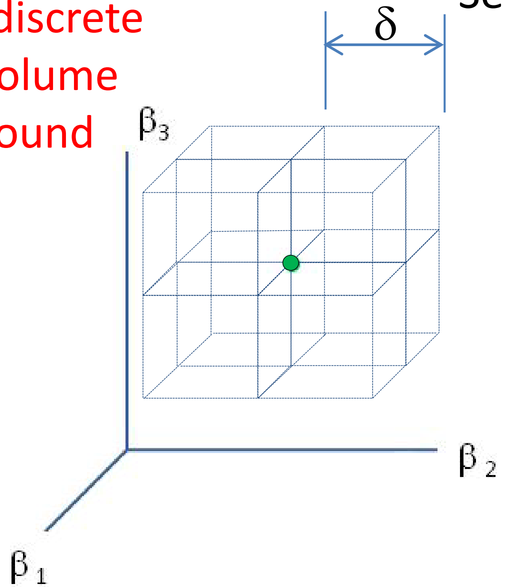
$||y_i - f(\{\beta^0_j, x_i\})||$: $\Delta\{\beta^0_j\}$: error is not minimum

t.gomez@csic.es www.us-biomat.com

The Gradient Descent approach for Inverse Problems.

Set a discretization step: δ

2. Set a discrete search volume (DSV) around IG



DATA

x_i , y_i

MODEL (*f*)

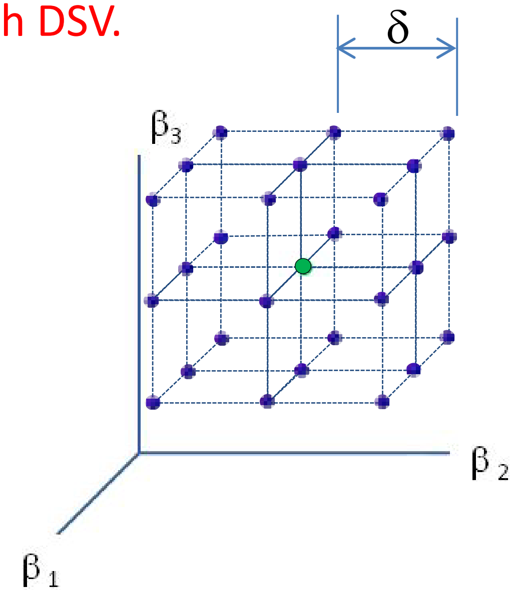
$y^*_i = f(\beta_j \pm \delta, x_i)$

$f(\{\beta^0_j \pm \delta, x_i\})$

$||y_i - f(\{\beta^0_j \pm \delta, x_i\})||$: $\Delta\{\beta^0_i \pm \delta\}$: error

The Gradient Descent approach for Inverse Problems.

3. Search DSV.



DATA

x_i , y_i

MODEL (*f*)

$y^*_i = f(\beta_j \pm \delta, x_i)$

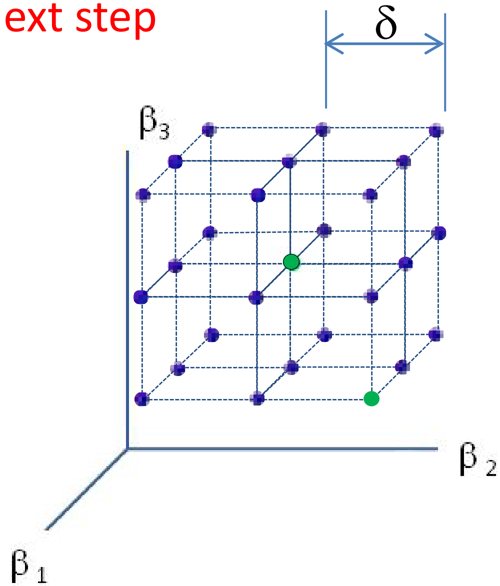
Dimension	Volume
N = 3:	27
N = 4:	81
N = 5:	243
N = 6:	729
N = 7:	2187
N = 8:	6561

$$f(\{\beta^0_j \pm \delta, x_i\})$$

$$||y_i - f(\{\beta^0_j \pm \delta, x_i\})|| : \quad \Delta\{\beta^0_i \pm \delta\}: \text{ error}$$

The Gradient Descent approach for Inverse Problems.

4. Find next step



Systematic approach:

Search all points in V

Take $\{\beta_i\}$ / $\Delta\{\beta_i\}_{min}$

Stochastic approach:

Random search of V

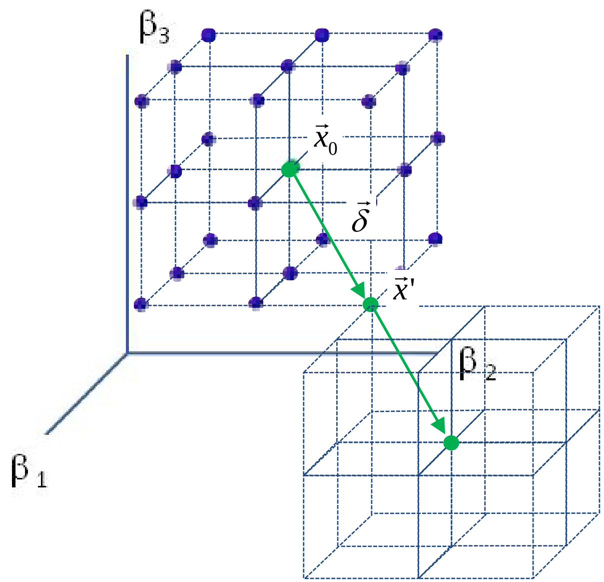
Stop searching when:

$\Delta\{\beta_i\} < \Delta\{\beta_0\}$

	Systematic	Stochastic
Computational cost:	Cte. 3^N	Var. $1 \rightarrow 3^N$
Incremental improve:	Optimum	< Optimum
Avoid local minima:	??	May be
Same solution:	Yes	NO

The Gradient Descent approach for Inverse Problems.

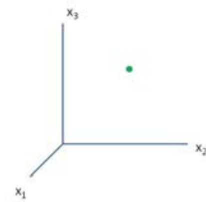
5. Take a step.

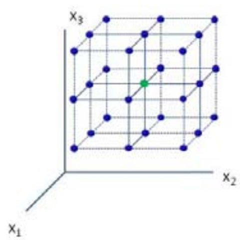


Take a step: $2\vec{\delta} = 2\left(\vec{\beta}' - \vec{\beta}_0\right)$

$$\vec{\beta}_0 = \vec{\beta}_0 + 2\vec{\delta}$$

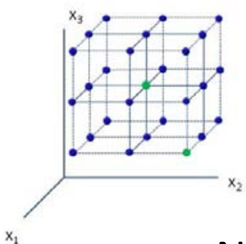
Initial guess $\{\beta_0\}$.





$f(\{\beta_0\})$ and $\Delta\{\beta_0\}$

Volume around V $\{\beta_0\}$: $f(\{\beta\})$ and $\Delta\{\beta\}$



Does any point reduce error?
[Systematic /Stochastic]

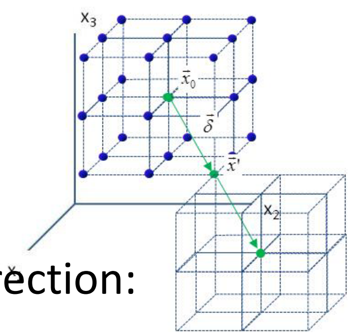
NO

$\Delta\{\beta_0\} < EL$

NO → No solution

YES → Solution

YES



Take one step in this direction:
$\{\beta'\} = \{\beta_0\} + 2\delta$

New guess $\{\beta_0\} = \{\beta'\}$

5. A Python Implementation of GDA for IPs.

… a simple code example built from scratch.

Main function:

This function performs the Gradient Descent search by calling Grad_ErrorML at each step

```
In [105]: def Walk_downGradientML(exp_dat, guess_0, params, x, delta, NoStepsOK,
                              target_func,Error_Func, MaxSteps = 500, Prec = 0.999,forwardCheck = 1):
              """ This function takes one step in the fitting procedure douwn the gradient in the error hiperspace.
              exp_dat: numpy_array, vector with the value of the function at xi
              params: tuple with the position in the list of the guess_0 elements to be used for the fitting
              guess_0: list, list with the initial guess parameters
              x: numpy_array, vector with the value of the independent variable (xi)
              delta: float, step size for the GD
              NoStepsOK: int, number of steps that improve the error in the GD before setting the direction of the nest step
                  NoStepsOK = 1 , the stochastic approach is used
                  NoStepsOK = 3 ** len(params) - 1, the conventional GD approach is used
              Prec: float, required minimum improvement in the error (Prec <= 1)
              target_Func: function model, user defined,
              Error_Func: function, user defined, function that calculates the error exp_data and
                 calculated value of the target function for the xi values and the actual value of the function parameters"""
```

Additional functions:

This function seach around the present value of guess parameters, and sets the next step: new guess parameters and error value at this new position

```
In [104]: def Grad_ErrorML(exp_dat, guess_0, params, x, delta, NoStepsOK, Prec,target_func,Error_Func):
    """ Computes the local gradient of the error function only for the parameters specified in LayerParam;
    exp_dat: numpy_array, vector with the value of the function at xi
    params: tuple with the position in the list of the guess_0 elements to be used for the fitting
    guess_0: list, list with the initial guess parameters
    x: numpy_array, vector with the value of the independent variable (xi)
    delta: float, step size for the GD
    NoStepsOK: int, number of steps that improve the error in the GD before setting the direction of the nest step
        NoStepsOK = 1 , the stochastic approach is used
        NoStepsOK = 3 ** len(params) - 1, the conventional GD approach is used
```

This function allows to deal with a n-parameters problem, where it is not necesary to know n beforehand:

```
In [102]: def trinarize(decimal):
    trinary = ''
    while decimal // 3 != 0:
        trinary = str(decimal % 3) + trinary
        decimal = decimal // 3
    return str(decimal) + trinary
```


t.gomez@csic.es www.us-biomat.com

Finally, we define a fitting object class with some features and functions to
More easily handle and display the results

```
In [98]: class Fitting(object):
    """ This defines a Fitting object, contains:
    best fit parameters [0],
    number of steps in GD to best fit [1]
    final error[2],
    path to fit [3],
    exec time of fitting [4]
    initial guess parameters [5]
    Error_log [6]
    delta_log [7]

    It also contains some functions to visualize the results:

    plot_error_log;    plot_delta_log;   plot_path;   plot_path2;   plot_pathN;   plot_cloud  plot_Bestcloud """
```

6. Examples.

Basic examples
Signals in time domain
Complex high dimensional problems

6. Examples (I)

Basic Examples.

