Part 2

Inverse Problems for Geophysicists: Introduction to Linear Inverse Problems and Regularization Methods

M D Sacchi
UNLP Problemas Inversos

Inverse Problems

- An Inverse Problem is a mathematical problem where one attempts to estimate models that explain observations. We often name the observations **d** (data)

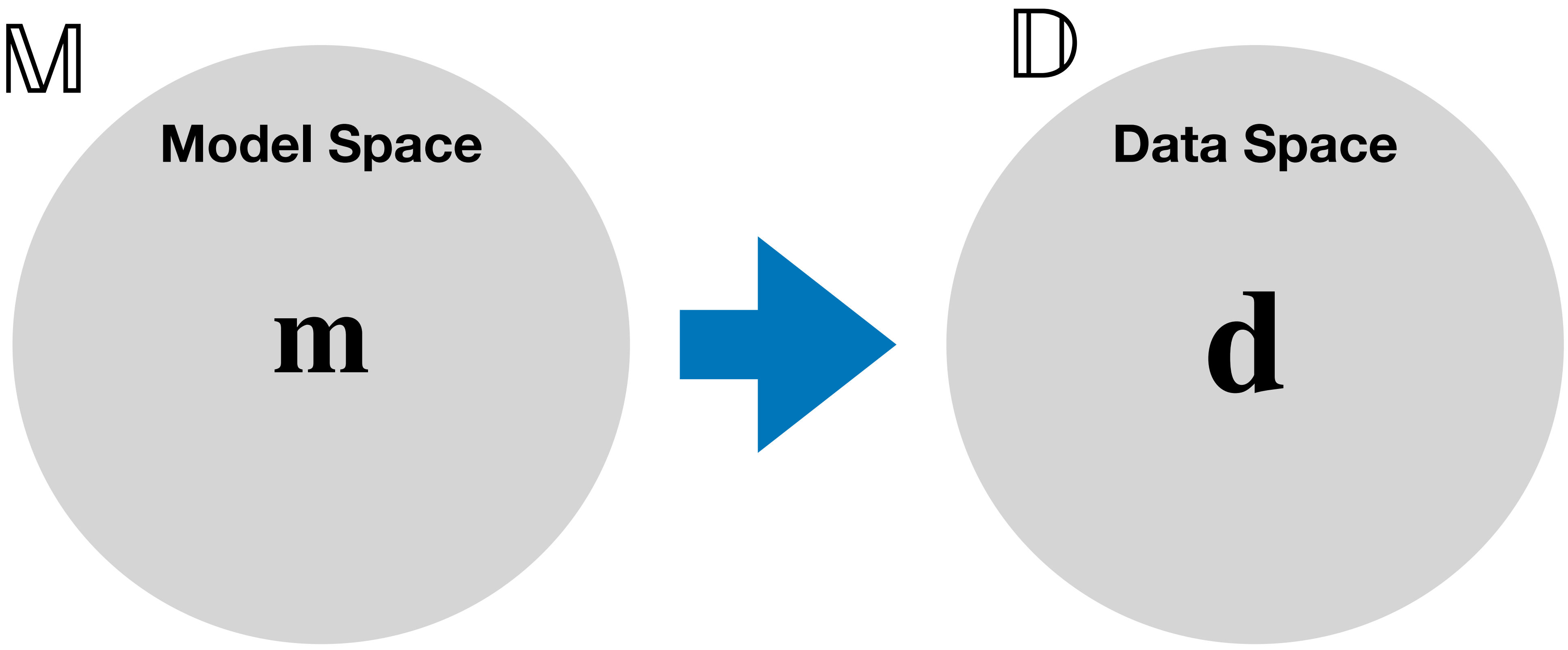
- Observations are generally measured on the surface of the earth and are discrete in time and space

- The subsurface is described by properties (density, velocity, reflectivity, resistivity, etc). These properties exist everywhere. We will refer to these properties as $m(x,y,z)$ or **m** (model parameters)

- $m(x,y,z)$ is the distribution of some property that can be discretized and represented via the vector **m**

