Intro to Inverse Problems in Exploration Seismology

M. D. Sacchi
Department of Physics
Institute for Geophysical Research
University of Alberta

PIMS Summer School 06 - Inversion and Imaging

Contact:

M.D.Sacchi

`sacchi@phys.ualberta.ca`

Notes:

`www-geo.phys.ualberta.ca/~sacchi/slides.pdf`

- Inverse Problems in Geophysics
- Reflection Seismology
- Introduction to Inverse Problems
- Inverse Problems in Reflection Seismology

Geophysics

- In Geophysics old and new physical theories are used to understand processes that occur in the Earth's interior (i.e., mantle convection, Geo-dynamo, fluid flow in porous media, etc) and to estimate physical properties that cannot be measured *in situ* (i.e., density, velocity, conductivity, porosity, etc)

What makes Geophysics different from other geosciences?

- Physical properties in the Earth's interior are retrieved from indirect measurements (observations)
- Physical properties are continuously distributed in a 3D volume, observations are sparsely distributed on the surface of the earth.

How do we convert indirect measurements into material properties ?

- Brute force approach
- Inverse Theory
- A combination of Signal Processing and Inversion Techniques

Another aspect that makes Geophysics **unique** is the problem of **non-uniqueness**

Toy Problem:

$$m_1 + m_2 = d$$

given one observation d find m_1 and m_2

Dealing with **non-uniqueness**

Non-uniqueness can be diminished by incorporating into the problem:

- Physical constraints

  Velocities are positives

  Causality

- Non-informative priors

  Bayesian inference, MaxEnt priors

- Results from Lab experiments

- Common sense = Talk to the Geologists [a]

[a] They can understand the complexity involved in defining an Earth model.

Reflection Seismology involves solving problems in the following research areas:

Wave Propagation Phenomena: we deal with waves

Digital Signal Processing (DSP): large amounts of data in digital format, Fast Transforms are needed to map data to different domains

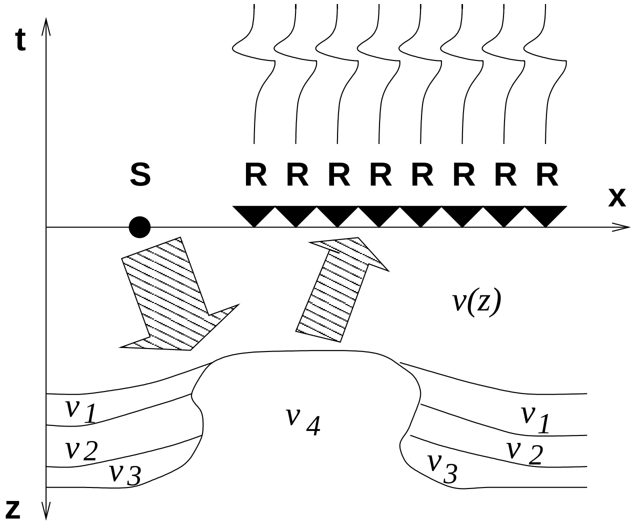
Computational Physics/Applied Math: large systems of equations, FDM, FEM, Monte Carlo methods, etc

High Performance Computing (HPC): large data sets (>Terabytes), large systems of equations need to be solved again and again and again

Inverse Problems: to estimate an Earth model

Statistics: source estimation problems, to deal with signals in low SNR environments, risk analysis, etc

The Seismic Experiment



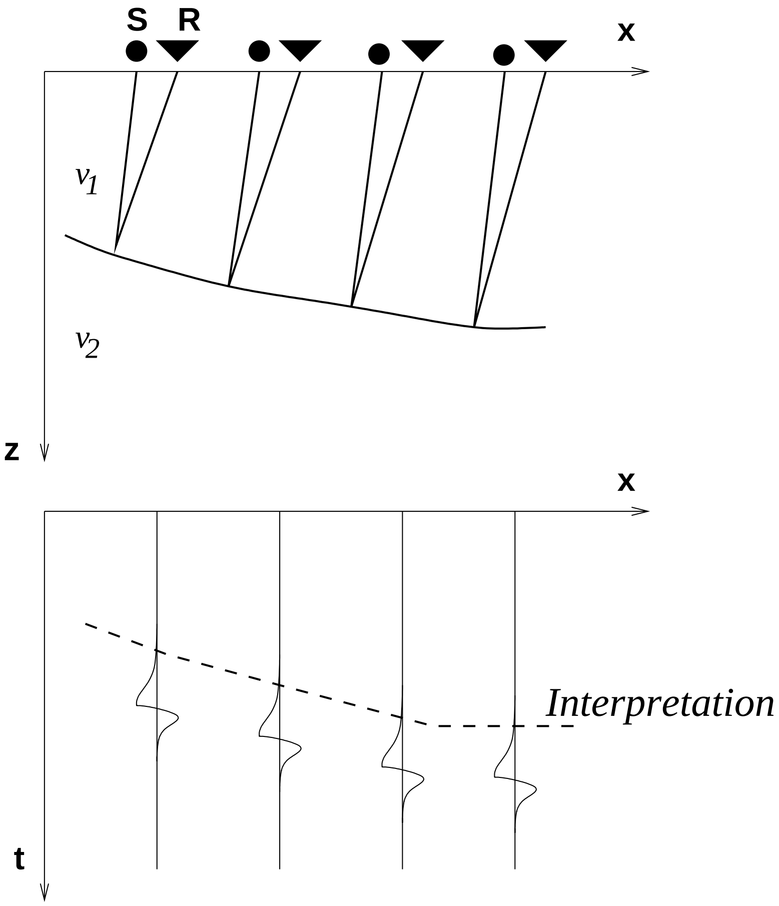
$D(\mathbf{x}_r, \mathbf{x}_s, t)$: Data (acoustic waves)

$v(x, z)$: Unknown media velocity

In general, our unknowns are the position of the boundaries defining different material properties.

The Seismic Experiment :/eps: Command not found.

The Seismic Experiment



Seismic data in $x - t$ provide a distorted image of the subsurface

Seismic Data Processing

Correct irregularities in source-receiver postioning

Compensate amplitude attenuation

Filter deterministic and stochatic noise

Guarantee source-receiver consistency

Reduce the data volume/improve the SNR

Enhance resolution

Provide an “interpretable” image of the subsurface

Reduce the data volume/improve the SNR

Redundancy in the data is exploited to improve the SNR

$$Data(r, s, t) \rightarrow Image(x, z)$$

Forward and Inverse Problems

Forward Problem:

$$\mathcal{F}m = d$$

$\mathcal{F}$: Mathematical description of the physical process under study
m: Distribution of physical properties
d: Observations

In the forward problem m and $\mathcal{F}$ are given and we want to compute d at discrete locations.

In general, $m(x)$ is a continuous distribution of some physical property (velocity is defined everywhere), whereas d is a discrete vector of observations $d_i, i = 1 : N$

Inverse Problem: Given $\mathcal{F}$ and d, we want to estimate the distribution of physical properties m.

In other words:

1 - Given the data

2 - Given the physical equations

3 - Details about the experiment →

What is the distribution of physical properties???

In general, the forward problem can be indicated as follows:

$$\mathcal{F} : M \rightarrow D$$

M: Model Space, a space whose elements consist of all possible functions or vectors which are permitted to serve as possible candidates for a physical model

D: Data Space, a space whose elements consists of vector of observations $d = (d_1, d_2, d_3, \ldots, d_N)^T$

Conversely, the inverse problem entails finding the mapping $\mathcal{F}^{-1}$ such that

$$\mathcal{F}^{-1} : D \rightarrow M$$

Linear and non-linear problems

The function $\mathcal{F}$ can be linear or non-linear.

If $\mathcal{F}$ is linear the following is true

given $m_1 \in M$ and $m_2 \in M$, then

$$\mathcal{L}(\alpha m_1 + \beta m_2) = \alpha \mathcal{L} m_1 + \beta \mathcal{L} m_2$$

for arbitrary constants α and β. Otherwise is non-linear.

Fredholm Integral equations of the first kind

$$d_j = \int_a^b g_j(x)\, m(x)\, dx, \qquad j = 1, 2, 3, \ldots, N$$

N: Number of observations

Example: Deconvolution problem

$$s(t_j) = \int r(\tau) w(t_j - \tau) d\tau$$

$s(t_j)$: observed samples of the seismogram

$r(t)$: earth's impulse response (reflectivity)

$w(t)$: seismic source wavelet

Questions:

Is there a solution ?

How do we construct the solution ?

Is the solution unique ?

Can we characterize the degree non-uniqueness?

To answer the above questions, we first need to consider what kind of data we have at hand:

a- Infinite amount of accurate data

b- Finite number of accurate data

c- Finite number of inaccurate data (the real life scenario!)

Linear Inverse Problems: The Discrete case

We will consider the discrete linear inverse problem with accurate data. Let us assume that we have finite amount of data d_j, $j = 1, \dots N$, the unknown model after discretization using layers or cells is given by the vector m with elements m_i, $i = 1, \dots M$. The data as a function of the model parameters are given by

$$\mathbf{d} = \mathbf{G}\mathbf{m}$$

where **G** is a $N \times M$ matrix that arises after discretizing the Kernel function of the Forward problem.

If $M > N$ many solution can solve the above system of equations.

Minimum norm solution

In this case we will try to retrieve a model with minimum norm. The problem is posed a follows:

Find model $\hat{\mathbf{m}}$ *that minimizes the following cost function*

$$J = \mathbf{m}^T \mathbf{m} \tag{1}$$

subject to data constraints

$$\mathbf{d} = \mathbf{G}\mathbf{m} \tag{2}$$

Each observation provides one constraint, therefore, we have N constraints. The constrained minimization problem is solved using Lagrange multipliers:

In this case we minimized the following objective function

$$J' = \mathbf{m}^T \mathbf{m} + \lambda^T (\mathbf{d} - \mathbf{G}\mathbf{m}) \tag{3}$$

where λ is a vector of Lagrange multipliers λ_i,

$i = 1, \ldots, N$.

Setting the derivative of J' with respect to the unknowns $\mathbf{m}$ and λ to zero gives rise to the following system of equations:

$$\begin{aligned} \mathbf{2m} + \mathbf{G}^T\lambda &= 0 & (4) \\ \mathbf{Gm} - \mathbf{d} &= \mathbf{0}\,. & (5) \end{aligned}$$

You can play for a while with the last two equations to get the minimum norm solution:

$$\hat{\mathbf{m}} = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{d}\,. \qquad (6)$$

Note that we have assume that $\mathbf{GG}^T$ is invertible. In other words $\mathbf{GG}^T$ has a complete set of positive eigenvalues.

Example

Suppose that some data d_j is the result of the following experiment

$$d_j = d(r_j) = \int_0^L e^{-\alpha(x-r_j)^2} m(x) dx \, . \qquad (7)$$

Note that r_j can indicate the position of an observation point (receiver). We can discretize the above expression using the trapezoidal rule:

$$d(r_j) = \sum_{k=0}^{M-1} \Delta x \, e^{-\alpha(r_j-x_k)^2} \;\; m(x_k) \, , \;\; j = 0 \dots N-1 \qquad (8)$$

Suppose that r_j are N observation distributed in $[0, L]$, then

$$r_j = j \, . \, L/(N-1) \, , \;\; j = 0 \dots N-1 \, .$$

The above system can be written down as

$$\mathbf{d} = \mathbf{Gm} \qquad (9)$$

In our example we will assume a true solution given by the profile **m** portrayed in Figure 1a. The corresponding data are displayed in Figure 1b. For this toy problem I have chosen the following parameters:

$$\alpha = 0.8, \quad L = 100., \quad N = 20, \quad M = 50$$

The following script is used to compute the Kernel **G** and generate the observations d:

Script 1: toy1.m

```
M = 50;             % Number of unknowns
m = zeros(M,1);     % True model
m(10:14,1) =  1.;
m(15:26,1) = -.3;
m(27:34,1) = 2.1;
N = 20;             % Observations
L = 100;            % Kernel G
alpha = .8
x = (0:1:M-1)*L/(M-1);
dx = L/(M-1);
r = (0:1:N-1)*L/(N-1);
for j=1:M
for k=1:N
G(k,j) = dx*exp(-alpha*abs(r(k)-x(j))^2);
end
end
d = G*m;            % Compute data
```

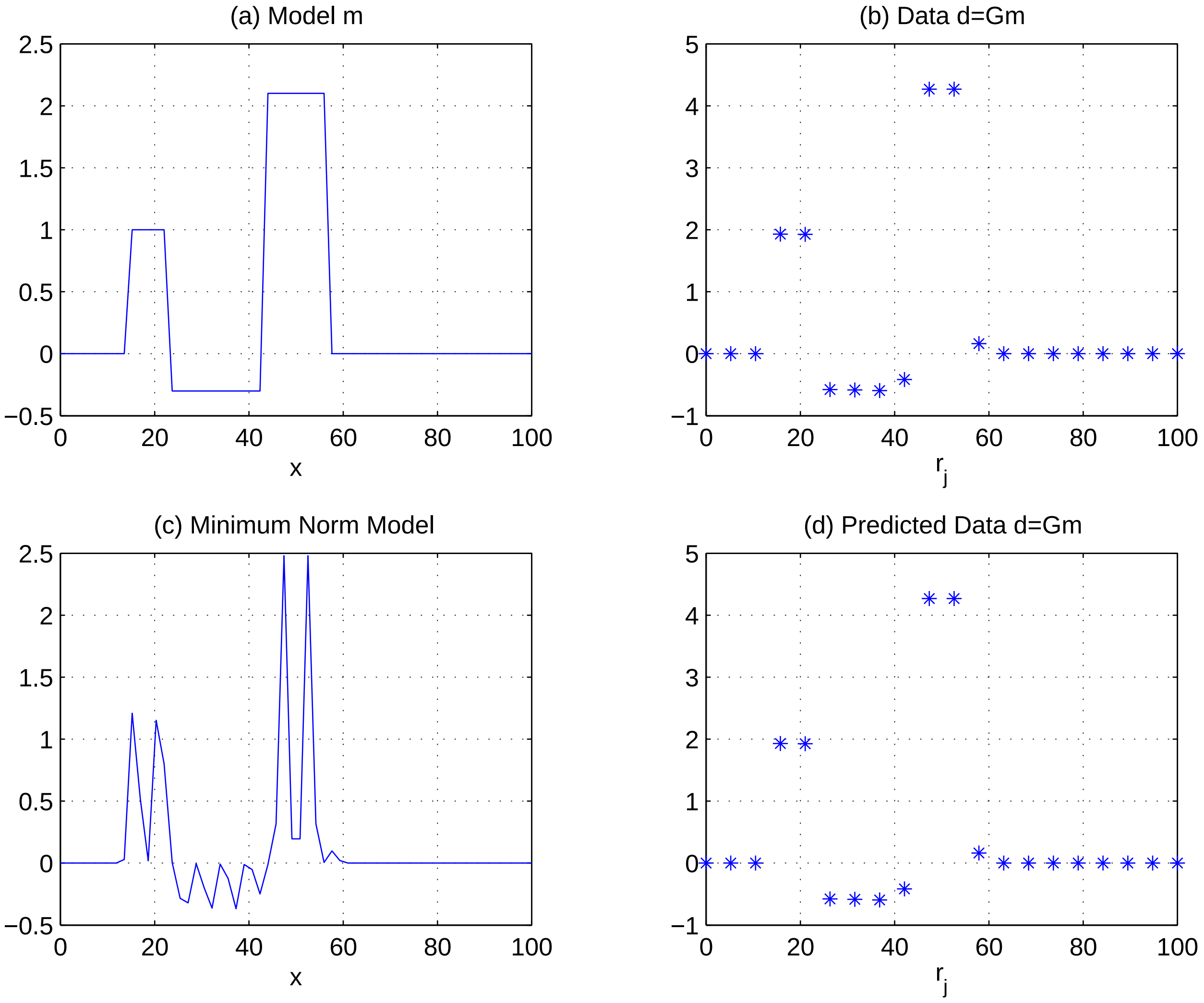


Figure 1: a) Model. d) Data. c) Minimum norm solution. d) Predicted data.

The MATLAB solution for the minimum norm solution is given by:

Script 2: `mns.m`

```
function [m_est,d_pred] = min_norm_sol(G,d);
%
% Given a discrete Kernel G and the data d, computes the
% minimum norm solution of the inverse  problem d = Gm.
% m_est: estimates solution (minimum norm solution)
% d_pred: predicted data

  m_est = G'*inv(G*G')*d;
  d_pred = G* m_est;
```

Let’s see how I made Fig.1 using Matlab

Script 3: makefigs.m

```
% Make figure 1 - 4 figures in the same canvas.
  figure(1); clf;
  subplot(221);
  plot(x,m)    ;title(’(a) Model m’); xlabel(’x’);grid;
  subplot(222);
  plot(r,d,’*’);title(’(b) Data d=Gm’); xlabel(’r_j’);grid
  subplot(223);
  plot(x,m_est);title(’(c) Minimum Norm Model ’);xlabel(’x’);grid;
  subplot(224);
  plot(r,d,’*’);title(’(d) Predicted Data d=Gm’); xlabel(’r_j’);grid;
```

Weighted Minimum norm solution

As we have observed the minimum norm solution is too oscillatory. To alleviate this problem we introduce a weighting function into the model norm. Let us first define a weighted minimum norm solution

$$J = ||\mathbf{W}\,\mathbf{m}||^2 = \mathbf{m}^T\,\mathbf{W}^T\,\mathbf{W}\,\mathbf{m}\,, \qquad (10)$$

this new objective function is minimized subject to data constraints $\mathbf{Gm} = \mathbf{d}$.

$$\hat{\mathbf{m}} = \mathbf{Q}\,\mathbf{G}^T(\mathbf{G}\,\mathbf{Q}\,\mathbf{G}^T)^{-1}\mathbf{d} \qquad (11)$$

where

$$\mathbf{Q} = (\mathbf{W}^T\,\mathbf{W})^{-1} \qquad (12)$$

This solution is called the minimum weighted norm solution.

In Figs. 2 and 3, I computed the minimum norm weighted solution using $\mathbf{W} = \mathbf{D}_1$ and $\mathbf{W} = \mathbf{D}_2$, that is the first derivative operator and the second derivative operator respectively.

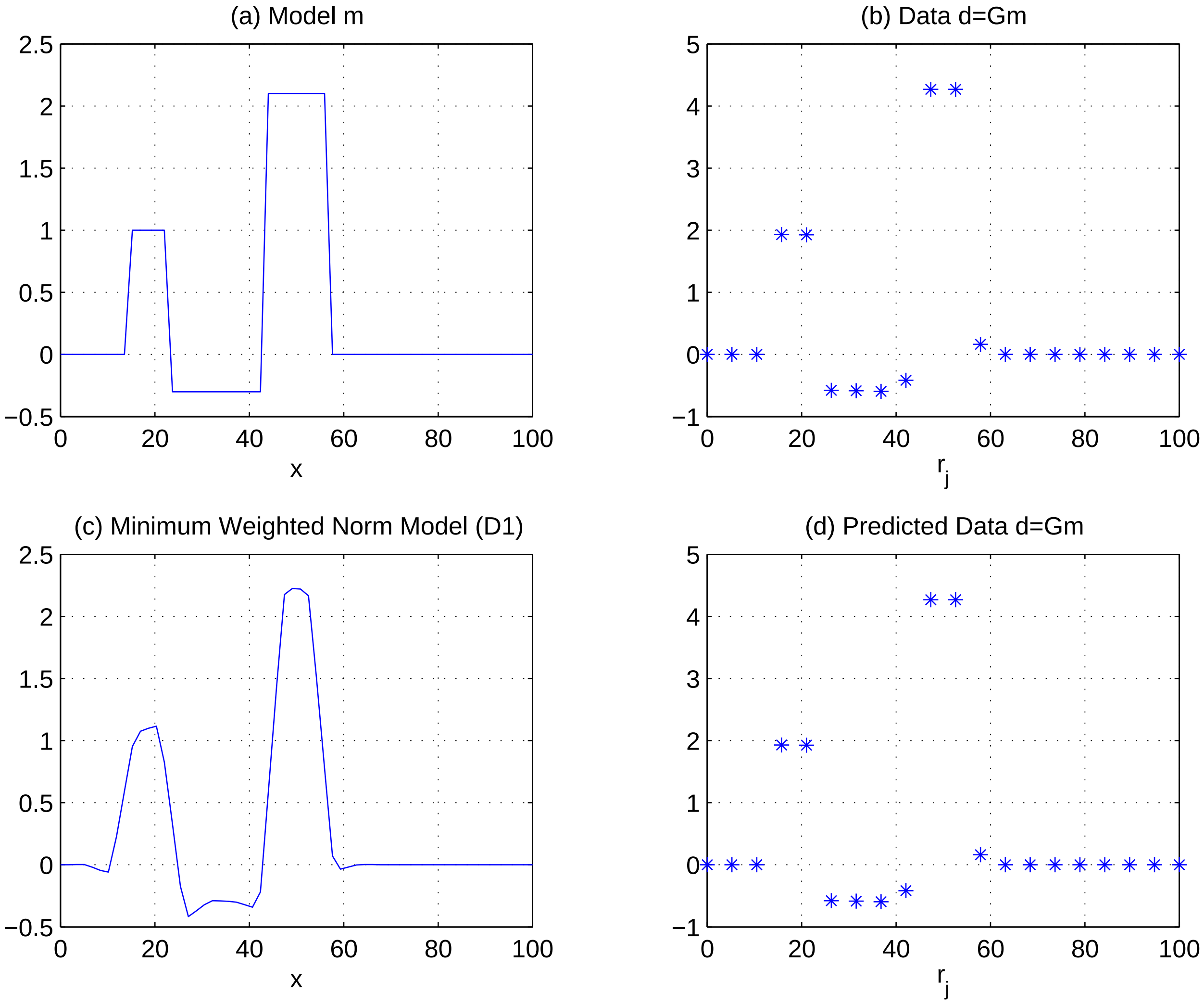


Figure 2: a) True model. b) Data (observations). c) Inversion using minimum weighted norm solution. The smoothing operator is $\mathbf{D}_1$ (First order derivative). d) Predicted data.

