THE INVERSE PROBLEM

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

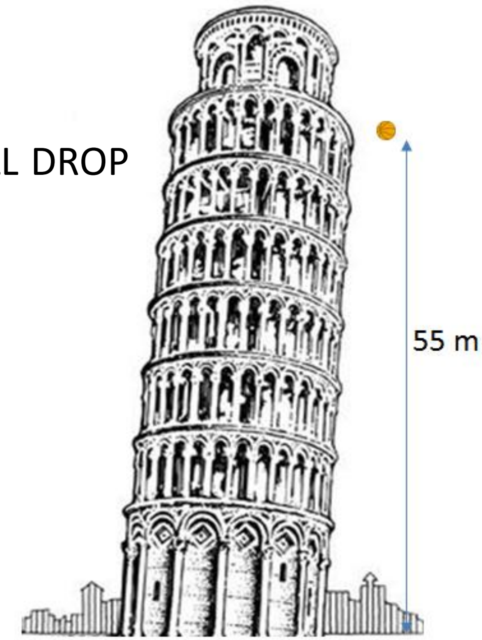
&

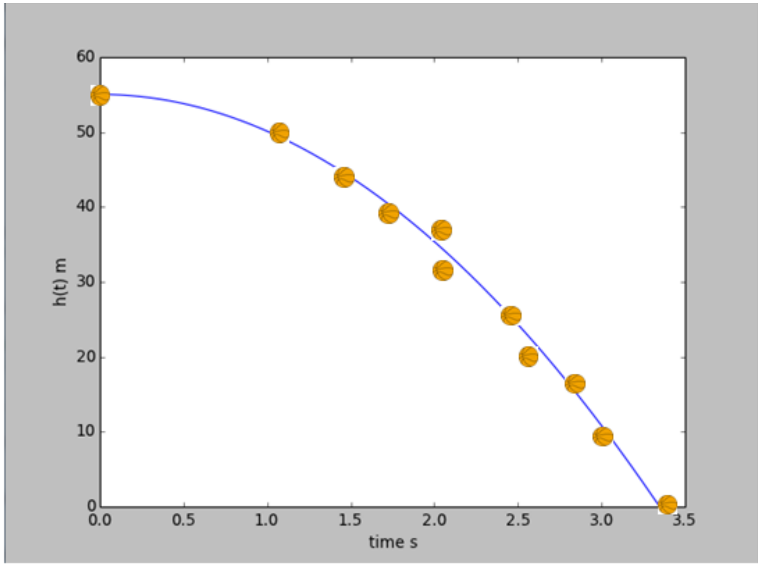
Model (law):
$h(t) = h_0 - 1/2\ g\ t^2$

PREDICT:

h at any t

BALL DROP





IP: Definition.

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

INVERSE PROBLEM

EXTRACT:

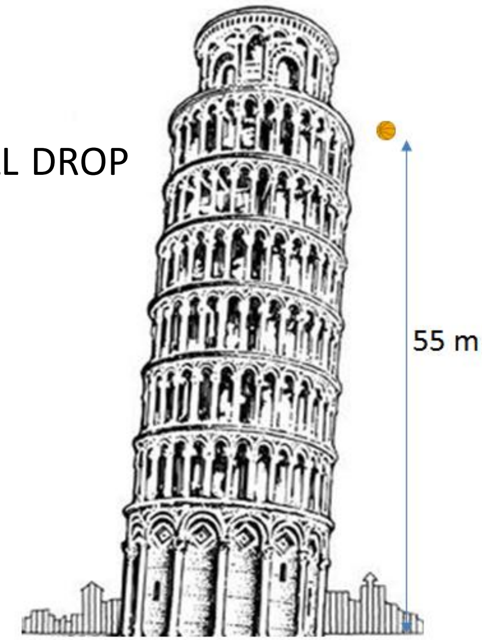
Model parameters: h_0, g

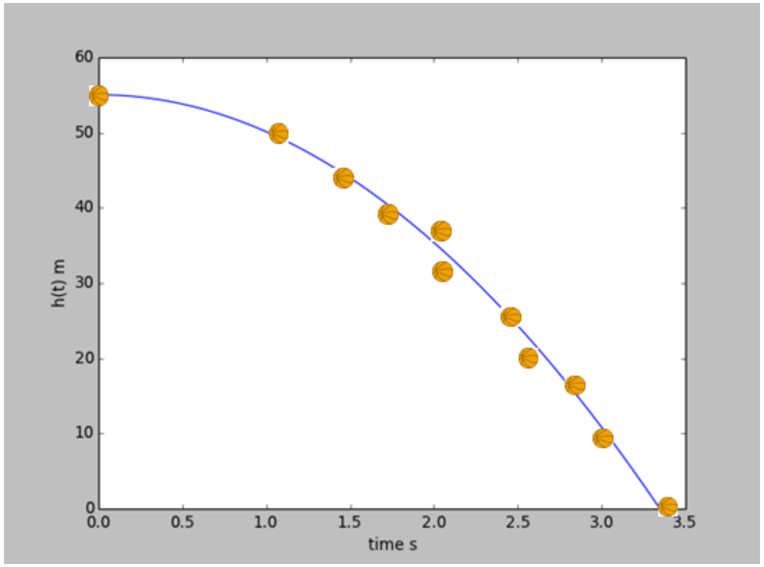
GIVEN

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

& h at some t

BALL DROP

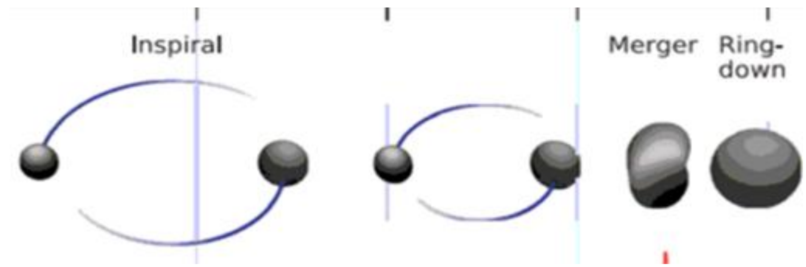




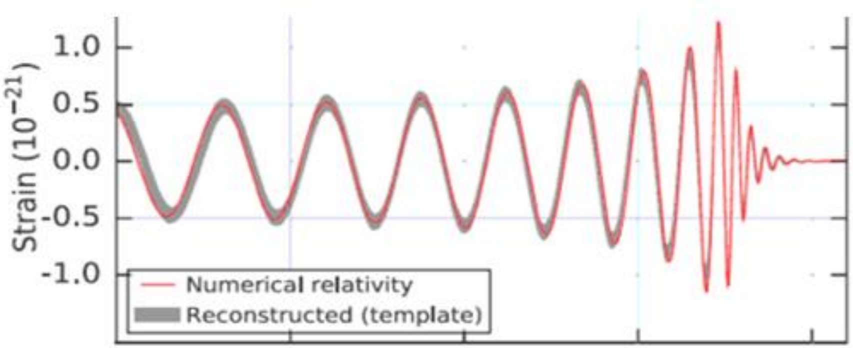
IP EXAMPLE:

GRAVITATIONAL WAVES

DIRECT PROBLEM

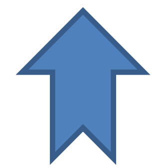


$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1+m_2)^{1/5}} = \frac{c^3}{G}\left[\frac{5}{96}\pi^{-8/3} f^{-11/3}\dot{f}\right]^{3/5},$$

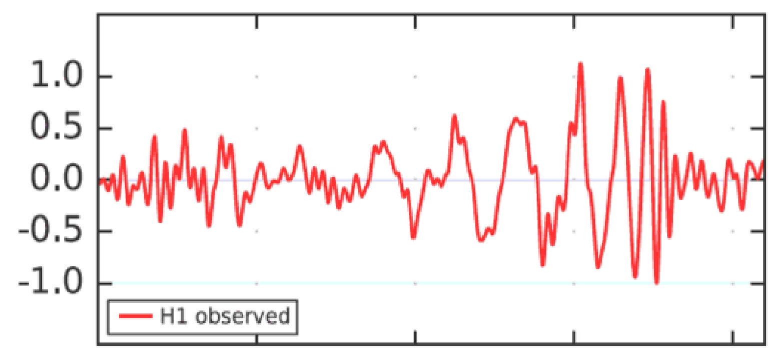


INVERSE PROBLEM

Parameters



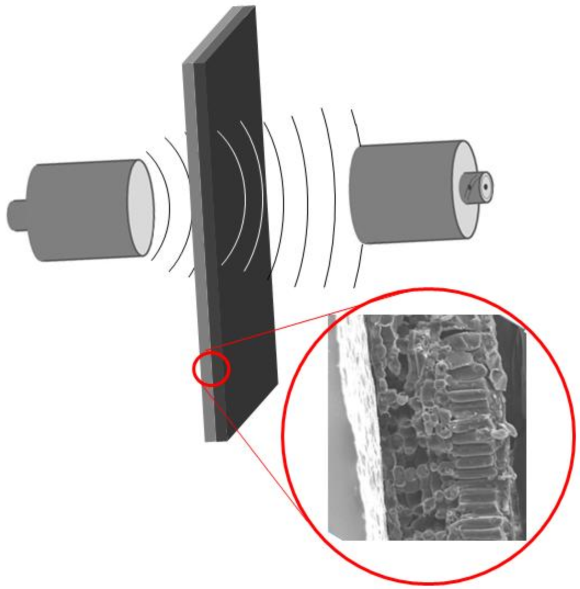
Hanford, Washington (H1)