Two spaces

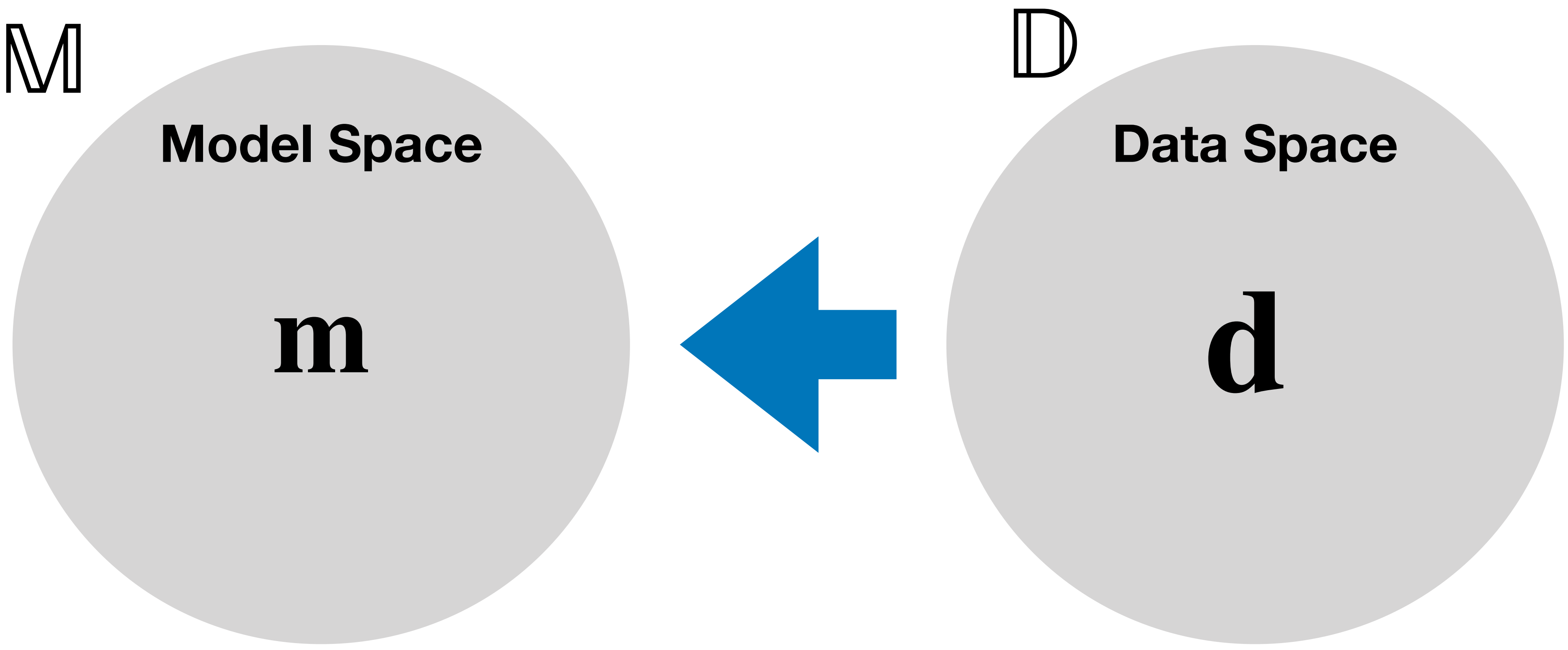
- Forward Problem



$$F[\mathbf{m}] = \mathbf{d}$$

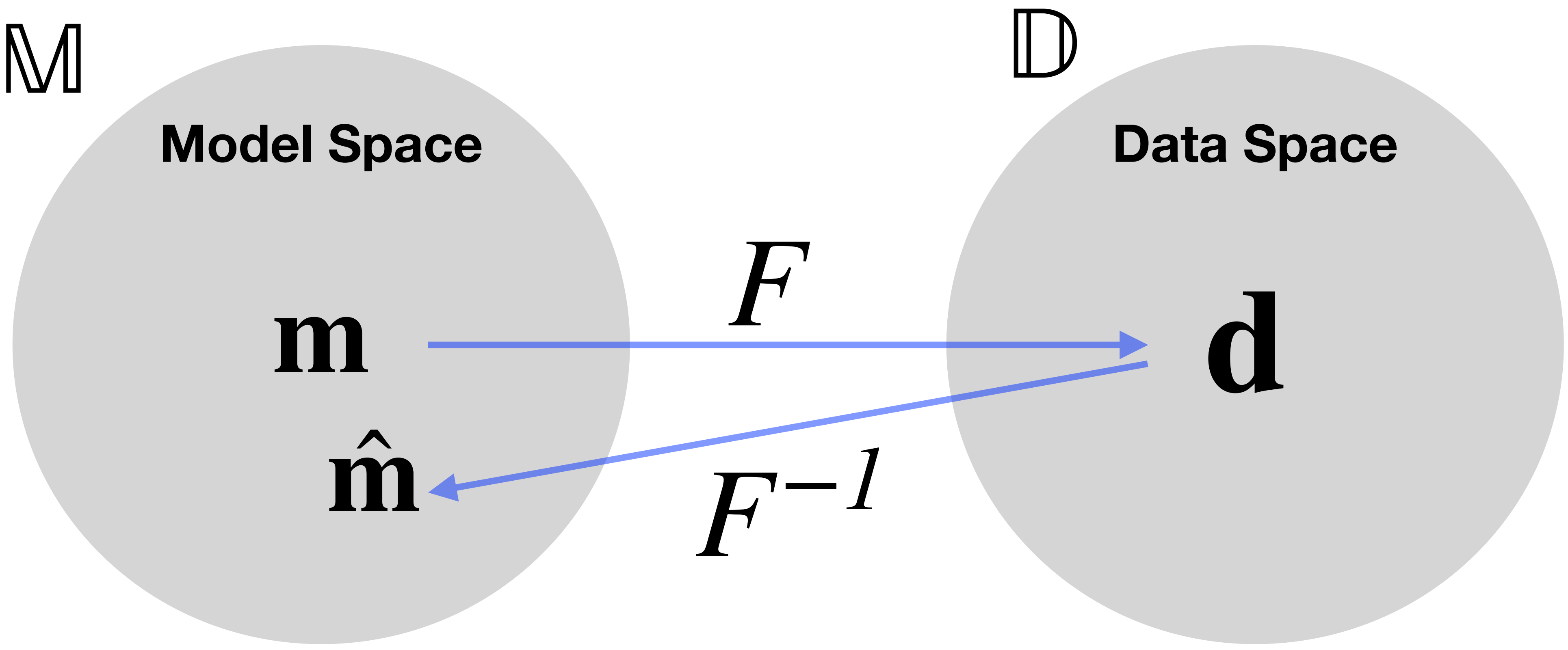
Two spaces

- Inverse Problem



$$F^{-1}[\mathbf{d}] = \mathbf{m}$$

It is more complicated...



$\hat{\mathbf{m}}$: **Solution**

Geophysics and Inverse Problems

Data (What you can measure)	Model (What you would like to know)	Method
Gravity anomalies	Density	Potential Field Methods
Electrical potential	Resistivity	Potential Field Methods
Electrical and Magnetic Field	Electrical conductivity	EM/MT methods
Magnetic Fields	Susceptibility	Potential Field Methods
Seismic Waves	Velocities	Seismic Methods

ILL-posed problems

- Well-posed problem. A problem is said to be well-posed when
  - There is a solution
  - The solution is unique
  - The solution is stable
- If one of the above is not true, the problem is called an ill-posed problem
- Typical geophysical inverse problems are ill-posed problems (2 and 3 are not true)

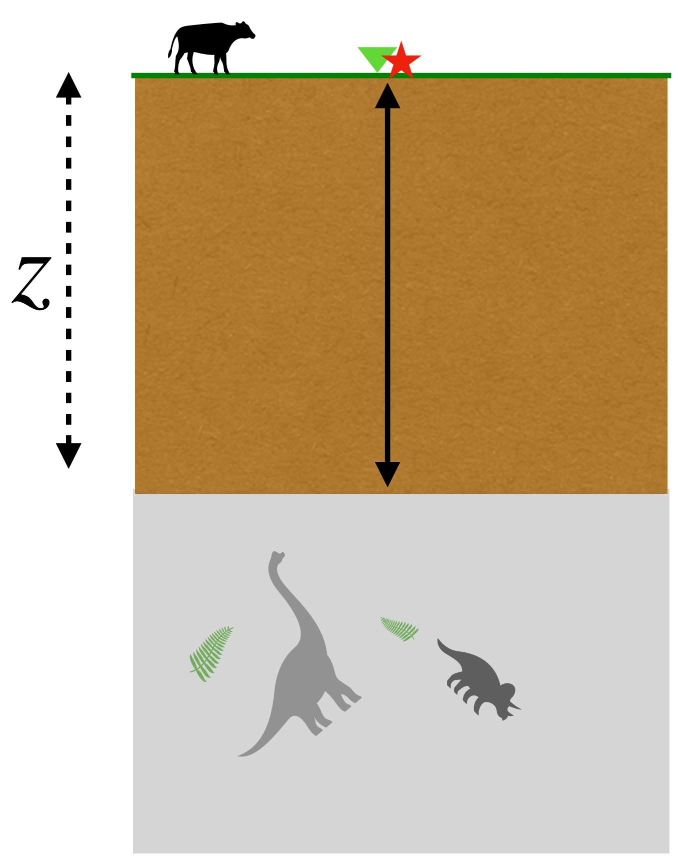
Jacques Hadamard, 1865-1963

ILL-posed problems

- There is a solution: YES
  - There is a solution otherwise we wouldn’t be here
  - e.g. Rocks have density causing gravity anomalies

ILL-posed problems

- The solution is unique: Generally NO
- For instance, Depth-Velocity ambiguity: $t = \dfrac{2z}{v}$
- Assume two-way traveltime for a vertical incidence plane wave



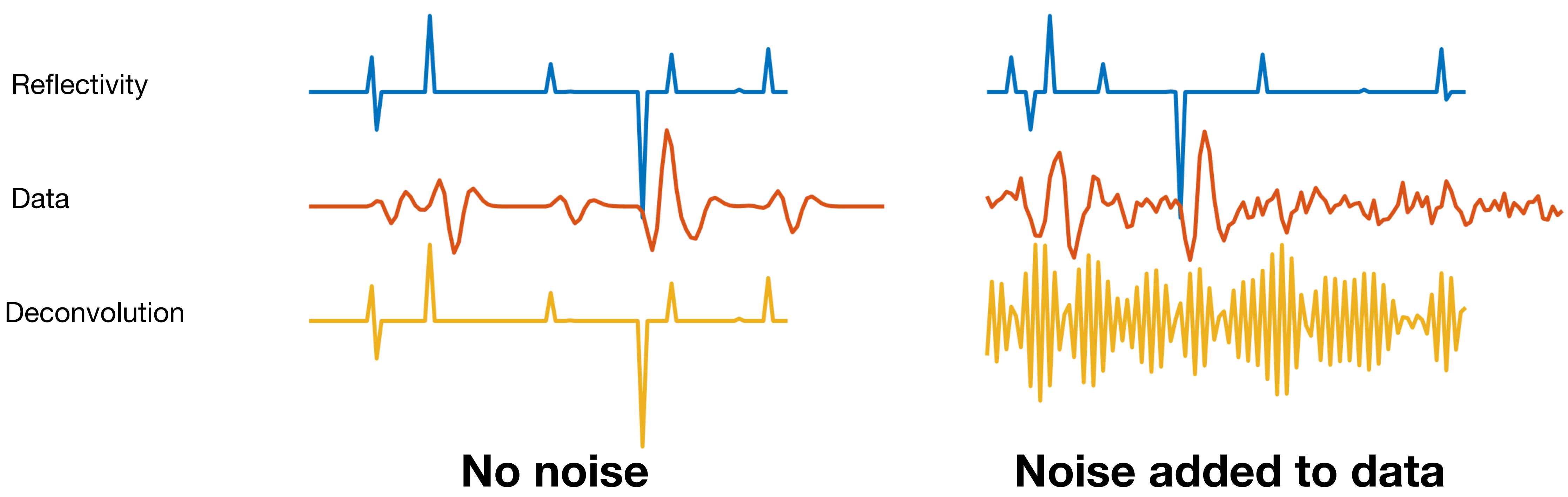
$$t = \frac{2 \times 1000m}{2000m/s} = \frac{2 \times 1500m}{3000m/s}$$