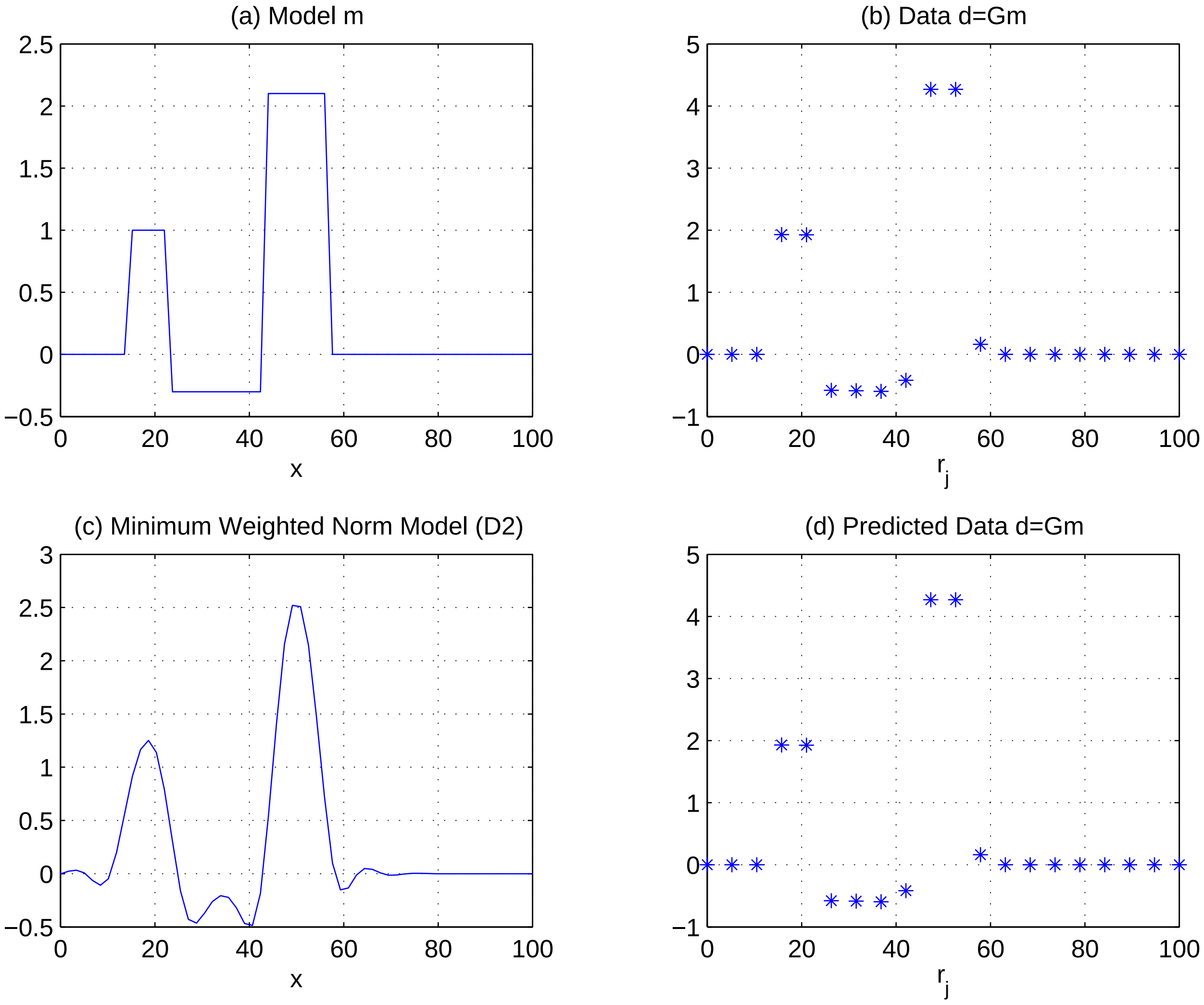


Figure 3: a) True model. b) Data (observations). c) Inversion using minimum weigher norm solution. The smoothing operator is $\mathbf{D}_2$ (Second order derivative). d) Predicted data.

This is the MATLAB script that I used to compute the solution with 1D first order derivative smoothing:

Script 3: `toy1.m`

```
W = convmtx([1,-1],M);              % Trick to compute D1
D1 = W(1:M,1:M);
Q = inv(D1'*D1);                    % Matrix of weights
m_est = Q*G'*inv(G*Q*G')*d;         % Solution
```

Similarly, you can use second order derivatives:

Script 4: `toy1.m`

```
W = convmtx([1,-2,1],M);
D2 = W(1:M,1:M);
Q = inv(D2’*D2);
m_est = Q*G’*inv(G*Q*G’)*d;
```

The Derivative operators $\mathbf{D}_1$ and $\mathbf{D}_2$ behave like high pass filters. If you are not convince compute the amplitude spectrum of a signal/model after and before applying one of these operators.

Question: We want to find smooth solutions then, why are we using High Pass operators?

Linear Inverse Problems: The Discrete case with non-accurate data

In this section we will explore the solution of the discrete inverse problem in situation where the data are contaminated with errors.

Attempting to exactly fit the data is not a good idea. In fact, rather than attempting an exact fit for each observation we will try to fit the data with certain degree of tolerance.

In the previous section we have worked with the Exact Data Problem, and define the minimum norm solution via the following problem:

$$\text{Minimize } J = \mathbf{m}^T\mathbf{m} \;\; [\text{This is the model norm}]$$

$$\text{Subject to } \;\mathbf{Gm} - \mathbf{d} = \mathbf{0}$$

When data are contaminated with errors the exact fitting goal is replaced by

$$\mathbf{Gm} - \mathbf{d} \approx \mathbf{0}$$

A model like the above can also be written as

$$\mathbf{Gm} = \mathbf{d} + \mathbf{e}$$

where **e** is used to indicate the error or noise vector. This quantity is unknown.

The minimum norm solution in this case becomes the solution of the following optimization problem:

$$\text{Minimize } J = \mathbf{m}^T \mathbf{m}$$

$$\text{Subject to } \; ||\mathbf{G}\mathbf{m} - \mathbf{d}||_2^2 = \epsilon$$

It is clear that now rather than having one constraint per observation we have a single global constraint for the complete set of observations.

Please, notice that in the previous equation we have used the l_2 norm as a measure of distance for the errors in the data; we will see that this also implies that the errors are considered to be distributed according to the normal law (Gaussian errors). Before continuing with the analysis, we recall that

$$||\mathbf{Gm} - \mathbf{d}||_2^2 = ||\mathbf{e}||_2^2$$

which in matrix/vector notation can be also expressed as

$$||\mathbf{e}||_2^2 = \mathbf{e}^T\mathbf{e}\,.$$

Coming back to our optimization problem, we now minimize the cost function J' given by

$$\begin{aligned} J' &= \mu \text{Model Norm} + \text{Misfit} \\ &= \mu \mathbf{m}^T\mathbf{m} + \mathbf{e}^T\mathbf{e} \\ &= \mu \mathbf{m}^T\mathbf{m} + (\mathbf{Gm} - \mathbf{d})^T(\mathbf{Gm} - \mathbf{d}) \end{aligned}$$

The solution is now obtained by minimizing J' with respect to the unknown $\mathbf{m}$. This requires some algebra and I will give you the final solution:

$$\begin{aligned} \tfrac{d\,J'}{d\mathbf{m}} &= 0 \\ &= (\mathbf{G}^T\mathbf{G}+\mu\mathbf{I})\mathbf{m}-\mathbf{G}^T\mathbf{d}=\mathbf{0}\,. \end{aligned}$$

The minimizer is then given by

$$\mathbf{m}=(\mathbf{G}^T\mathbf{G}+\mu\mathbf{I})^{-1}\mathbf{G}^T\mathbf{d}\,. \tag{13}$$

This solution is often called the *damped least squares solution.* Notice that the structure of the solution looks like the solution we obtain when we solve a least squares problem. A simple identity permits one to make equation (13) look like a minimum norm solution:

Identity $\quad (\mathbf{G}^T\mathbf{G}+\mathbf{I})^{-1}\mathbf{G}^T = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T+\mathbf{I})^{-1}\,.$

Therefore, equation (13) can be re-expressed as

$$\mathbf{m} = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T+\mu\mathbf{I})^{-1}\mathbf{d}\,. \tag{14}$$

It is important to note that the previous expression reduces to the minimum norm solution for exact data when $\mu = 0$.

About μ

The importance of μ can be seen from the cost function J'

$$J' = \mu \text{Model Norm} + \text{Misfit}$$

- Large μ means more weight (importance) is given to minimizing the misfit over the model norm.
- Small μ means that the model norm is the main term entering in the minimization; the misfit becomes less important.
- You can think that we are trying to simultaneously achieve two *goals*:

  Norm Reduction (Stability - we don't want higly oscillatory solutions)

  Misfit Reduction (We want to honor our observations)

We will explore the fact that these two goals cannot be simultaneously achieved, and, this is why we often call μ a trade-off parameter.

The parameter μ receives different names according to the scientific background of the user:

1. Statisticians: Hyper-parameter
2. Mathematicians: Regularization parameter, Penalty parameter
3. Engineers: Damping term, Damping factor, Stabilization parameter
4. Signal Processing: Ridge regression parameter, Trade-off parameter

Trade-off diagrams

A trade-off diagram is a display of the model norm versus the misfit for different values of μ.

Say we have computed a solution for a value μ this gives the model estimate:

$$\mathbf{m}(\mu) = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})^{-1}\mathbf{G}^T\mathbf{d}$$

This solution can be used to compute the model norm and the misfit:

$$\text{Model Norm } J(\mu) = \mathbf{m}(\mu)^T\mathbf{m}(\mu)$$

$$\text{Misfit}(\mu) = (\mathbf{Gm}(\mu) - \mathbf{d})^T(\mathbf{Gm}(\mu) - \mathbf{d})$$

by varying μ, we get a curve of $J(\mu)$ versus Misfit(μ).

Example

We use the toy example given by `Script 1` to examine the influence of the trade-off parameter in the solution of our inverse problem. But first, I will add noise to the synthetic data generated by script (`toy.m`). I basically add a couple of lines to the code:

Script 5

```
% Add noise to synthetic data
%
 dc = G*m;                      % Compute clean data
 s = 0.1;                       % Standart error of the noise
 d = dc + s*randn(size(dc)); % Add noise
```

The Least Squares Minimum Norm solution is obtained with the following script

Script 6

```
% LS Min Norm Solution
%
I = eye(N);
m_est = G'*((G*G'+mu*I)\d);
```

Don't forget our identity; the solution can also be written in the following form:

Script 6'

```
% Damped LS Solution
%
I = eye(M);
m_est = (G'*G+mu*I)\ (G'*d);
```

Q - Script 6 or Script 6'?? Which one would you use??

Inversion of the noisy data using the least squares minimum norm solution for $\mu = 0.05$ and $\mu = 5$. are portrayed in Figures 4 and 5, respectively. Notice, that a small value of μ leads to data over-fitting. In other words, we are attempting to reproduce the noisy observations like if they were accurate observations. This is not a good idea since over-fitting can force the creation of non-physical features in the solution (highly oscillatory solutions).

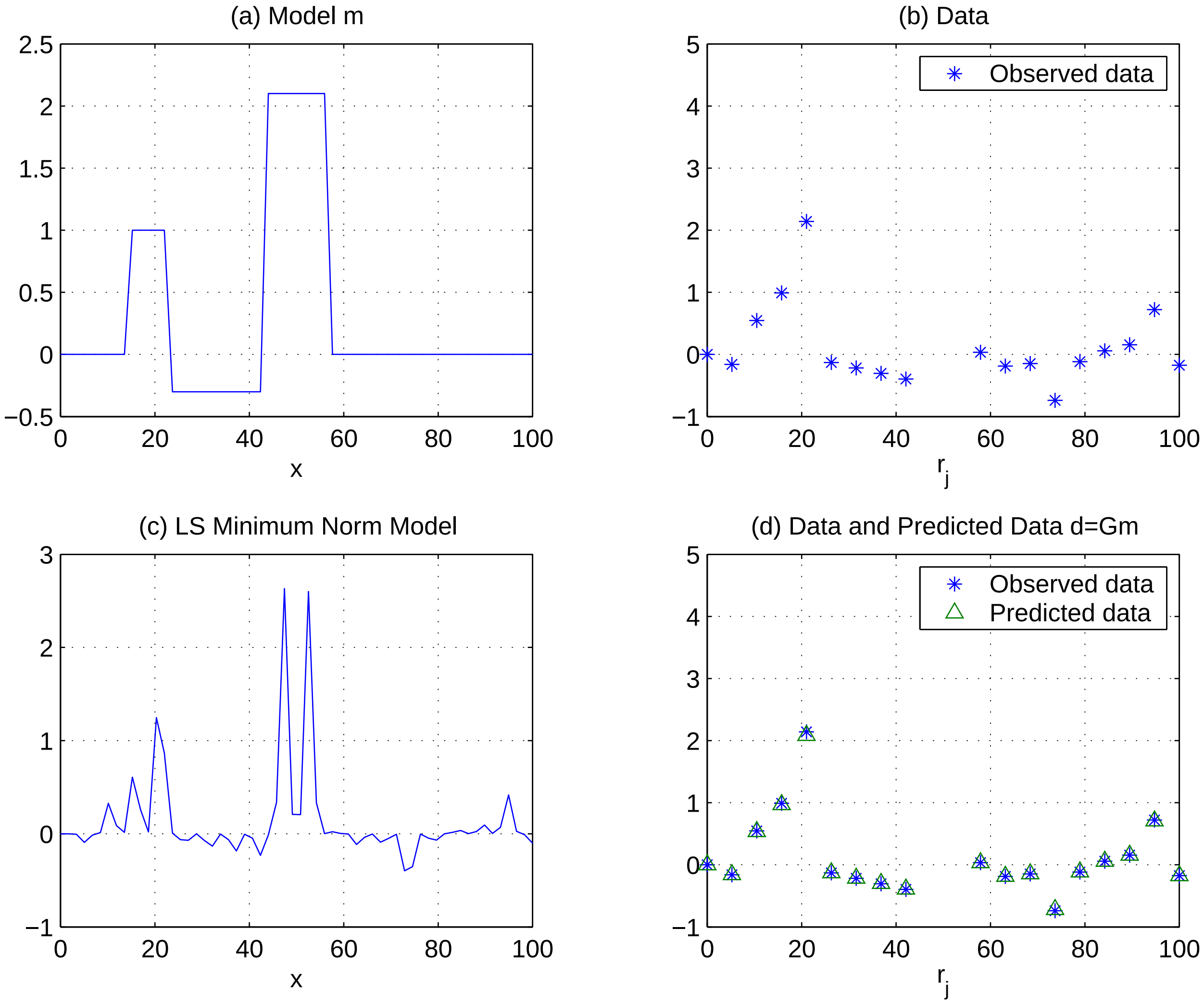


Figure 4: a) Model. d) Data. c) LS Minimum norm solution. d) Predicted data and observed data. In this example I used a small trade-off parameter $\mu = 0.005$. Notice that the predicted data reproduce quite well the noisy observations. Not a good idea since you do not want to fit the noise.

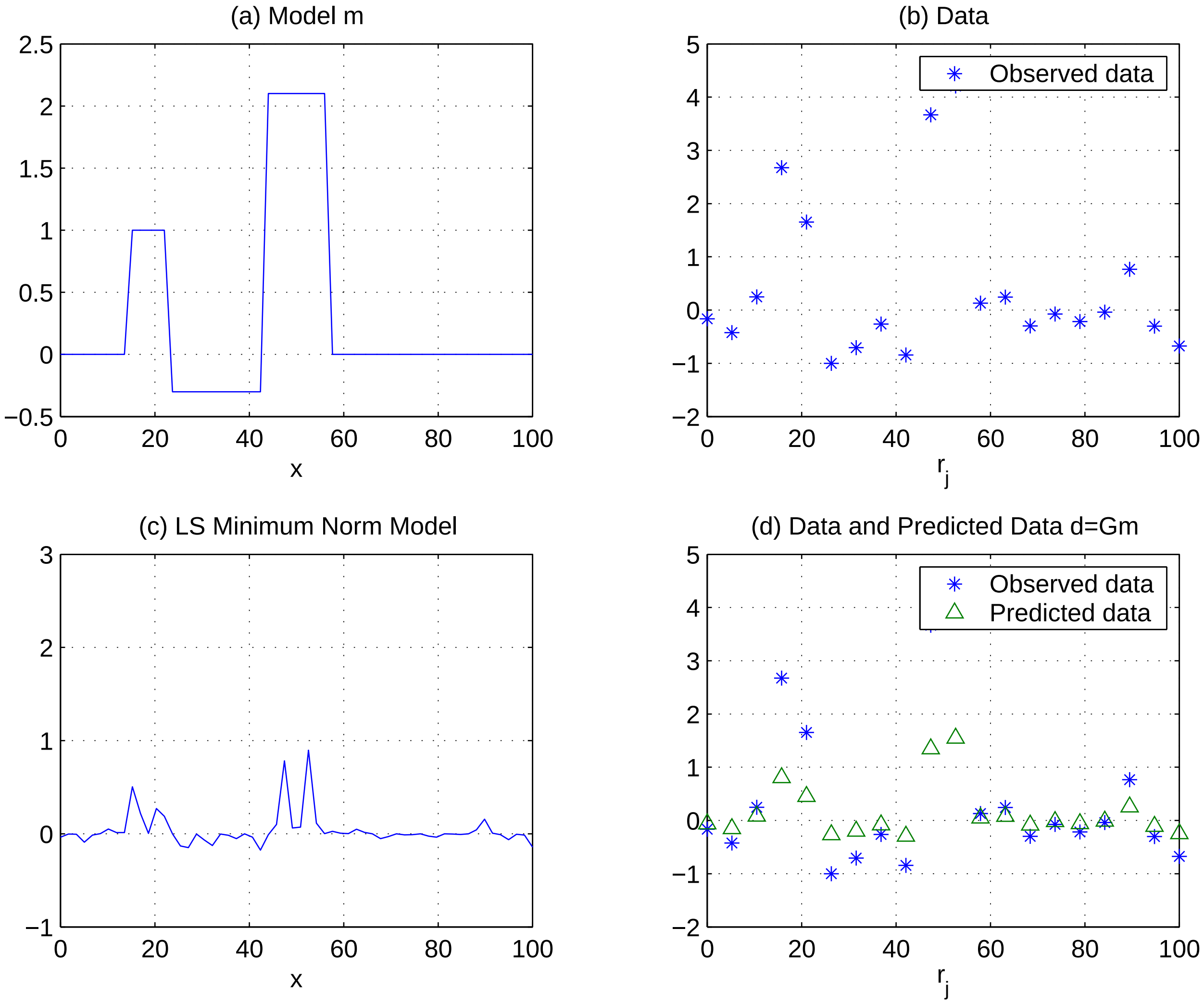


Figure 5: a) Model. d) Data. c) LS Minimum norm solution. d) Predicted data and observed data. In this example I used a small trade-off parameter $\mu = 5$. Notice that the predicted data does not try reproduce the noisy observations.