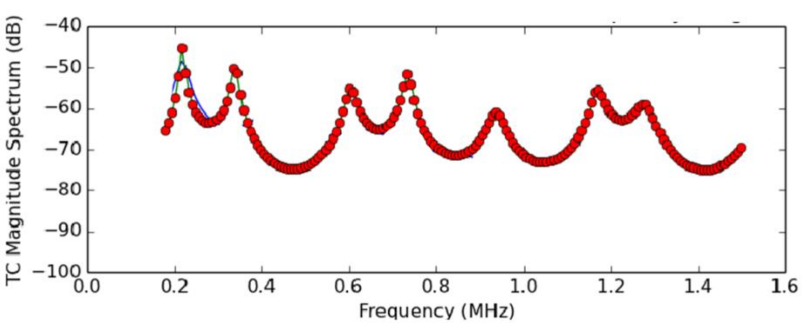


IP EXAMPLE:
ECHOGRAPHIC SIGNALS

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

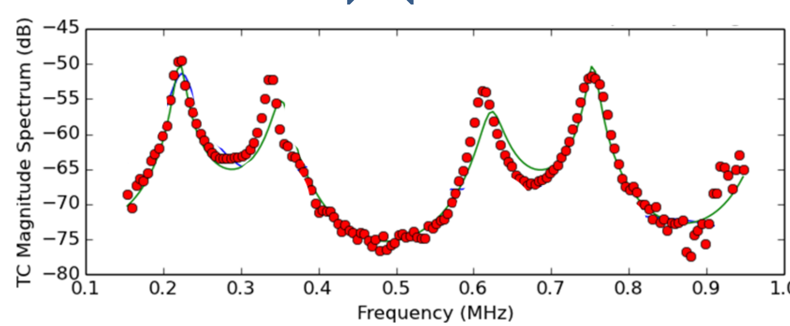
DIRECT PROBLEM





INVERSE PROBLEM

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



2. IP APPLICATIONS

1. Obtention of model parameters .

2. Model confirmation.

3. Model selection.
   Ockham's razor

3. IP Main Elements

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

Inverse problem

Model (f):

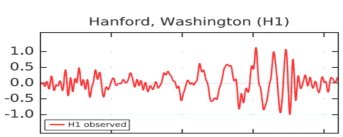
$y^*_i = f(\beta_j, x_i)$

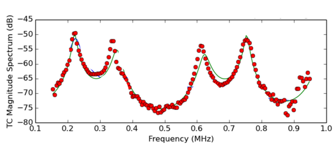
Error Function

ε

Data

x_i , y_i





Optimization algorithm

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

ε_{min}

Output

Model Parameters:

β_j.

IP *vs* ML

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}

GUESS MODEL OUTPUT FOR NEW DATA

OUTPUT

MODEL PARAMETERS:

β_j.

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.

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

t.gomez@csic.es

4. IP Optimization Algorithms

t.gomez@csic.es

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.

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

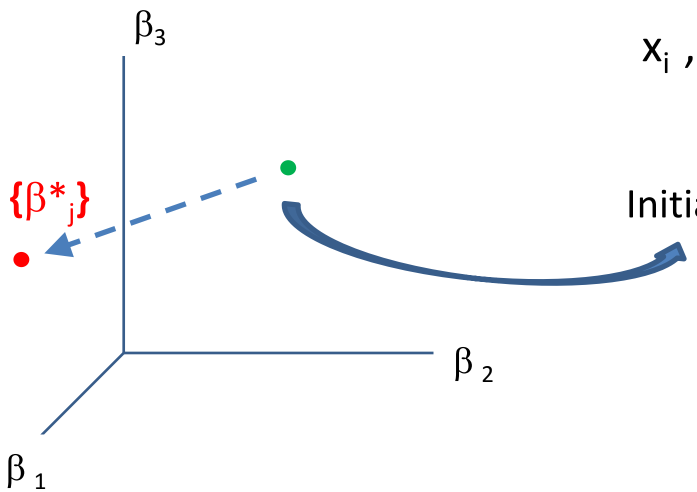
DATA

x_i , y_i

MODEL (f)

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

Solution: $\{\beta^*_j\}$



Initial guess: β^0_1, β^0_2, β^0_3

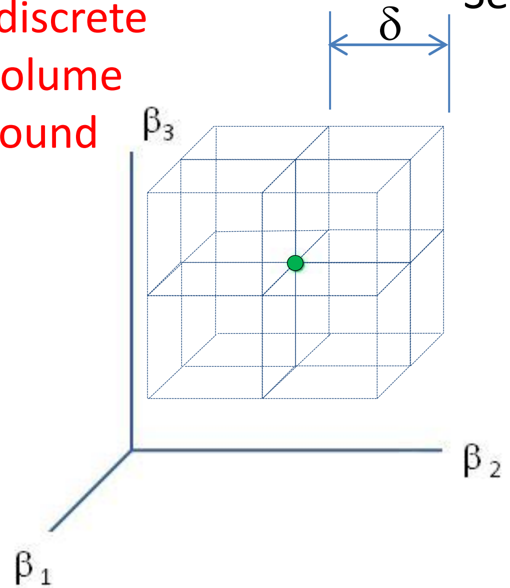
$f(\{\beta^0_j, x_i\})$

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

The Gradient Descent approach for Inverse Problems.

Set a discretization step: δ

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



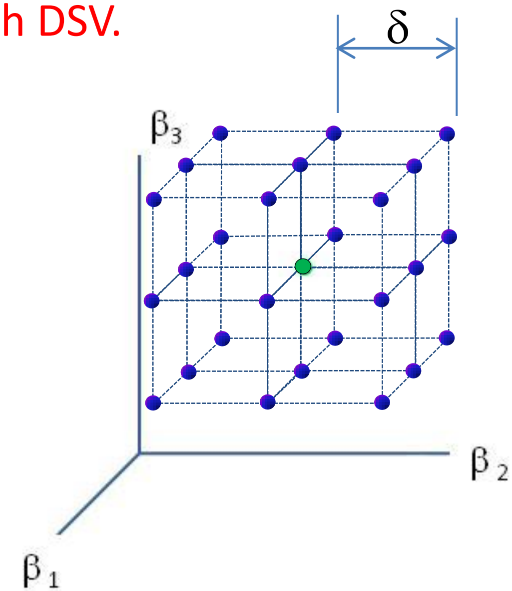
DATA	**MODEL (*f*)**
$x_i\ ,\ y_i$	$y^*_i = f(\beta_j \pm\delta,\ x_i\,)$

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

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

The Gradient Descent approach for Inverse Problems.

3. Search DSV.



DATA

x_i , y_i

MODEL (f)

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

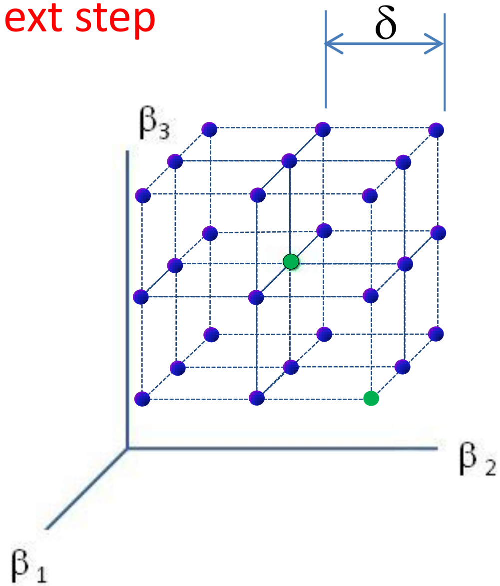
Dimension	Volume
N = 3:	27
N = 4:	81
N = 5:	243
N = 6:	729
N = 7:	2187
N = 8:	6561

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

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

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

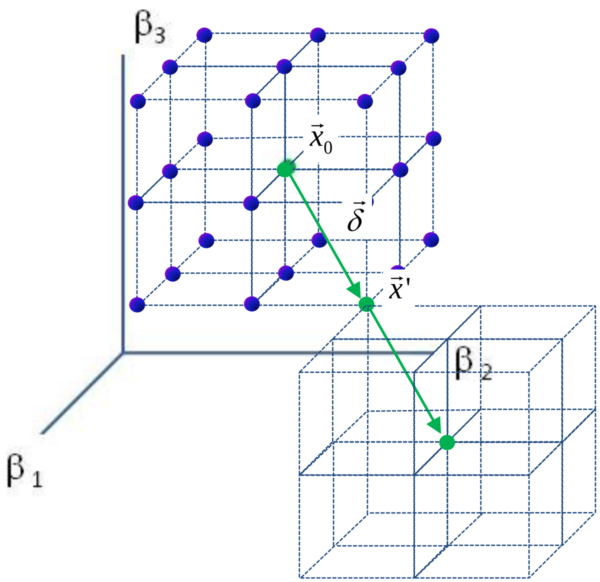
Stop searching when:

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

	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.

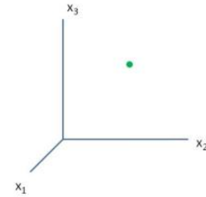
5. Take a step.

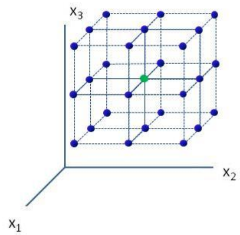


Take a step: $2\vec{\delta} = 2\left(\vec{\beta}' - \vec{\beta}_0\right)$

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

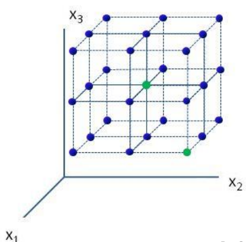
Initial guess $\{\beta_0\}$.



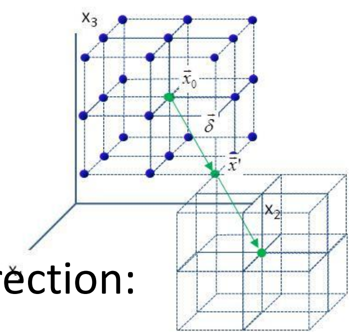


$f(\{\beta_0\})$ and $\Delta\{\beta_0\}$

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



Does any point reduce error?
[Systematic /Stochastic]



NO

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

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.

CODE DESCRIPTION

https://github.com/PyDataMadrid2016/Conference-Info/

20160409_1530_The_solution_of_inverse_problems

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

6. Examples.

Basic examples
Signals in time domain
Complex high dimensional problems

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

6. Examples (I)

Basic Examples.