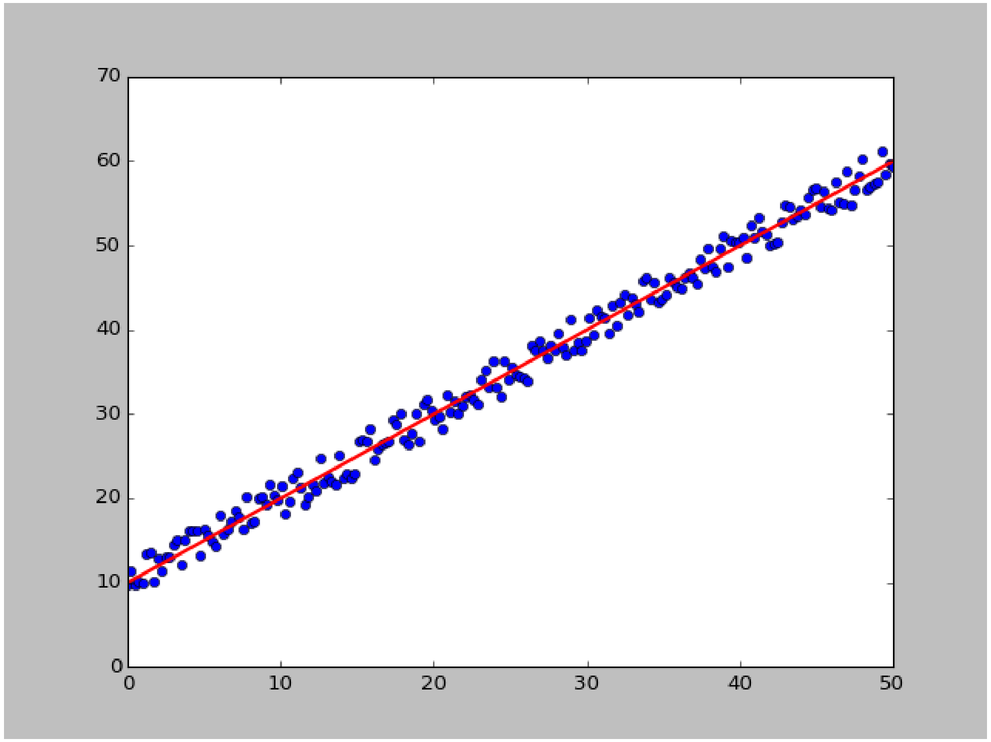
Linear regression
Polynomial fitting

t.gomez@csic.es www.us-biomat.com

Example 1: Linear regression
Run #1

Data:
$y^* = a\,x + b + \text{noise}$
$a,b = (1.0, 10.0)$

Model:
$y = a\,x + b$

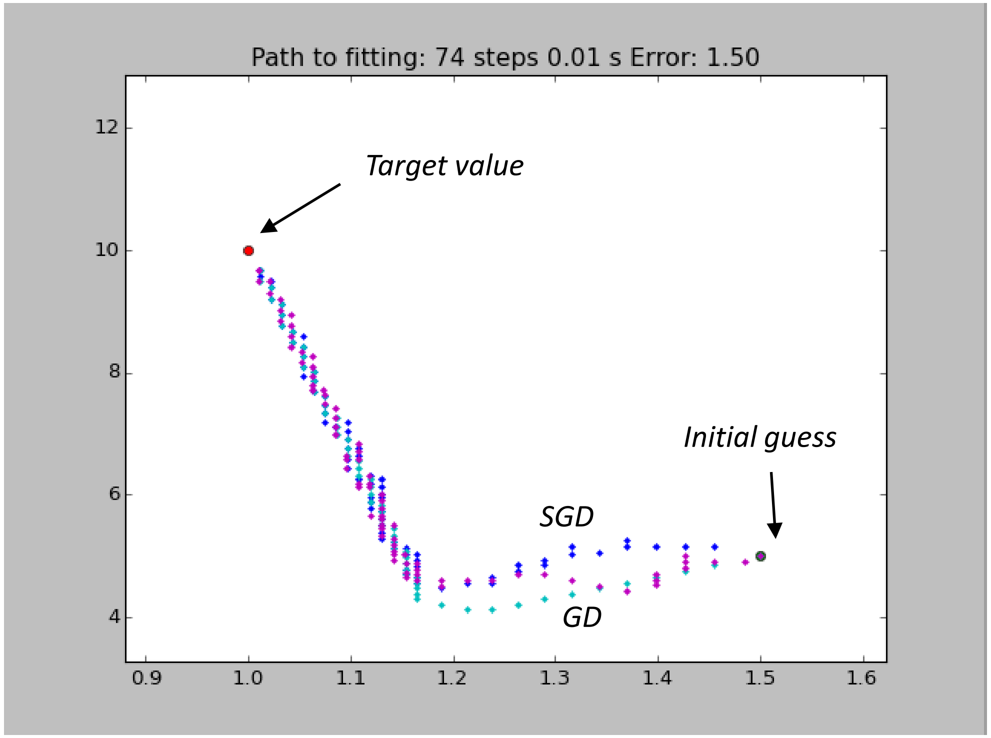


Stochastic Gradient Descent (SGD)
Discretization step (DS): $\delta = 0.01$

Example 1: Linear regression
Run #1

Path to fitting…
(in the parameters space)

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)



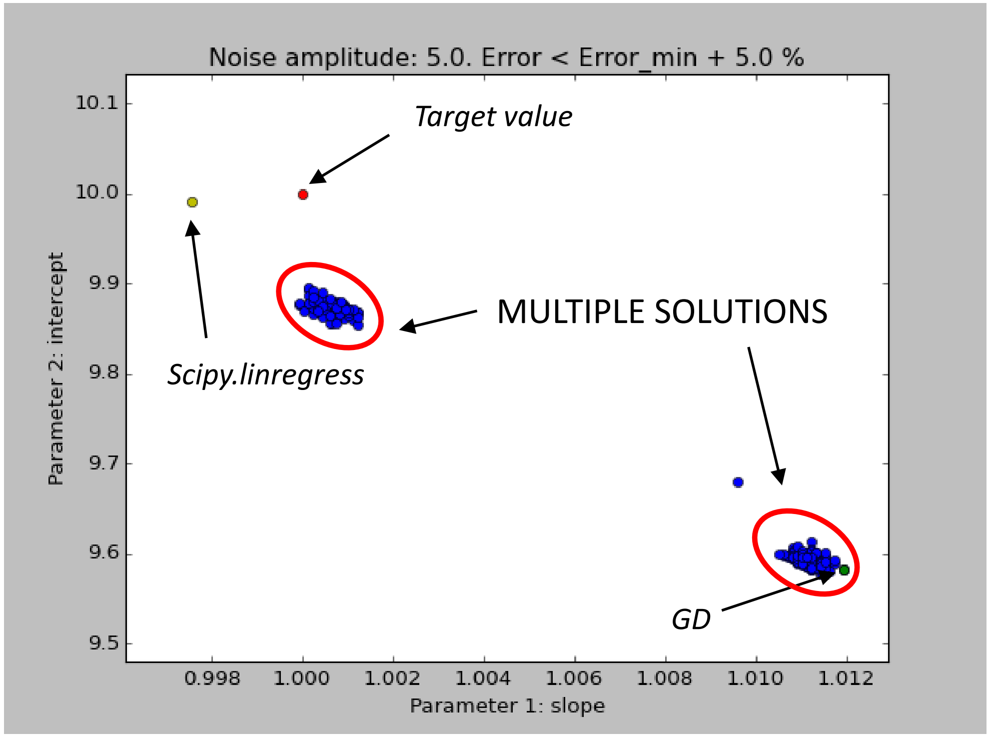
GG.plot_pathN((GG2,GG3),(0,1),True,pa,pb)

Example 1: Linear regression
Run #1

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)

Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 5%



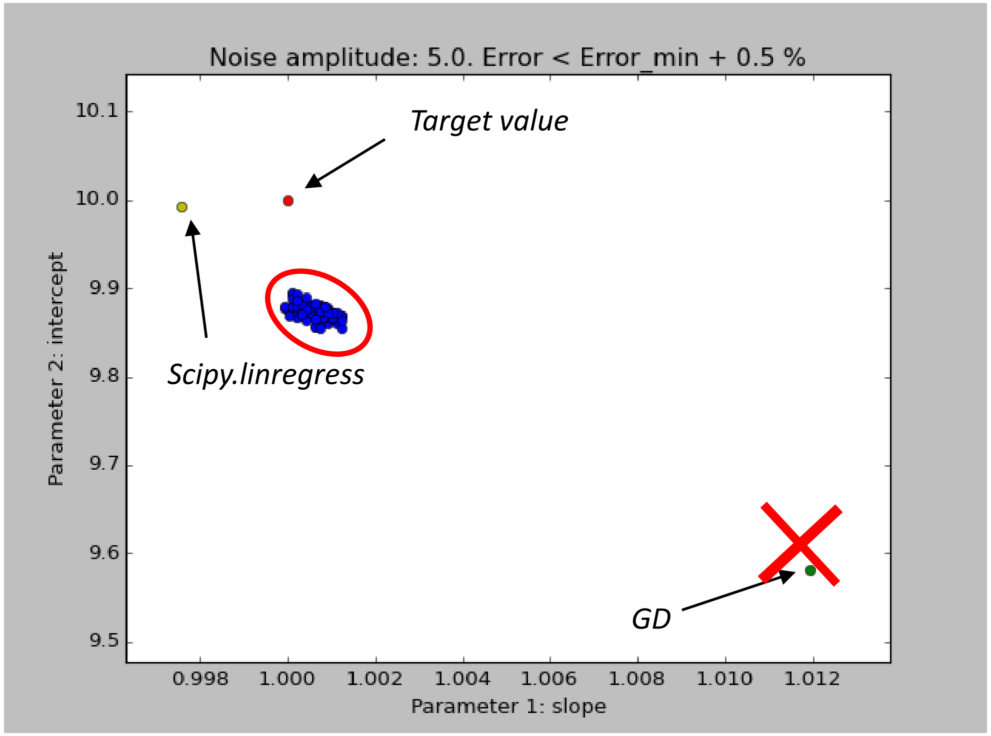
GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #1

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)

Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 0.5%



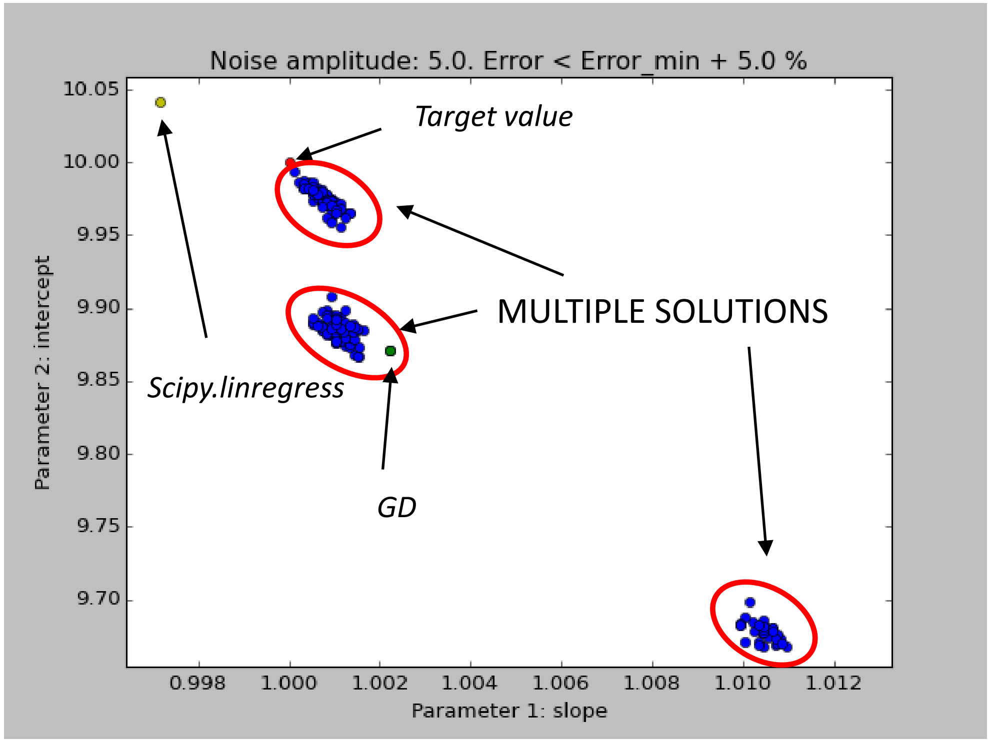
GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #2