Script 7: trade.m

```
% Compute trade-off curves
% mu is varied in log scale
%
mu_min= log10(0.5);
mu_max = log10(100);
Nmu = 11;

for k=1:Nmu;
 mu(k) = mu_min + (mu_max-mu_min)*(k-1)/(Nmu-1);
 mu(k) = 10^mu(k);
 m_est = G’*((G*G’+mu(k)*I)\d); % Compute min norm LS solution
 dp = G*m_est;                  % Compute estimated data
 Misfit(k) = (dp-d)’*(dp-d);
 Model_Norm(k) = m_est’*m_est;
end;

%Fancy plot
 figure(1); clf;
 plot(Model_Norm,Misfit,’*’); hold on;
 plot(Model_Norm,Misfit    );

 for k=1:Nmu
  say = strcat(’   \mu=’,num2str(mu(k)));
  text(Model_Norm(k),Misfit(k),say);
 end;
 xlabel(’Model Norm’)
 ylabel(’Misfit’); grid
```

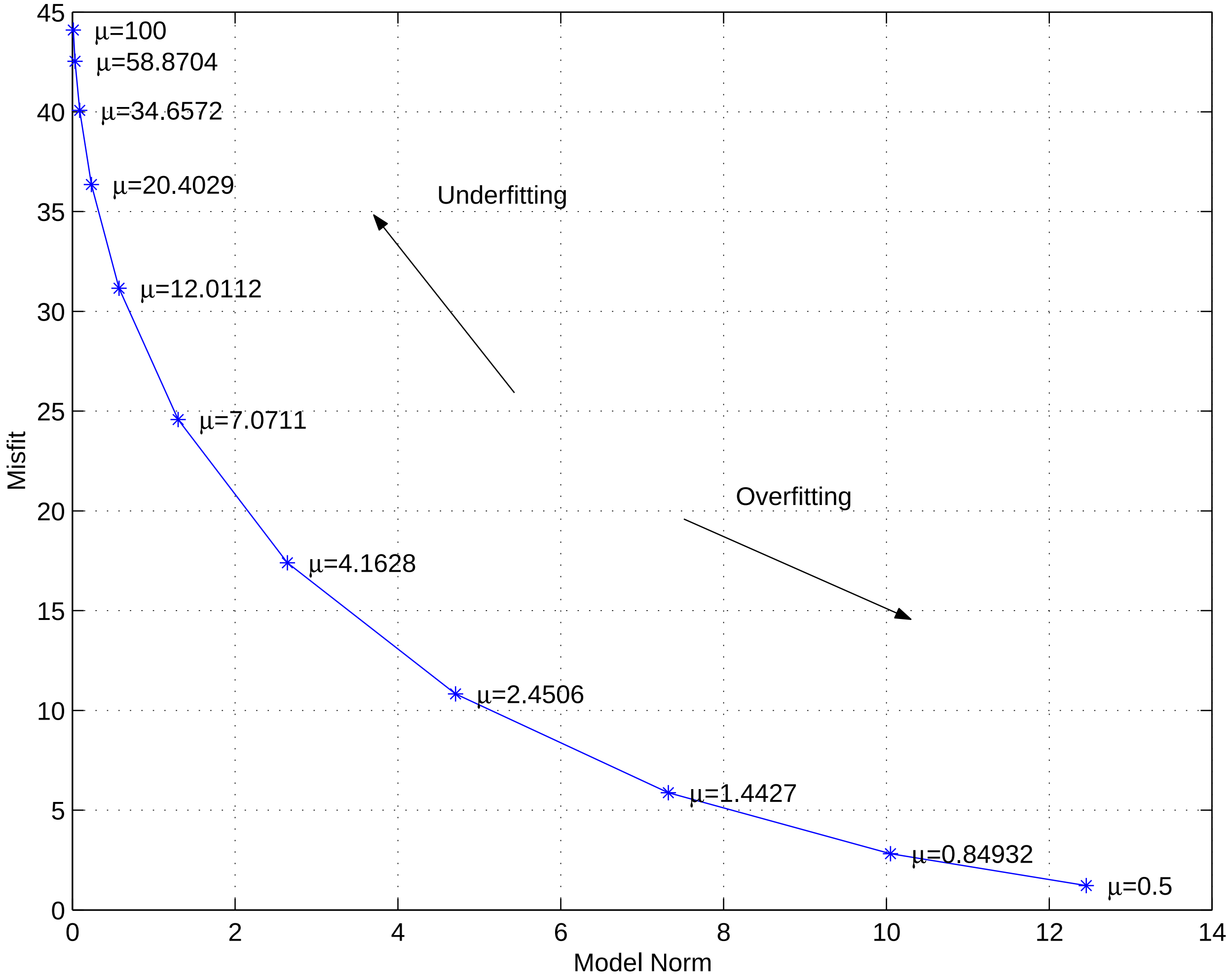


Figure 6: Trade-off diagram for our toy example. Which value of μ would you pick?

Smoothing in the presence of noise

In this case we minimize:

$$\begin{aligned} J' &= \mu \text{Model Norm} + \text{Misfit} \\ &= \mu(\mathbf{Wm})^T(\mathbf{Wm}) + \mathbf{e}^T\mathbf{e} \\ &= \mu\mathbf{m}\mathbf{W}^T\mathbf{W}\mathbf{m} + (\mathbf{Gm}-\mathbf{d})^T(\mathbf{Gm}-\mathbf{d}) \end{aligned}$$

where **W** can be a matrix of first or second order derivatives as shown before. The minimization of J' leads to a least squares weighted minimum norm solution

$$\begin{aligned} \frac{d\,J'}{d\mathbf{m}} &= 0 \\ &= (\mathbf{G}^T\mathbf{G} + \mu\mathbf{W}^T\mathbf{W})\mathbf{m} - \mathbf{G}^T\mathbf{d} = \mathbf{0}\,. \end{aligned}$$

or,

$$\mathbf{m} = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{W}^T\mathbf{W})^{-1}\mathbf{G}^T\mathbf{d}$$

Exercise: Try to rewrite last equation in a way that looks like a minimum norm solution, this is a solution that contains the operator $\mathbf{G}\mathbf{G}^T$.

Script 8: mwnls.m

```
% Trade-off parameter
%
 mu = 5.;
%
% First derivative operator
%
 W = convmtx([1,-1],M);
 W1 = W(1:M,1:M);
 m_est = (G'*G+mu*W1'*W1)\(G'*d);
 dp = G*m_est;
 Model_Norm = m_est'*m_est;
 Misfit = (d-dp)'*(d-dp);
```

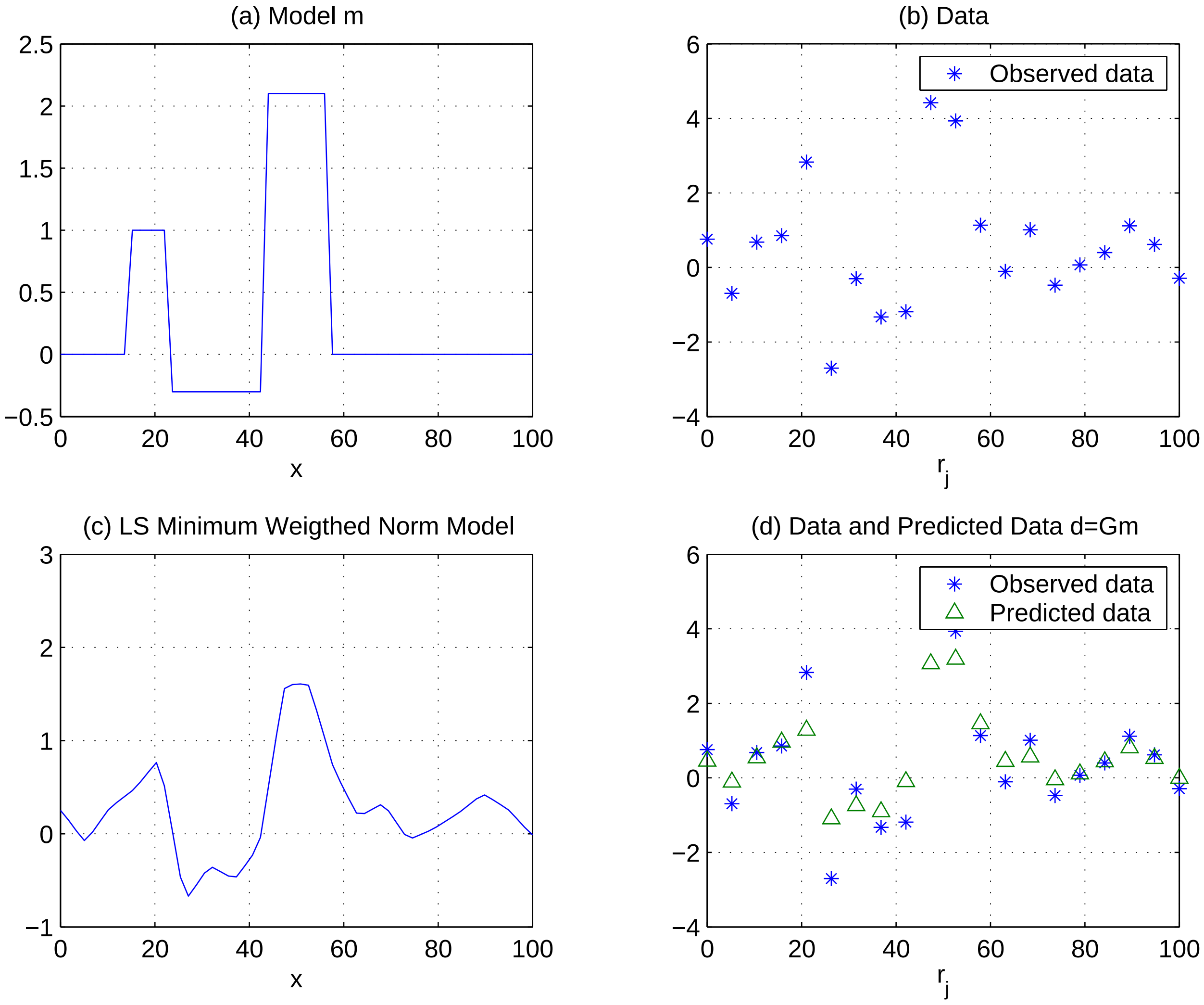


Figure 7: a) Model. d) Data. c) LS Minimum Weighted norm solution. d) Predicted data and observed data. In this example I used a trade-off parameter $\mu = 5$. Notice that the predicted data does not try reproduce the noisy observations. In this example the model norm is $||\mathbf{Wm}||_2^2$ where $\mathbf{W}$ is the first order derivative operator $\mathbf{D}_1$. This example was computed with **Script 8**

Edge Preserving Regularization (EPR)

Smoothing tends to blurr edges. How can we preserve edges in our inverted models ???

Minimize

$$J_{EP} = ||\mathbf{G}\mathbf{m} - \mathbf{d}||^2 + \mu \Phi(\mathbf{m})$$

where J is an edge preserving potential of the form

$$\Phi(\mathbf{m}) = \sum_i \ln(1 + (\mathbf{D}_1 \mathbf{m}/\delta)_i^2)$$

Notice that now the regularization term (Φ) is non-quadratic.

The EP solution involves an iterative solution of the following system:

$$\mathbf{m}^k = (\mathbf{G}^T\mathbf{G} + \mu \mathbf{Q}^{k-1})^{-1}\,\mathbf{G}^T\mathbf{d}$$

where **Q** is a diagonal matrix that depends on **m**. k indicates iteration number.

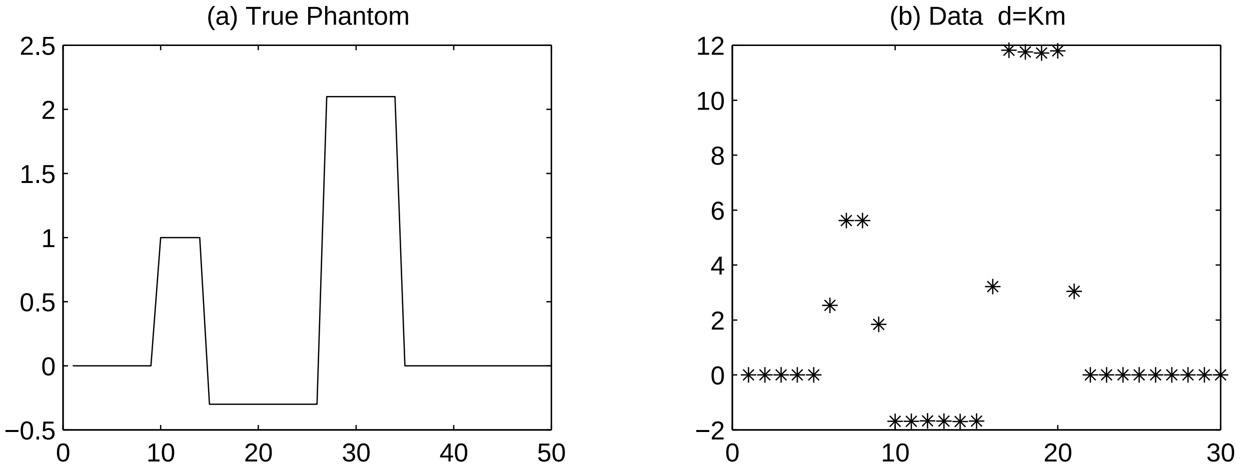


Figure 8: a) Model. d) Data.

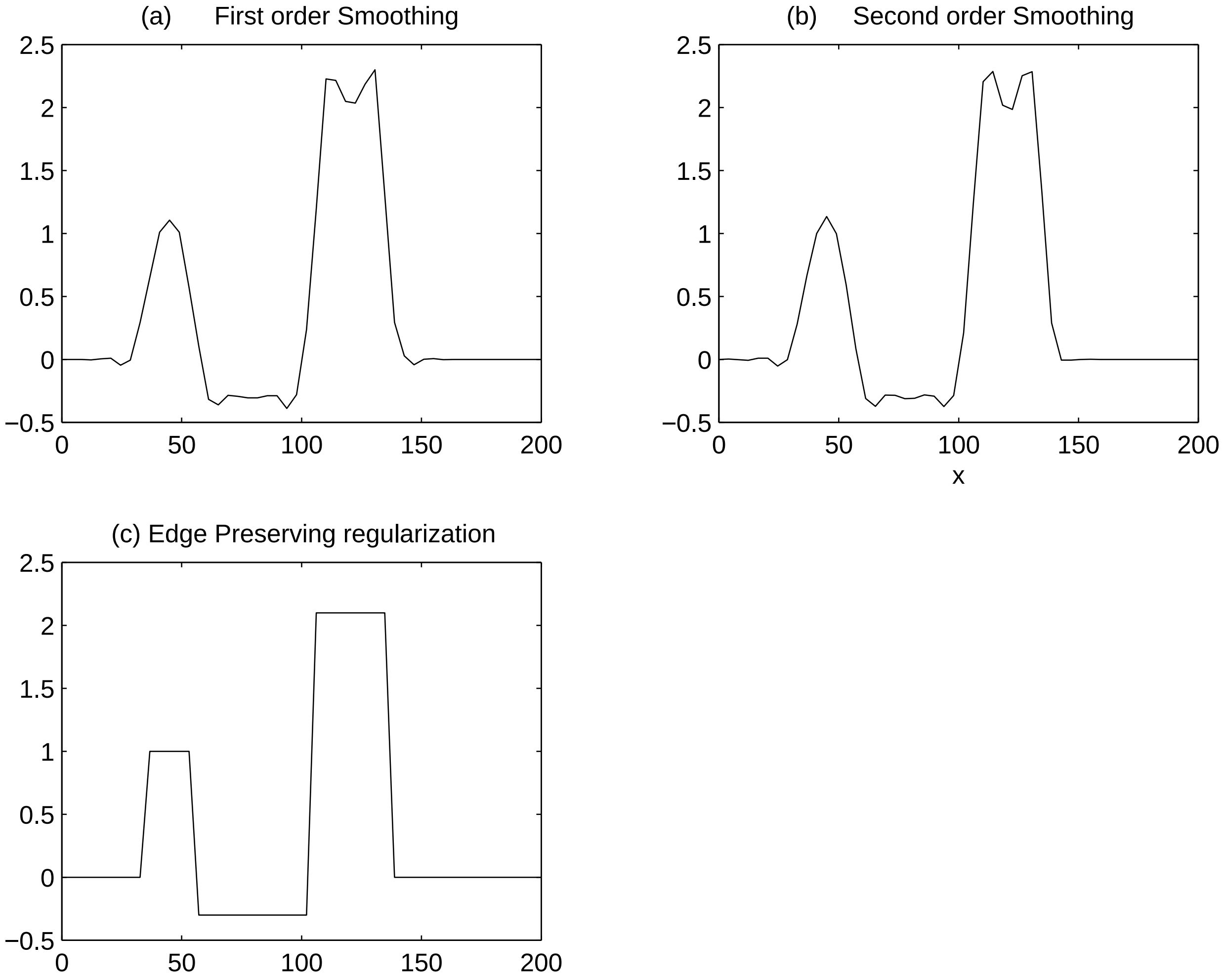


Figure 9: a) and b) Smooth solution obtained with quadratic regularization (first and second order derivative smoothing) c) Edge preserving solution.

Email me for the EPR scripts: `sacchi@phys.ualberta.ca`

Large sparse system of equations

Sparse matrices

A sparse matrix is a matrix where most elements are equal to zero.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

It is clear that a more convenient way of storing the matrix **G** is via the following scheme

$$\begin{array}{llll} G_{1,2} \rightarrow & i(1) = 1 & j(1) = 2 & g(1) = 1 \\ G_{1,4} \rightarrow & i(2) = 1 & j(2) = 4 & g(2) = 2 \\ G_{3,3} \rightarrow & i(3) = 2 & j(3) = 3 & g(3) = -1 \end{array}$$

Now instead of saving $G_{i,j}$, $i = 1, 2$, $j = 1, 4$ we save the non-zero elements of the matrix in sparse format $i(k), j(k), g(k), k = 1, K$, where K is the number of non-zero elements of the matrix **G**.

Matrix-vector multiplication in sparse format

We would like to evaluate the following matrix-vector multiplication:

$$\mathbf{d} = \mathbf{G}\mathbf{m}$$

the last equation entails performing the following sum

$$d_i = \sum_{j=1}^{M} G_{i,j} m_j, \;\; i = 1, N$$

As always, we need a computer code to perform the sum:

$$\boxed{\texttt{fullmult.m}}$$

```
d = zeros(N,1)         % Allocate a vector of zeros
  for i=1:N
    for j=1:M
      d(i) = d(i) + G(i,j)*m(j)
       end
       end
```

Let's see how many operations have been used to multiple **G** times **m**

```
Number of sums = M*N
Number of products = N*M
```

If **G** is sparse and we use the sparse format storage, the matrix-vector multiplication can be written as:

`sparsemult.m`

```
d = zeros(N,1)      % Allocate a vector of zeros
  for k=1:K
    d(i(k)) = d(i(k)) + g(k)*m(j(k))
      end
```

Now the total number of operations is given by

```
Number of sums = K
Number of products = K
```

It is clear that only when sparsity of the matrix $K/(N \times M) << 1$ there is an important computational saving.

Conjugate Gradients

The CG method is used to find the minimum of the system

$$J' = ||\mathbf{A}\mathbf{x} - \mathbf{y}||$$

where $\mathbf{A}$ is an $N \times M$ matrix. CG used the following iterative scheme:

Choose an initial solution $\mathbf{x}_0$ and compute $\mathbf{s}_0 = \mathbf{y} - \mathbf{A}\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}_0)$, and $\mathbf{q}_0 = \mathbf{A}_{\mathbf{p_0}}$. Then

$$\begin{aligned}
\alpha_{k+1} &= \mathbf{r}_k^T\mathbf{r}_k/\mathbf{q}_k^T\mathbf{q} \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_{k+1}\mathbf{p}_k \\
\mathbf{s}_{k+1} &= \mathbf{s}_k - \alpha_{k+1}\mathbf{q}_k \\
\mathbf{r}_{k+1} &= \mathbf{A}^T\mathbf{s}_{k+1} \qquad (*) \\
\beta_{k+1} &= \mathbf{r}_k^T\mathbf{r}_k/\mathbf{r}_k^T\mathbf{r} \\
\mathbf{p}_{k+1} &= \mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k \\
\mathbf{q}_{k+1} &= \mathbf{A}\mathbf{p}_{k+1} \qquad (**)
\end{aligned}$$

where $k = 0, 1, 2, ..$ is the iteration number.

Theoretically, the minimum is found after M iterations (steps). In general, we will stop in a number of iterations $M' < M$ an be happy with an approximate solution. It is clear that all the computational cost of minimizing J' using CG is in the lines I marked $*$ and $**$.

How do we use the CG algorithm to minimize the cost function $J = ||\mathbf{Gm} - \mathbf{d}||^2 + \mu||\mathbf{m}||^2$?. If we choose $\mathbf{A}$ and $\mathbf{y}$ as follows

$$\mathbf{A} = \left(\begin{array}{c} \mathbf{G} \\ \sqrt{\mu}\mathbf{I}_M \end{array} \right) \tag{15}$$

$$\mathbf{y} = \left(\begin{array}{c} \mathbf{d} \\ \mathbf{0}_M \end{array} \right) \tag{16}$$

then, J and J' have the same minimizer [a]. Therefore, when minimizing the cost J we replace $\mathbf{A}$ and $\mathbf{y}$ by eqs (15) and (16) and use CG to minimize the cost J'.

[a] Please, prove it!

testcg.m

```
% Make an Example (y = Ax)
N = 20;
M = 10;
A=randn(N,M);
x = ones(M,1);
y = A*x;

% Try to get back x from y

% Initialization
x =zeros(M,1);
s=y-A*x;
p=A’*s;
r=p;
q=A*p;
old = r’*r;
max_iter = 10
% CG loop
for k=1:max_iter
 alpha = (r’*r)/(q’*q);
 x= x +alpha*p;
 s = s-alpha*q;
 r= A’*s;
 new = r’*r;
 beta = new/old;
 old = new;
 p = r + beta*p;
 q = A*p;
end
```

An excellent tutorial for CG methods:

Jonathan Richard Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994. [http://www-2.cs.cmu.edu/~jrs/jrspapers.html]

Imaging vs. inversion

Imaging - Where?
Inversion - Where and What ?