Linear regression
Polynomial fitting

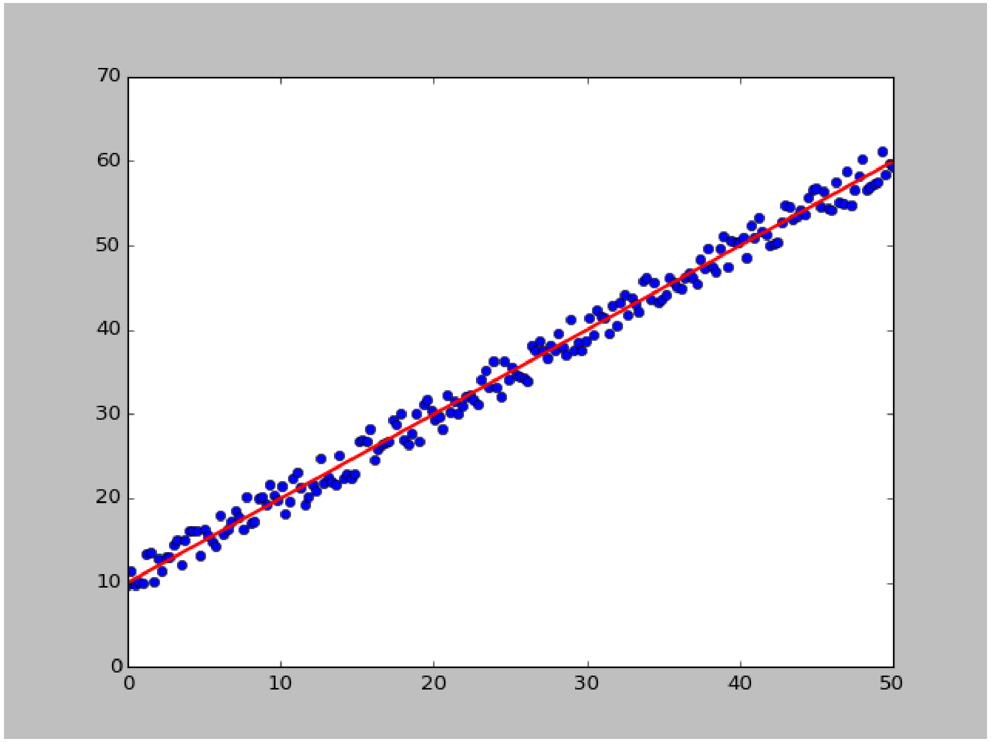
Example 1: Linear regression
Run #1

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

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

Objective:

Find value of function parameters: (a,b) that minimize the error (L2 norm) between calculated (y) and data (y*).

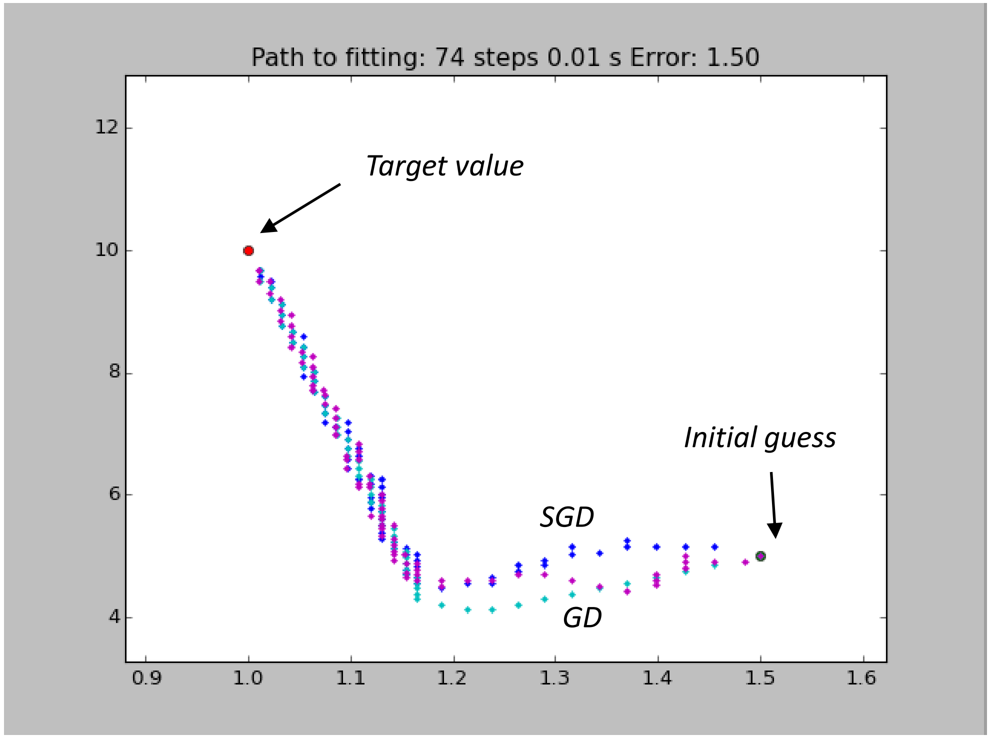


Stochastic Gradient Descent (SGD)
Discretization step (DS): $\delta = 0.01$

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)



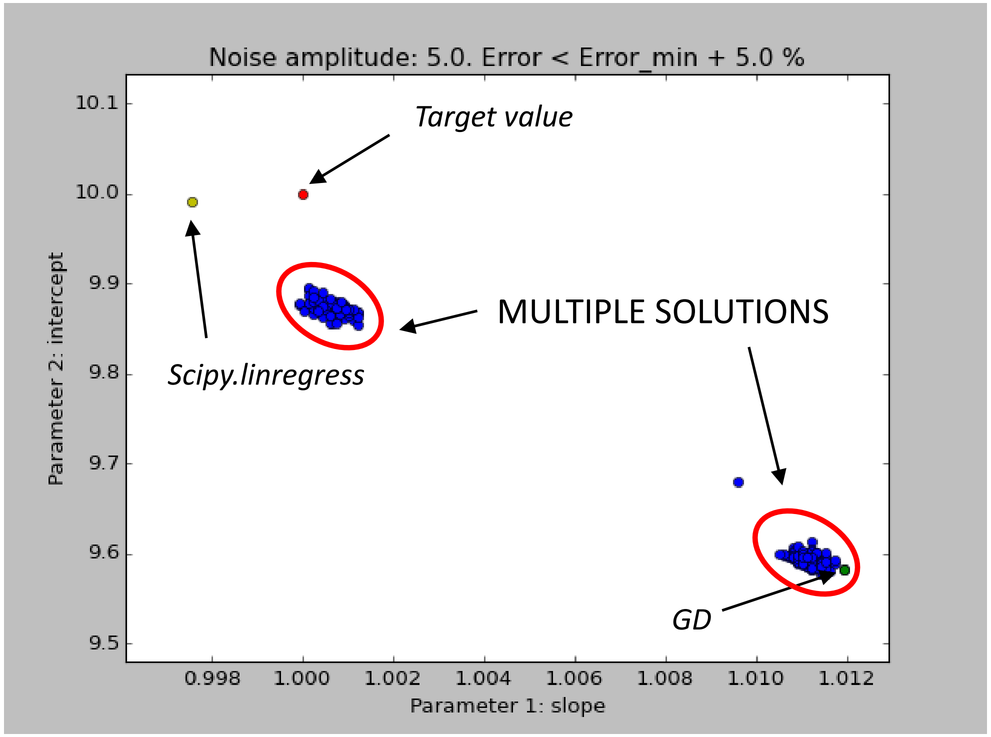
GG.plot_pathN((GG2,GG3),(0,1),True,pa,pb)

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%



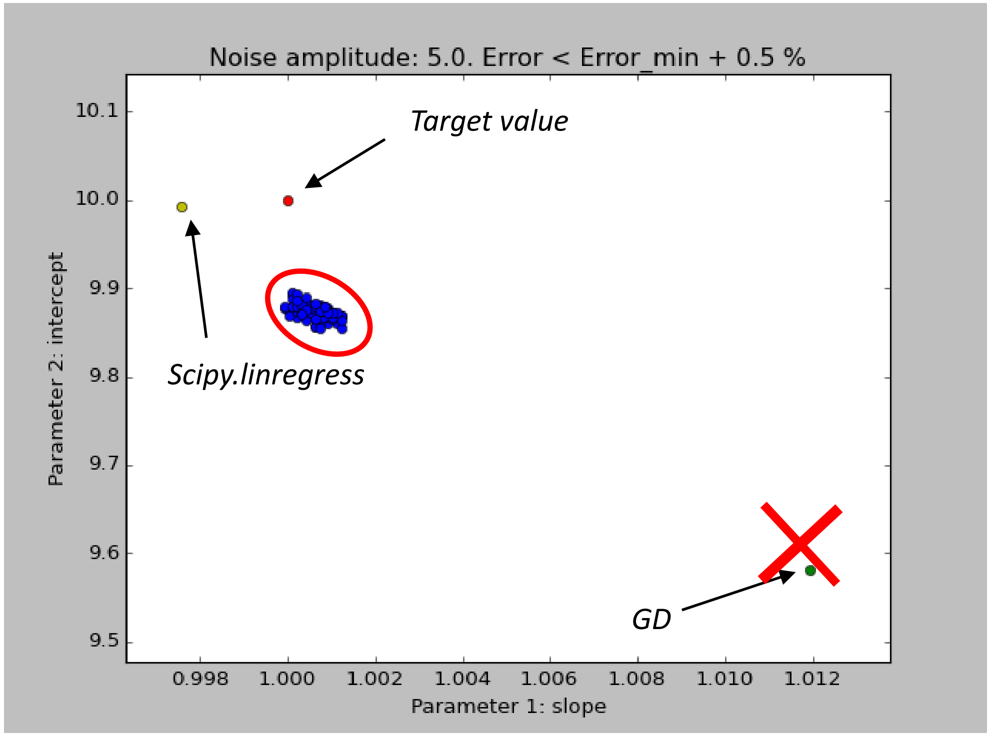
GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