Sol 1	Sol 2
1000m	1500m
2000m/s	3000m/s

Clearly, there is an infinite number of solutions

ILL-posed problems

- The solution is stable: Not true
  - A small perturbation in the data causes a large perturbation in the solution
  - e.g. Deconvolution



ILL-posed problems

- The solution is stable: Not true
  - A small perturbation can cause a large perturbation in the solution
  - Assume perfect data and an invertible operator

$$d = F(m) \rightarrow \hat{m} = F^{-1}(d)$$

  - Assume perturbed data (inaccurate data)

$$\hat{m} + \delta m = F^{-1}(d + n)$$

ILL-posed problems

- The solution is stable: Not true
  - Solution for perturbed data

$$\hat{m} + \delta m = F^{-1}(d + n)$$

  - The solution is not stable. The latter means

$$if \, |n| = small \, , \, |\delta m| = large$$

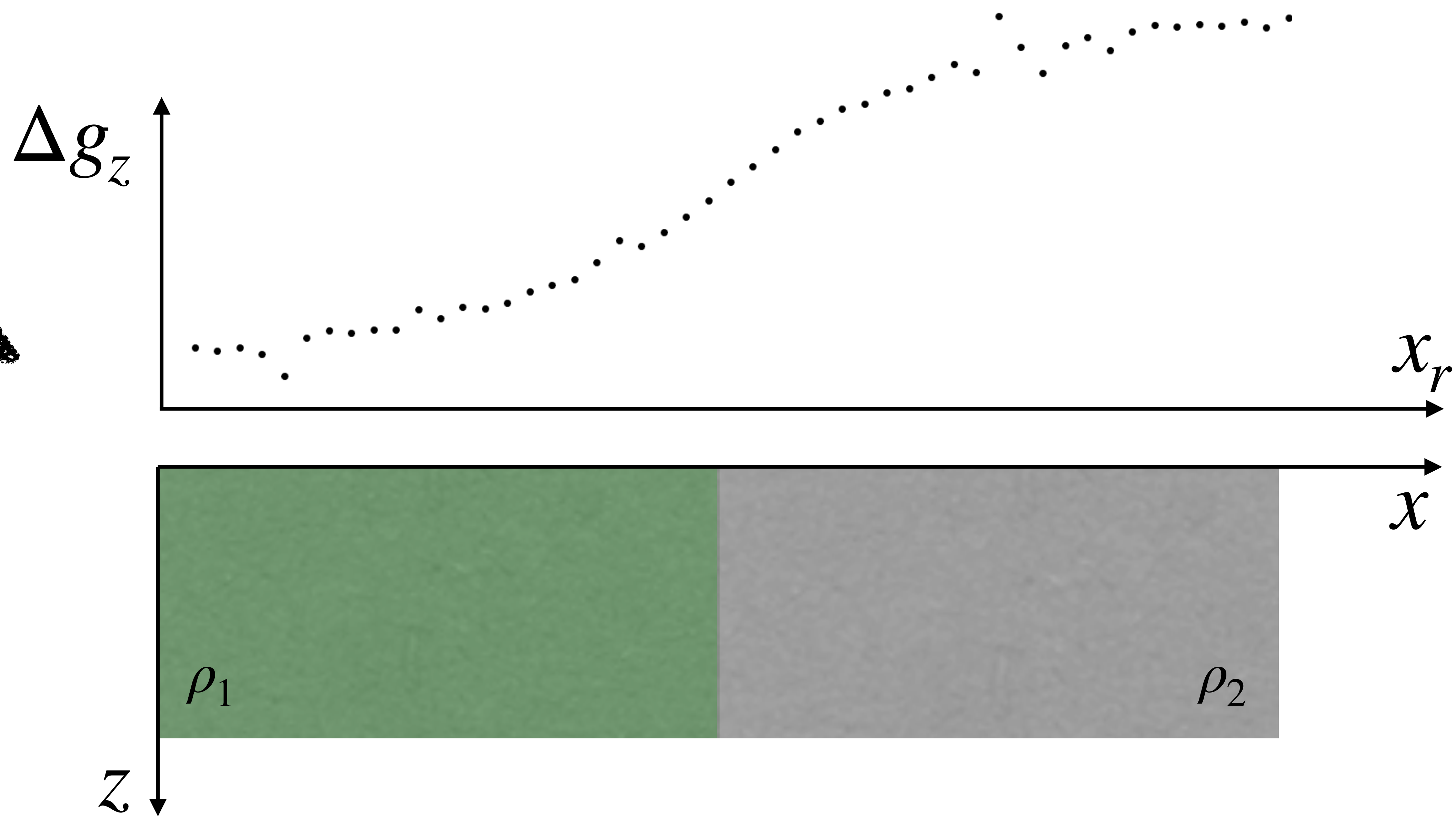
  - This is the main problem we face in geophysical inversion and often in data processing. You can think that somewhere one is trying to divide by small numbers and hence, whatever you are doing to your data leads to “noise amplification”.

Why geophysical problems are ill-posed?

- Not enough data (insufficient spatial data or insufficient bandwidth)
- Presence of noise also leads to unstable solutions
- A central characteristic of the forward problem:
    - The Forward Problem smoothes the unknown subsurface properties. Seismic waves, gravity anomalies, electrical potential, etc are nice smooth functions of space and time. Subsurface properties, on the other hand, might or might not be smooth!
- The next slide illustrates the aforementioned concept