Imaging and Inversion using the distorted Born appoximation

Start with the acoustic wave equation (Helmholtz equation)

$$\nabla^2 u(\mathbf{x},\mathbf{s},\omega) + \frac{\omega^2}{c(\mathbf{x})} u(\mathbf{x},\mathbf{s},\omega) = -\delta(\mathbf{x}-\mathbf{s}) W(\omega)$$

$$\frac{1}{c(\mathbf{x})^2} = \frac{1}{c_0(\mathbf{x})^2} + f(\mathbf{x})$$

$c(\mathbf{x})$: Velocity of the medium (Unknown)

$c_0(\mathbf{x})$: Known background velocity model (Macro Model)

$f(\mathbf{x})$: Acostic potential (Unknown)

$$\begin{aligned}\nabla^2 u(\mathbf{x},\mathbf{s},\omega) + \tfrac{\omega^2}{c_0(\mathbf{x})} u(\mathbf{x},\mathbf{s},\omega) &= \\ -\delta(\mathbf{x}-\mathbf{s})W(\omega) - & \quad \tfrac{f(\mathbf{x})\omega}{c_0(\mathbf{x})} u(\mathbf{x},\mathbf{s})\end{aligned}$$

$$\nabla^2 G(\mathbf{x},\mathbf{s},\omega) + \frac{\omega^2}{c_0(\mathbf{x})} G(\mathbf{x},\mathbf{s},\omega) = -\delta(\mathbf{x}-\mathbf{s})W(\omega)$$

$G(\mathbf{x},\mathbf{y},\omega)$: Green function for the background medium

Lippman-Schwinger equation:

$$\begin{aligned}u(\mathbf{r},\mathbf{s},\omega) = &\quad W(\omega)G(\mathbf{r},\mathbf{s},\omega) \\ + &\quad \omega^2 \int G(\mathbf{r},\mathbf{x},\omega)\, f(\mathbf{x})\, u(\mathbf{x},\mathbf{s},\omega)\, d^3\mathbf{x}\end{aligned}$$

$$\begin{aligned} u(\mathbf{r},\mathbf{s},\omega) = \quad & W(\omega)G(\mathbf{r},\mathbf{s},\omega) \\ + \quad & \omega^2 \int G(\mathbf{r},\mathbf{x},\omega)\, f(\mathbf{x})\, u(\mathbf{x},\mathbf{s},\omega)\, d^3\mathbf{x} \end{aligned}$$

$$u(\mathbf{r},\mathbf{s},\omega) = u_{inc}(\mathbf{r},\mathbf{s},\omega) + u_{sc}(\mathbf{r},\mathbf{s},\omega)$$

If $u_{sc}(\mathbf{r},\mathbf{s},\omega) \approx u_{inc}(\mathbf{r},\mathbf{s},\omega) = W(\omega)G(\mathbf{r},\mathbf{s},\omega)$

$\rightarrow$

Single scattering approximation about the background medium

$$\begin{aligned} u_{sc}(\mathbf{r},\mathbf{s},\omega) \quad & = \\ \omega^2 W(\omega) \quad & \int G(\mathbf{r},\mathbf{x},\omega)\, f(\mathbf{x})\, G(\mathbf{x},\mathbf{s},\omega)\, d^3\mathbf{x} \end{aligned}$$

or, in compact notation:

$$u = \mathcal{B} f$$

- $c_0 = constant$ - First Born Approximation
- $c_0 = c_0(\mathbf{x})$ - Distorted Born Approximation
- Validity of the approximation requires $c(\mathbf{x}) \approx c_0(\mathbf{x})$
- $G(\mathbf{x}, \mathbf{y}, \omega)$ can be written explicitly only for very simple models
- In arbirtrary backgrounds we can use the First Order Asymtotic approximation (Geometrical Optics)

$$G(\mathbf{x}, \mathbf{y}, \omega) = A(\mathbf{x}, \mathbf{y}) e^{i\omega\, \tau(\mathbf{x}, \mathbf{y})}$$

Forward/Adjoint pairs:

$$u = \mathcal{B} f$$

$$\tilde{f} = \mathcal{B}' u$$

where

$$\langle u, \mathcal{B} f \rangle = \langle \mathcal{B}'\, u, f \rangle$$

$$\begin{aligned} \tilde{f}(\mathbf{x}) &= \\ \int\int\int & W(\omega)^*\, \omega^2\, G^*(\mathbf{x}, \mathbf{r}, \omega)\, u(\mathbf{r}, \mathbf{s}, \omega)\, G^*(\mathbf{s}, \mathbf{x}, \omega) d\omega\, d^2\mathbf{r}\, d^2\mathbf{s} \end{aligned}$$

Imaging with the adjoint operator:

$$\tilde{f}(\mathbf{x}) = \text{distorted version of } f(\mathbf{x})$$

Alternative: Try to invert $f(\mathbf{x})$,

$$u = \mathcal{B} f \,, \qquad \hat{f} = \mathcal{B}' u$$

$$\tilde{f} = \mathcal{B}' \, \mathcal{B} \, f$$

$$\tilde{f}(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') d^3\mathbf{x'}$$

Try to adjust the adjoint and forward pairs to obtain:

$$K(\mathbf{x}, \mathbf{x}') \approx \delta(\mathbf{x} - \mathbf{x}')$$

Forward-adjoint operators with WKBJ Green functions

(Solution in wavenumber domain)

$C_0 = C_0(z)$

$$u(\mathbf{k}_y, \mathbf{k}_h, \omega) = \int \frac{\omega^2}{4} \frac{f(\mathbf{k}_y, z) e^{i \int_0^z k_z(z')dz'}}{(k_{rz}(0) k_{sz}(0) k_{rz}(z) k_{sz}(z))^{1/2}} dz,$$

$$\tilde{f}(\mathbf{k}_y, z) = \int \int \frac{\omega^2}{4} \frac{u(\mathbf{k}_y, \mathbf{k}_h, \omega) e^{-i \int_0^z k_z(z')dz'}}{(k_{rz}(0) k_{sz}(0) k_{rz}(z) k_{sz}(z))^{1/2}} d\omega d^2 k_h,$$

with

$$k_z(z) = \frac{\omega}{c(z)} \sqrt{1 - \frac{|\mathbf{k}_y + \mathbf{k}_h|^2 c^2(z)}{4\omega^2}} + \frac{\omega}{c(z)} \sqrt{1 - \frac{|\mathbf{k}_y - \mathbf{k}_h|^2 c^2}{4\omega^2}}$$

Recursive implementation (Downward continuation)

$$\begin{aligned} u(\mathbf{k}_y, \mathbf{k}_h, z, \omega) &= u(\mathbf{k}_y, \mathbf{k}_h, \omega) e^{-i \int_0^z k_z(z') dz'} \\ &\approx u(\mathbf{k}_y, \mathbf{k}_h, \omega) e^{-i \sum_{z'=0}^{z} k_z(z') \Delta z} \end{aligned}$$

$$u(\mathbf{k}_y, \mathbf{k}_h, z+\Delta z, \omega) = u(\mathbf{k}_y, \mathbf{k}_h, z, \omega) e^{-i k_z(z) \Delta z}$$

Lateral variant background velocity $c_0 = c_0(\mathbf{x}, z)$ (Splitting[a])

$$e^{-ik_z(x,z)\Delta z} \approx e^{-ik_z(z)\Delta z} \times e^{-i\omega \Delta S \Delta z}$$

$$\Delta S = \frac{2}{c_0(z)} - \frac{1}{c_0(\mathbf{y}+\mathbf{h}, z)} - \frac{1}{c_0(\mathbf{y}-\mathbf{h}, z)}$$

$c_0(z)$: mean velocity at depth z

$c_0(\mathbf{x}, z)$: background velocity

The algorithm recursively downward continues the data with $c_0(z)$ in $\mathbf{k}_h$, $\mathbf{k}_y$

The split-step correction is applied in $\mathbf{y}$, $\mathbf{h}$.

[a] Feit and Fleck'78, Light Propagation in graded-index fibers, Appl. Opt. 17

The Algorithm

$$u(\mathbf{k}_y, \mathbf{k}_h, z+\Delta z, \omega) = \mathcal{F}[e^{-i\omega \Delta S \Delta z} \times \mathcal{F}^{-1}[u(\mathbf{k}_y, \mathbf{k}_h, z, \omega) \times e^{-}$$

- $\mathcal{F}$ is a 2D or 4D Fourier transform
- Algorithm complexity $\approx$ two FFTs per depth step

Imaging and Inversion

Imaging:

$$\tilde{f} = \mathcal{B}' u$$

Inversion (analytical solutions): Modify the adjoint operator to collapse the PSF into a delta function

$$f_{inv} = \mathcal{B}^{\dagger} u$$

Inversion (numerical solution)

Minimize

$$\Phi = ||u - \mathcal{B} f||^2 + \mu \, ||f||^2$$

Inversion (numerical solution) - CG Method

- CG: Conjugate Gradients to solve large linear systems of equation (in this case linear operators)
- Easy to implement if you have $\mathcal{B}$ and $\mathcal{B}'$
- Don't need to iterate forever

CG Scheme

To solve the problem $||\mathcal{B}x - y||^2$, with initial solution x_0

Set initial values: $r = y - \mathcal{B}\,x_0$, $g = \mathcal{B}'\,r$, $s = g$

```
For i = 1:imax
    $v = \mathcal{B}s$   ! Modeling
    $\delta = ||v||^2$
    $\alpha = \gamma/\delta$
    $x = x + \alpha\, s$
    $r = r - \alpha\, v$
    $g = \mathcal{B}'\, r$   ! Imaging
    $s = g + \beta\, s$
Enddo
```

A note for code developers: Operators

`= subroutines/functions`

$$\mathcal{B} = \mathcal{A}_1 \mathcal{A}_2 \dots \mathcal{A}_n$$

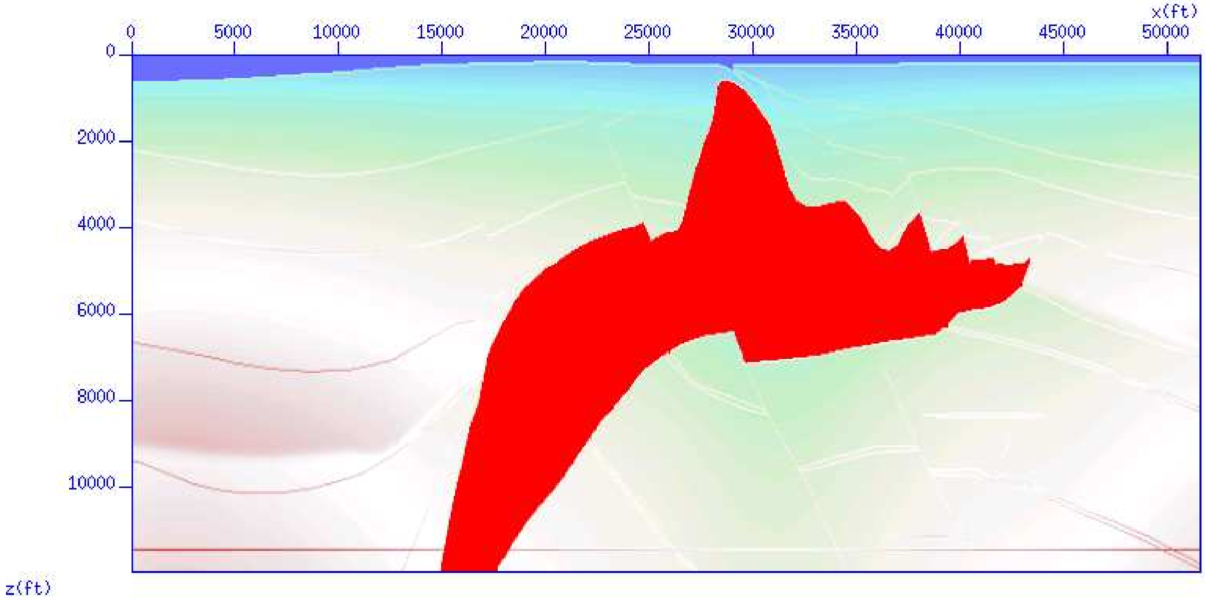
$$\mathcal{B}' = \mathcal{A}_n' \dots \mathcal{A}_2' \mathcal{A}_1'$$

Definition of forward/adjoint pairs:

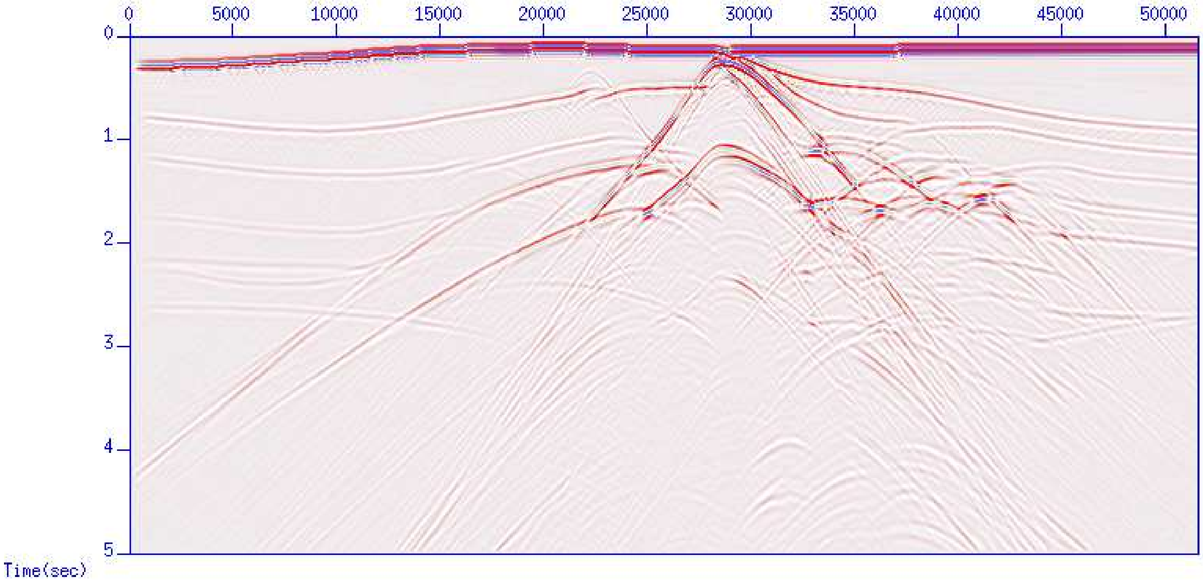
$$\langle u, \mathcal{B} f \rangle = \langle \mathcal{B}' \, u, f \rangle$$

The latter can be numerically tested.

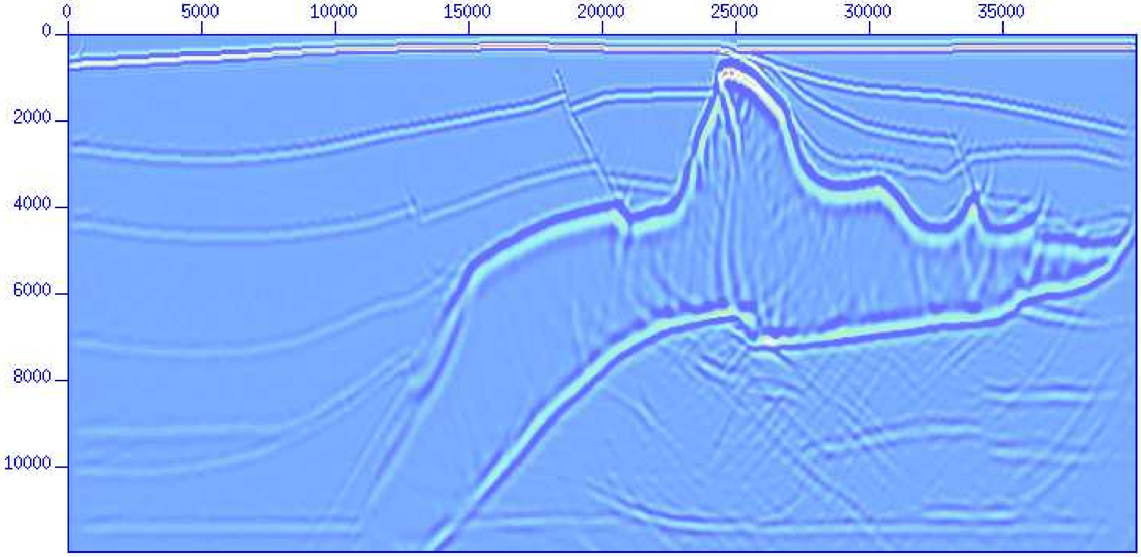
SEG/EAGE Sub Salt Model - Velocity



SEG/EAGE Sub Salt Model - Zero offset data



SEG/EAGE Sub Salt Model - Born+Split-step Imaging / adjoint operator



Edge Preserving Regularization

$$J = ||u_{sc} - \mathcal{B}f||^2 + \beta_x \sum_k \mathcal{R}[(D_x f)_k] + \beta_z \sum_k \mathcal{R}[(D_z f)_k] \, .$$

$$\mathcal{R}(x) = ln(1 + x^2)$$

with spatial derivative operators:

$$(D_x f)_{i,j} = (f_{i,j+1} - f_{i,j})/\delta_x$$

$$(D_z f)_{i,j} = (f_{i+1,1} - f_{i,j})/\delta_z$$

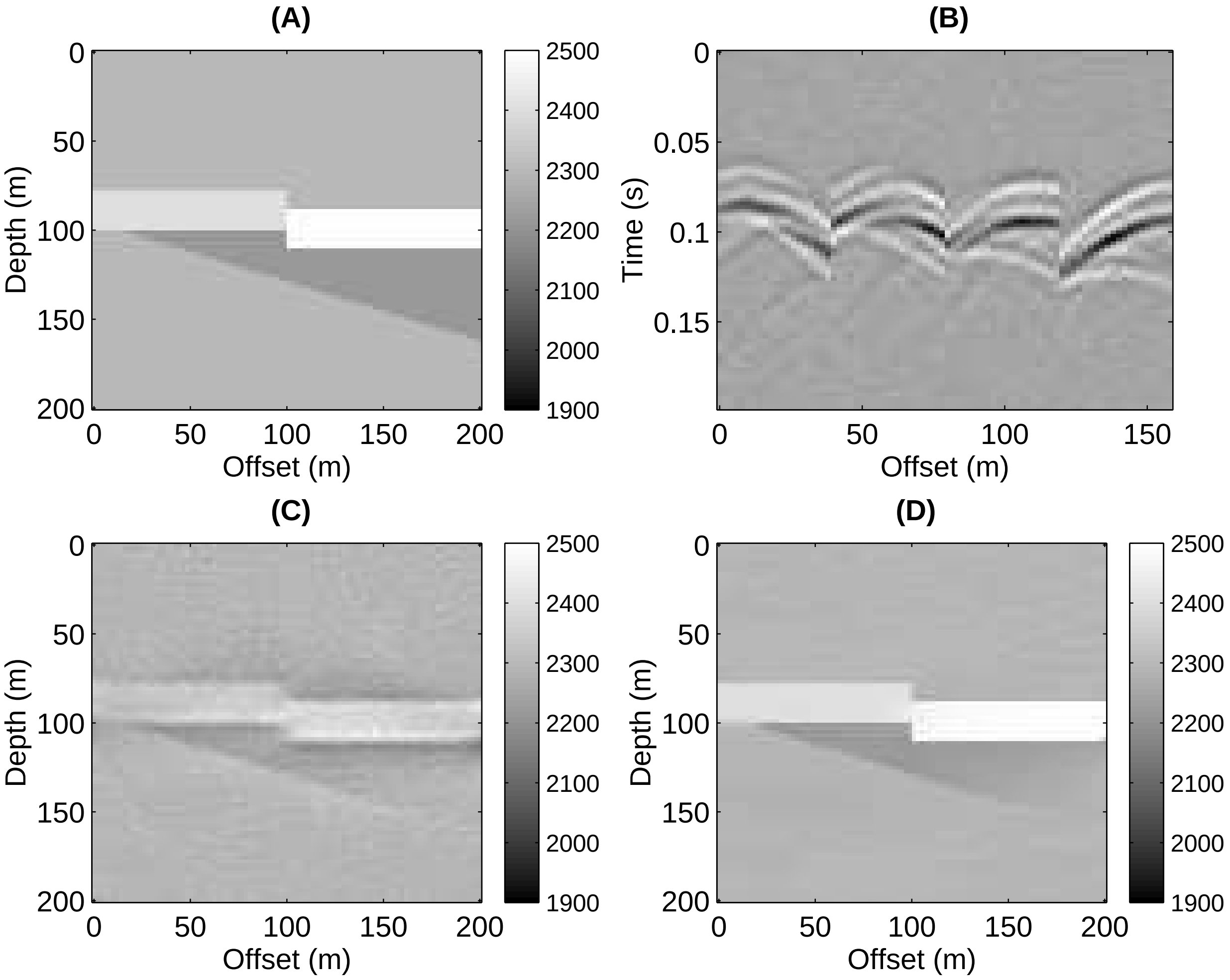


Figure 10: (A) true model. (B) Seismic waveforms (4 sources). (C) Damped least-squares solution. (D) EPR.