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%

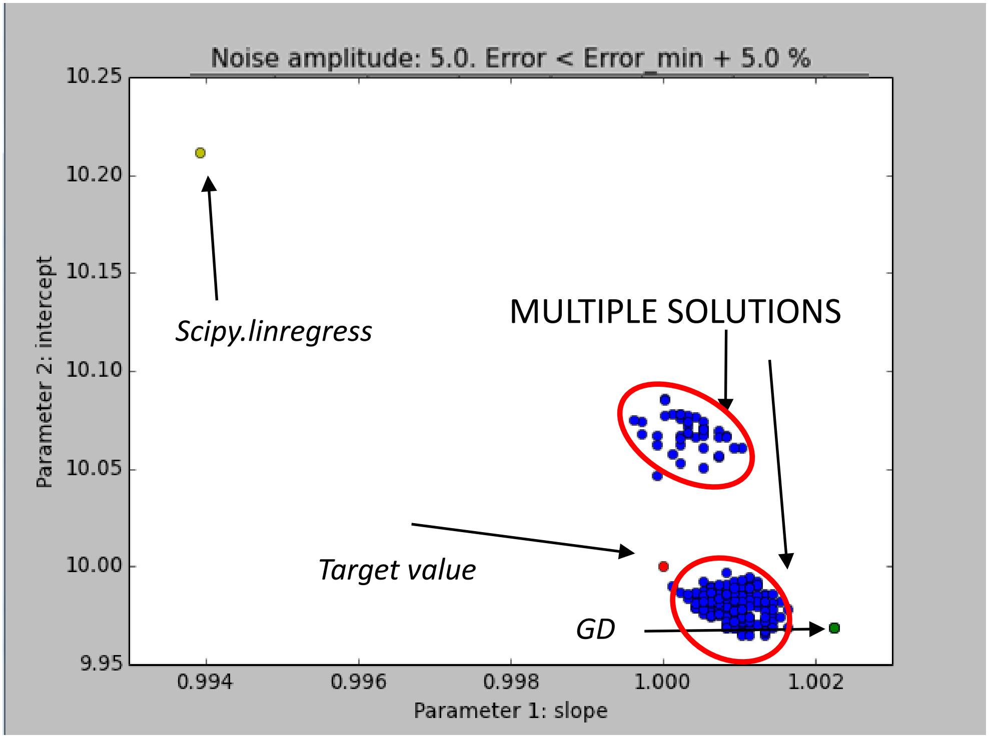


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 1: Linear regression
Run #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%

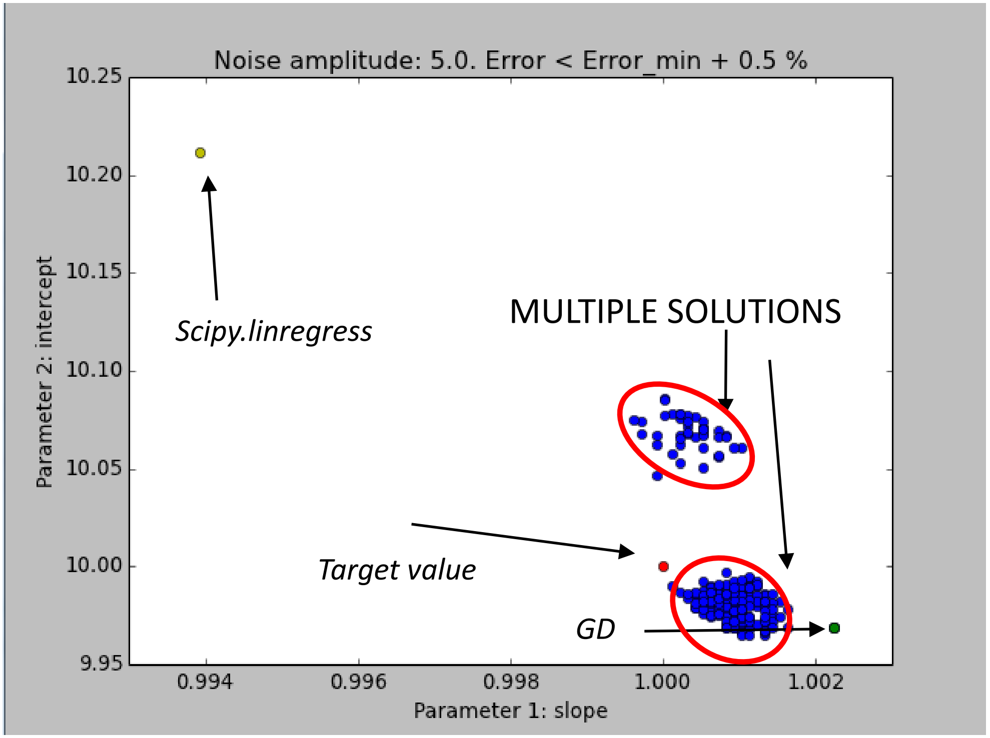


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

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%

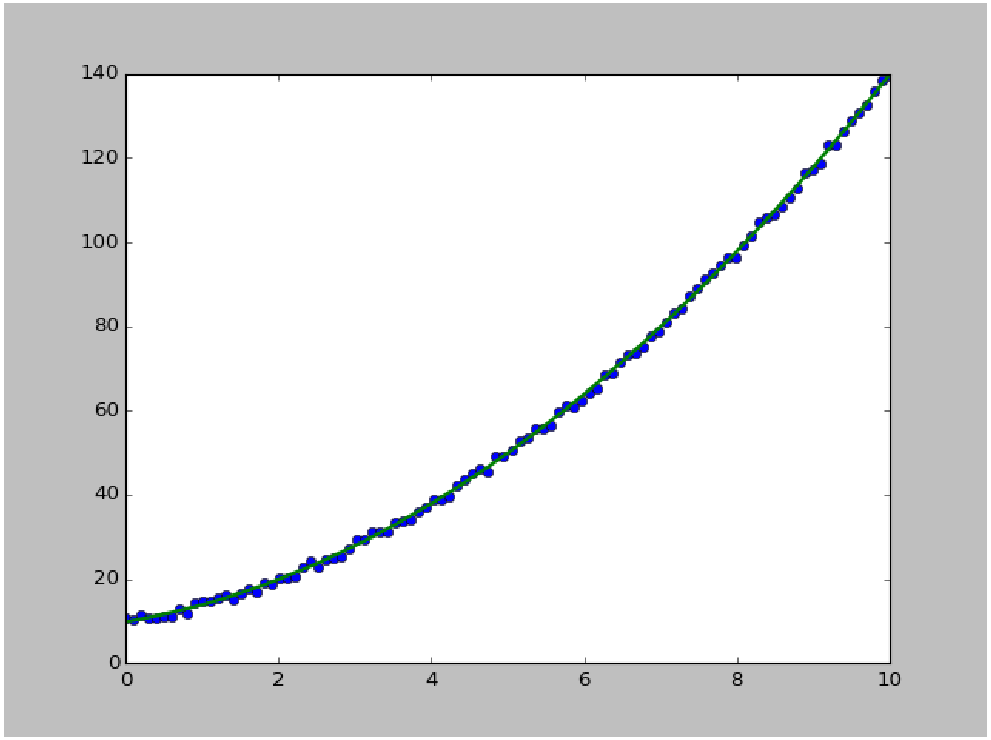


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

Example 2: Polynomial order 2

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

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

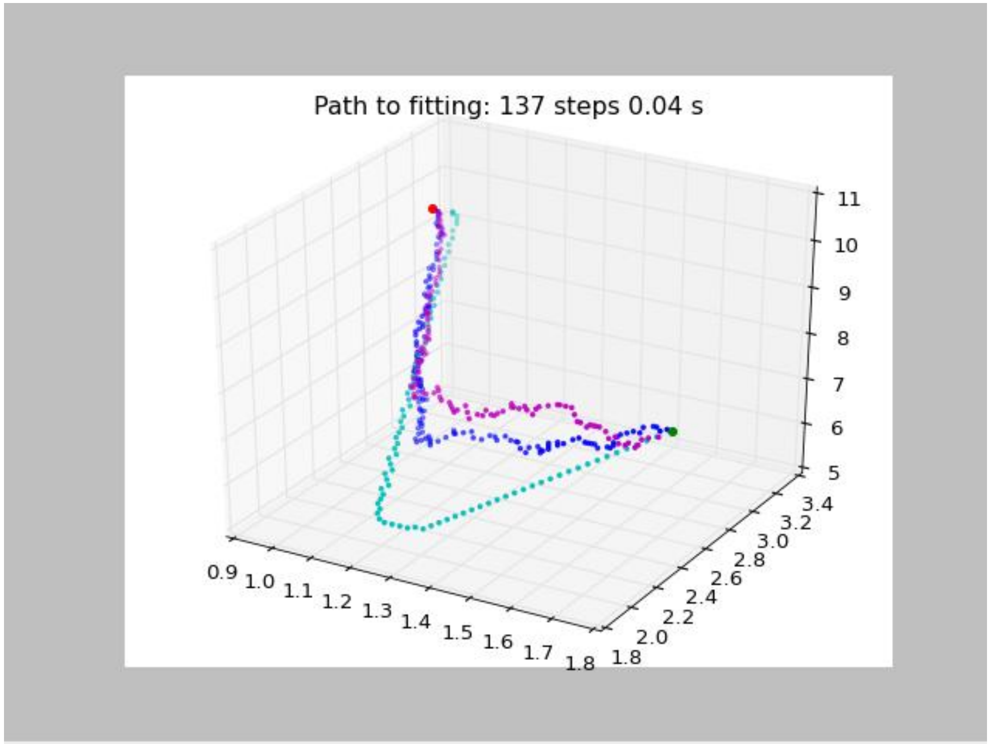


Stochastic Gradient Descent (SGD)
$\delta = 0.005$

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)



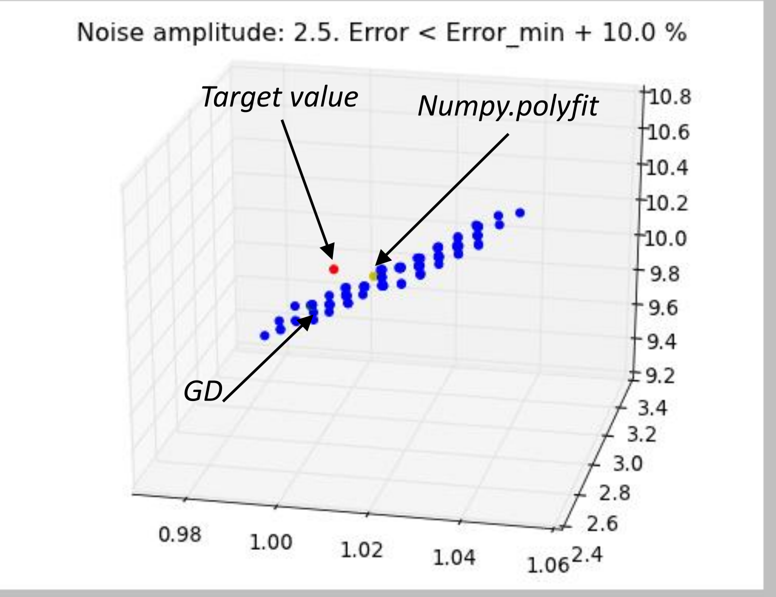
Stochastic Gradient Descent (SGD)
$\delta = 0.005$