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)

Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 5%



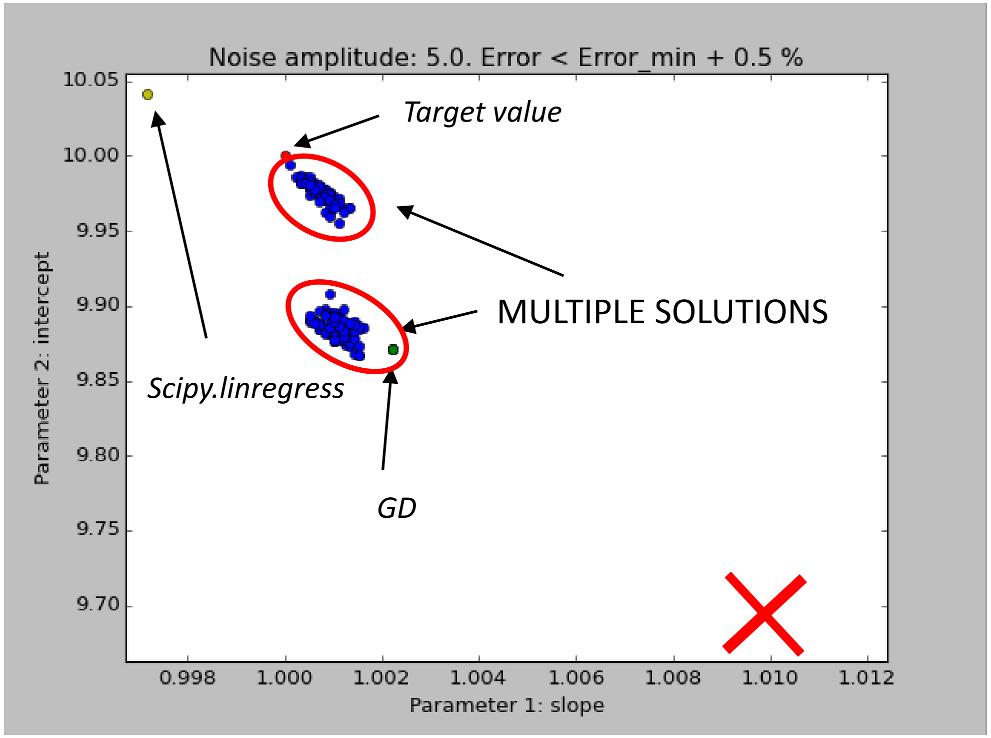
GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #2

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)

Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 5%

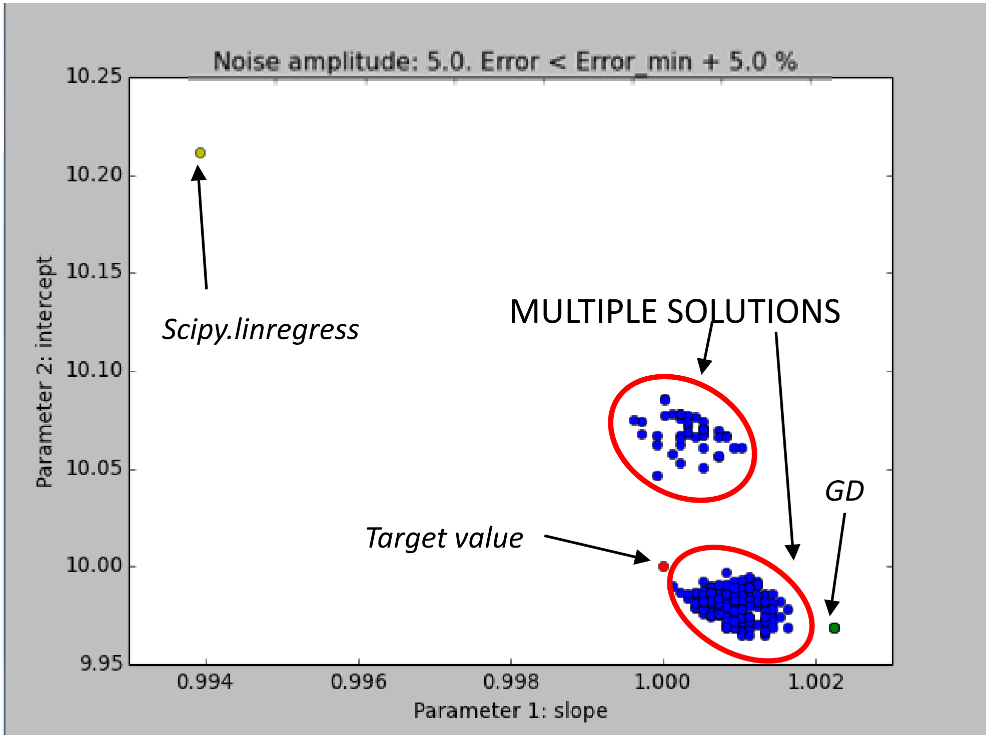


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #3

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)
Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 0.5%

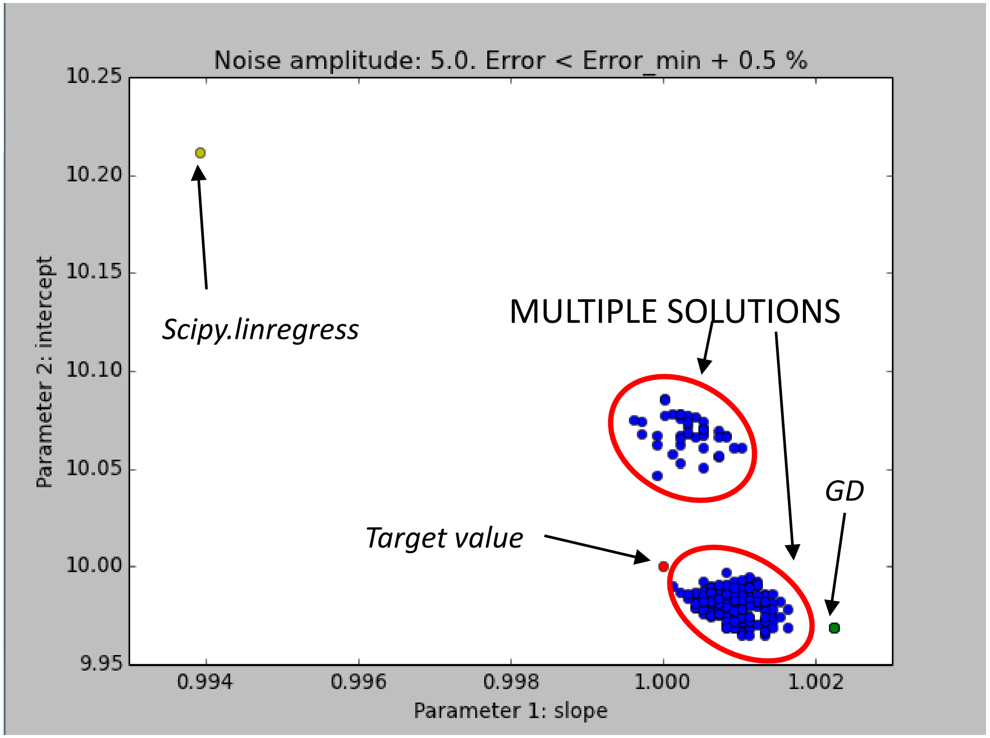


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #3

Target value: (1.0, 10.0)
Initial guess: (1.5, 5.0)
Run *GD*
Run *scipy.linregress*
Run *SGD* 200 times…

Only take Error < Error_min + 5%

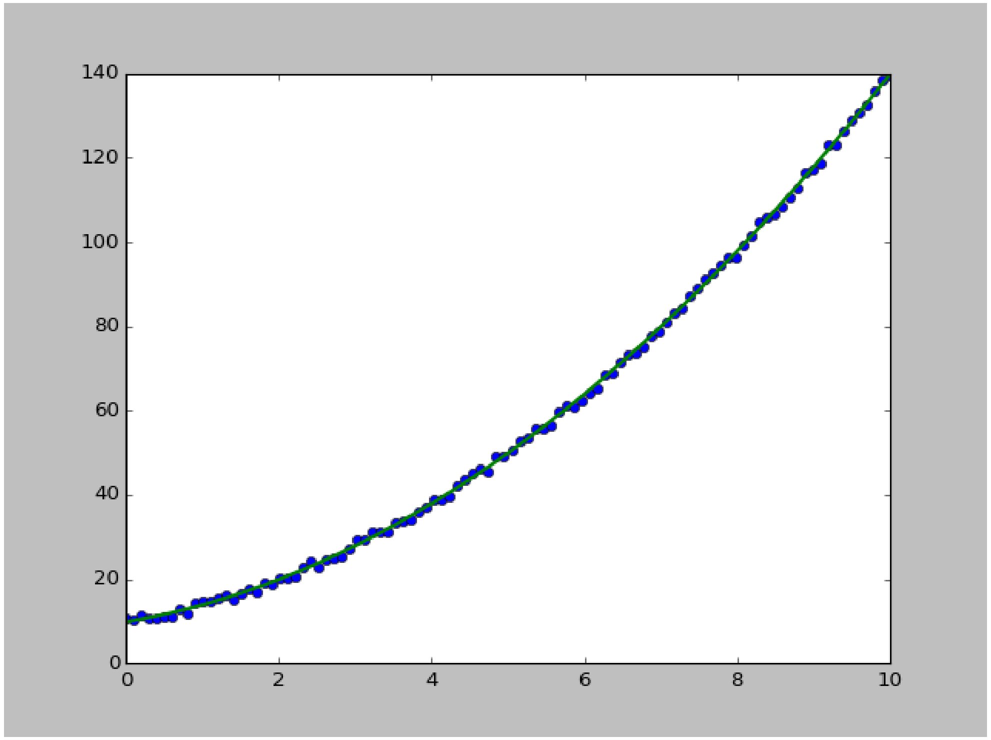


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 2: Polynomial order 2

Data:
$y^* = a\,x^2 + bx + c + \text{noise}$

Model:
$y = a\,x^2 + bx + c$

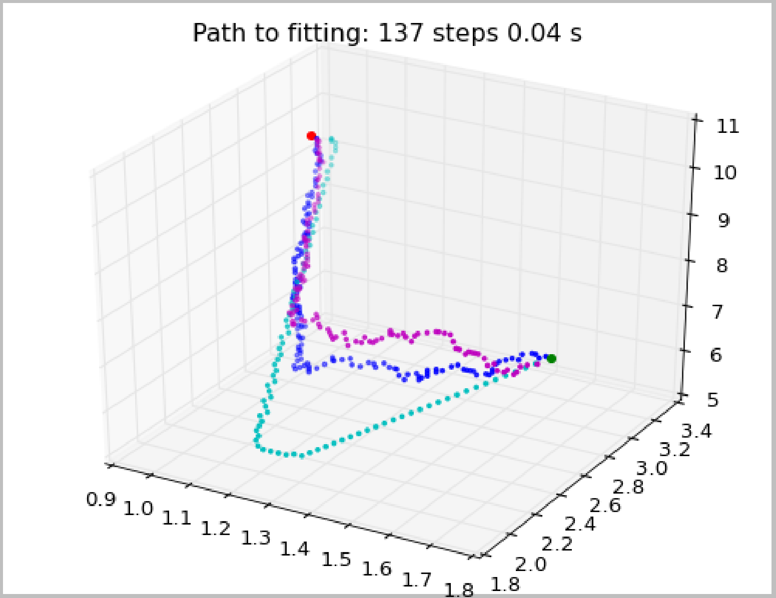


Stochastic Gradient Descent (SGD)
$\delta = 0.005$

Example 2: Polynomial order 2

Path to fitting…
(in the parameters space)