Example: Inversion of gravity anomalies

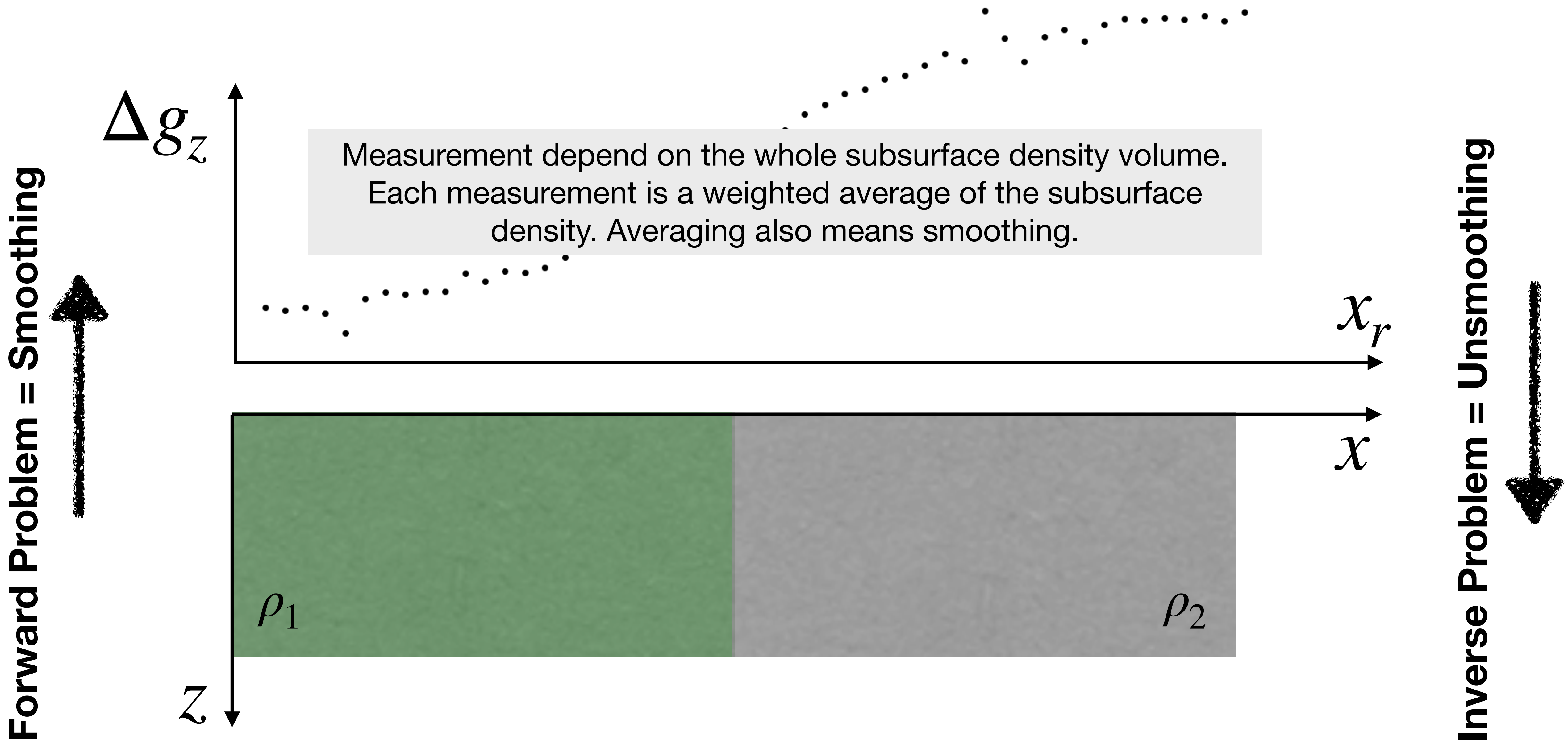
Forward Problem = Smoothing



Inverse Problem = Unsmoothing

$$\Delta g_z(x_r, z_r = 0) = \int_{x,y} G(x, y \,|\, x_r, z_r = 0)\, \Delta\rho(x, y)\, dx\, dy$$

Example: Inversion of gravity anomalies



$$\Delta g_z(x_r, z_r = 0) = \int_{x,y} G(x, y \,|\, x_r, z_r = 0)\, \Delta\rho(x, y)\, dx\, dy$$

Key points

- Previous example shows an interesting feature of the Forward Problem:
  - Observations (data) are a weighted average of density. The averaging kernel smoothes the density anomaly (The physics, in this case given by Newton's gravitational attraction, is the cause of this behaviour)
  - Inverse Problem: Recovering the density from the data entails the opposite of smoothing (un-smoothing); an unstable operation
  - *Smoothing* = is a low pass operation = Stable
  - *Un-smoothing* = is a high pass operation = Unstable
  - Solving an inverse problem in many cases entails controlling instability

Key points

- Subsurface properties (such as density anomalies) are a function of space (that exists everywhere). The observed gravity anomalies are given by a finite number of observations

$$d(\mathbf{r}_j) = \int G(\mathbf{x} \,|\, \mathbf{r}_j) m(\mathbf{x}) d\mathbf{x} \qquad j = 1 \ldots N$$

- So inversion, theoretically, attempt to go from a finite number of observations to a function

$$\mathbf{d} \in R^N, m(\mathbf{x}) \in R^\infty$$

- There is a natural underdeterminacy in an inverse problem

Linear inverse problems

- Many inverse problems in seismology entail solving integral equations. Others entail solving PDEs. We will start with simple problems that can be written as integral equations that after discretization lead to discrete system of equations.
  - Integral equations
  - Linear discrete system of equations
  - Regularization methods
  - Connection to exploration seismology:
    - Deconvolution
    - AVO inversion
    - Linearized seismic imaging (Forward and Adjoint Operators)
      - Migration & Least-Squares migration

Linear inverse problems

- Fredholm integral equation of 1st kind

$$d(r_j) = \int_X G(r_j, x)\, m(x)\, dx \qquad j = 1 \ldots N$$

- We can discretize the model

$$x_k = x_0 + (k-1)\Delta x \qquad k = 1 \ldots M$$

$$d(r_j) = \sum_{k=1}^{M} G(r_j, x_k)\, m(x_k)\, \Delta x \qquad j = 1 \ldots N$$

- Which leads to a system of equations

$$\mathbf{d} = \mathbf{L}\mathbf{m}, \quad \mathbf{d} \in R^N, \quad \mathbf{m} \in R^M$$

Examples

$$d(r_j) = \int_X G(r_j, x)\, m(x)\, dx \qquad j = 1 \ldots N$$

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

Convolution

$$T(r_j) = \int_0^{r_j} s(l)\, dl$$

Travel-time tomography

$$d(\omega, s_i, r_j) = \int_x \int_z B(\omega, x, y \,|\, s_i, r_j) m(x, z) dx\, dz$$

Born imaging

TWO IMPORTANT MATRICES (OPERATORS)

Forward and Adjoint Operators

- **Two special operations (or operators?)**

  Forward : $\mathbf{d} = \mathbf{L}\mathbf{m}$

  Transpose or adjoint : $\tilde{\mathbf{m}} = \mathbf{L}^T\,\mathbf{d}$

- $\mathbf{L}$ is a linear operator (Forward modelling operator).

- Why are them special?

  - Iterative solvers for large inverse problems only need to evaluate $\mathbf{L}[\,\cdot\,]$ and $\mathbf{L}^T[\,\cdot\,]$
    We pretend these are Matrices but in reality these are linear operators (codes)

  - I usually interpret operators as matrices. In reality, operators are codes that apply an operation *on the flight* (implicit form)

  - *We will see these operators everywhere today*

Forward and Adjoint Operators

- **When $\mathbf{L}$ is a real matrix**

$$\mathbf{L}' = \mathbf{L}^T$$

- When $\mathbf{L}$ is a complex matrix

$$\mathbf{L}' = \mathbf{L}^H$$

Forward and Adjoint Operator

- **Two special operations (or operators)**

$$\textbf{Forward} : \mathbf{d} = \mathbf{L}\mathbf{m} \quad (1)$$

$$\textbf{Transpose or adjoint} : \tilde{\mathbf{m}} = \mathbf{L}^T \mathbf{d} \quad (2)$$

- Replace (1) into (2)

$$\tilde{\mathbf{m}} = \mathbf{L}^T \mathbf{L}\mathbf{m}$$

- Questions:
  - Is the *adjoint model* a good representation of the *true model* ?
  - Can I remove the distortion ?

See: Migration vs. Least-squares Migration: *Least-squares migration of incomplete reflection data. Nemeth et al. GEOPHYSICS (1999), 64(1)*

Forward and Adjoint Operators

- **Two special operations (or operators)**

Forward : $\mathbf{d} = \mathbf{L}\mathbf{m}$ **(1)**

Conjugate transpose or adjoint : $\tilde{\mathbf{m}} = \mathbf{L}^T\,\mathbf{d}$ **(2)**

(2) into (1)

$$\tilde{\mathbf{m}} = \mathbf{L}^T\,\mathbf{L}\mathbf{m}$$

$\tilde{\mathbf{m}}$ is a distorted version of $\mathbf{m}$

Forward and Adjoint Operators

Forward : $\mathbf{d} = \mathbf{L}\mathbf{m}$

Conjugate transpose or adjoint : $\tilde{\mathbf{m}} = \mathbf{L}^T\,\mathbf{d}$

$$\tilde{\mathbf{m}} = \mathbf{L}^T\,\mathbf{L}\mathbf{m}$$

$$\tilde{\mathbf{m}} \approx \mathbf{m} \;\; \textit{if} \;\; \mathbf{L}^T\mathbf{L} \approx \mathbf{I}$$

The adjoint, in some instances, is good enough to estimate $\mathbf{m}$. An example is RTM which can be considered the adjoint of Born modelling

Forward and Adjoint Operators

Forward : $\mathbf{d} = \mathbf{L}\mathbf{m}$

Conjugate transpose or adjoint : $\tilde{\mathbf{m}} = \mathbf{L}^T \mathbf{d}$

$$\tilde{\mathbf{m}} = \mathbf{L}^T \mathbf{L}\mathbf{m}$$

$$\mathbf{L}^T\mathbf{L} \approx \textbf{diag}(\mathbf{L}^T\mathbf{L}) = \mathbf{D}, \qquad \hat{\mathbf{m}} = \mathbf{D}^{-1}\tilde{\mathbf{m}}$$

This is an inexpensive way of partially correcting the adjoint to make it look closer to the desired solution

Forward and Adjoint Operators

Example: Averaging

$$d = [1/3 \;\; 1/3 \;\; 1/3] \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \mathbf{Lm}$$