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%

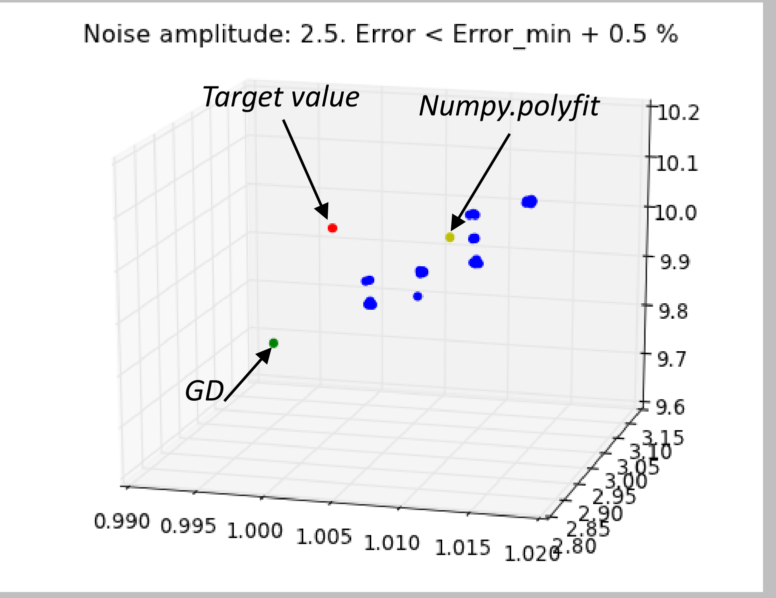


GG.plot_Bestcloud(tuple(GG_cloud),error_lim,(0,1),True,pa,pb)

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

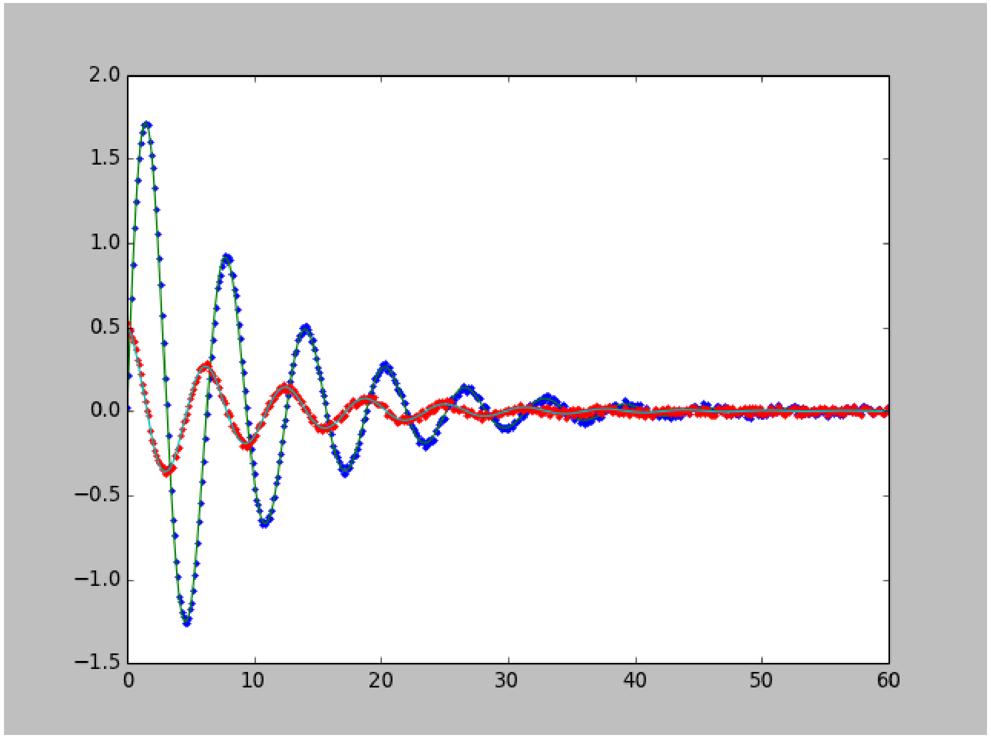
Example 3: Damped oscillator

Data:

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

Model:

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



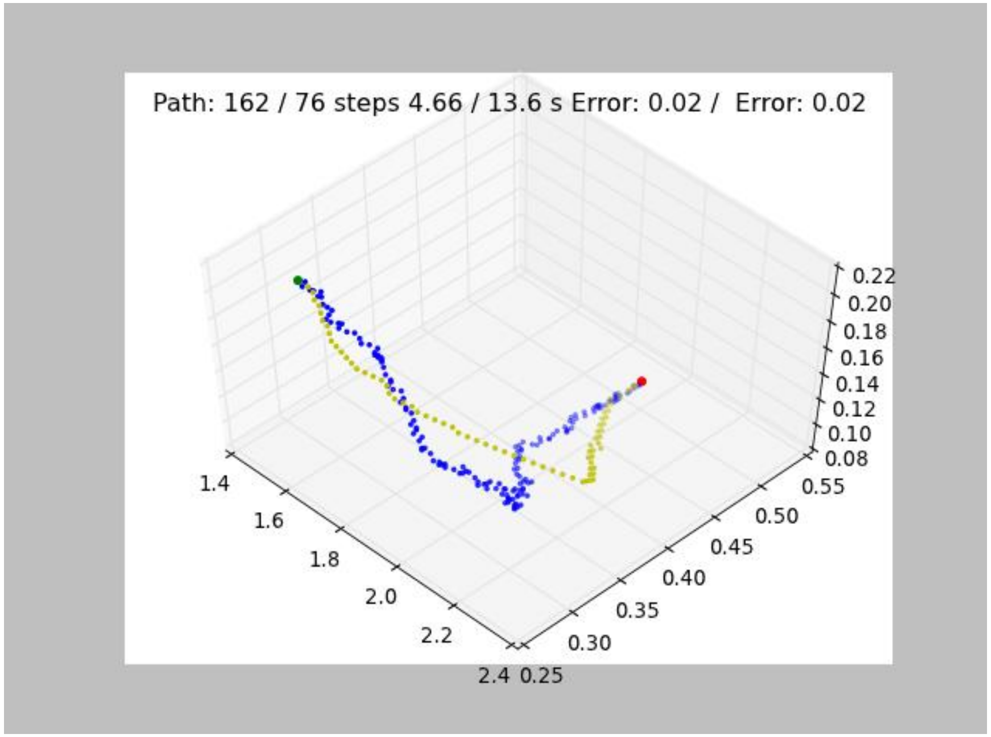
Stochastic Gradient Descent (SGD)

$\delta = 0.005$

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)
$\delta = 0.005$

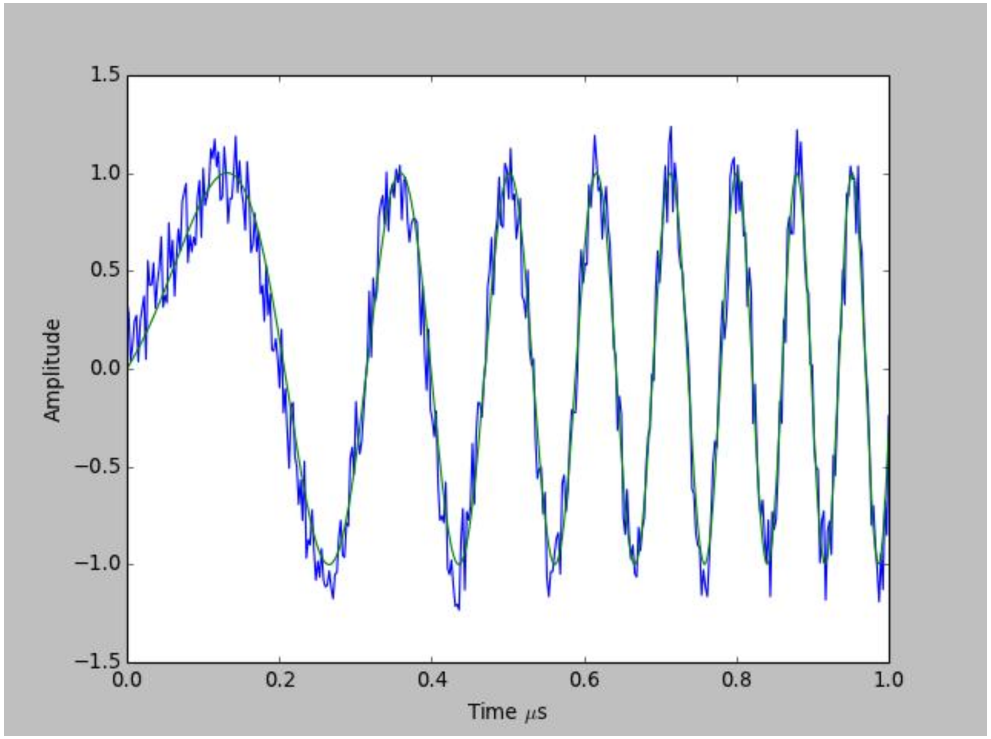
GG.plot_path2(GG2,(0,1),True,pa,pb)