Target value: (1.0, 3.0 ,10.0)
Initial guess: (1.7, 2.7, 7.0)



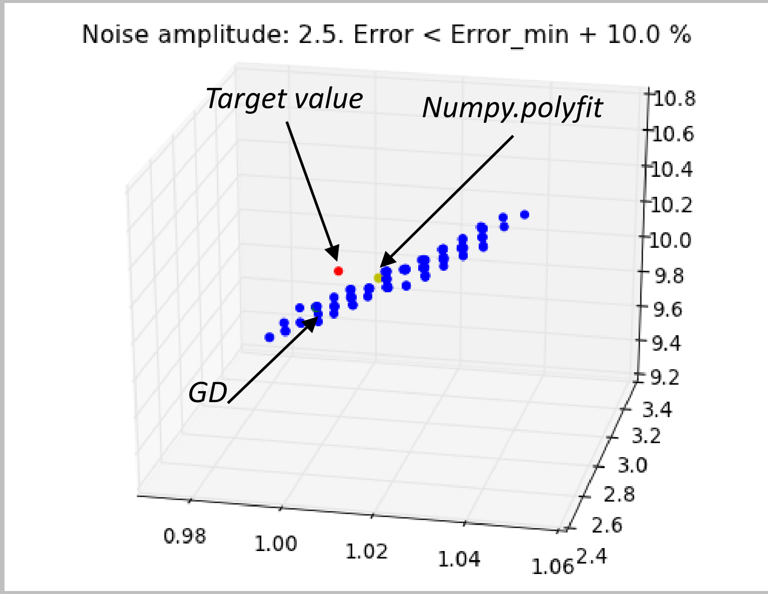
Stochastic Gradient Descent (SGD)
$\delta = 0.005$

GG.plot_path2(GG2,(0,1),True,pa,pb)

Example 2: Polynomial order 2

Target value: (1.0, 3.0 10.0)
Initial guess: (1.7, 2.7, 7.0)
Run *GD*
Run *numpy.polyfit*
Run *SGD* 200 times…

Only take Error < Error_min + 10%

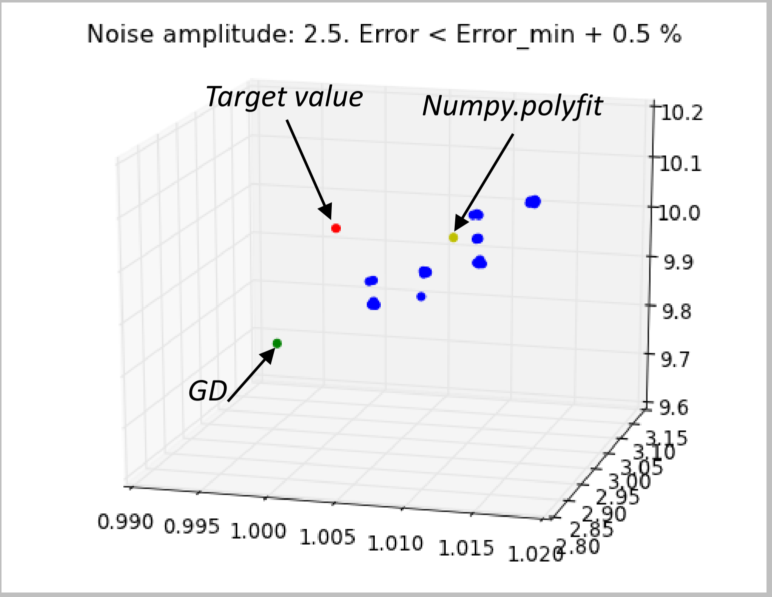


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 2: Polynomial order 2

Target value: (1.0, 3.0 10.0)
Initial guess: (1.7, 2.7, 7.0)
Run *GD*
Run *numpy.polyfit*
Run *SGD* 200 times…

Only take Error < Error_min + 0.5%



GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

6. Examples (II)

Signals in the time domain.

Damped oscillator
Linear chirp
Linear-gained chirp
Mass chirp (gravitational waves)

t.gomez@csic.es www.us-biomat.com

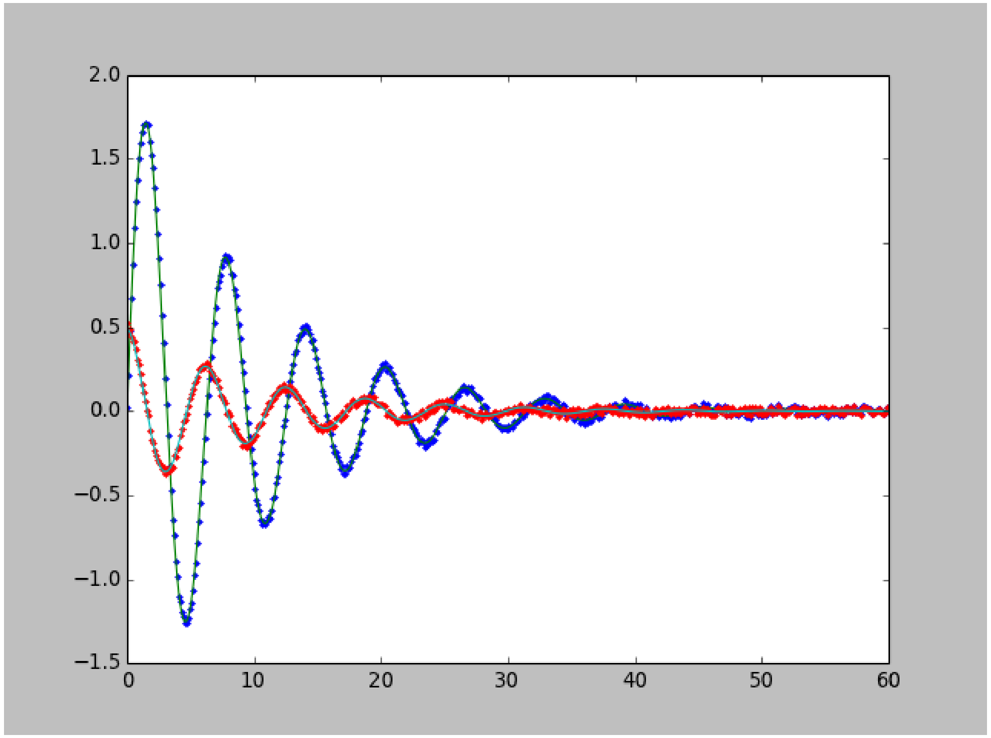
Example 3: Damped oscillator

Data:

y* = (a sinx + j b cosx) exp(-cx)+ noise

Model:

y = (a sinx + j b cosx) exp(-cx)



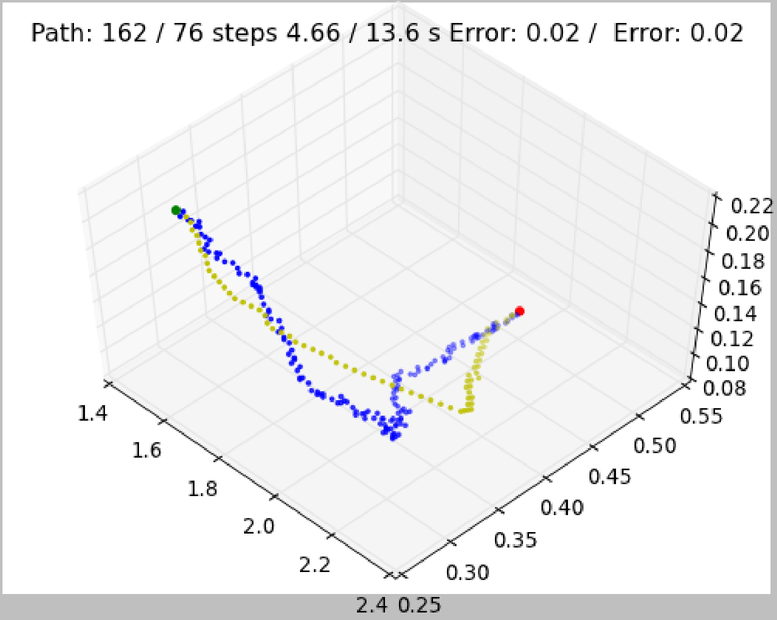
Stochastic Gradient Descent (SGD)

δ = 0.005

Example 3: Damped oscillator

Target value: (2.0, 0.5, 0.1)
Initial guess: (1.5, 0.3, 0.2)

Run *GD*
Run *SGD* 50 times…

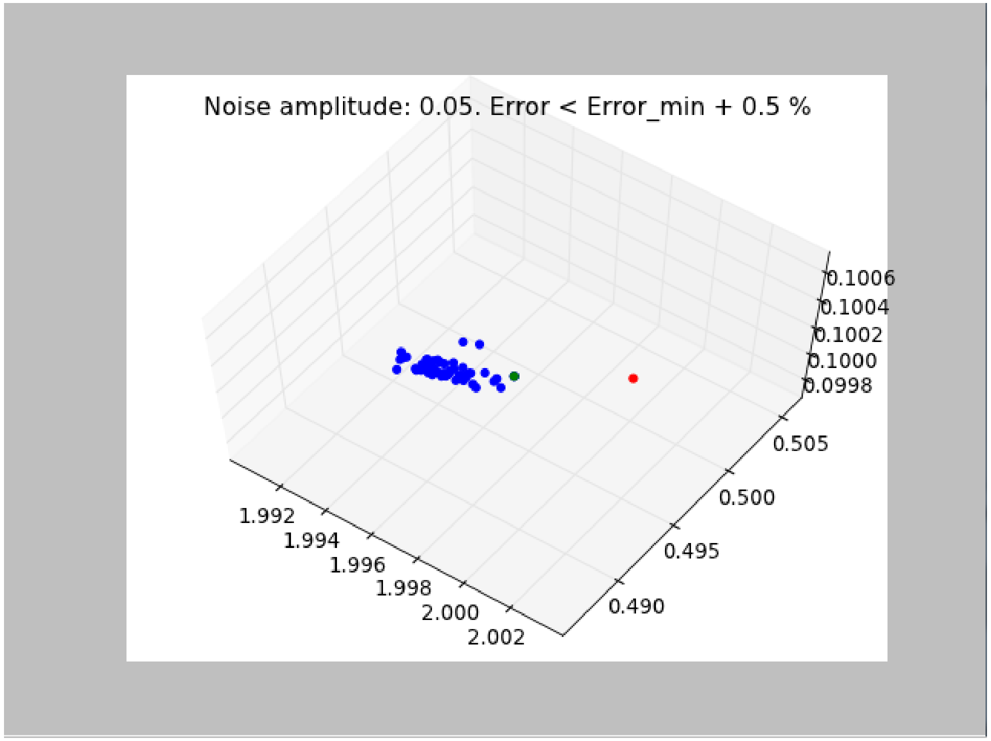


Stochastic Gradient Descent (SGD)
$\delta = 0.005$

GG.plot_path2(GG2,(0,1),True,pa,pb)