From Inverse Filters to LS Migration

Mauricio Sacchi

Institute for Geophysical Research & Department of Physics
University of Alberta, Edmonton, AB, Canada

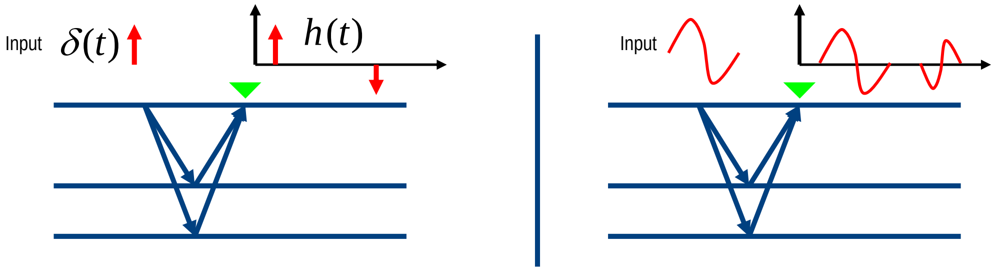
Linear systems and Invariance

- Impulse response (hitting the system with an impulse)

$$y(t) = \int h(t-\tau)x(\tau)d\tau = h(t) * x(t)$$

$$x(t) = \delta(t) \longrightarrow \boxed{h(t)} \longrightarrow h(t)$$

$$x(t) \longrightarrow \boxed{h(t)} \longrightarrow y(t)$$

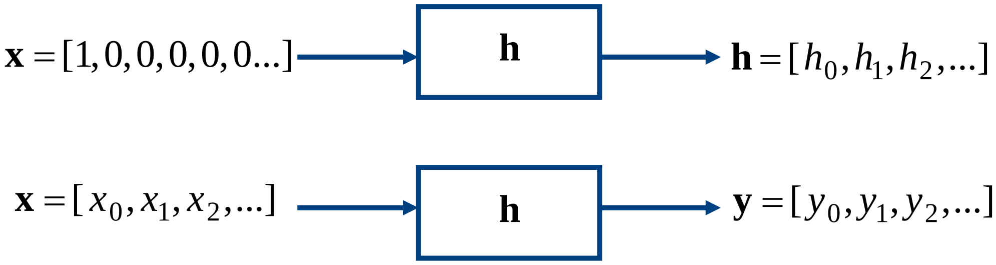


Linear systems and Invariance - Discrete case

- Convolution Sum

$$y_n = \sum_k h_{k-n} x_k = h_n * x_n$$

- Signals are time series or vectors



$$\mathbf{x} = [1,0,0,0,0,0...] \longrightarrow \boxed{\mathbf{h}} \longrightarrow \mathbf{h} = [h_0, h_1, h_2, ...]$$

$$\mathbf{x} = [x_0, x_1, x_2, ...] \longrightarrow \boxed{\mathbf{h}} \longrightarrow \mathbf{y} = [y_0, y_1, y_2, ...]$$

Discrete convolution

- Formula

$$y_n = \sum_k h_{n-k} x_k = h_n * x_n$$

- Finite length signals

$$x_k, \quad k = 0, NX - 1$$

$$y_k, \quad k = 0, NY - 1$$

$$h_k, \quad k = 0, NH - 1$$

- How do we do the convolution with finite length signals?
  - Computer code
  - Matrix times vector
  - Polinomial multiplication
  - DFT

Discrete convolution

```
% Initialize output

  y(1: NX + HH - 1) = 0

% Do convolution sum

  for i = 1: NX
    for j  =  1: NH
     y(i+ j - 1)  =  y(i + j - 1)  + x(i) h(j)
    end
  end
```

Discrete convolution

Example:

$$x = [x_0, x_1, x_2, x_3, x_4], \quad NX = 5$$

$$h = [h_0, h_1, h_2], \quad NH = 3$$

$$y_n = \sum_k h_{k-n} x_k = h_n * x_n$$

$$y_0 = x_0 h_0$$

$$y_1 = x_1 h_0 + x_0 h_1$$

$$y_2 = x_2 h_0 + x_1 h_1 + x_0 h_2$$

$$y_3 = x_3 h_0 + x_2 h_1 + x_1 h_2$$

$$y_4 = x_4 h_0 + x_3 h_1 + x_2 h_2$$

$$y_5 = x_4 h_1 + x_3 h_2$$

$$y_6 = x_4 h_2$$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} x_0 & 0 & 0 \\ x_1 & x_0 & 0 \\ x_2 & x_1 & x_0 \\ x_3 & x_2 & x_1 \\ x_4 & x_3 & x_2 \\ 0 & x_4 & x_3 \\ 0 & 0 & x_4 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \end{pmatrix}$$

An finally: the LS filter design method…

- Inverse filtering using series expansion results in long filters with low convergence in some cases (*a=0.99*)
- Truncation errors could deteriorate the performance of the filter
- An alternative way of designing filter involves using the method of LS (inversion):

Given $x = (1, a)$

find $h = (h_0, h_1, h_2, \cdots h_N)$

Such that $h * x \approx (1, 0, 0, 0..)$

- We have posed the problem of designing and inverse filter as an inverse problem.

Inversion of a dipole

Find a filter of length 3 that inverts the dipole:

Dipole□□

Desire output□□

$$\begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & a & 1 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

In matrix form: $\mathbf{W\,h} \approx \mathbf{g}$

We form an error function to minimize $e = \|\,\mathbf{W\,h} - \mathbf{g}\,\|_2^2$

Minimize the error function by taking derivatives with respect to the unknowns $\nabla e = \mathbf{0} = \mathbf{W^T Wh} - \mathbf{W^T g}$

LS filter $\mathbf{h} = (\mathbf{W^T W})^{-1}\mathbf{W^T g}$

Actual output $\hat{\mathbf{g}} = \mathbf{W\,h}, \quad (\hat{g}_k = w_k * h_h)$

Inversion of a wavelet

Find a filter of length 3 that inverts the *wavelet:*

Wavelet

Desire output□□

$$\begin{pmatrix} w_0 & 0 & 0 \\ w_1 & w_0 & 0 \\ w_2 & w_1 & w_0 \\ w_3 & w_2 & w_1 \\ w_4 & w_3 & w_2 \\ 0 & w_4 & w_3 \\ 0 & 0 & w_4 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

In matrix form:

$$\mathbf{W}\,\mathbf{h} \approx \mathbf{g}$$

$$e = \|\,\mathbf{W}\,\mathbf{h} - \mathbf{g}\,\|_2^2$$

$$\nabla e = \mathbf{0} = \mathbf{W}^{\mathbf{T}}\mathbf{W}\mathbf{h} - \mathbf{W}^{\mathbf{T}}\mathbf{g}$$

$$\mathbf{h} = (\mathbf{W}^{\mathbf{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathbf{T}}\mathbf{g}$$

$$\hat{\mathbf{g}} = \mathbf{W}\ \mathbf{h}, \quad (\hat{g}_k = w_k * h_h)$$

Ideal and Achievable Inversion of a Wavelet: Pre-whitening

- Trade-off curves and Pre-whitening

Trade-off parameter

$$\mathbf{W}\,\mathbf{h} \approx \mathbf{g}$$

$$e = \|\,\mathbf{W}\,\mathbf{h} - \mathbf{g}\,\|_2^2 + \mu\,\|\,\mathbf{h}\,\|_2^2$$

$$\nabla e = \mathbf{0} = \mathbf{W}^{\mathbf{T}}\mathbf{W}\mathbf{h} - \mathbf{W}^{\mathbf{T}}\mathbf{g} + \mu\mathbf{g}$$

$$\mathbf{h} = (\mathbf{W}^{\mathbf{T}}\mathbf{W} + \mu\,\mathbf{I})^{-1}\mathbf{W}^{\mathbf{T}}\mathbf{g}$$

$$\hat{\mathbf{g}} = \mathbf{W}\ \mathbf{h}, \quad (g_k = w_k * h_h)$$

$$\mathbf{R} = \mathbf{W}^{\mathbf{T}}\mathbf{W} = \begin{pmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{pmatrix}$$

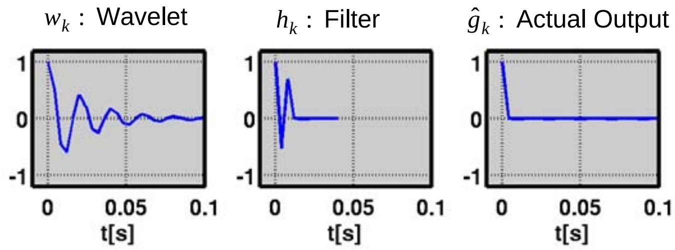
R_k : Autocorrelation coefficient

$\mu = R_0.P/100,\quad P$: %of pre – whitening

Ideal and Achievable Inversion of a Wavelet: Pre-whitening

- Pre-whitening *P=0%*

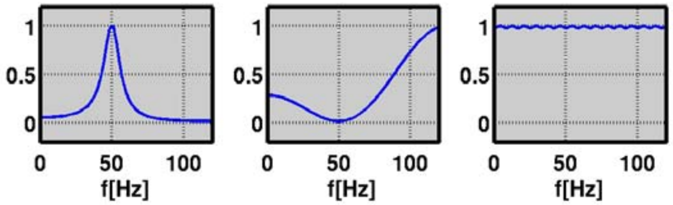
Time Domain



High Gain at High Frequencies - Noise will be amplified

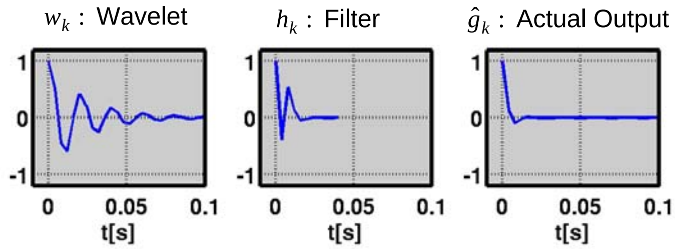
Frequency Domain

Normalized Amplitude Spectrum

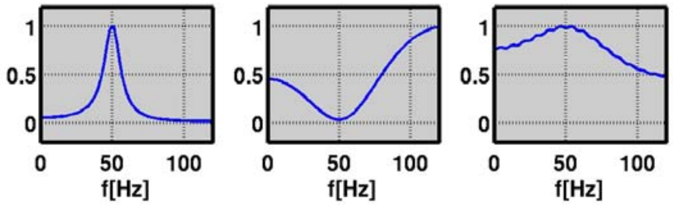


Ideal and Achievable Inversion of a Wavelet: Pre-whitening

- *P=10%*

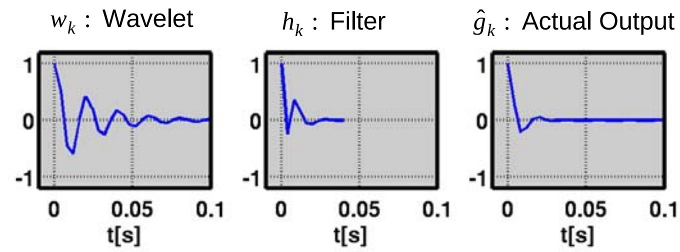


With Pre-whitening we can control the gain of the inverse filter

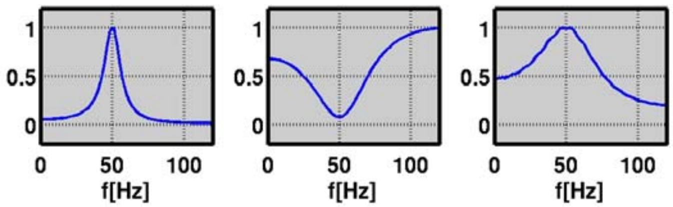


Ideal and Achievable Inversion of a Wavelet: Pre-whitening

- *P=20%*



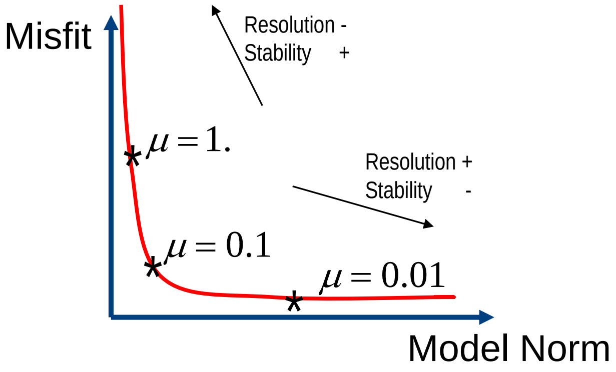
With Pre-whitening we can control the gain of the inverse filter



Ideal and Achievable Inversion of a Wavelet: Pre-whitening & Trade-off curve

$$e = \| \mathbf{W}\,\mathbf{h} - \mathbf{g} \|_2^2 + \mu \| \mathbf{h} \|_2^2$$

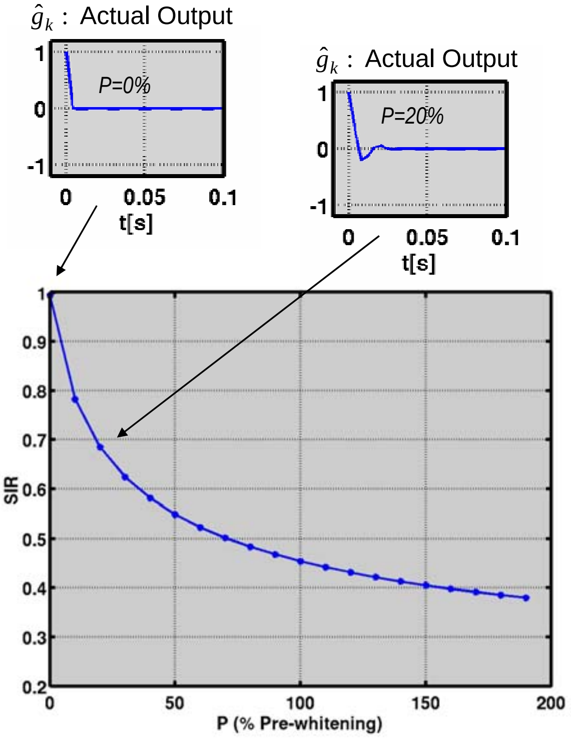
$$= \text{Misfit} + \mu\, \text{Model Norm}$$



How good is the output ? SIR

- Signal-to-interference ratio (SIR)

$$\mathrm{SIR} = \frac{\max(|\hat{g}_k|)}{\sum_k |\hat{g}_k|}$$



LS Inversion of non-minimum phase wavelets

- Previous analysis was OK for minimum phase wavelets - we tried to convert a front loaded signal into anther front loaded signal (spike)
- For non-minimum phase wavelets we introduce an extra degree of freedom (*Lag*)

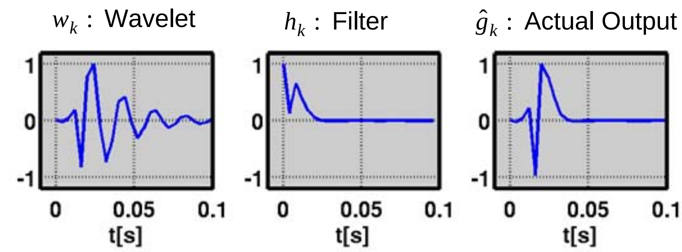
$$\begin{pmatrix} w_0 & 0 & 0 \\ w_1 & w_0 & 0 \\ w_2 & w_1 & w_0 \\ w_3 & w_2 & w_1 \\ w_4 & w_3 & w_2 \\ 0 & w_4 & w_3 \\ 0 & 0 & w_4 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \downarrow$$

In this example *Lag=3*

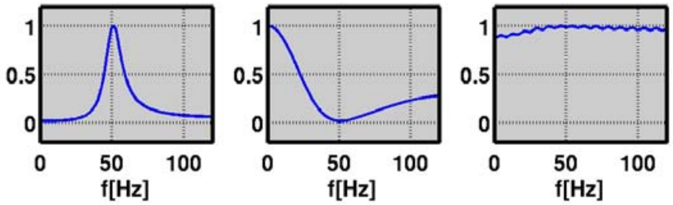
- Try different lags until you get a nice “spiky” output

LS Inversion of non-minimum phase wavelets

- *P=1%, Lag=1, SIR=0.23*

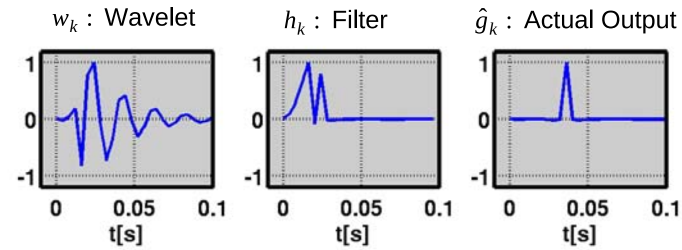


The phase of the actual output is wrong

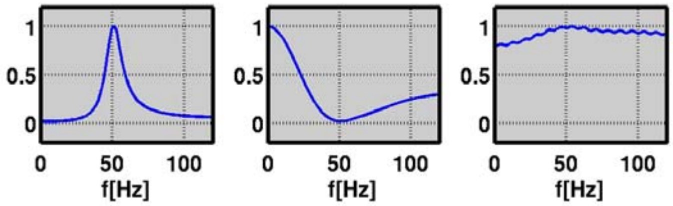


LS Inversion of non-minimum phase wavelets

- *P=1%, Lag=10, SIR=0.97*

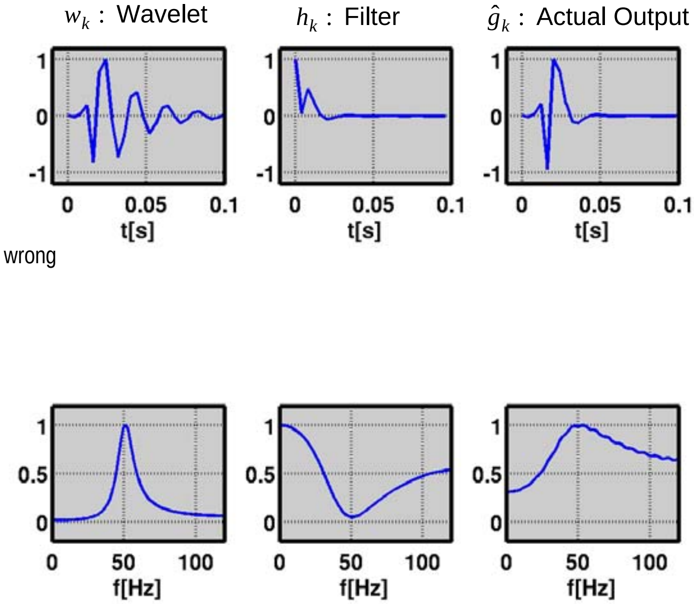


The phase of the actual output is correct



LS Inversion of non-minimum phase wavelets

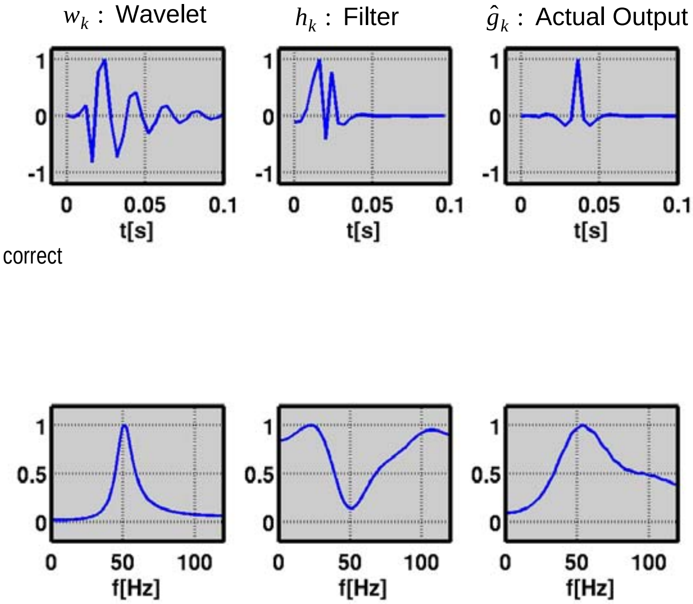
- *P=20%, Lag=1, SIR=0.22*



The phase of the actual output is wrong

LS Inversion of non-minimum phase wavelets

- *P=20%, Lag=10, SIR=0.56*



The phase of the actual output is correct

Where does the wavelet come from?

- Observations
  - Finding the inverse filter requires the matrix R. This is the autocorrelation matrix of the wavelet.
  - If the reflectivity is assumed to be white then the autocorrelation of the trace is equal (within a scale factor) to the autocorrelation of the wavelet
  - *In Short:* Since you cannot measure the wavelet or its autocorrelation, we replace the autocorrelation coefficients of the wavelet by those of the trace and then, we hope that the validity of the white reflectivity assumption holds!!

$$\mathbf{R} = \mathbf{W}^{\mathbf{T}}\mathbf{W} = \begin{pmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{pmatrix}$$

$$\boxed{\mathbf{R}^{Trace} = c\,\mathbf{R}^{Wavelet}}$$

Why is that ?