Example 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)
$\delta = 0.005$

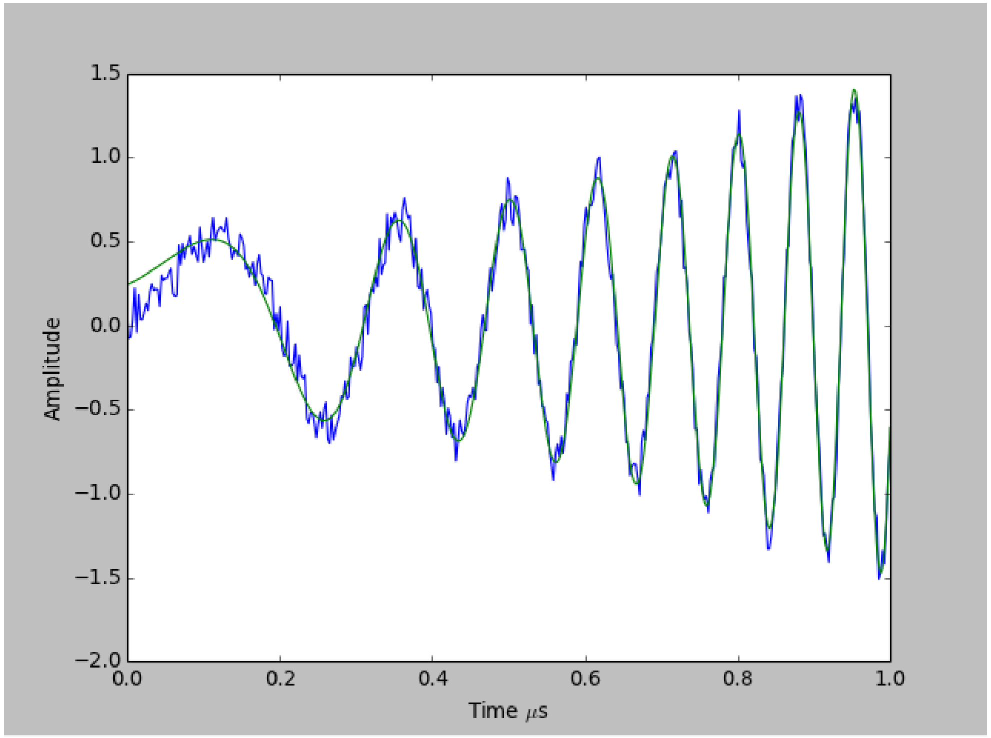


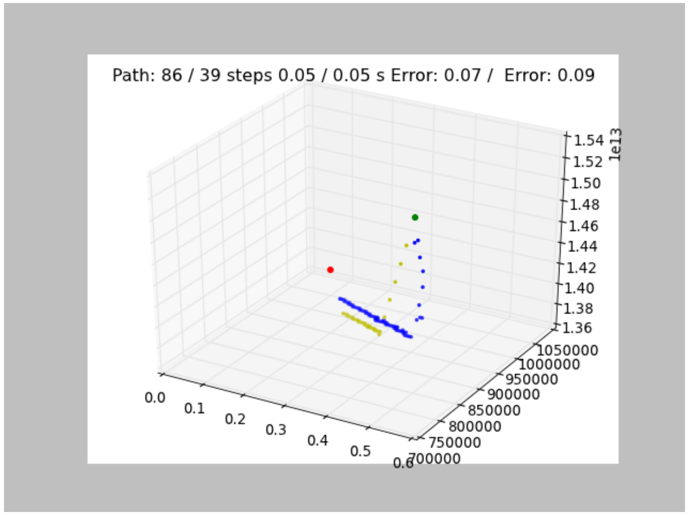
Example 4:
Gained linear chirp

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

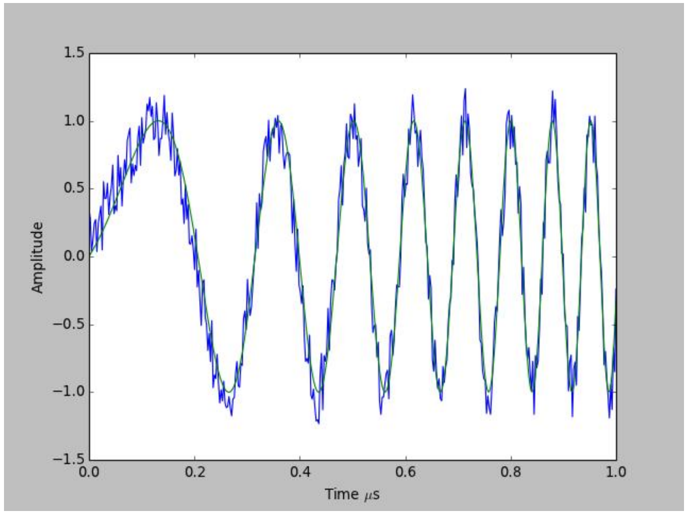
Initial guess error:
$B/T \approx 10\%$
$f_1 \approx 20\%$
$\phi_0 \approx 0.4$ rad

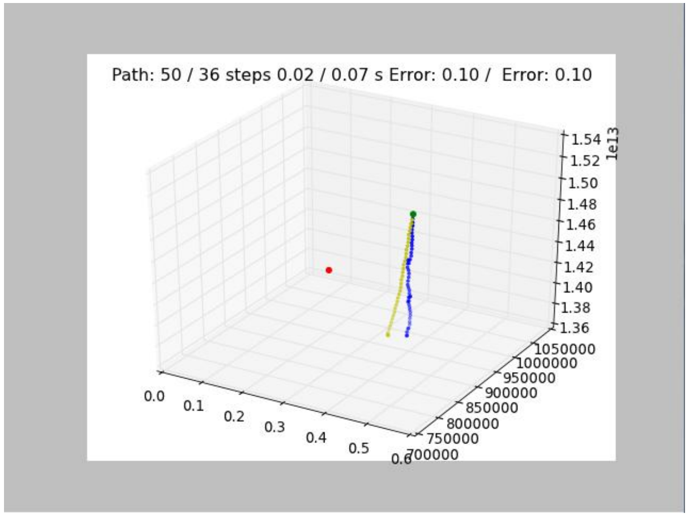
Stochastic Gradient Descent (SGD)
$\delta = 0.005$



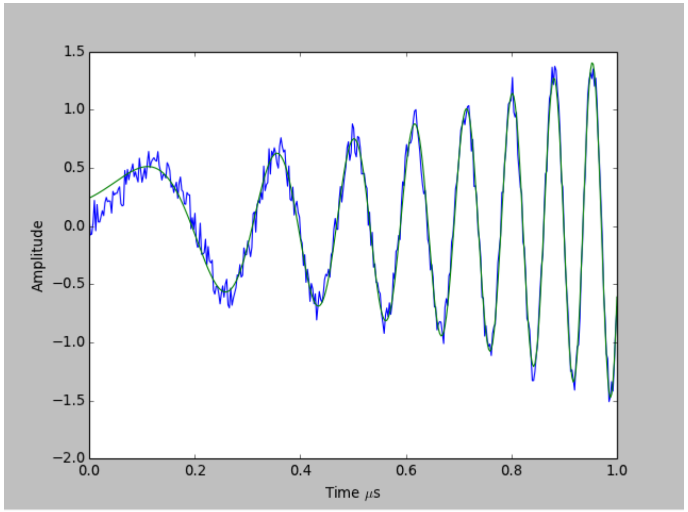


Linear chirp

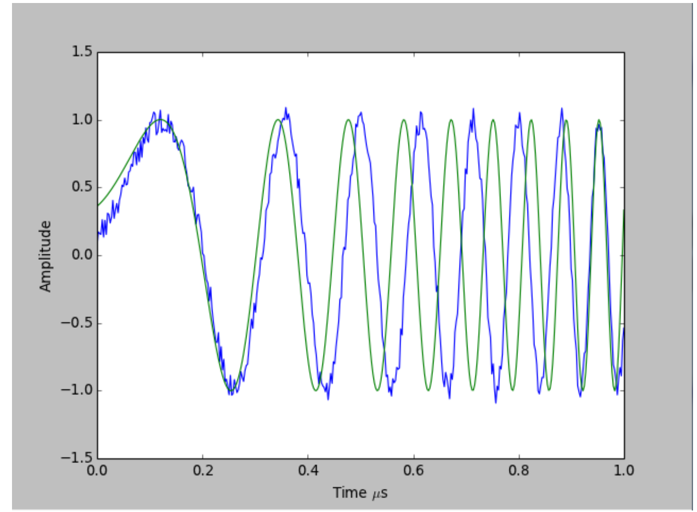




Gained linear chirp



Linear chirp



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

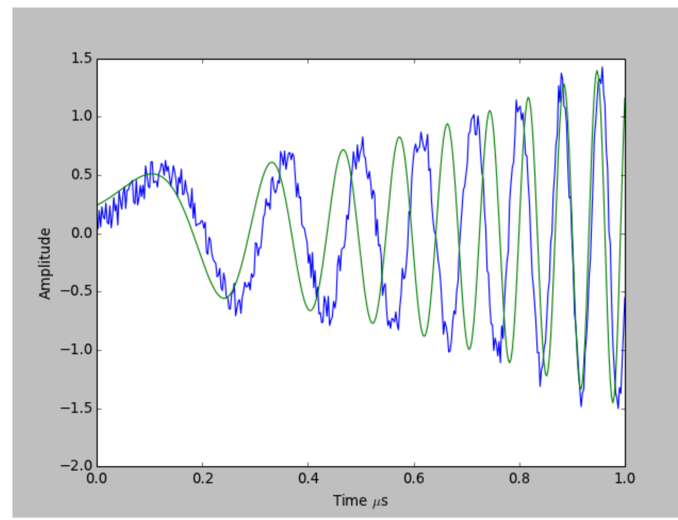
Initial guess error:

B/T ≈ 20% → trapped in loc. min.

f_1 ≈ 20%

$\phi_0 \approx 0.4$ rad

Gained linear chirp



Increase δ: 0.005 → 0.2

Transform: FFT

Time domain → frequency 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

FFT

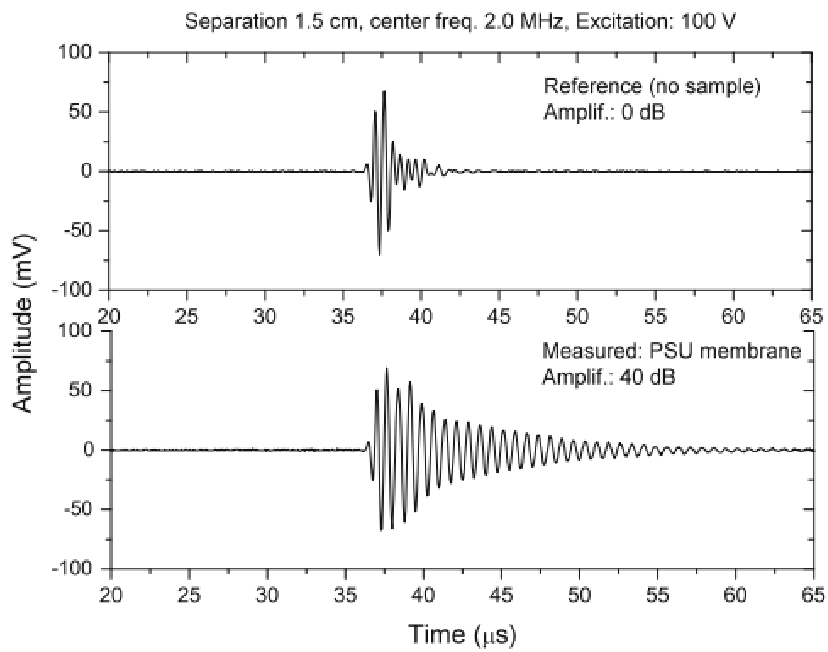
$S(t) = f(\beta, t)$ → $S^*(\omega) = f^*(\beta, \omega)$