Example 3: Damped oscillator

Target value: (2.0, 0.5, 1.0)
Initial guess: (1.5, 0.3, 0.2)



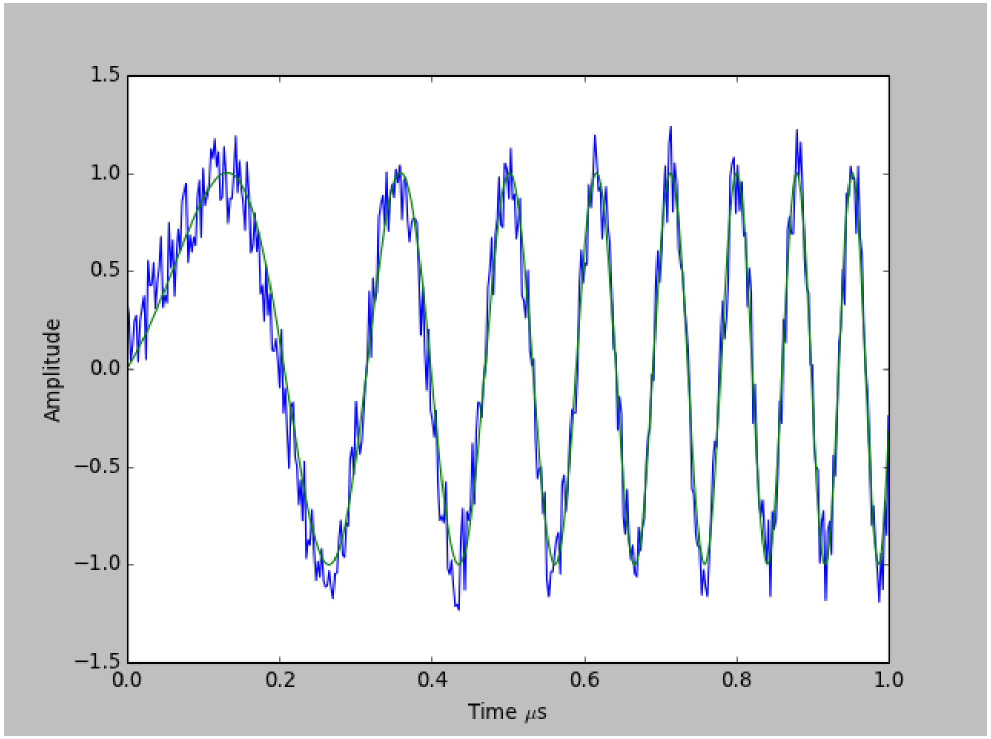
Stochastic Gradient Descent (SGD)
$\delta = 0.005$

GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 4:
Linear chirp

$S(t) = w(t)\sin[\phi_0 + 2\pi f_1 t + \pi\ B/T\ t^2]$

Initial guess error:
B/T ≈ 10%
f_1 ≈ 20%
ϕ_0 ≈ 0.4 rad



Stochastic Gradient Descent (SGD)
$\delta = 0.005$

Example 4:
Gained linear chirp

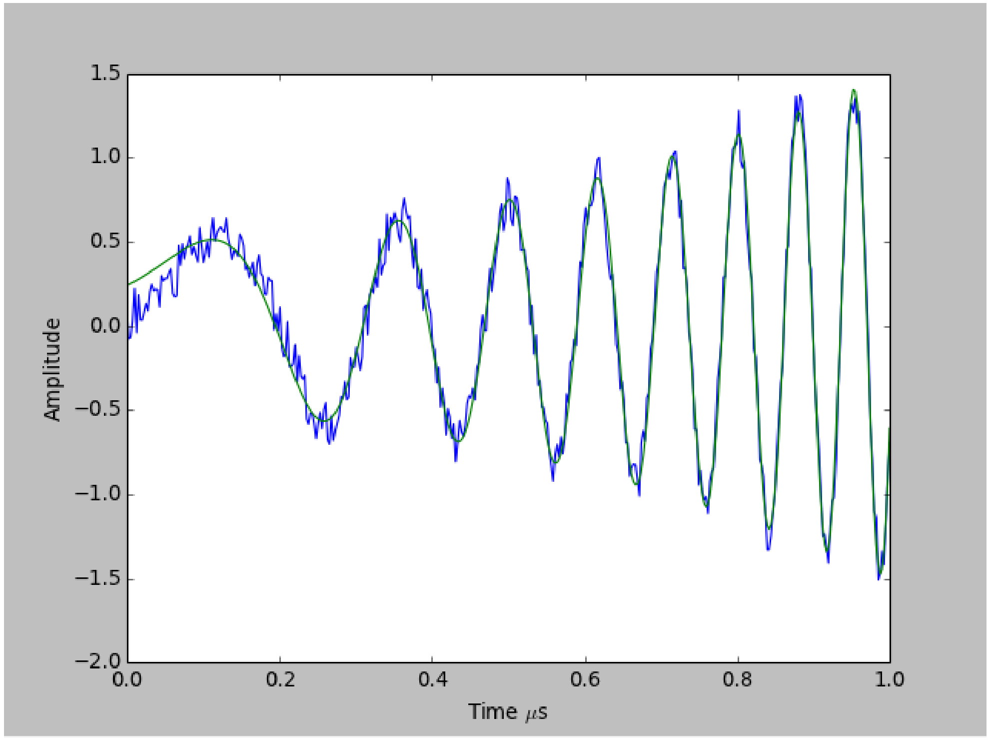
$S(t) = w(t)\sin[\phi_0 + 2\pi f_1 t + \pi\, B/T\, t^2]$

Initial guess error:

$B/T \approx 10\%$

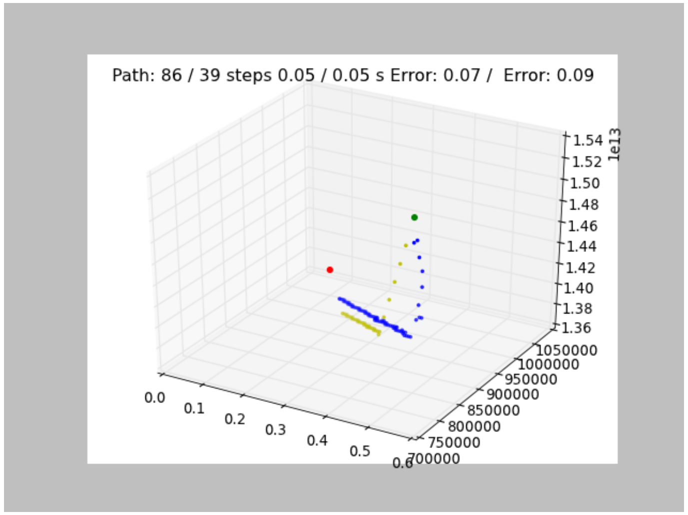
$f_1 \approx 20\%$

$\phi_0 \approx 0.4$ rad

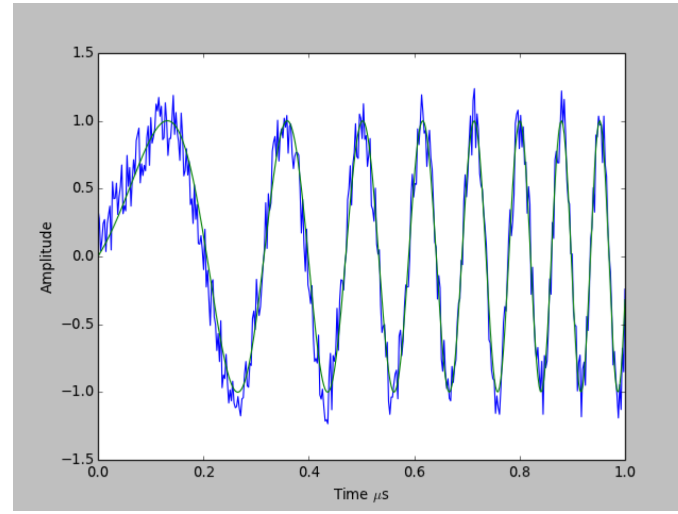


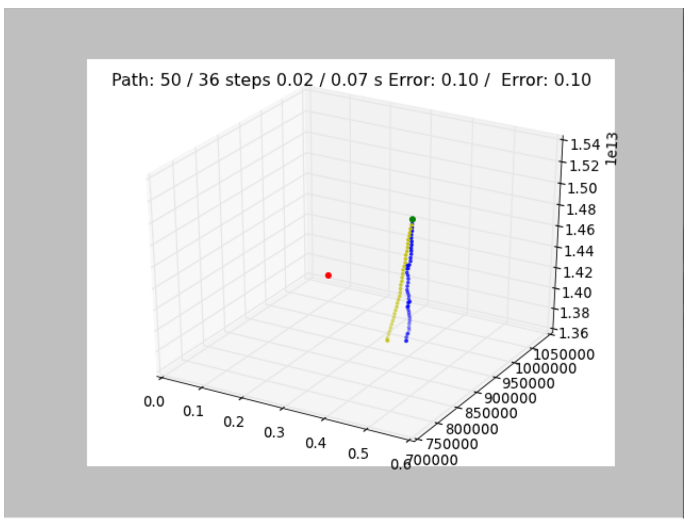
Stochastic Gradient Descent (SGD)