Autocorrelation of the trace *s* $\quad R_k^s = \sum_t s_{t+k} s_k$

Convolution model $\quad s_k = w_k * r_k$

We can show this one $\quad R_k^s = R_k^r * R_k^w$

If reflectivity is white (Geology does not have memory?) $\quad R_k^r = \begin{cases} c \text{ if} & k = 0 \\ 0 \text{ every where else} \end{cases}$

Then…… $\quad \boxed{R_k^s = cR_k^w}$

Color ?

Autocorrelation of the trace *s* $R_k^s = \sum_t s_{t+k} s_k$

Convolution model $s_k = w_k * r_k$

We can show this one $R_k^s = R_k^r * R_k^w$

If the reflectivity is non-white (Geology does have memory) $R_k^w = (R_k^r)^{-1} * R_k^s$

Rosa, A. L. R., and T. J. Ulrych, 1991, Processing via spectral modeling: Geophysics, 56, 1244-1251.

Haffner, S., and S. Cheadle, 1999, Colored reflectivity compensation for increased vertical resolution: 69th Annual International Meeting, SEG, Expanded Abstracts, 1307-1310.

Saggaf, M.M., and E. A. Robinson, 2000, A unified framework for the deconvolution of traces of nonwhite reflectivity: Geophysics, 65, 1660-1676.

Practical aspects of deconvolution

- We often compute deconvolution operators from the data (we by-pass the wavelet)
- This is what we called Wiener filter or spiking filter
- Inversion is done with *a fast algorithm* that exploits the fact that the autocorrelation matrix has a special structure (Toeplitz Matrix, *Levinson Recursion*)

$$\mathbf{h} = (\mathbf{W}^{\mathrm{T}}\mathbf{W} + \mu\,\mathbf{I})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{g}$$

$$\hat{\mathbf{g}} = \mathbf{W}\,\mathbf{h}, \quad (\hat{g}_k = w_k * h_h)$$

I started with a wavelet

$$\mathbf{h} = (\mathbf{R}^{Trace} + \mu\,\mathbf{I})^{-1}\mathbf{v}$$

$$\mathbf{v} = \mathbf{W}^{\mathrm{T}}\mathbf{g} = \mathbf{W}^{\mathrm{T}}\begin{pmatrix}1\\0\\0\end{pmatrix} = \begin{pmatrix}w_0\\0\\0\\0\end{pmatrix}, \qquad \mathbf{v} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$\hat{r}_k = h_k * s_k$$

Estimate reflectivity

I started with the trace

Filter will not have the right scale factor… not a problem!!

Practical aspects of deconvolution: Noise

- How to control the noise?
  - Pre-whitening

$$s_k = w_k * r_k + n_k$$

$$\hat{r}_k = h_k * s_k = h_k * (w_k * r_k + n_k)$$

$$= \underbrace{h_k * w_k}_{\hat{g}_k} * r_k + \underbrace{h_k * n_k}_{b_k}$$

1 - This is the actual output

2- This is filtered/amplified noise (Noise after filtering)

•1- should become a spike

•2- should go to zero

•1-2 cannot be simultaneously achieved

•The pre-whitening is included in the design of *h* and is the parameter that controls *1-2*

Practical aspects of deconvolution: Noise

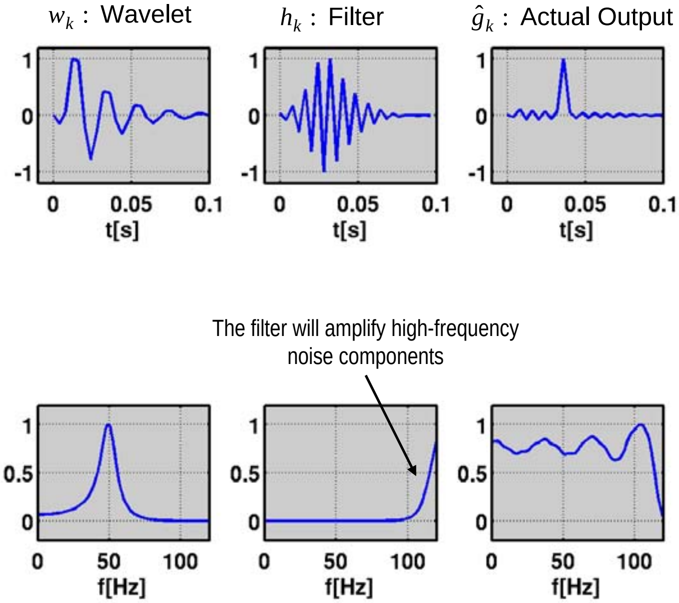
Filter Design

P=0%

Noise = 1%

•Noise variance is 1% of the max amplitude of the noise-free signal

•Spectra are normalized



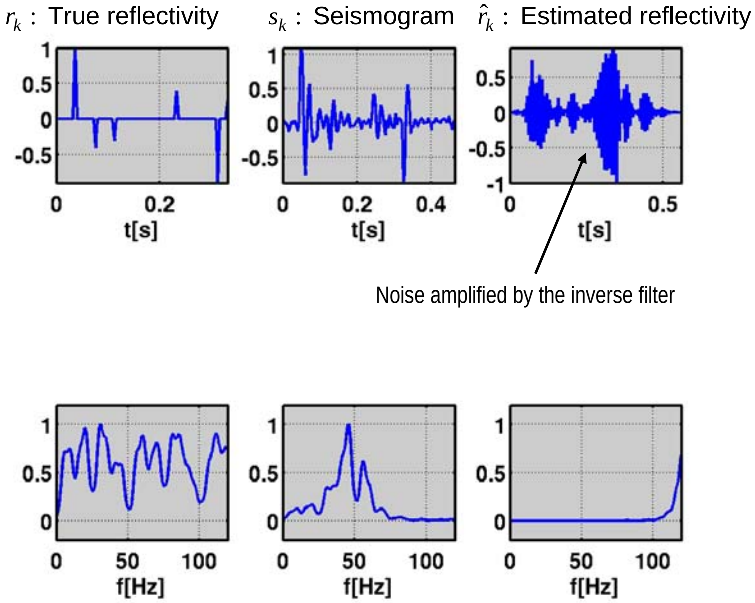
The filter will amplify high-frequency noise components

Practical aspects of deconvolution: Noise

Filter Application

P=0%

Noise = 1%



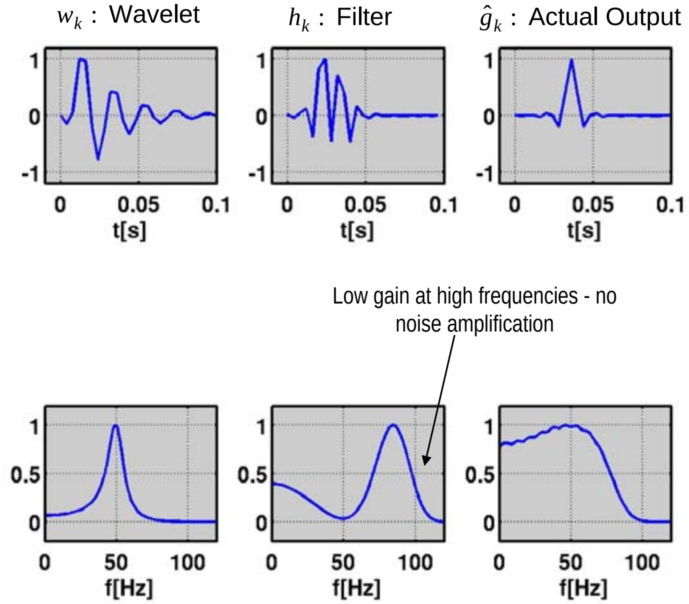
Noise amplified by the inverse filter

Practical aspects of deconvolution: Noise

Filter design

P=10%

Noise = 1%



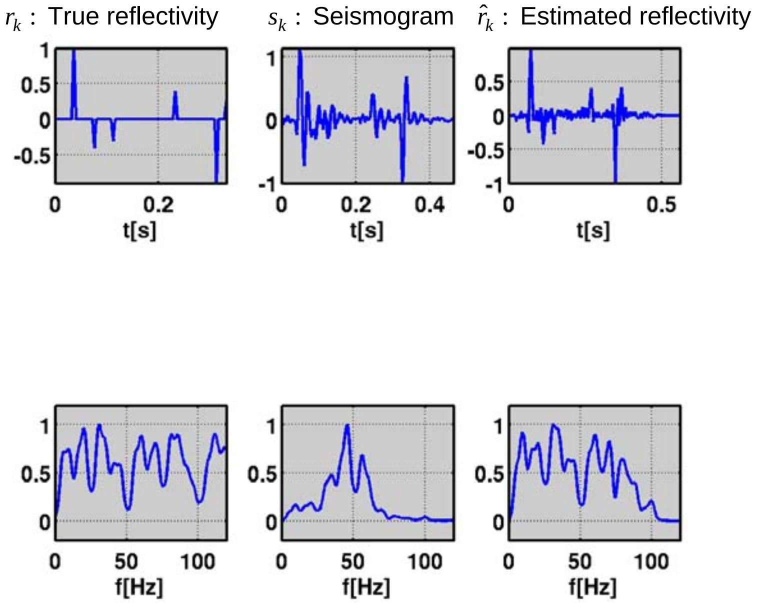
Low gain at high frequencies - no noise amplification

Practical aspects of deconvolution: Noise

Filter Application

P=10%

Noise = 1%



Practical aspects of deconvolution: Noise

The previous results can be interpreted in the freq. domain

$$\left.\begin{aligned} s_k &= w_k * r_k + n_k \\ \hat{r}_k &= h_k * s_k = h_k * (w_k * r_k + n_k) \\ &= h_k * w_k * r_k + h_k * n_k \end{aligned}\right\} \text{Time}$$

$$\left.\begin{aligned} S(\omega) &= W(\omega).R(\omega) + N(\omega) \\ \hat{R}(\omega) &= H(\omega).S(\omega) = H(\omega).[W(\omega).R(\omega)] + N(\omega)) \\ &= H(\omega).W(\omega).R(\omega) + H(\omega).N(\omega) \end{aligned}\right\} \text{Frequency}$$

$$\boxed{|\hat{R}(\omega)| = |H(\omega)|.|W(\omega)|.|R(\omega)| + |H(\omega)|.|N(\omega)|}$$

To Reproduce the examples

- http://www-geo.phys.ualberta.ca/saig/SeismicLab/SeismicLab/

```
function [f,o] = spiking(d,NF,mu);
%SPIKING: Spiking deconvolution using Levinson recursion
```

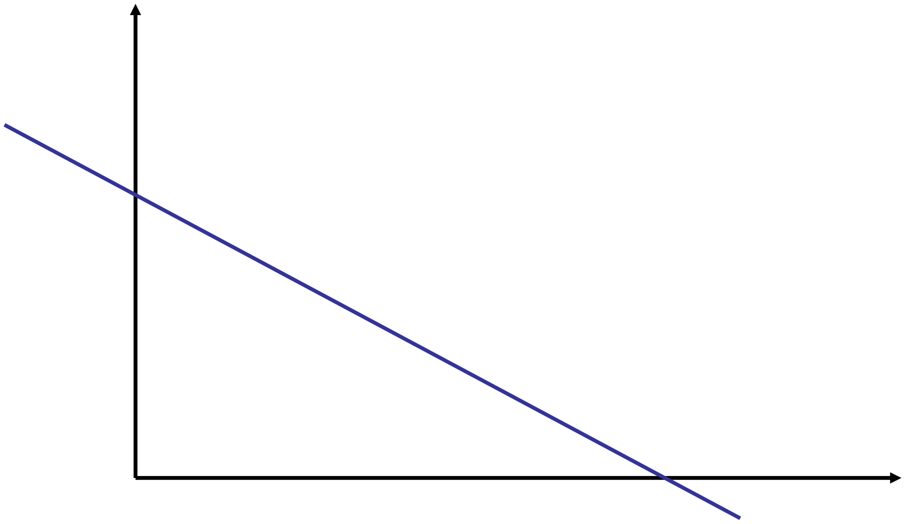
```
function [f,o] = ls_inv_filter(w,NF,Lag,mu);
%LS_INV_FILTER: Given a wavelet compute the LS inverse filter
```

```
function [K] = kurtosis(x)
%KURTOSIS: Compute the kurtosis of a time series or
%         of a multichannel series
```

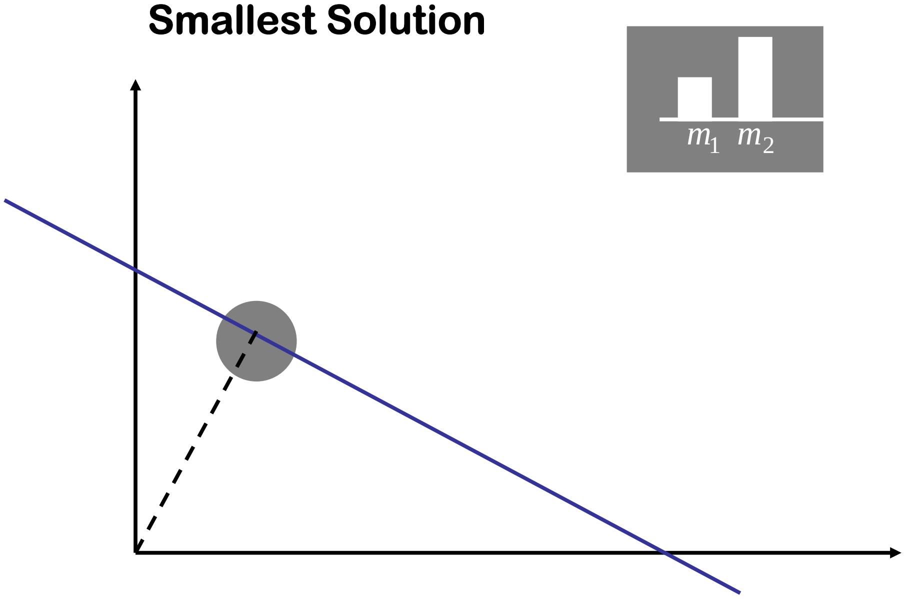
Non-Uniqueness

- A simple problem

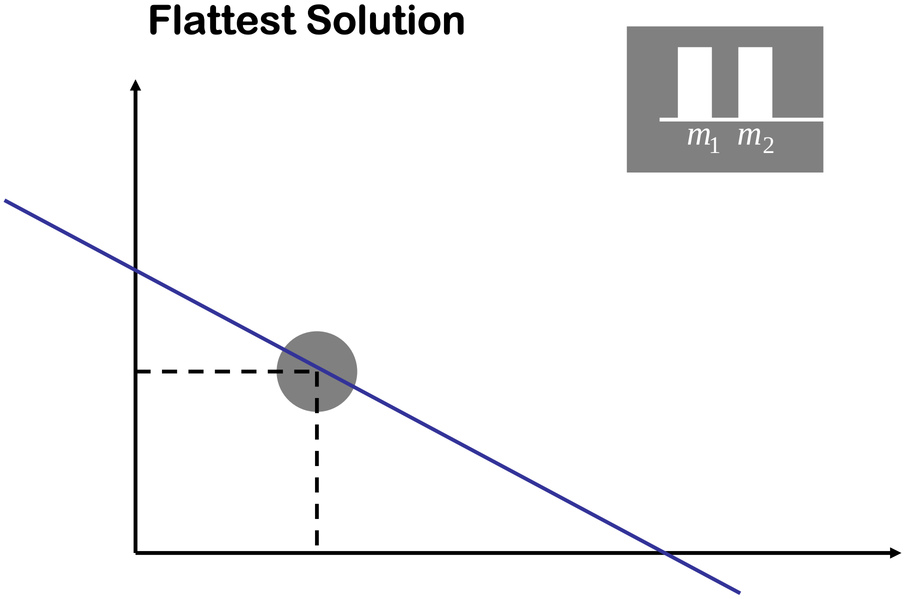
Simple example



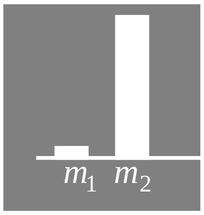
Smallest Solution

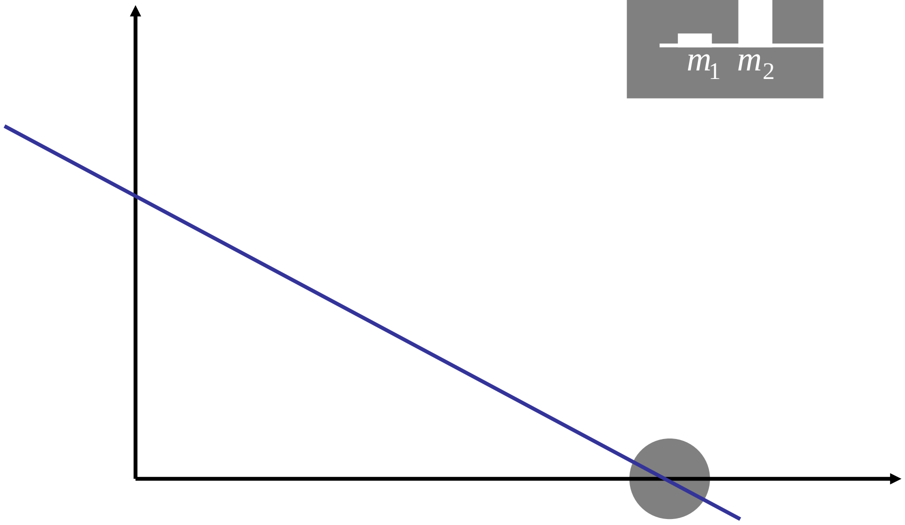


Flattest Solution



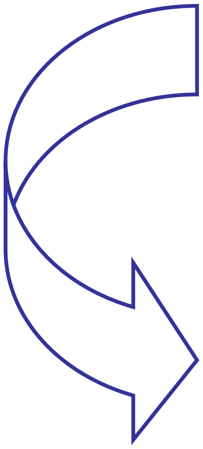
Sparse Solution





Deconvolution

- debluring
- equalization



Increase resolution - ability to
“see” thin layers

Seismic Deconvolution

•Assume wavelet is know and we want to estimate a high resolution volume of the variability of the reflectivity or impedance

•In conventional deconvolution we attempt to equalize sources/receivers

•In seismic inversion we go after images of the reflectivity or impedance parameters or attributes with high frequency content

High Frequency Imaging

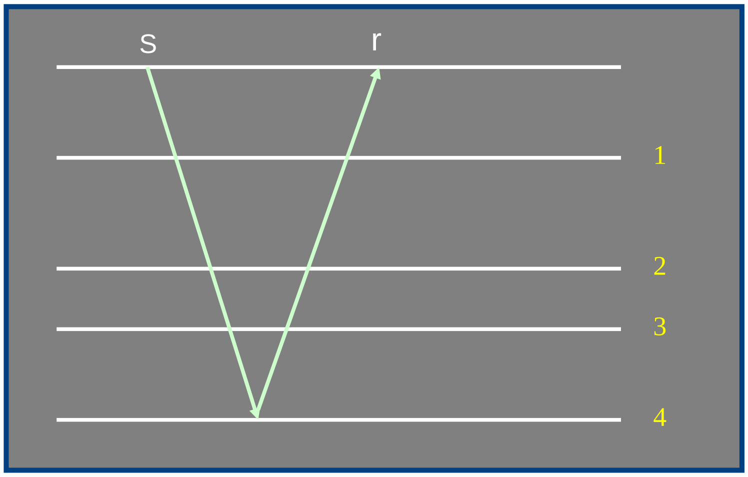
High Frequency Restoration

BW extrapolation

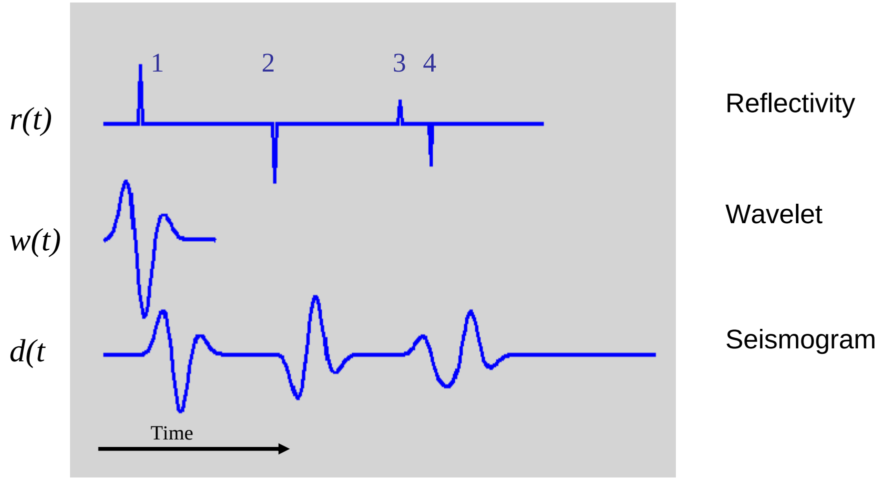
All names used to indicate we are trying to squeeze high frequencies out of the seismic data

Seismic Deconvolution

Depth
or Time



Seismic Deconvolution



Seismic Deconvolution

Naïve Solution (Just try to fit the data…)

$$(W'W)^{-}$$

Small perturbations on d produce large perturbations on r

Seismic Deconvolution

Low Resolution Solution:

Some sort of “backprojection”