Averaging operator

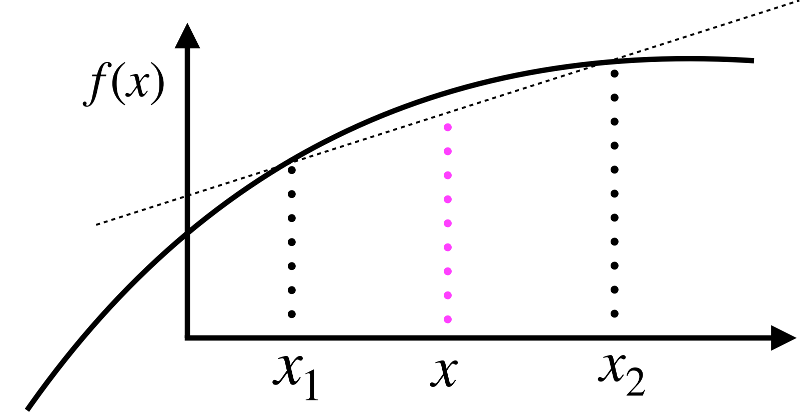
$$\tilde{\mathbf{m}} = \begin{bmatrix} \tilde{m}_1 \\ \tilde{m}_2 \\ \tilde{m}_3 \end{bmatrix} = \begin{bmatrix} d/3 \\ d/3 \\ d/3 \end{bmatrix} = \mathbf{L}^T d$$

Adjoint of the averaging operator

Question: Can you recover $\mathbf{m}$ from $\tilde{\mathbf{m}} = \mathbf{L}^T \mathbf{Lm}$?

Forward and Adjoint Operators

Example: **Linear Interpolator (this is used a lot!!)**



Linear Interpolator (LI)

$$f(x) = a\,f(x_1) + b\,f(x_2)$$

$$f(x) = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \end{bmatrix}$$

$$\begin{bmatrix} \tilde{f}(x_1) \\ \tilde{f}(x_2) \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} f(x)$$

Adjoint of LI

$$\tilde{f}(x_1) = af(x)$$
$$\tilde{f}(x_2) = bf(x)$$

Forward: 2 samples make 1, Adjoint: 1 sample makes 2

DISCRETE LINEAR INVERSE PROBLEMS IN EXPLICIT FORM (WITH MATRICES)

Discrete Linear Inverse Problems

$$\mathbf{d} = \mathbf{L}\mathbf{m}, \quad \mathbf{L} \in \mathscr{R}^{N \times M}$$

- A) $N>M$: More observations than data = Overdetermined problem
- B) $N<M$: Less observations than data = Underdetermined problem
- Let's consider Problem B)
  - Underdetermined Problem with Accurate Data
  - Underdetermined Problem with Noisy (inaccurate) Data

Discrete Linear Inverse Problems

(1) Accurate data $\mathbf{d} = \mathbf{L}\mathbf{m}$

(2) Inaccurate data $\mathbf{d} = \mathbf{L}\mathbf{m} + \mathbf{e} \rightarrow \mathbf{d} \approx \mathbf{L}\mathbf{m}$

Consider discrete data and discrete vector of model parameters

Discrete Linear Inverse Problem with Accurate Data

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

$$\mathbf{d} \in R^N, \qquad \mathbf{m} \in R^M$$

We assume an under-determined problem $M > N$

- We have more unknowns than observations.
- Hence, we have an infinite number of solutions.
- One needs to pick one. Which one?

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- Among all possible solution select the one with minimum Euclidian norm. In other words,

$$\textbf{Minimize } \|\mathbf{m}\|_2^2$$

$$\textbf{Subject to } \mathbf{d} = \mathbf{L}\mathbf{m}$$

- The latter is solved using the Method of Lagrange Multipliers. We minimized an augmented cost function

$$J = \|\mathbf{m}\|_2^2 + \mathbf{b}^T(\mathbf{L}\mathbf{m} - \mathbf{d})$$

- $\mathbf{b} \in R^N$ is the vector of Lagrange Multipliers, one multiplier per observation

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- To solve the problem we need to minimize the cost respect to $\mathbf{m}$ and $\mathbf{b}$

$$J = \|\mathbf{m}\|_2^2 + \mathbf{b}^T(\mathbf{Lm} - \mathbf{d})$$

- Which leads to the so called minimum norm solution (I will do it in the whiteboard)

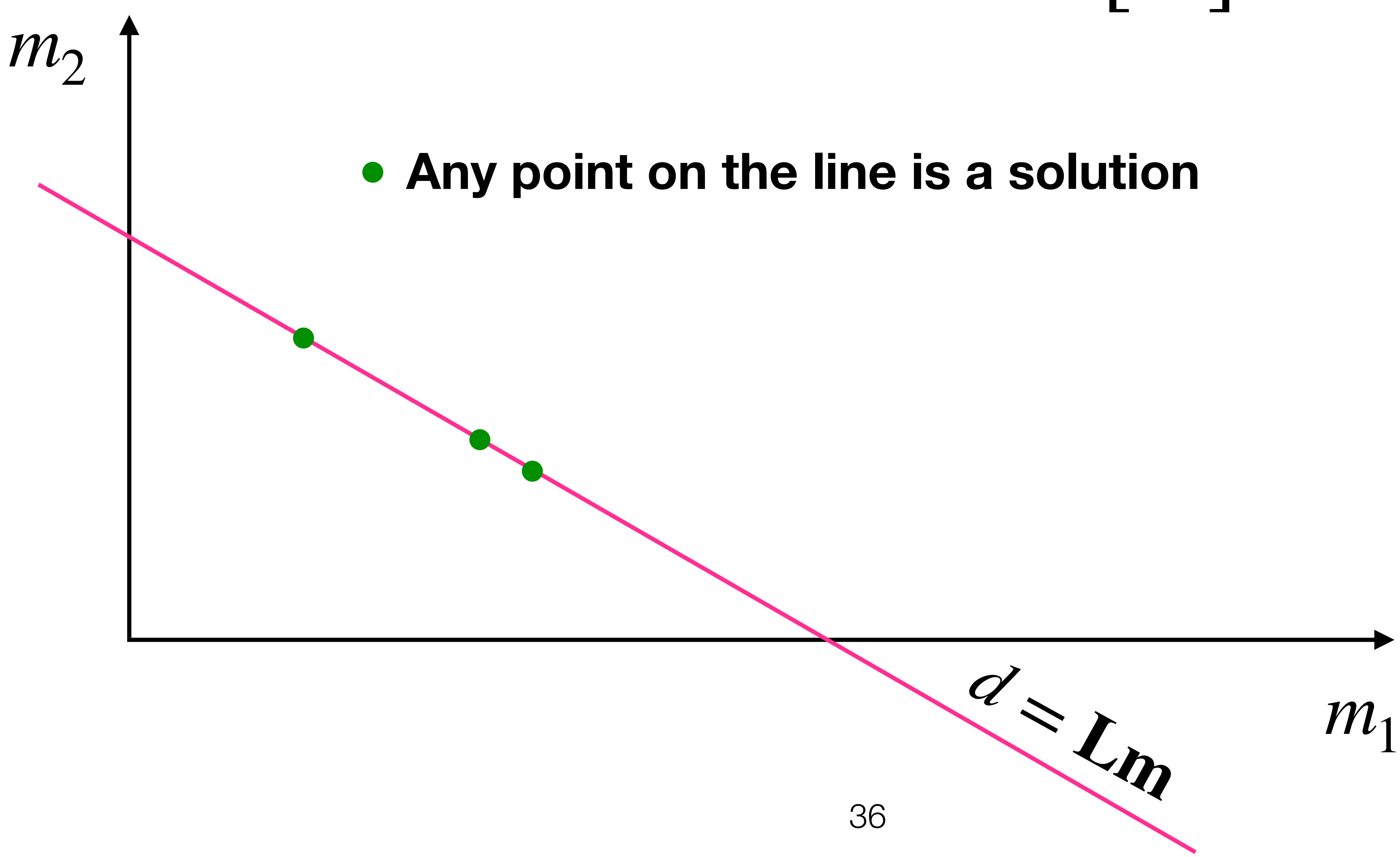
$$\mathbf{m}_{mn} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^{\mathbf{T}})^{-1}\mathbf{d}$$

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- $N=1$, $M=2$ (2 unknowns and 1 equation)

$$d = \mathbf{Lm} = \begin{bmatrix} L_{11} & L_{12} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