$\delta = 0.005$

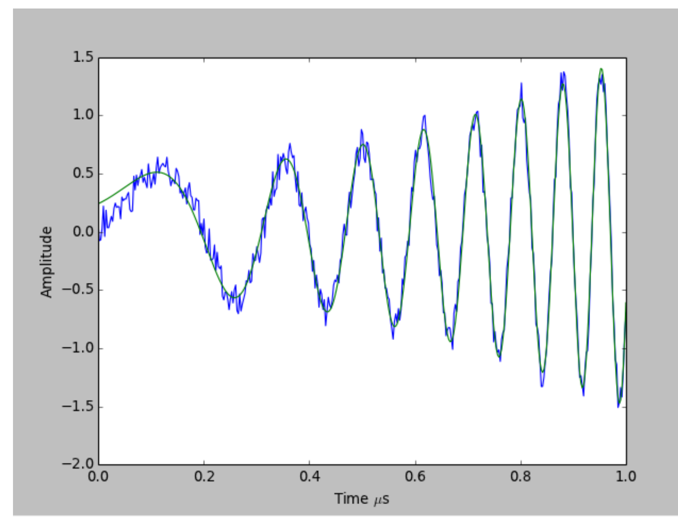


Linear chirp

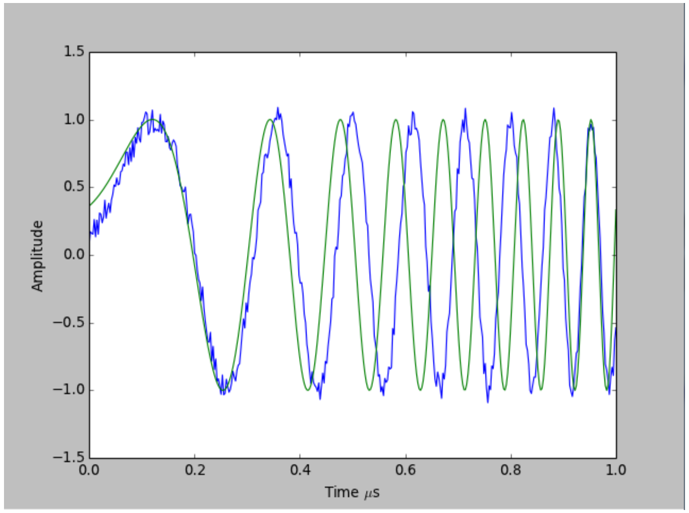




Gained linear chirp



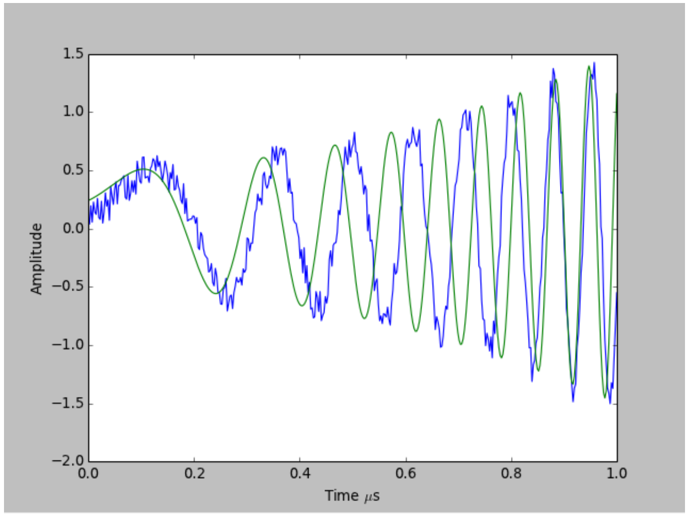
Linear chirp



$S(t) = w(t)\sin[\phi_0 + 2\pi f_1 t + \pi\, B/T\, t^2]$

Initial guess error:
B/T ≈ 20% → trapped in loc. min.
f_1 ≈ 20%
ϕ_0 ≈ 0.4 rad

Gained linear chirp



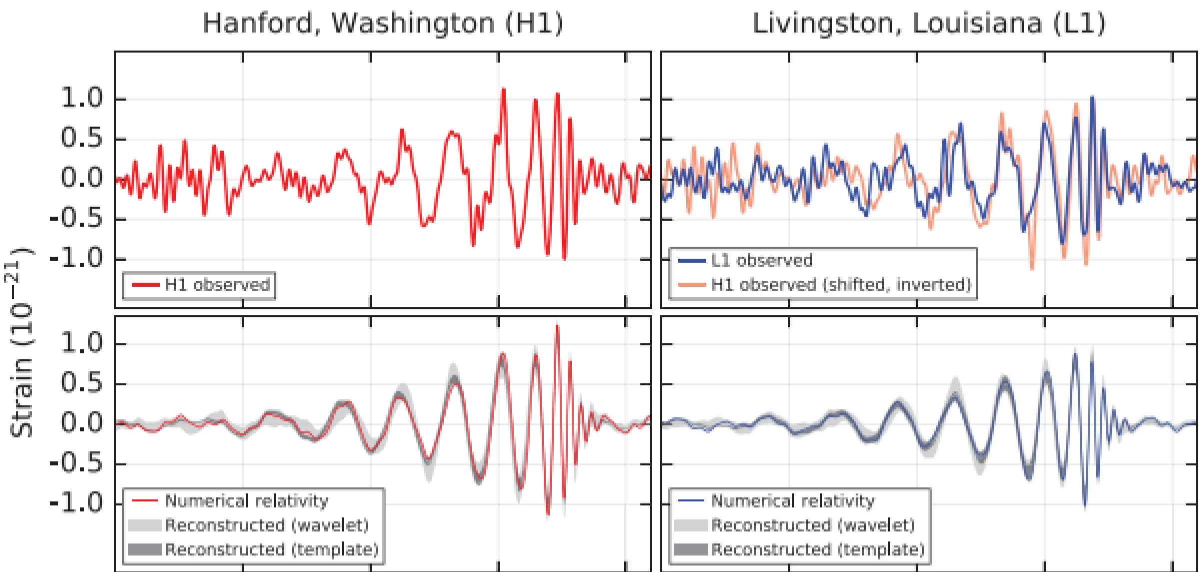
Increase δ: 0.005 → 0.2

Transform: FFT
Time domain → frequency domain

t.gomez@csic.es www.us-biomat.com

Original LIGO signals (time domain)

Best fitting signals (time domain)

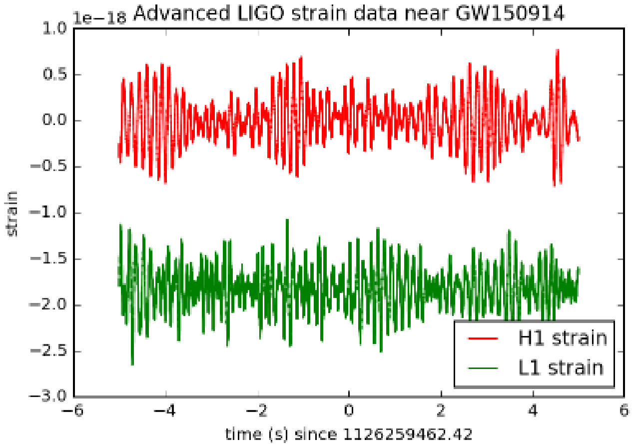


(time domain)

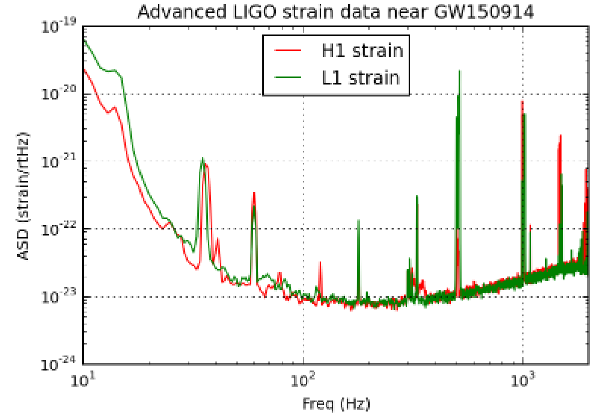
detector in the 35–350 Hz band. Solid lines show a numerical relativity waveform for a system with parameters consistent with those recovered from GW150914 [37,38] confirmed to 99.9% by an independent calculation based on [15]. Shaded areas show 90% credible

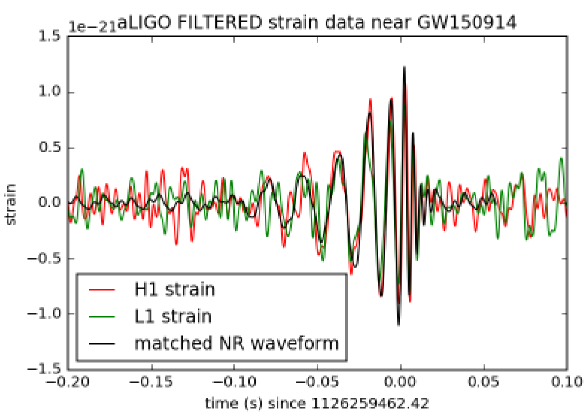
$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} = \frac{c^3}{G}\left[\frac{5}{96}\pi^{-8/3} f^{-11/3} \dot{f}\right]^{3/5},$$

Original LIGO signals (time domain)

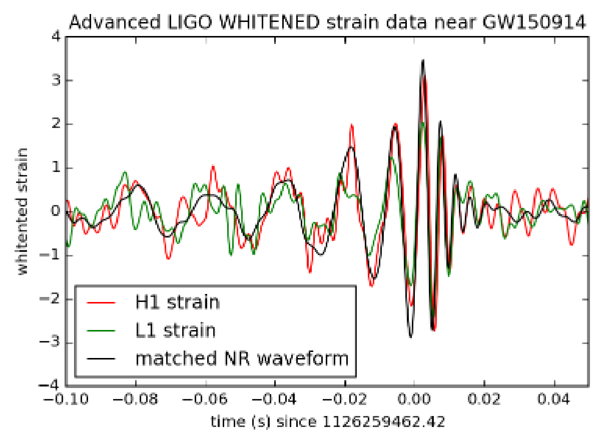


Frequency domain)





Final LIGO signals (time domain)



6. Examples (III)

Signals in the frequency domain.

Large dimensional problems

Echographic signal: single layer
Echographic signal: layered plate

Example 5:
Ultrasonic transmission through a layer of tissue

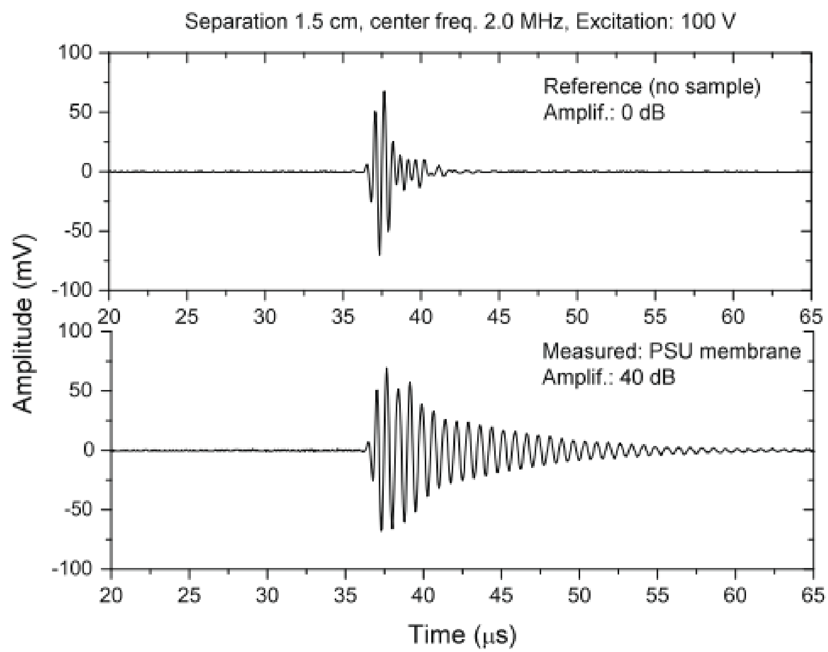
$S(t) = f(\beta, t)$ $\xrightarrow{\text{FFT}}$ $S^*(\omega) = f^*(\beta, \omega)$

ω : frequency

$\beta = \{t, \rho, v, \alpha, n\}$

t: thickness
ρ *:* density
v: us velocity
α *:* us attenuation
n: variation of α with freq.

Dimension = 5 (V_{search} = 242 pts.)