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

FFT

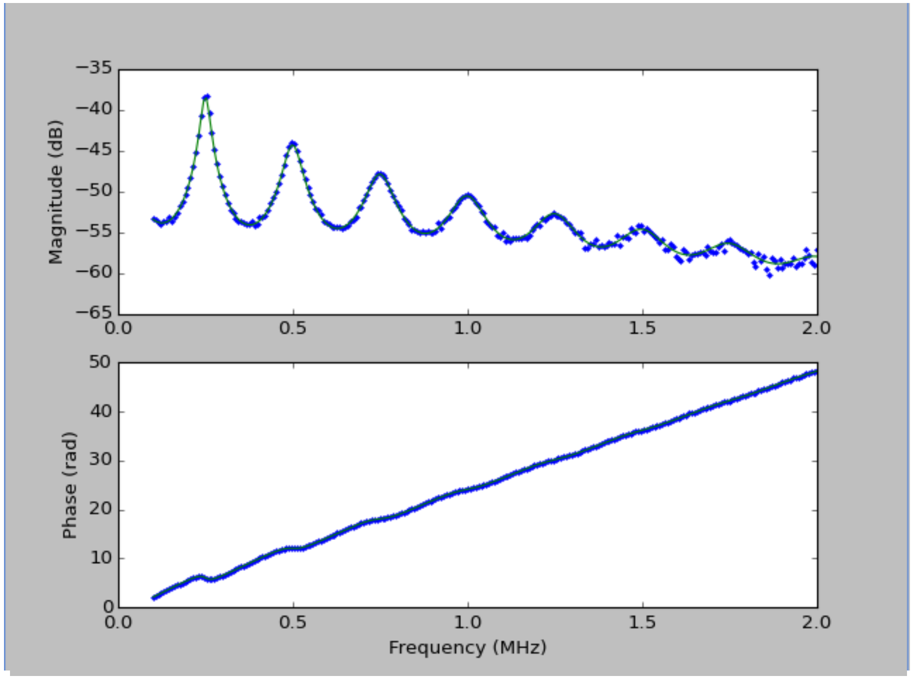
$S(t) = f(\beta,t)$ → $S^*(\omega) = f^*(M,L)$

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

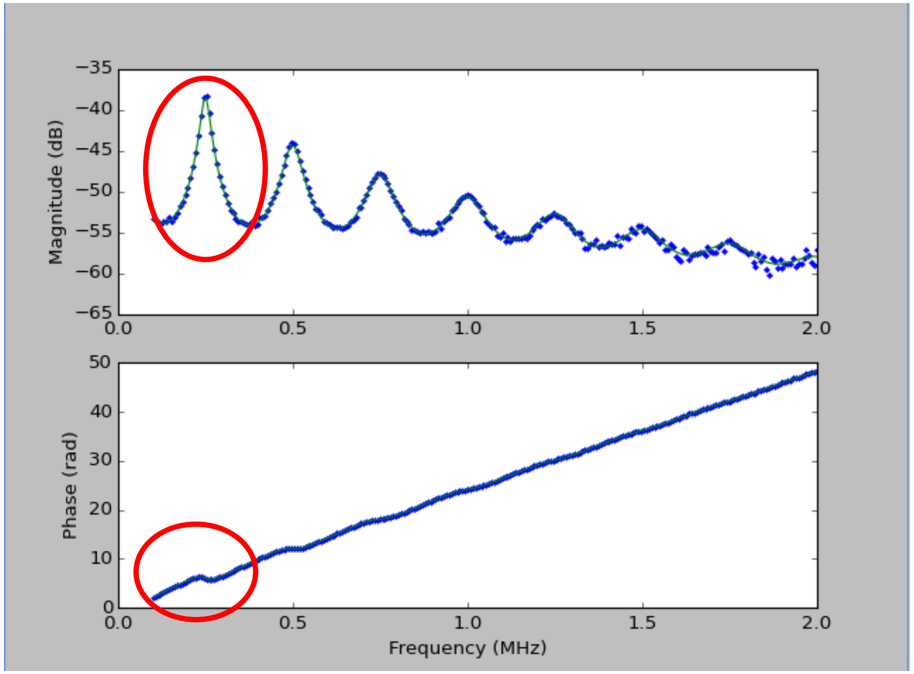
ρ : density

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

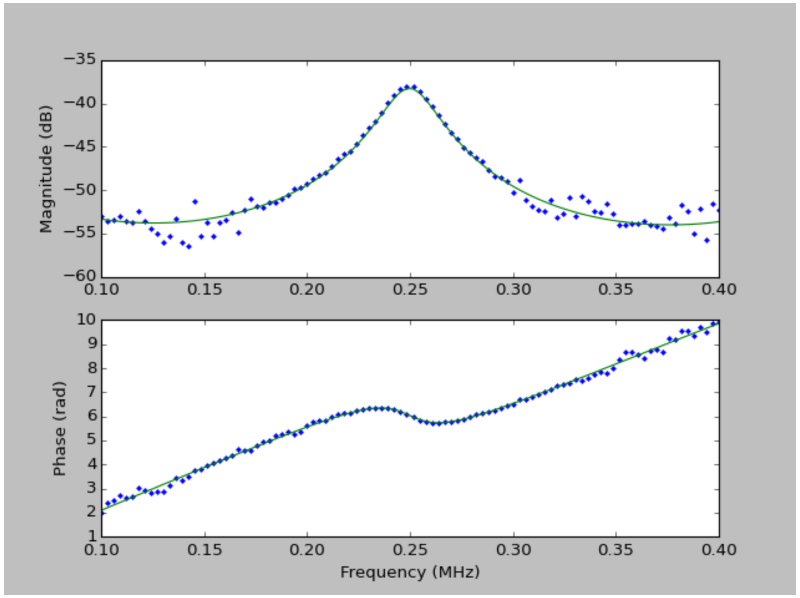
ρ *:* density

v: us velocity

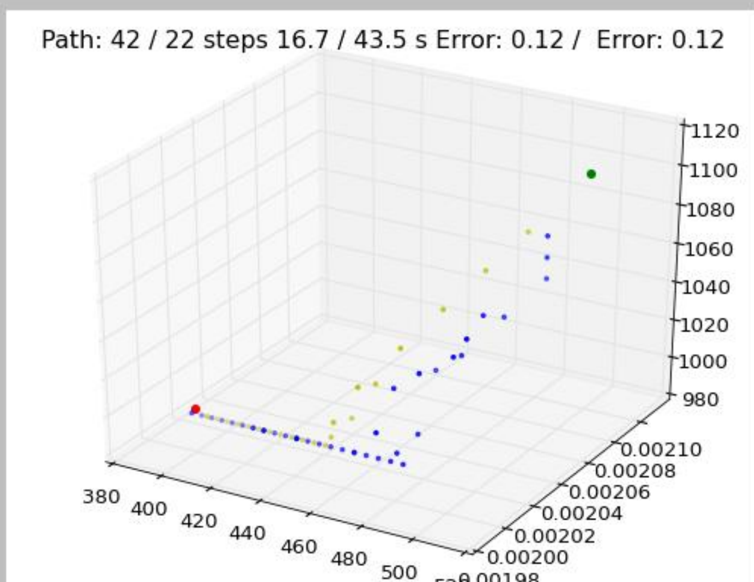
α *:* us attenuation

n: variation of α with freq

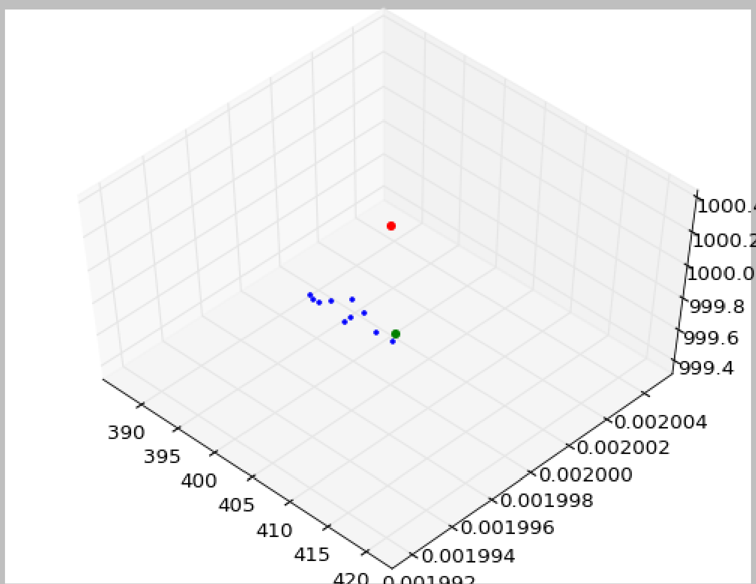
Dimension = 4 (V_{search} = 81 pts.)