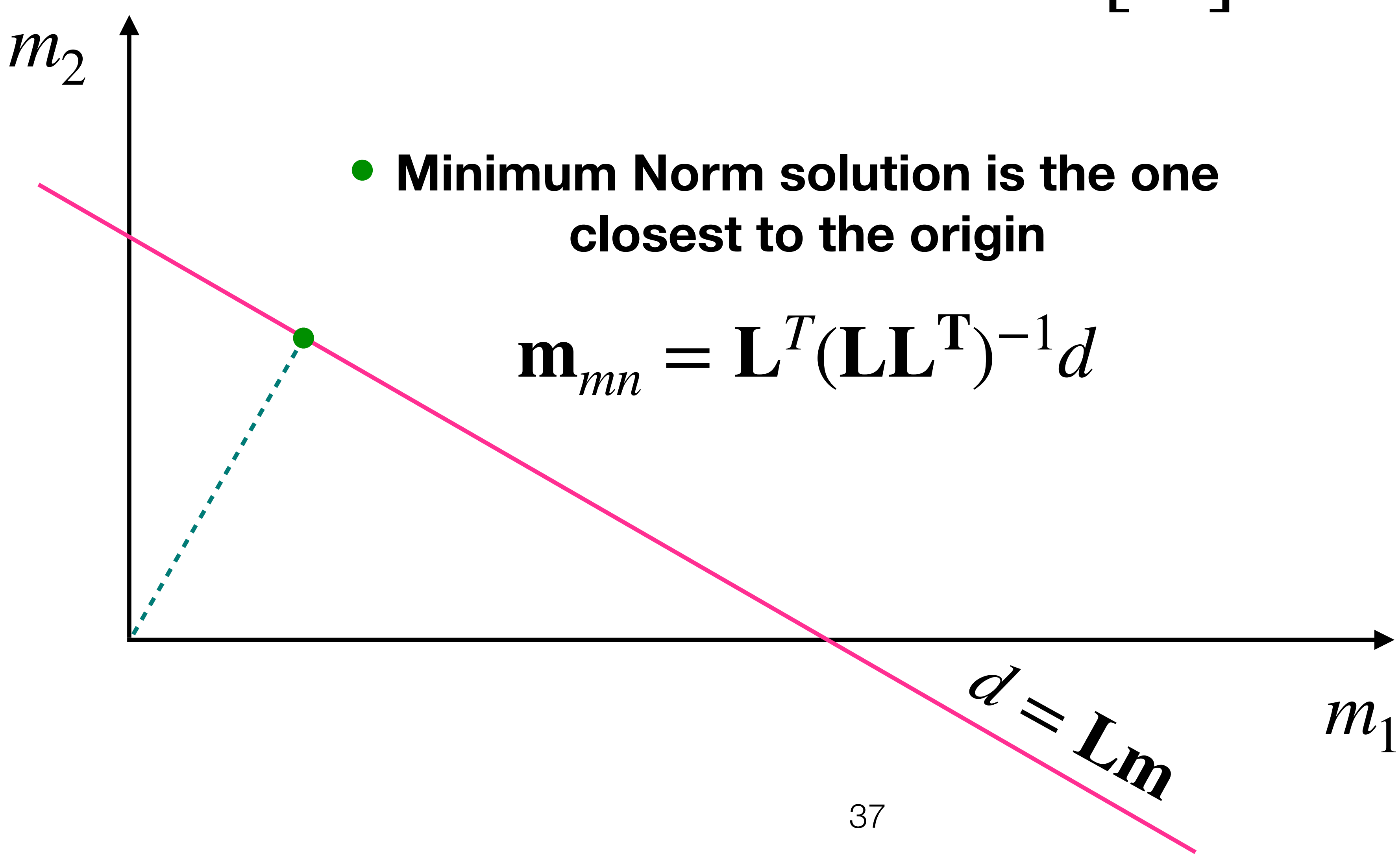
- **Any point on the line is a solution**



Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- $N=1$, $M = 2$ (2 unknowns and 1 equation)

$$d = \mathbf{Lm} = \begin{bmatrix} L_{11} & L_{12} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$



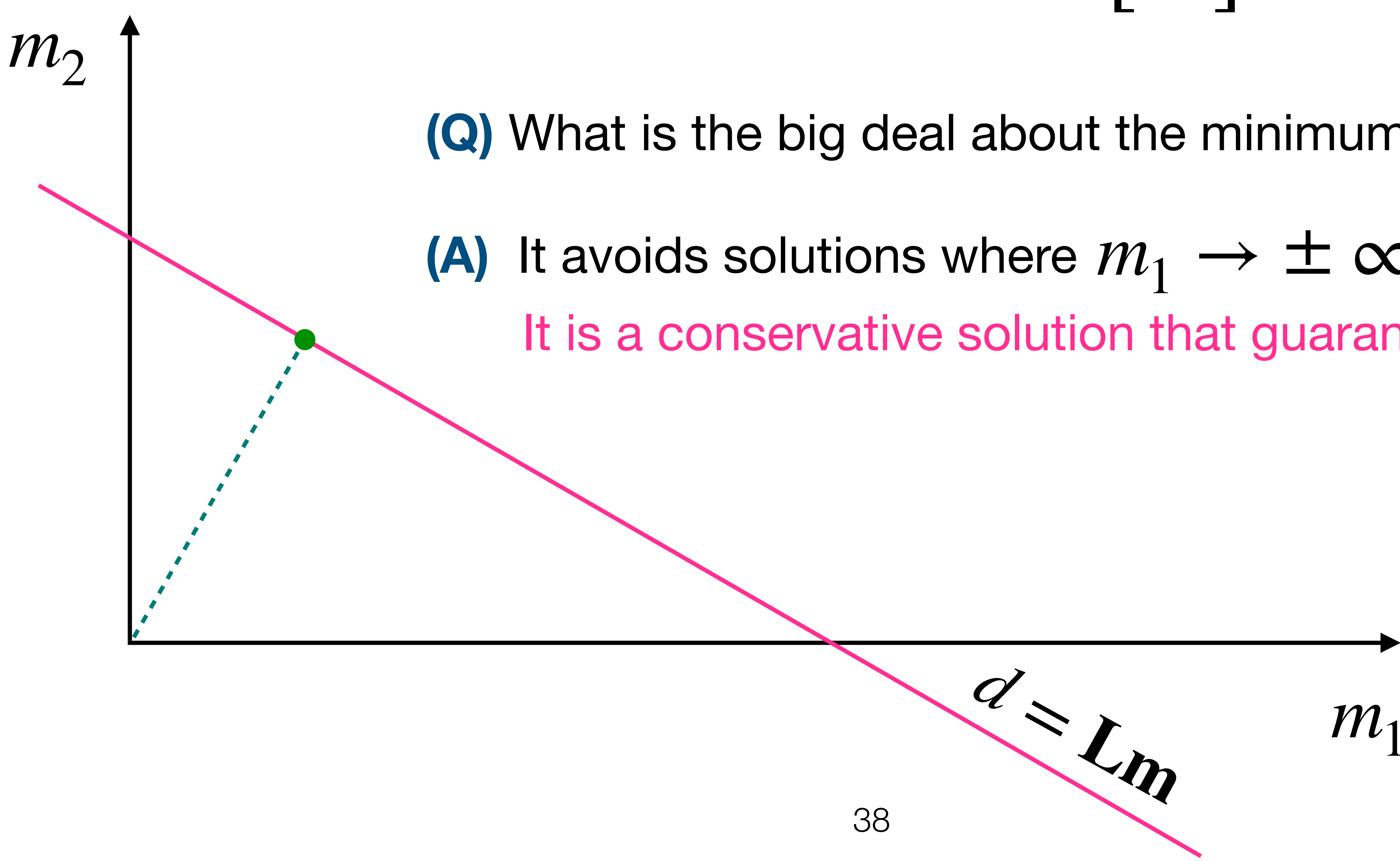
- **Minimum Norm solution is the one closest to the origin**

$$\mathbf{m}_{mn} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^\mathbf{T})^{-1} d$$

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- $\mathrm{N}=1, \mathrm{M}=2$ (2 unknowns and 1 equation)

$$d = \mathbf{Lm} = \begin{bmatrix} L_{11} & L_{12} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$



(Q) What is the big deal about the minimum norm solution?