Example 5:
Ultrasonic transmission through a layer of tissue

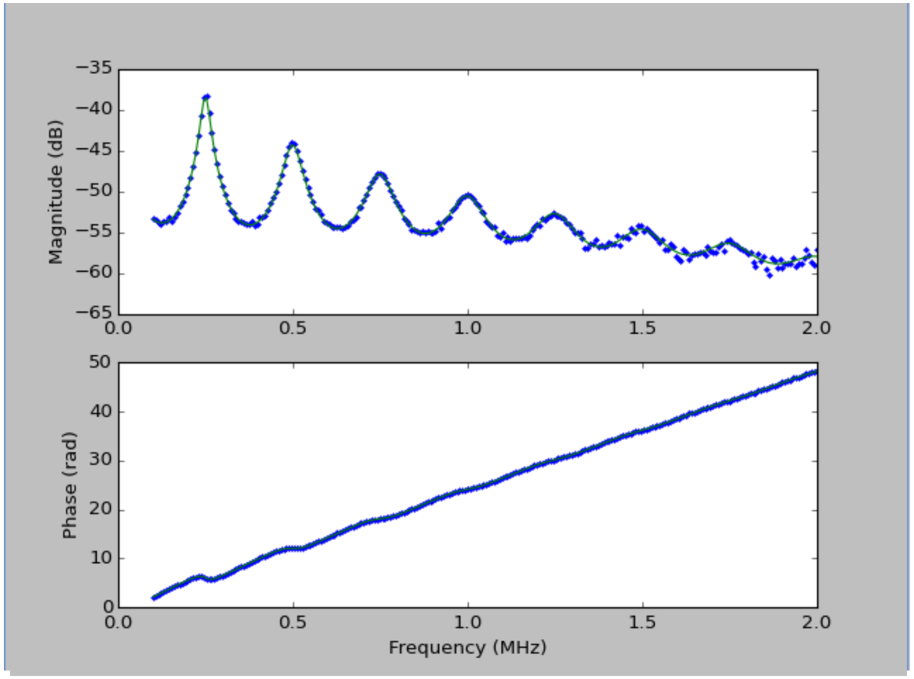
$S(t) = f(\beta, t)$ $\xrightarrow{\text{FFT}}$ $S^*(\omega) = f^*(M,L)$

ω : frequency

$\beta = \{t, \rho, v, \alpha, n\}$

t: thickness
ρ *:* density
v: us velocity
α *:* us attenuation
n: variation of α with freq.

Dimension = 5 (V_{search} = 242 pts.)



Example 5:
Ultrasonic transmission through a layer of tissue

FFT

$S(t) = f(M,L)$ → $S^*(\omega) = f^*(M,L)$

ω : frequency

M: medium*: Zm*

L: layer of tissue *t*, ρ, *v*, α, *n*

t: thickness

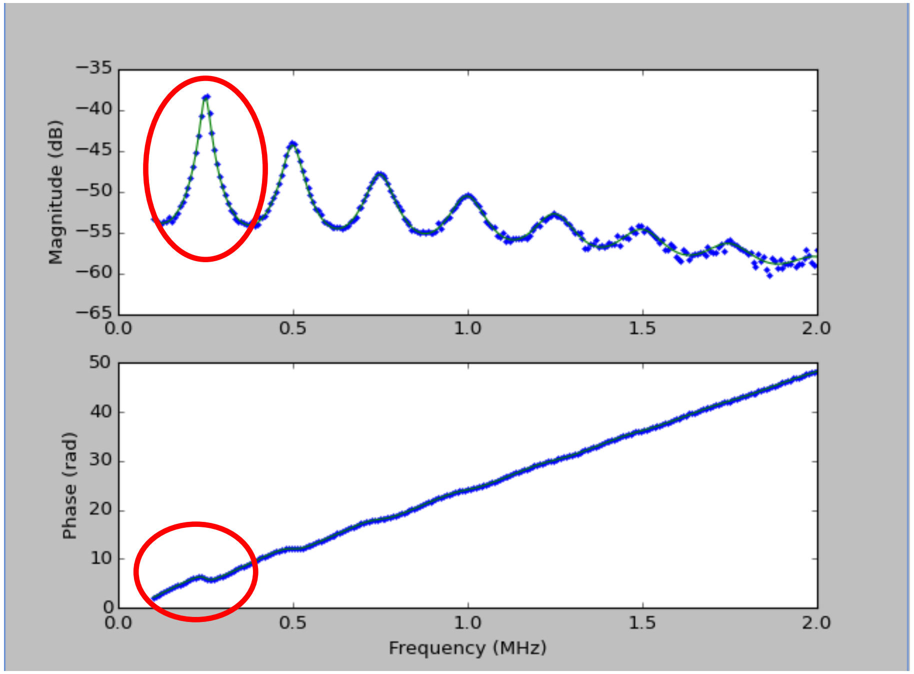
ρ *:* density

v: us velocity

α *:* us attenuation

n: variation of α with freq

Dimension = 4 (V_{search} = 81 pts.)



Example 5:
Ultrasonic transmission through a layer of tissue

FFT

S(*t*) = *f(M,L)* → S*(ω) = *f*(M,L)*

ω : frequency

M: medium*: Zm*

L: layer of tissue *t, ρ, v, α, n*

t: thickness

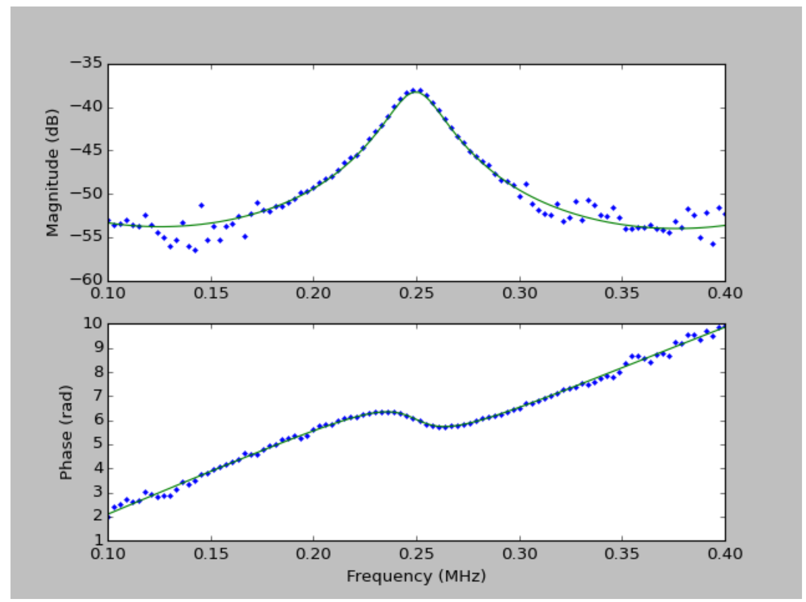
ρ *:* density

v: us velocity

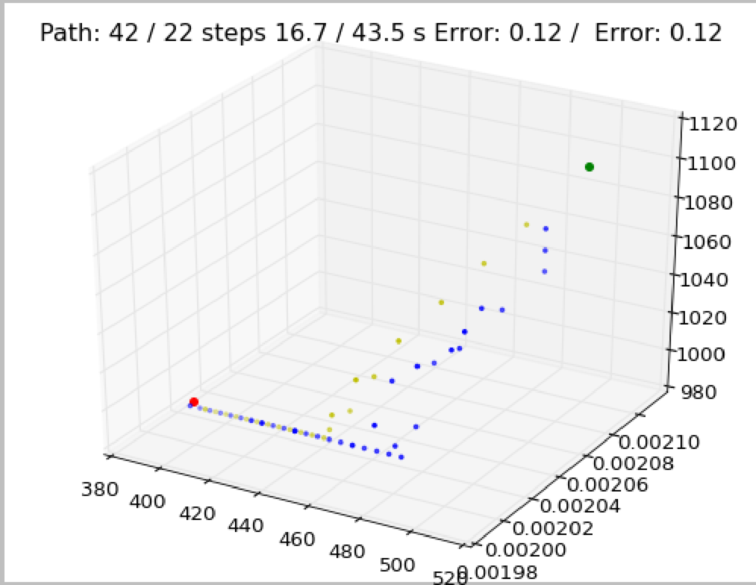
α *:* us attenuation

n: variation of α with freq

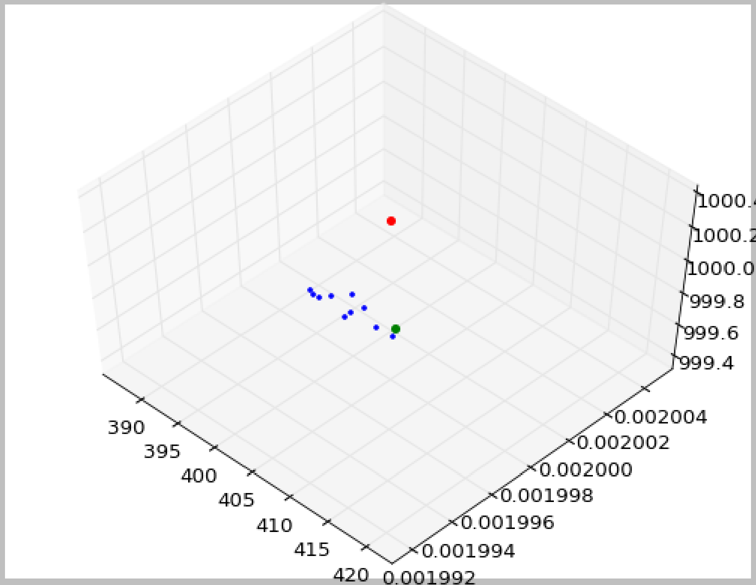
Dimension = 4 (V_{search} = 81 pts.)