Seismic Deconvolution

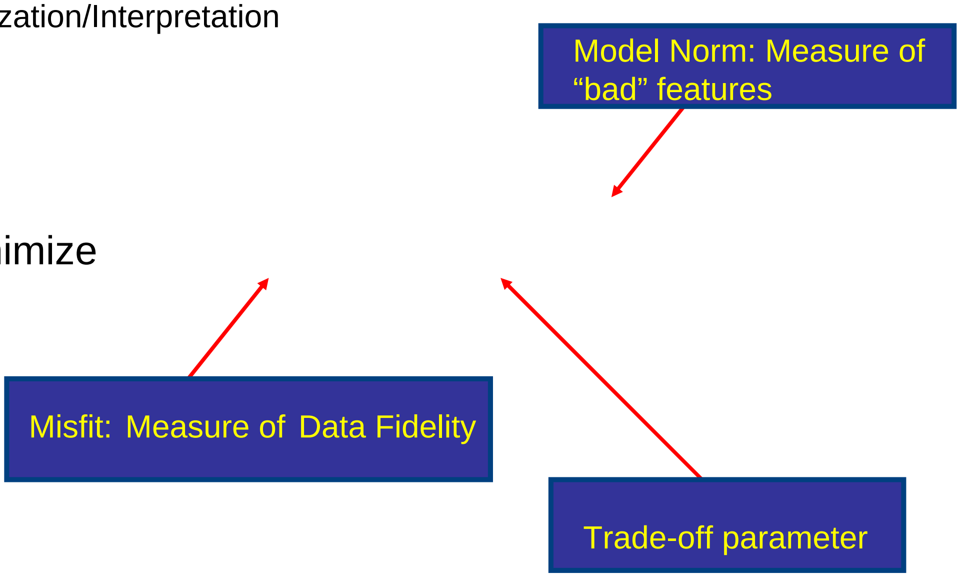
Quadratic Regularization

$\| Lr \|_2^2$

Seismic Deconvolution

Quadratic
Regularization/Interpretation

Minimize

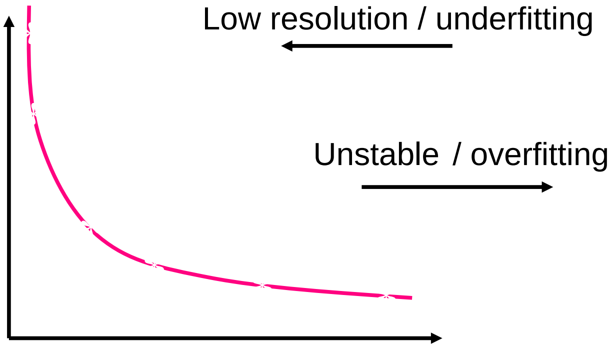

Model Norm: Measure of “bad” features

Misfit: Measure of Data Fidelity

Trade-off parameter

Seismic Deconvolution

Quadratic
Regularization/Interpretation



Trade-off curve / L-curve

Seismic Deconvolution

Quadratic Regularization

Typical regularizers

zero order quadratic regularization (smallest model)

First order quadratic regularization (flattest model)

Second order quadratic regularization (smoothest model)

Are high pass operators (First and Second order Derivatives)

Seismic Deconvolution: Trying to squeeze resolution via mathematical tricks..

When in Doubt, Smooth

Sir Harold Jeffreys (Quoted by Moritz, 1980; Taken from Tarantola, 1987)

But…, I don’t want to smooth!

A geophysicist trying to resolve thin layers

Let’s see how one can avoid smoothing and resolution degradation..

Seismic Deconvolution: Trying to squeeze resolution via mathematical tricks..

Super-resolution via non-quadratic regularization

L1 norm → Non Quadratic Norm

L2 norm → Quadratic Norm

Cauchy regularizer → Non Quadratic Norm

Let's study the Cauchy norm invrsion

Seismic Deconvolution: Non-quadratic norms and sparse deconvolution

- •Non-quadratic norms can be used to retrieve sparse models
- •Sparse models area associated to broad-band solution
- •A sparse-spike solution can be used to retrieved sparse reflectivity series which is equivalent to finding blocky impedance profiles
- •Historically the problems arose in the context of impedance inversion (80's)

- Claerbout, J. F., and F. Muir, 1973, Robust modeling with erratic data: Geophysics, 38, 826-844.

- Taylor, H. L., S. C. Banks, and J. F. McCoy, 1979, Deconvolution with the L-one norm: Geophysics, 44, 39-52.

- Oldenburg, D. W., T. Scheuer, and S. Levy, 1983, Recovery of the acoustic impedance from reflection seismograms: Geophysics, 48, 1318-1337.

Cauchy Norm Deconvolution

Super-resolution via non-quadratic regularization

IRLS

Cauchy Norm Deconvolution

The problem is non-linear, the reflectivity depends on the reflectivity…

Iterative Re-weighted Least Squares

Cauchy Norm Deconvolution

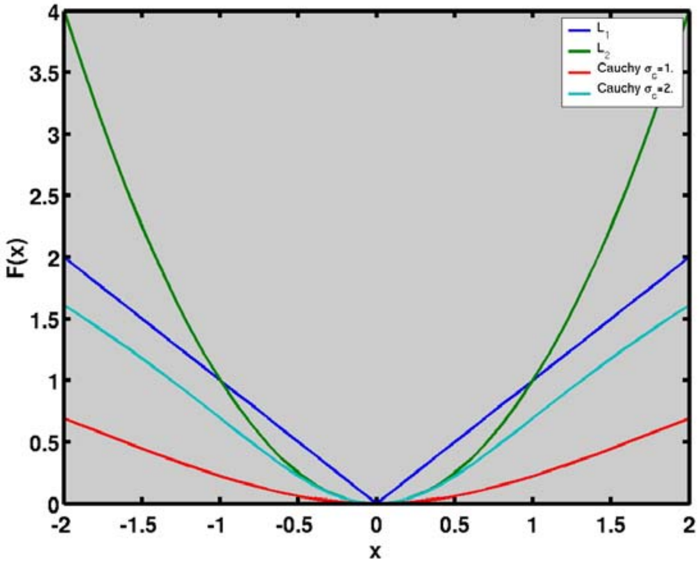
Fror large inverse problems, IRLS require some changes. The inversion can be performed by an semi-iterative solver :

- Conjugate Gradients (CG)
- Gauss-Seidel
- Pre-conditioned CG
- LSQR

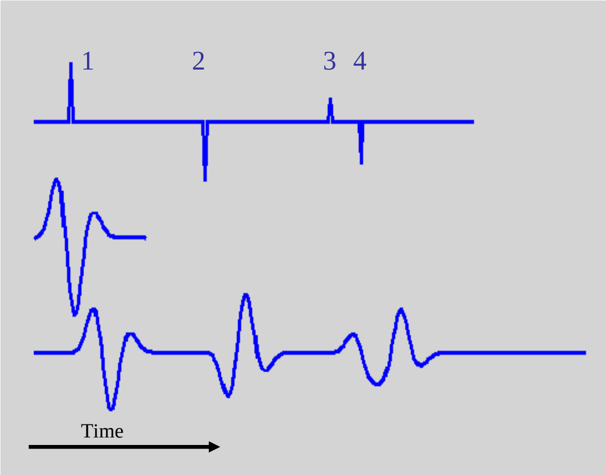
Iterative solvers permit one to compute an approximation to the solution of the linear system

Iterative Re-weighted Least Squares

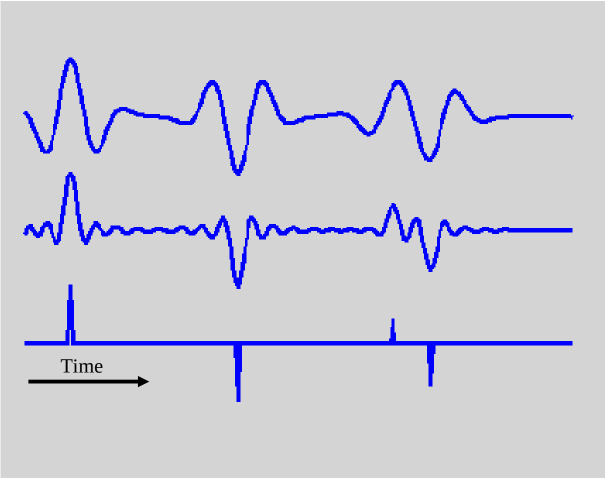
General Form



Seismic Deconvolution
Back to the original problem: estimation of r(t)



Seismic Deconvolution

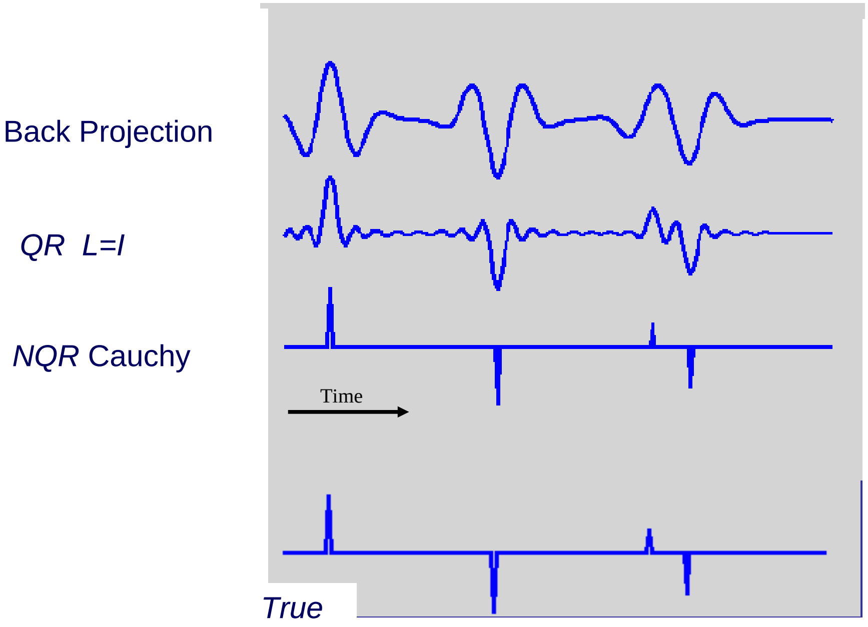


Back Projection

QR L=I

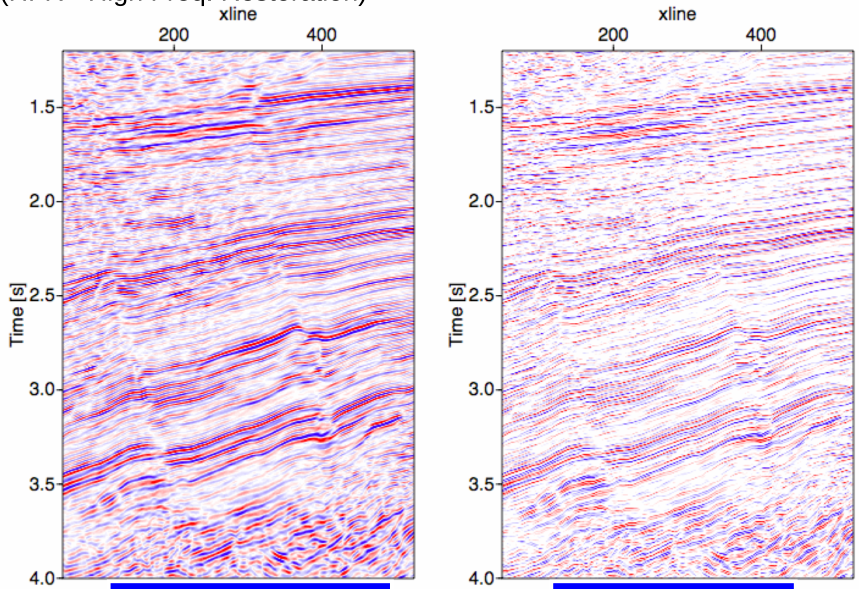
NQR Cauchy

Seismic Deconvolution



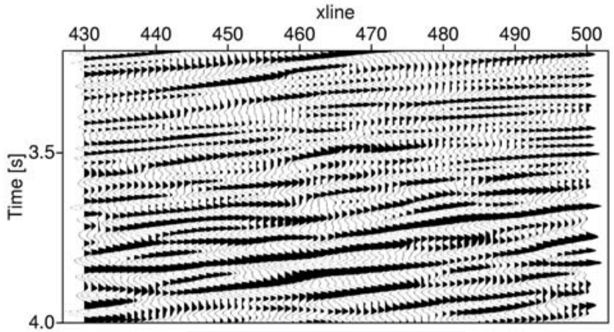
3D Seismic Section from somewhere

(HFR - High Freq. Restoration)

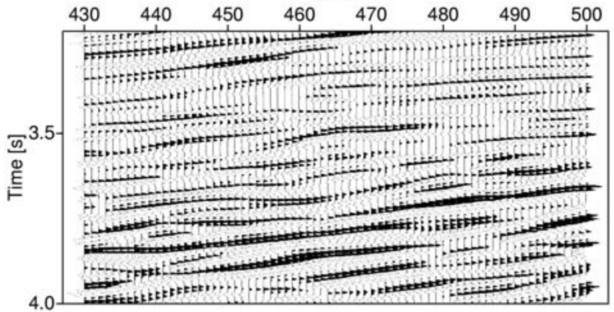


Conventional Deconv

High Res. Deconv



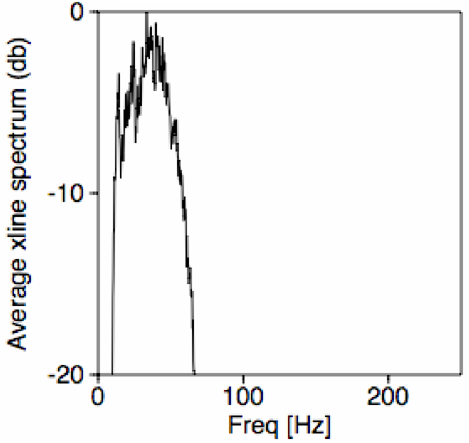
Conventional Deconv



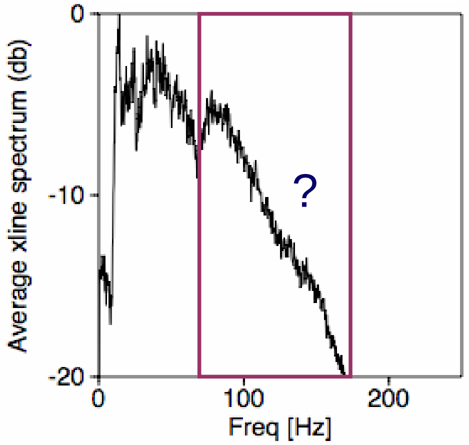
High Res. Deconv

Amplitude spectrum

Conventional Deconv



High Res. Deconv



Ongoing Research: High Freq recovery and scaling problem

Cauchy Norm Deconvolution: Algorithm

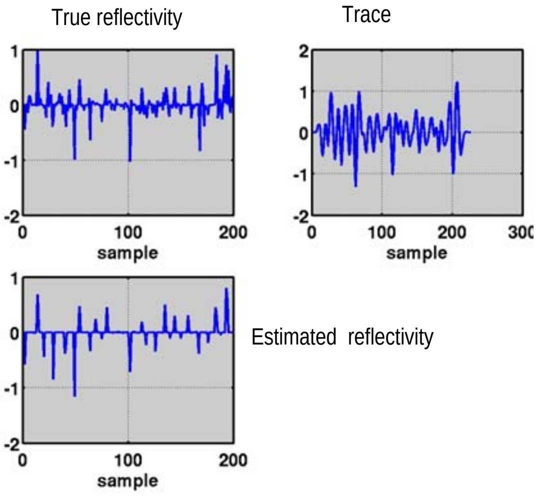
```
r = zeros(N,1);
sc=0.01;
mu =0.01
iter_max = 10;

R = W'*W;
iter_max;

for k=1:iter_max;

 Q = diag(1./(sc^2+r.^2));
 Matrix = R + mu*Q;
 r = inv(Matrix) * W'*s;

end;
```



Cauchy Norm Deconvolution: Algorithm

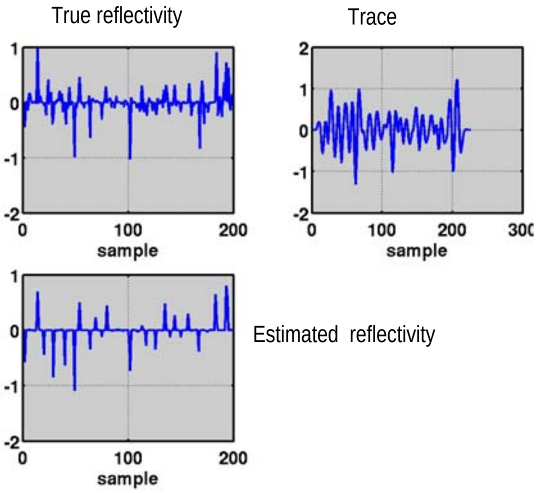
```
r = zeros(N,1);
sc=0.01;
mu = .01
iter_max = 10;
R = W'*W;

for k=1:iter_max;

 Q =diag(1./(sc^2+r.^2));
 Matrix = R + mu*Q;
 r = inv(Matrix) * W'*s;

 sc = 0.01*max(abs(r));

end;
```



Adaptive version

High frequency imaging methods….

Other methods exits - they all attempt to retrieve a sparse reflectivity sequence:

Atomic Decompostion/Matching Pursuit / Basic Pursuit (like an L1)

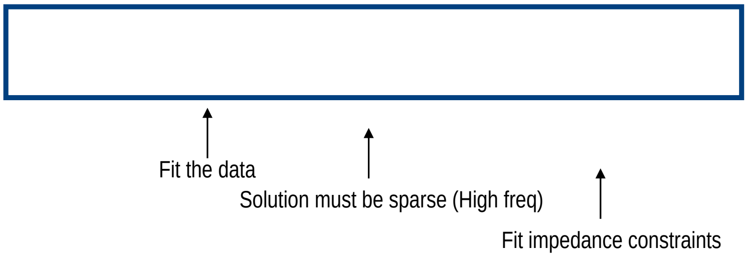
Chopra, S., J. Castagna, and O. Portniaguine, 2006, Seismic Resolution and Thin-Bed Reflectivity Inversion: Recorder, 31, 19-25. (www.cseg.ca)

Portniaguine, O., and J. P. Castagna, 2004, Inverse spectral decomposition: 74th Annual International Meeting, SEG, Expanded Abstracts, 1786-1789.