(A) It avoids solutions where $m_1 \to \pm\infty, m_2 \to \pm\infty$

It is a conservative solution that guarantees stability

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

Problem

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

Minimum norm solution

$$\mathbf{m}_{mn} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{d}$$

We can replace the data back into the solution

$$\mathbf{m}_{mn} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{L}\mathbf{m}$$

Model resolution matrix is not identity

$$\mathbf{m}_{mn} = \mathbf{R}_m\mathbf{m}$$

$$\mathbf{R}_m = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{L} \neq \mathbf{I}$$

So, unless the “true solution” is the minimum norm solution, we cannot recover the “true solution”…

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

Problem

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

Minimum norm solution

$$\mathbf{m}_{mn} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{d}$$

Predicted data

$$\mathbf{d}^{pred} = \mathbf{L}\mathbf{m}_{mn} = \mathbf{L}\mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{d}$$

Data resolution matrix is the identity

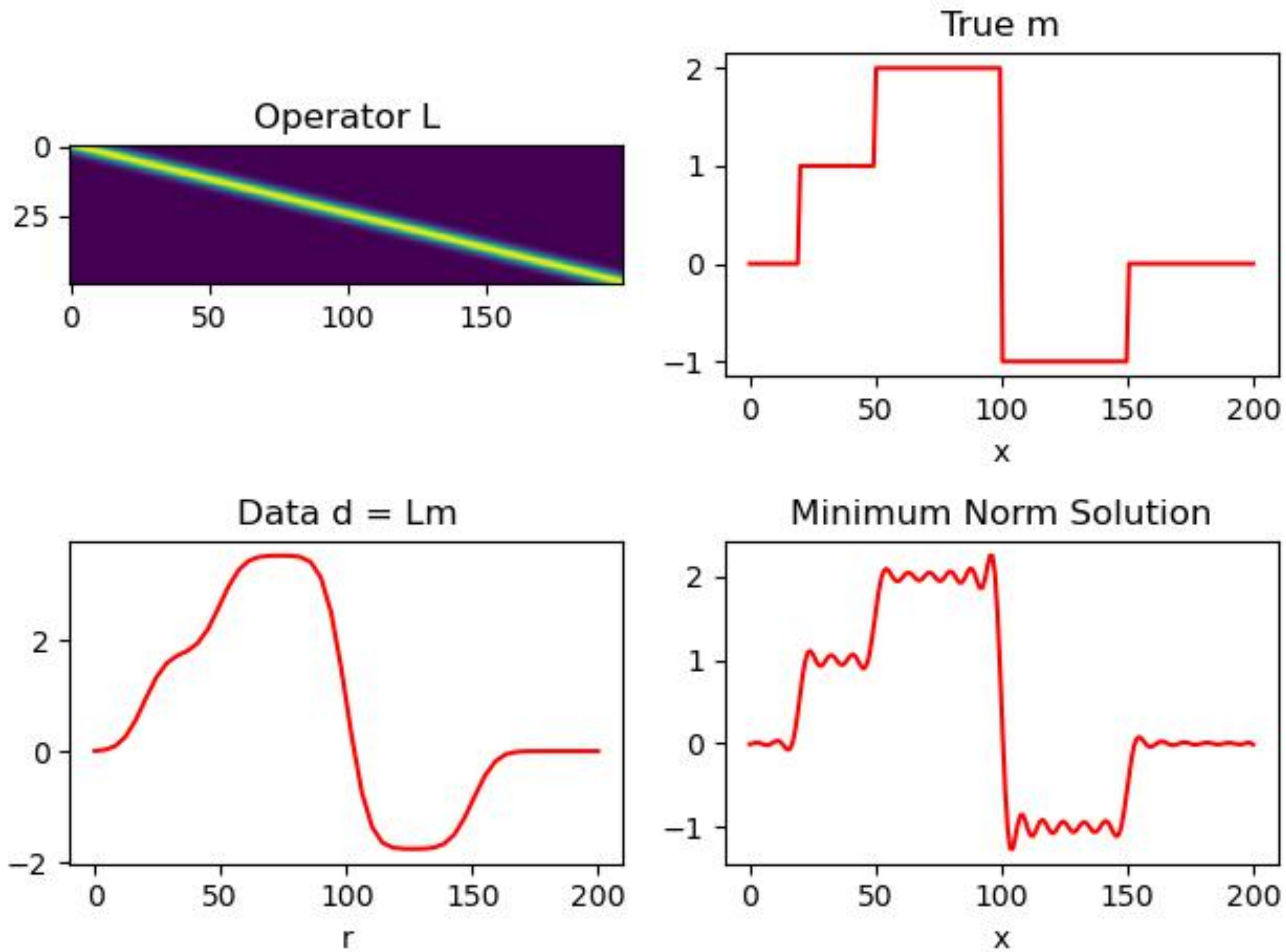
$$\mathbf{d}^{pred} = \mathbf{R}_d\mathbf{d} = \mathbf{d}$$

$$\mathbf{R}_d = \mathbf{L}\mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1} = \mathbf{I}$$

It is obvious that I can recover the data exactly because the data was used as an exact constraint when we constructed the minimum norm solution.

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

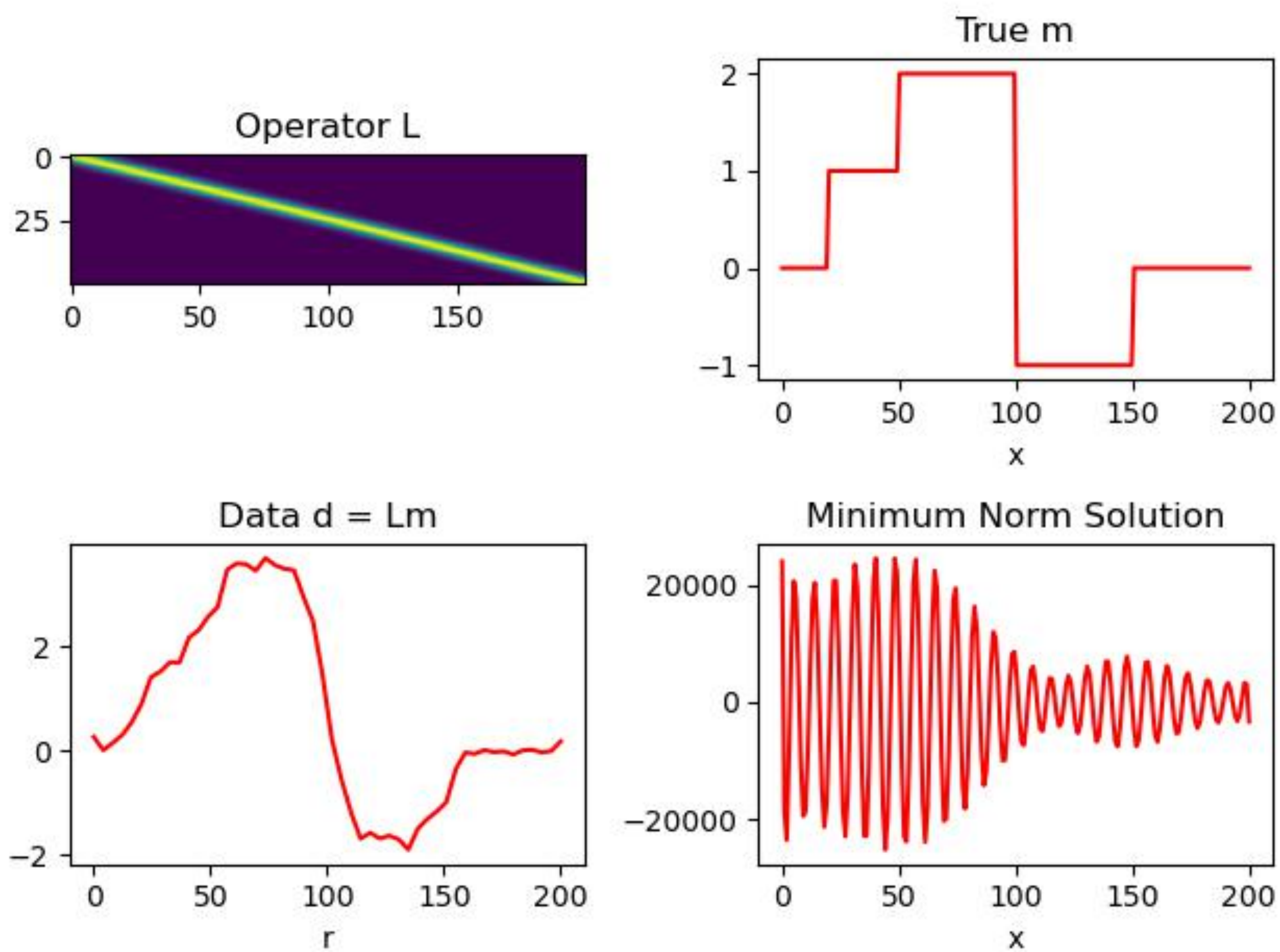
- Example $d(r_j) = \int_{x_{min}}^{x_{max}} A e^{-\alpha (r_j - x)^2} m(x) dx$