Example 5:
Ultrasonic transmission through a layer of tissue



GG.plot_path2(GG2,(0,1),True,pa,pb)



GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 5:
Ultrasonic transmission through a layer of tissue

$S(t) = f(M,L)$ —FFT→ $S^*(\omega) = f^*(M,L)$

ω : frequency

M: medium*: Zm*

L: layer of tissue t, ρ, v, α, n

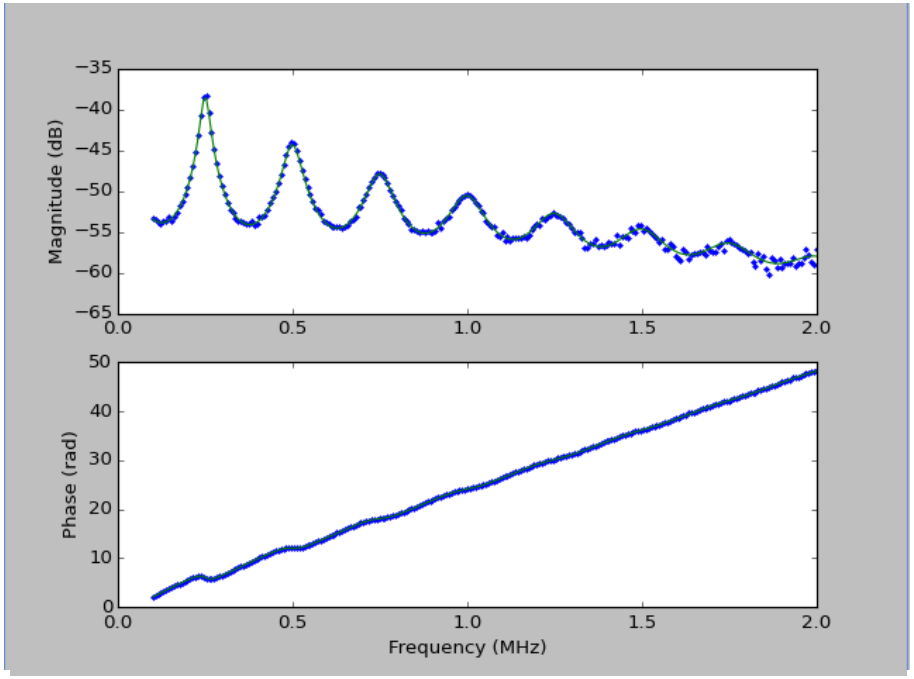
t: thickness

ρ *:* density

v: us velocity

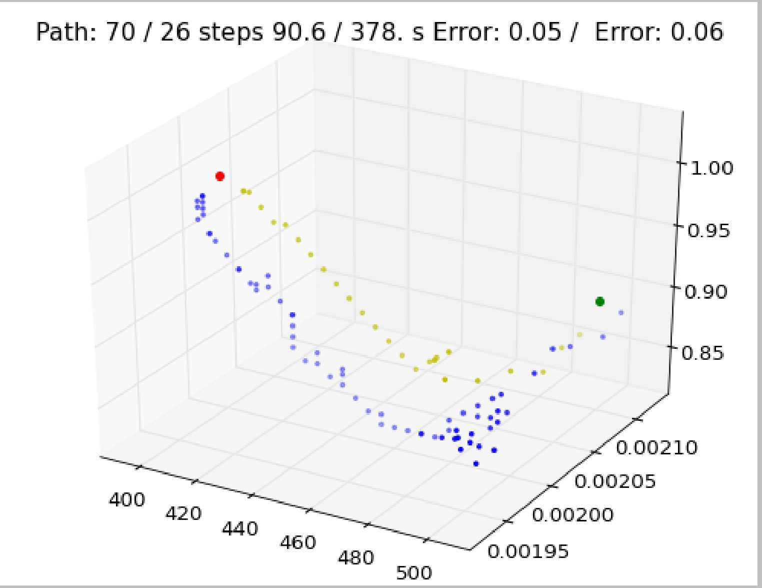
α *:* us attenuation

n: variation of α with freq

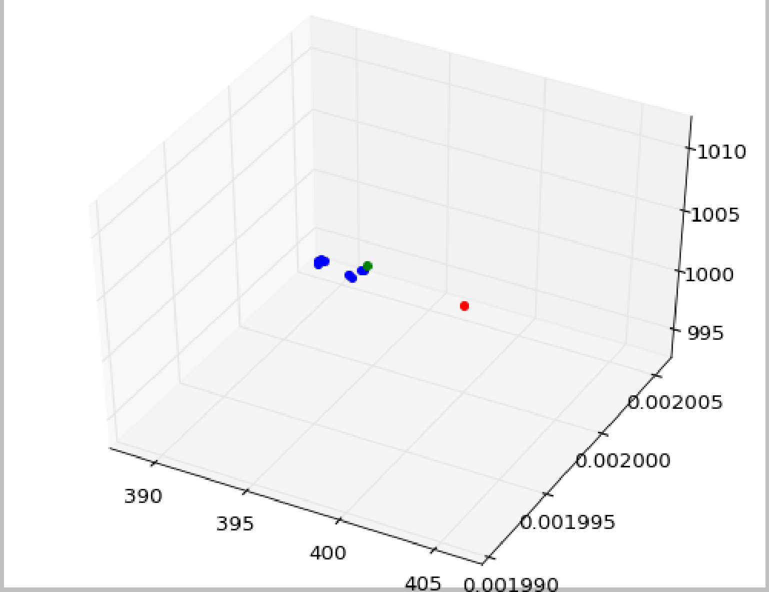


Dimension = 5 (V_{search} = 242 pts.)

Example 5:
Ultrasonic transmission through a layer of tissue



GG.plot_path2(GG2,(0,1),True,pa,pb)



GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Parameters separation/aggregation:

β_i , i = 1…N; 3^N-1

P subsets of parameters (example P = 2):

$\beta^1{}_i$, i = 1…K; 3^K-1

$\beta^2{}_i$, i = 1…L; 3^L-1

M = K + L

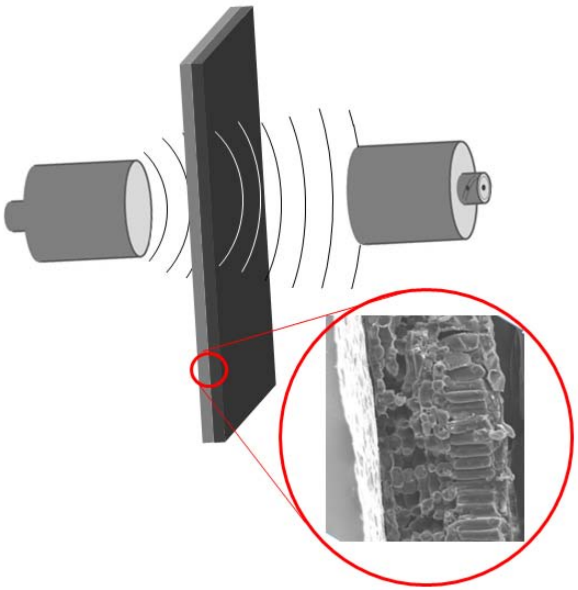
Perform SGDA on $\beta^1{}_i$ + early stop

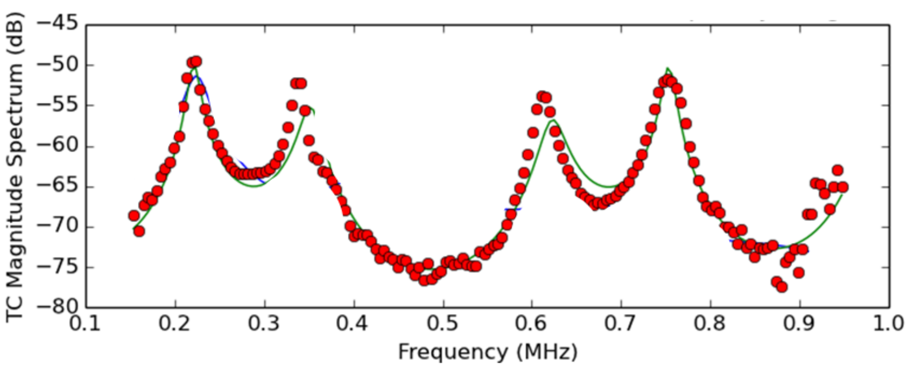
Perform SGDA on $\beta^2{}_i$ + early stop

t.gomez@csic.es www.us-biomat.com

Example 6:
Ultrasonic transmission through a layered tissue

10 parameters in the model:
IG for them all
VS = 59049





Simplify the data set (reduction)

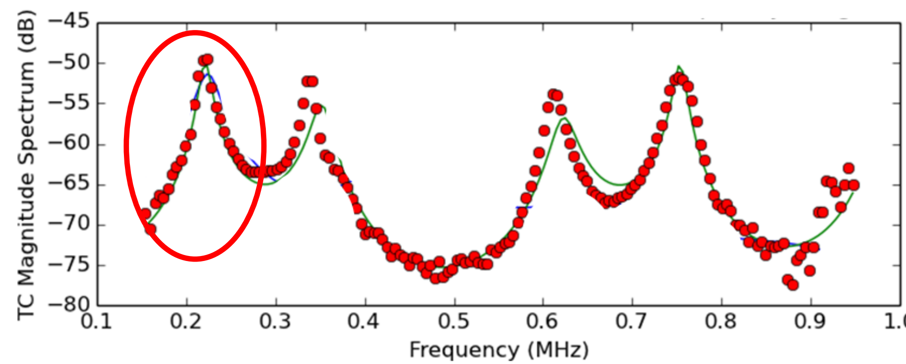
Simplify the model
(one layer: 5 params)

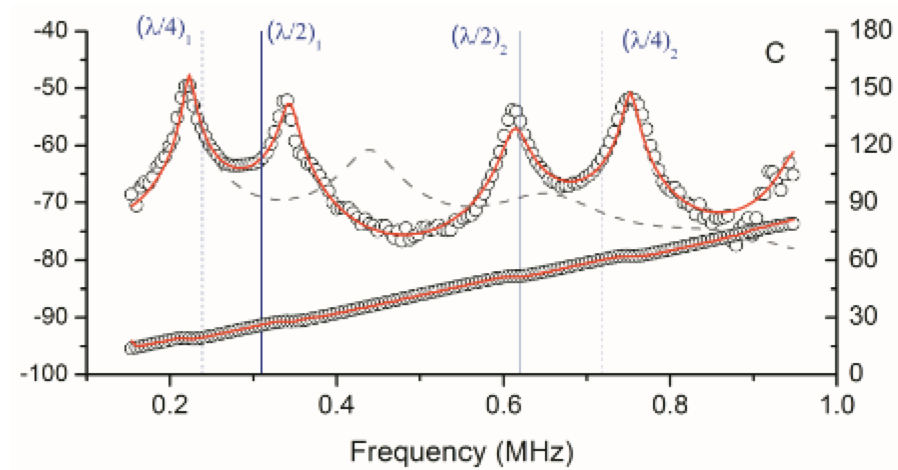
Apply parameters gathering
(3 + 2)

Solve the new IP (- - - -)
early stop

Generate IG for whole IP
5 params → 10 params

Solve the whole IP





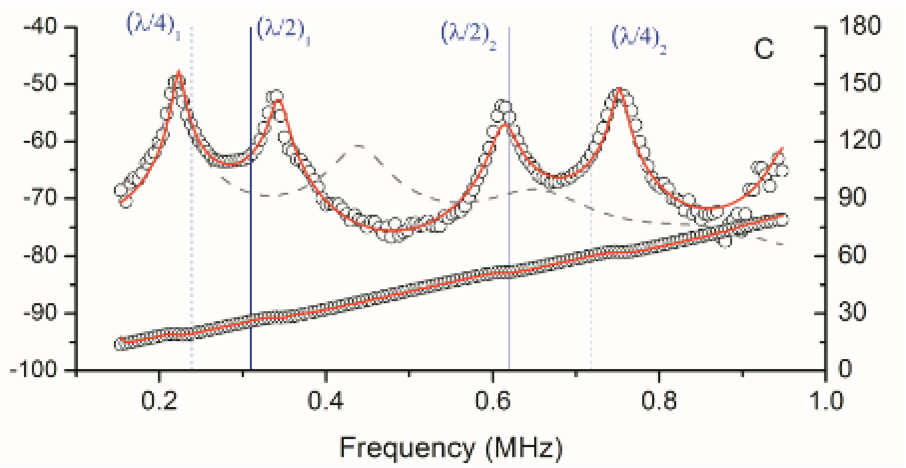
Solve the whole IP

Ill-posed problem
(infinite solutions)

Adding constrains:
(total thickness is know)
10 $\rightarrow$ 9

Ill-posed problem
(not unique solution).

IG
Further constrains.



Data

KNOWLEDGE

Q&A

t.gomez@csic.es www.us-biomat.com