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

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*

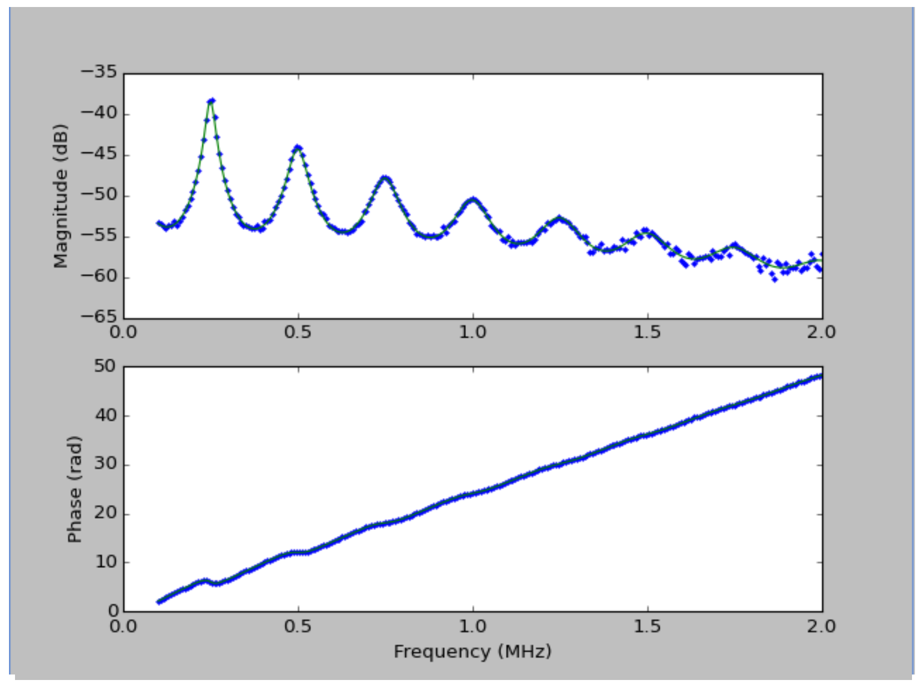
t: thickness

ρ *:* density

v: us velocity

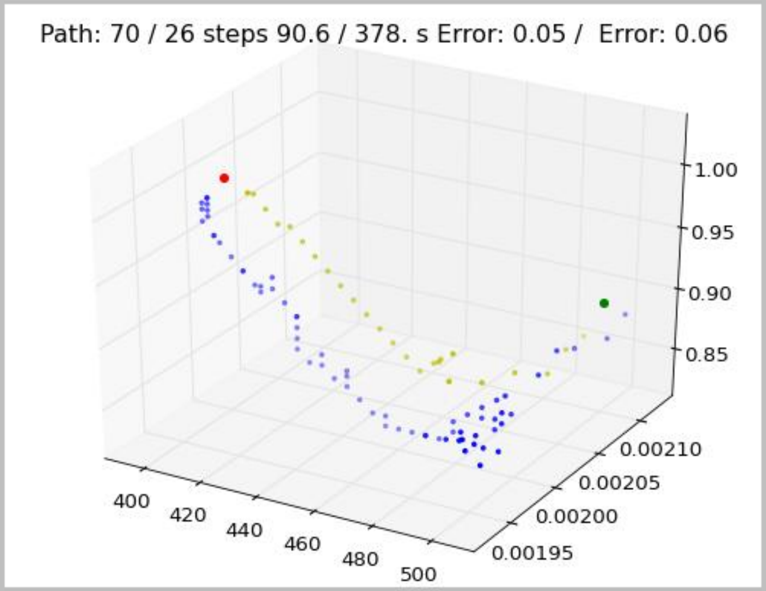
α *:* us attenuation

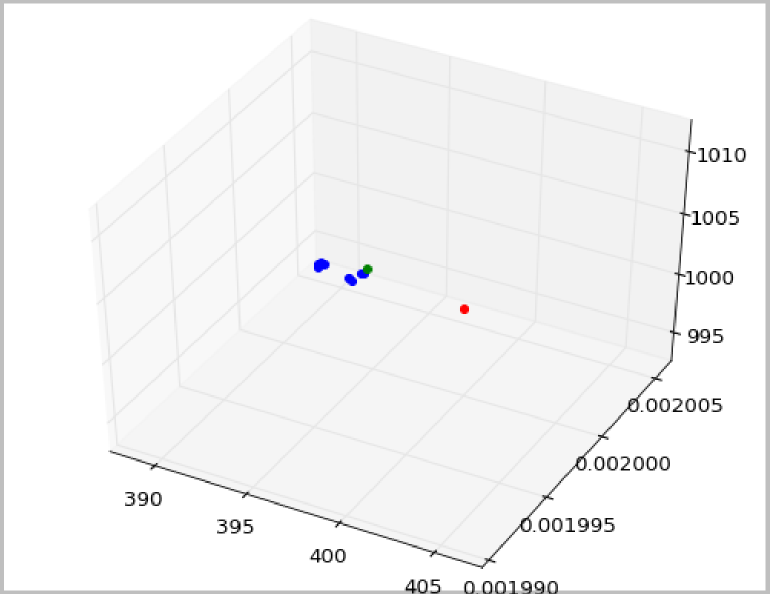
n: variation of α with freq



Dimension = 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):

β^1_i, i = 1…K; 3^K-1

β^2_i, i = 1…L; 3^L-1

M = K + L

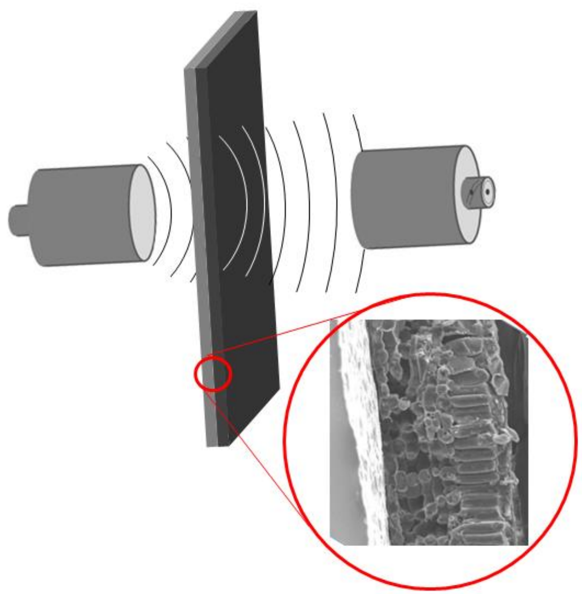
Perform SGDA on β^1_i + early stop

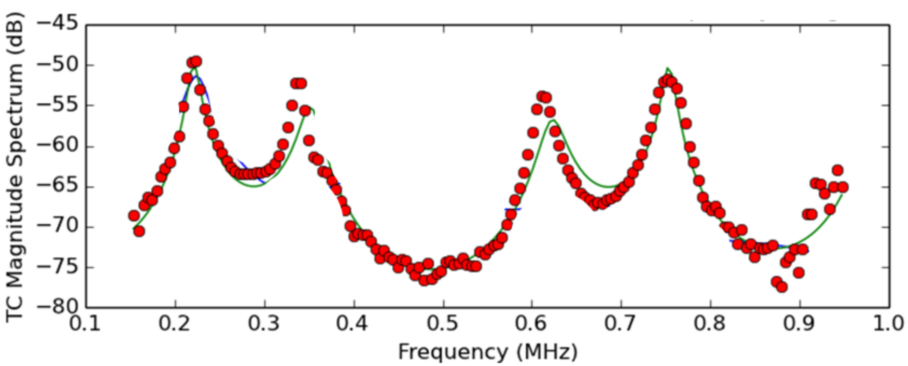
Perform SGDA on β^2_i + early stop

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)

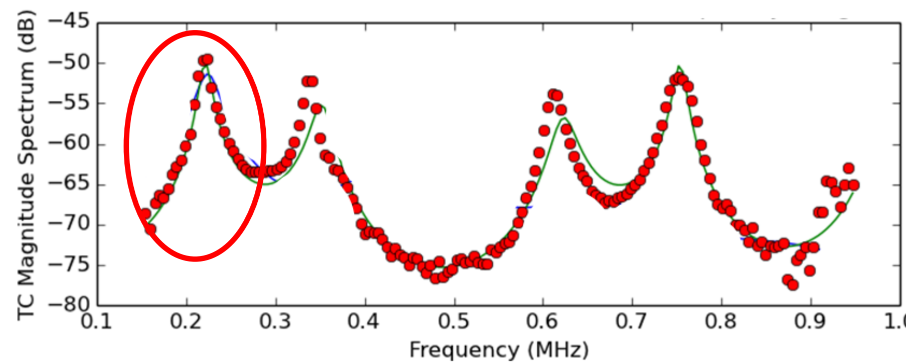
Simplify the model
(one layer: 5 params)

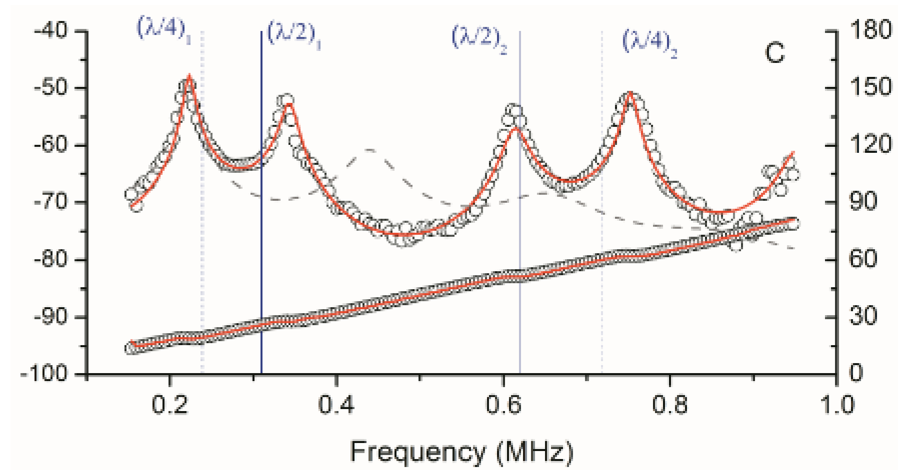
Apply parameters gathering
(3 + 2)

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

Generate IG for whole IP
5 params → 10 params

Solve the whole IP





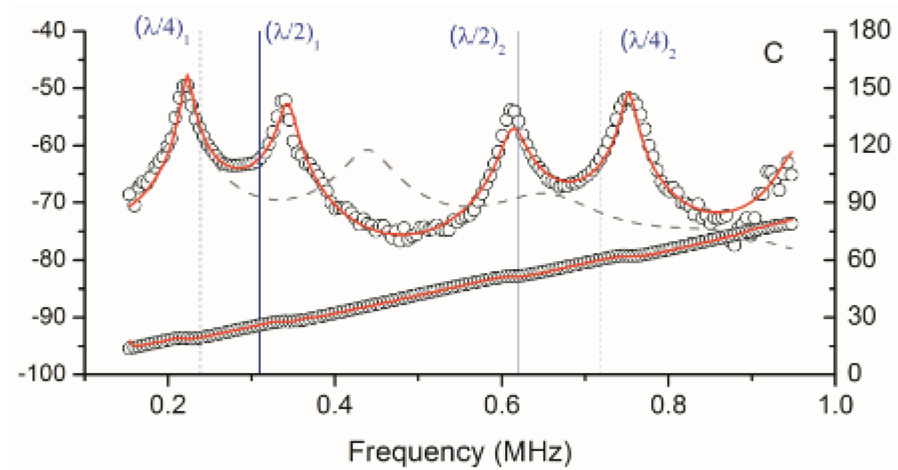
Solve the whole IP

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

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

Q&A

BUT NOT: IP[Q&A] = A&Q→ ILL-POSED

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