Cauchy Norm Deconvolution: with impedance constraints

Reflectivity as a function of P-impedance

Approximation

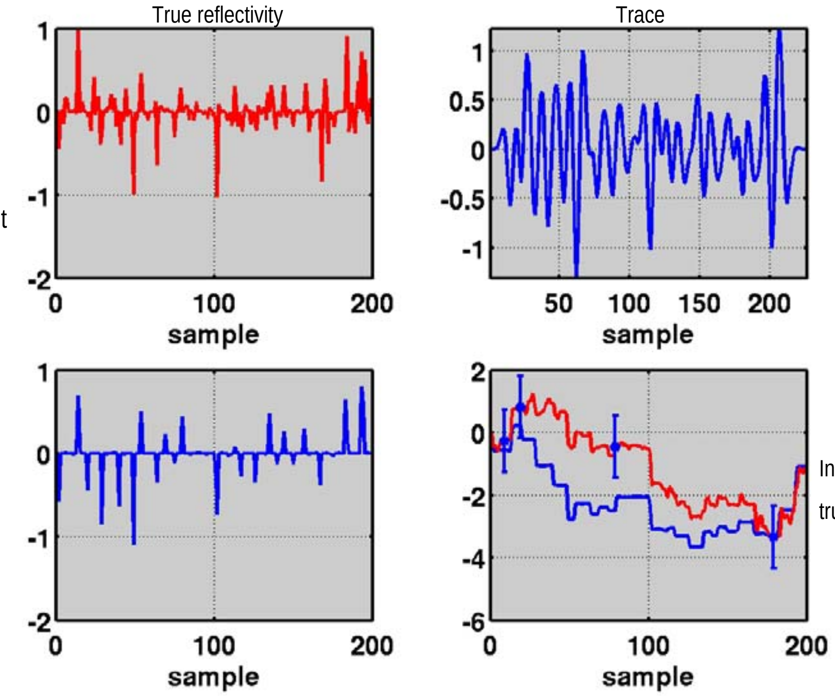


Fit the data

Solution must be sparse (High freq)

Fit impedance constraints

Cauchy Norm Deconvolution: with impedance constraints

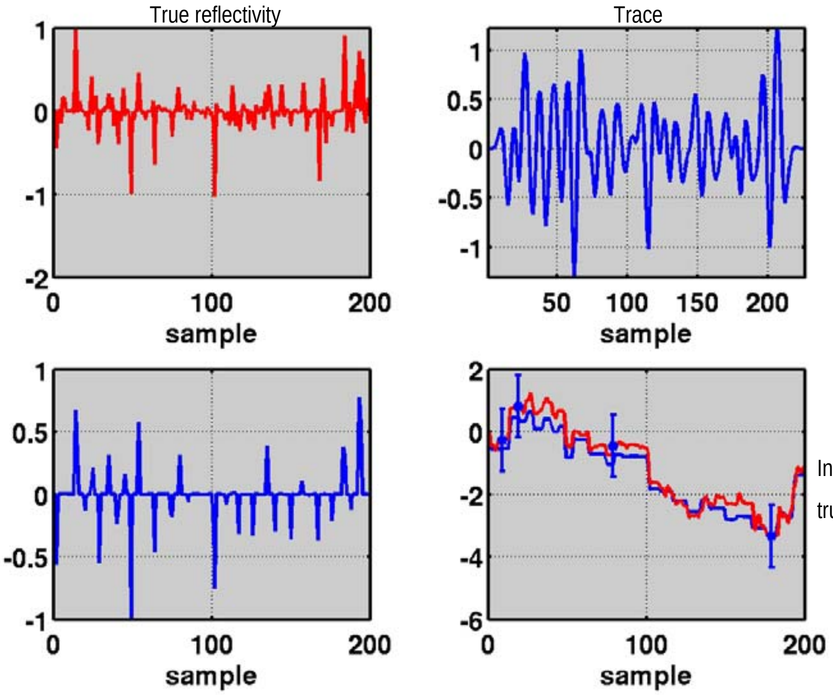

True reflectivity

Trace

Constraints are not honored

Inverted impedance & true impedance (red)

Cauchy Norm Deconvolution: with impedance constraints


True reflectivity

Trace

Constraints are honored

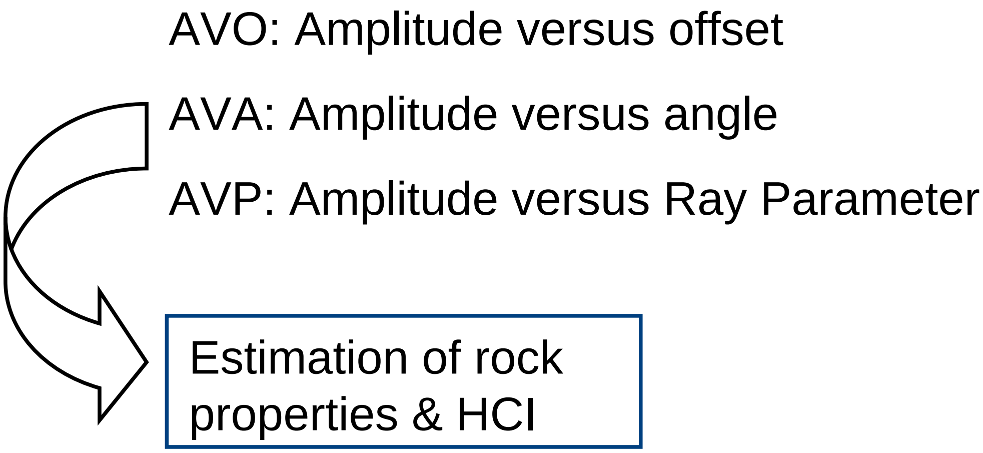
Inverted impedance & true impedance (red)

Cauchy Norm Deconvolution: with impedance constraints - Algorithm

```
beta=0.25
iter_max=10
R = W'*W+beta*C'*C;
r = zeros(N,1);

for k=1:iter_max;
  Q = diag(1./(sc^2+r.^2));
  Matrix = R + mu*Q;
  r = inv(Matrix) * (W'*s+beta*C'*psi)
  sc = 0.01*max(abs(r));
 end;
```

AVO



AVO: Amplitude versus offset

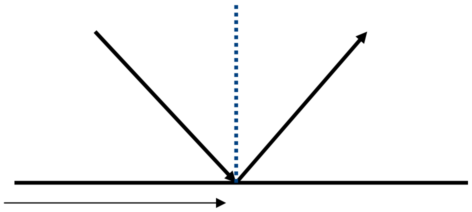
AVA: Amplitude versus angle

AVP: Amplitude versus Ray Parameter

Estimation of rock properties & HCI

Example: Multi-channel Seismic Deconvolution

Sensing a reflector



Now we consider a suite of seismograms
as a function of angle of incidence

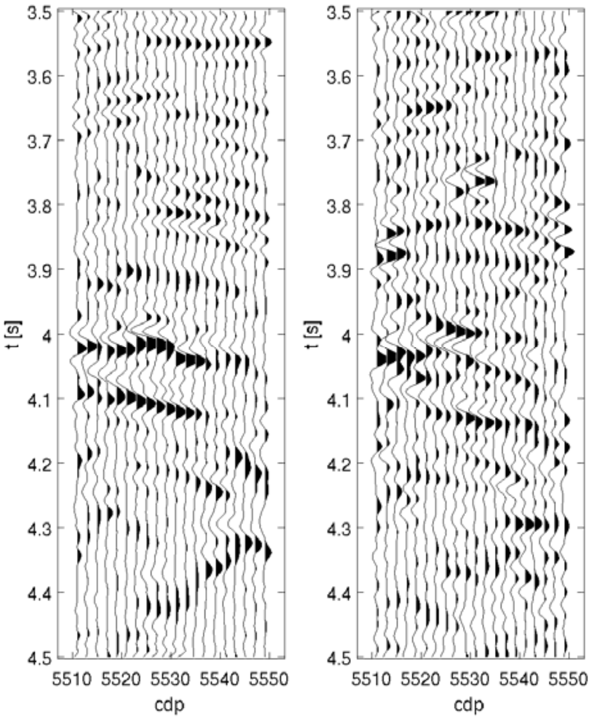
Goal: Estimation of AVO classes

Models for clay/sand interfaces
for consolidated and
unconsolidated sands

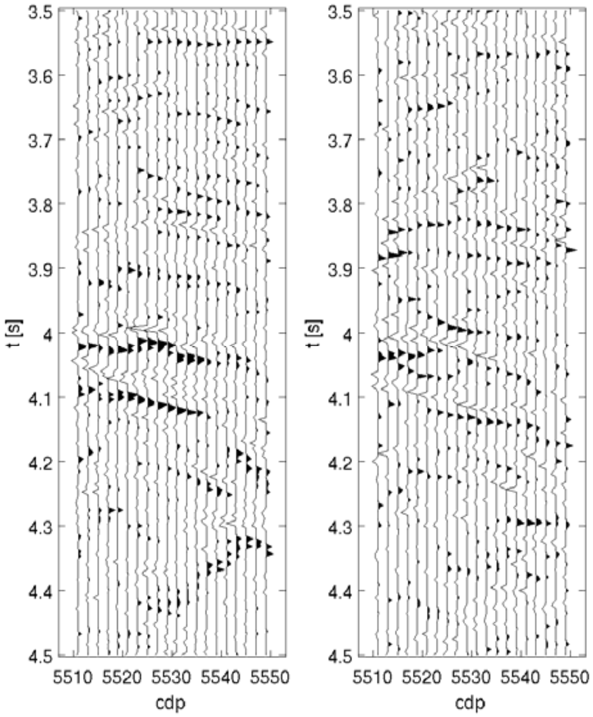
QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Quadratic regularization

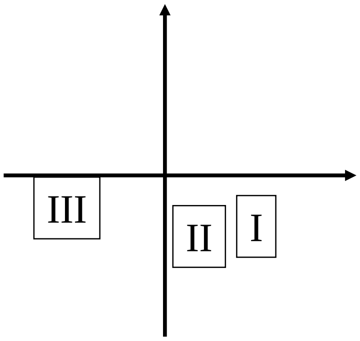
L=I



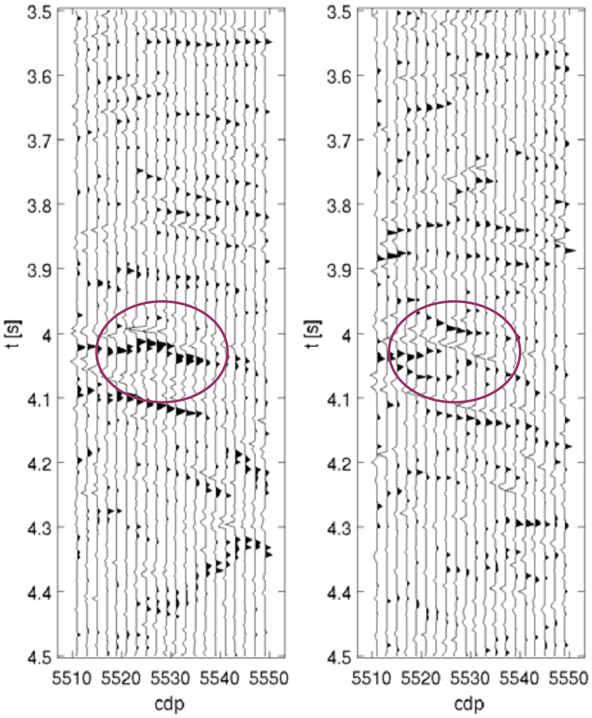
Cauchy Solution



Interpretation/
AVA Analysis



Identify gas
bearing sand lens



Large inverse problems: Pre-stack depth Imaging

Structural Imaging -- ***Where ?***

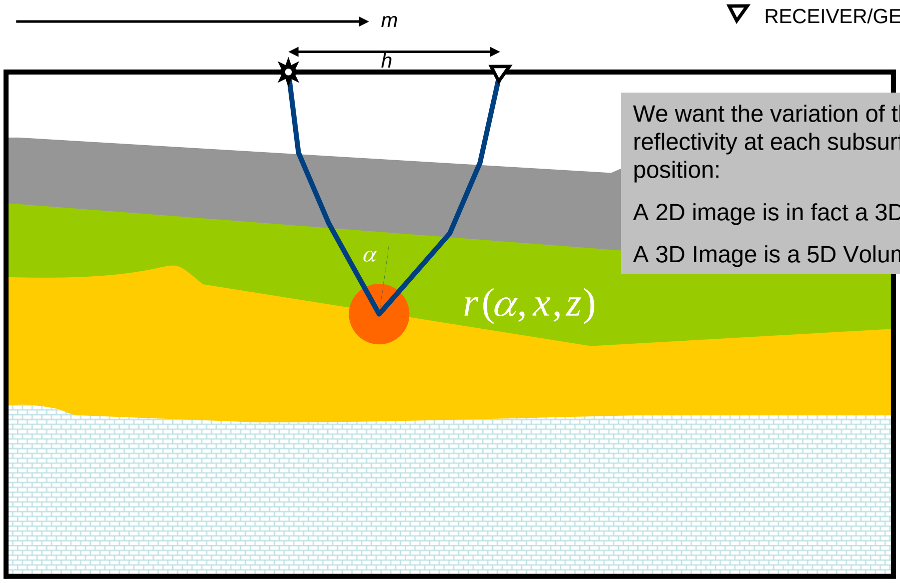
Imaging Angle Dependent Reflectivity -- ***Where and What ?***

Wang J., Kuehl H. and Sacchi M.D., 2005, High-resolution wave-equation AVA imaging: Algorithm and tests with a data set from the Western Canadian Sedimentary Basin: *Geophysics*, 70, 891-899

Imaging - Large scale problems

✸ SOURCE (s)

▽ RECEIVER/GEOPHONE (g)



We want the variation of the reflectivity at each subsurface position:

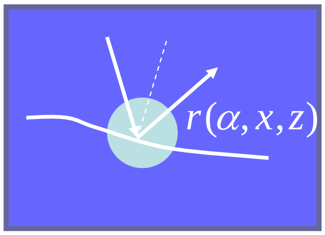
A 2D image is in fact a 3D volume

A 3D Image is a 5D Volume

m: midpoint
h: offset

Wang, Kuehl and Sacchi, Geophysics 2005

Imaging in Operator Form..



d: is now pre-processed seismic data

Migration/Inversion Hierarchy..

Modeling

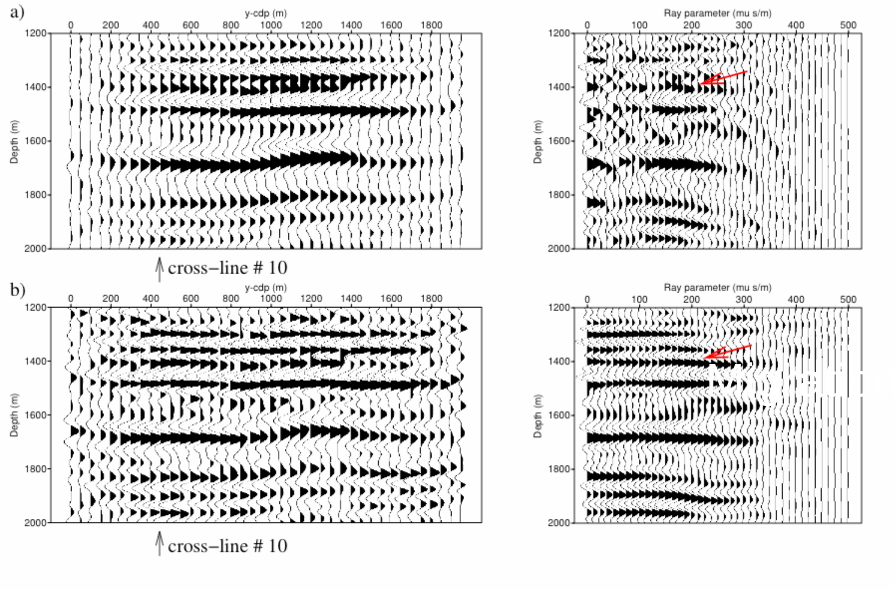
Migration

Inversion

Regularized Imaging

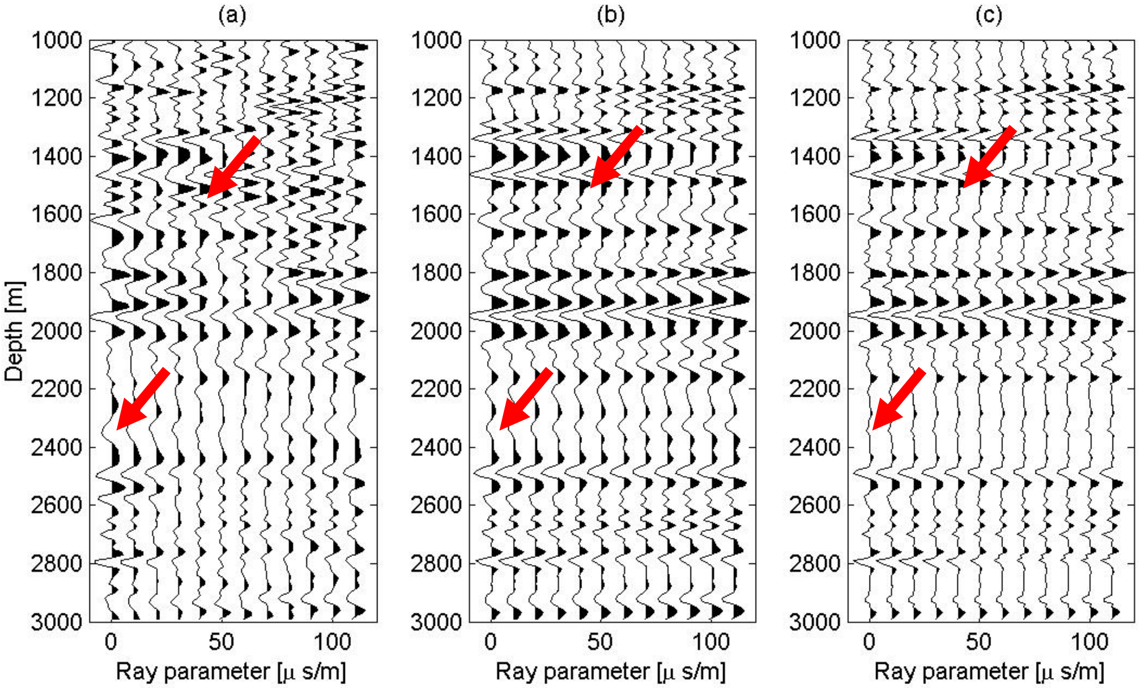
Remarks:

- We don’t have *A*, we only have a code that knows how to apply *A* to *m* and, another, that knows how to apply *A’* to *d*
- *A* and *A’* are built to pass the dot product test
- *Importance of sampling Matrix W for reducing footprints (Kuehl, 2002, PhD Dissertation)*



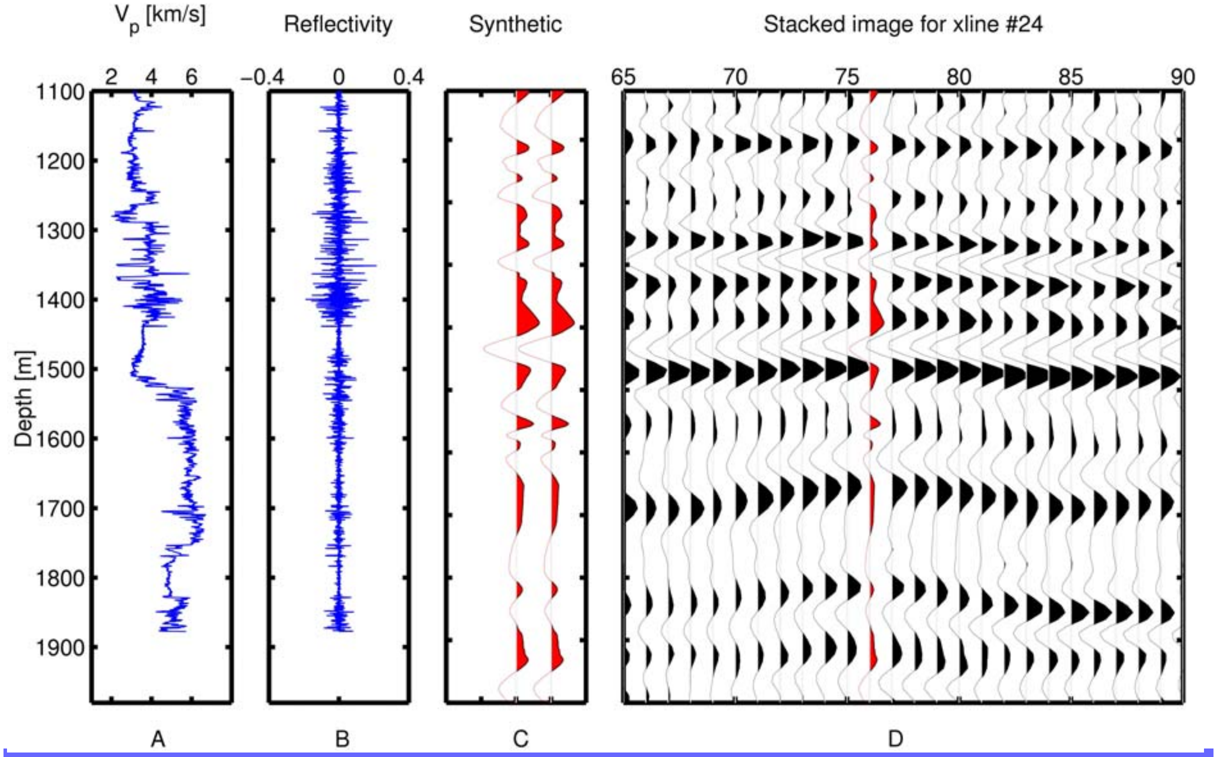
Migration

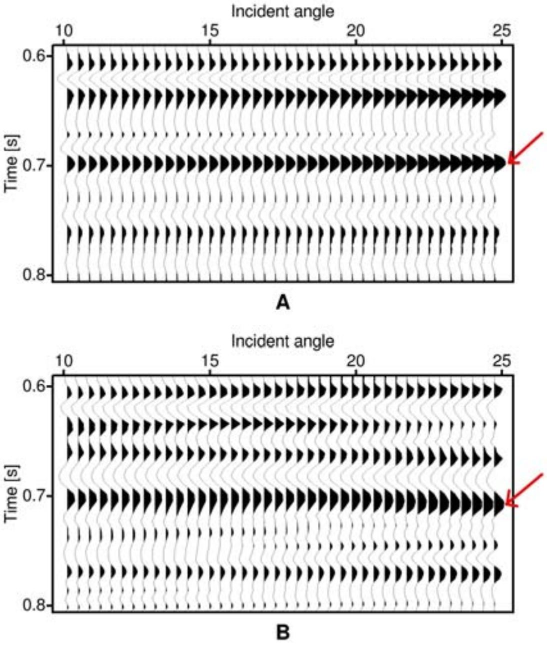
Inversion

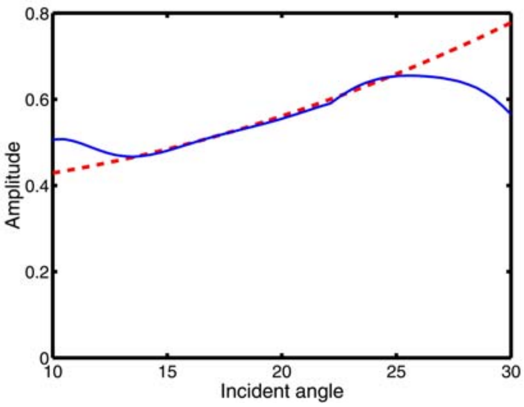


Migration **QR** **NQR**

Synthetic trace







Red: synthetic
Blue: inverted

A:synthetic B: inverted