`demo_1.ipynb`

Discrete Linear Inverse Problem with Accurate Data: Minimum Norm Solution

- Example $$d(r_j) = \int_{x_{min}}^{x_{max}} A e^{-\alpha(r_j - x)^2} m(x)dx + \sigma n(r_j)$$



`demo_2.ipynb`

This is not working because I am fitting the data “exactly” and the data now contains noise!

Discrete Linear Inverse Problem with Accurate Data: Weighted minimum Norm Solution

- Among all possible solution select the one with minimum Weighted Euclidian norm. In other words,

$$\textbf{Minimize } \|\mathbf{Wm}\|_2^2$$

$$\textbf{Subject to } \mathbf{d} = \mathbf{Lm}$$

- The latter is solved using the Method of Lagrange Multipliers. We minimized an augmented cost function

$$J = \|\mathbf{Wm}\|_2^2 + \mathbf{b}^T(\mathbf{Lm} - \mathbf{d})$$

- $\mathbf{b} \in R^N$ is the vector of Lagrange Multipliers, one multiplier per observation

Discrete Linear Inverse Problem with Accurate Data: Weighted minimum Norm Solution

- To solve the problem we need to minimize the cost respect to $\mathbf{m}$ and $\mathbf{b}$

$$J = \|\mathbf{W}\mathbf{m}\|_2^2 + \mathbf{b}^T(\mathbf{L}\mathbf{m} - \mathbf{d})$$

- Which leads to the so called weighted minimum norm solution (I will do it in the whiteboard)

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

$$\mathbf{m}_{wmn} = \mathbf{Q}\mathbf{L}^T(\mathbf{L}\mathbf{Q}\mathbf{L}^T)^{-1}\mathbf{d}$$

Linear Inverse Problems with inaccurate data

- Including noise into the problem

$$\mathbf{d} = \mathbf{L}\mathbf{m} + \mathbf{e}$$

$$\mathbf{d} : N \times 1 \qquad \mathbf{m} : M \times 1 \qquad \mathbf{L} : N \times M$$

- $\mathbf{e}$ represents "nice" Gaussian additive noise

- Given the vector of observed data $\mathbf{d}$, one wishes to estimate the vector of model parameters $\mathbf{m}$

Linear Inverse Problems with inaccurate data: Regularized least-squares solution

Cost function for least-squares problems

$$J = \|\mathbf{e}\|_2^2 = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2$$

The principle is simple, find $\mathbf{m}$ that minimize the sum of the squares of the errors

$$\nabla J = 0 \rightarrow (\mathbf{L}^T\mathbf{L})\mathbf{m} = \mathbf{L}^T\mathbf{d}$$

To compute the solution we now have to invert a matrix. Assume the matrix is invertible then

$$\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\mathbf{d} \qquad M << N$$

Careful here: This is valid for overdetermined problems where the matrix to invert is full rank

Linear Inverse Problems with inaccurate data: Regularized least-squares solution

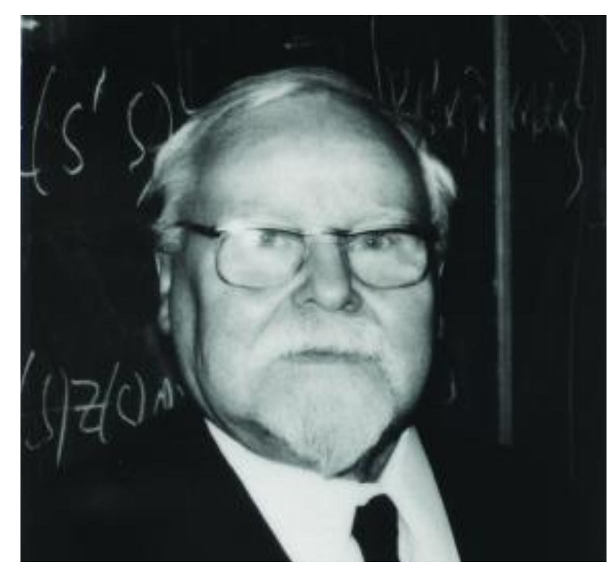
The latter is a naive solution because inverse problems are ill-posed and regularization is needed and, in general, they are not well-posed over-determined problems. In other words, one cannot safely compute: $(\mathbf{L}^T\mathbf{L})^{-1}$

- Either is non invertible
- Or the Matrix has a large condition number (It will amplify noise)
- Condition Number $\kappa = \dfrac{\lambda_{max}(\mathbf{L}^T\mathbf{L})}{\lambda_{min}(\mathbf{L}^T\mathbf{L})}$

Matrix computations
GH Golub, CF Van Loan, 2013 (4th edition)

Regularization (a solution to ill-conditioning)

- Main idea of regularization methods:
  - Take an ill-posed problem and turn into a well-posed problem by introducing constraints that lead to a stable solution. The solution often depends on a trade-off parameter. Therefore, regularization methods create a family of solutions with different properties (e.g. smoothness). One can modify the solution by changing the tradeoff parameter
  - Often called Tikhonov regularization



A. N. Tikhonov (1906-1993)

Regularization

- Replace cost function to minimize

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 \longleftarrow \text{Cost with no regularization term}$$

- by

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu \|\mathbf{W}\,\mathbf{m}\|_2^2$$

- μ Is the infamous trade-off parameter or regularization parameter. Why infamous?

- $\mathbf{W}$ is a matrix/operator of weights

Regularization

- Evaluate the solution, as usual, by minimizing a cost function

$$\mathbf{m}_{sol} = \underset{\mathbf{m}}{\operatorname{argmin}}\; J$$
$$= \underset{\mathbf{m}}{\operatorname{argmin}}\, \{ \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu \|\mathbf{W}\,\mathbf{m}\|_2^2 \}$$

- Taking derivatives and equating them to zero

$$\nabla J = 0 \rightarrow \mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{R})^{-1}\mathbf{L}^T\mathbf{d}$$

$$\mathbf{R} = \mathbf{W}^T\mathbf{W}$$

Regularization

- Anatomy of the cost function

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{W}\,\mathbf{m}\|_2^2}_{Model\ Norm}$$

$$J = \underbrace{\|\mathbf{e}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{u}\|_2^2}_{Model\ Norm}$$

- To minimize J is equivalent to simultaneously minimize $\mathbf{e}$ and $\mathbf{u}$

Regularization

- To minimize J is equivalent to simultaneously minimize $\mathbf{e}$ and $\mathbf{u}$

$$\mathbf{e} = \mathbf{d} - \mathbf{L}\mathbf{m} \approx \mathbf{0}$$ *Minimize the residuals*

$$\mathbf{u} = \mathbf{W}\mathbf{m} \approx \mathbf{0}$$ *Minimize the bad features of* $\mathbf{m}$

- $\mathbf{W}$ is a high-pass operator therefore it penalizes roughness. The vector $\mathbf{u}$ represents amplified bad features of $\mathbf{m}$

- Examples of $\mathbf{W}$ are first and second order derivatives

Regularization with first order derivative smoothing

$$\mathbf{W} = \mathbf{D}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

$$\frac{\partial f(x)}{\partial x} \leftrightarrow ikF(k)$$

The derivative operator is high pass because it amplifies high frequencies

Regularization with second order derivative smoothing

$$\mathbf{W} = \mathbf{D}_2 = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

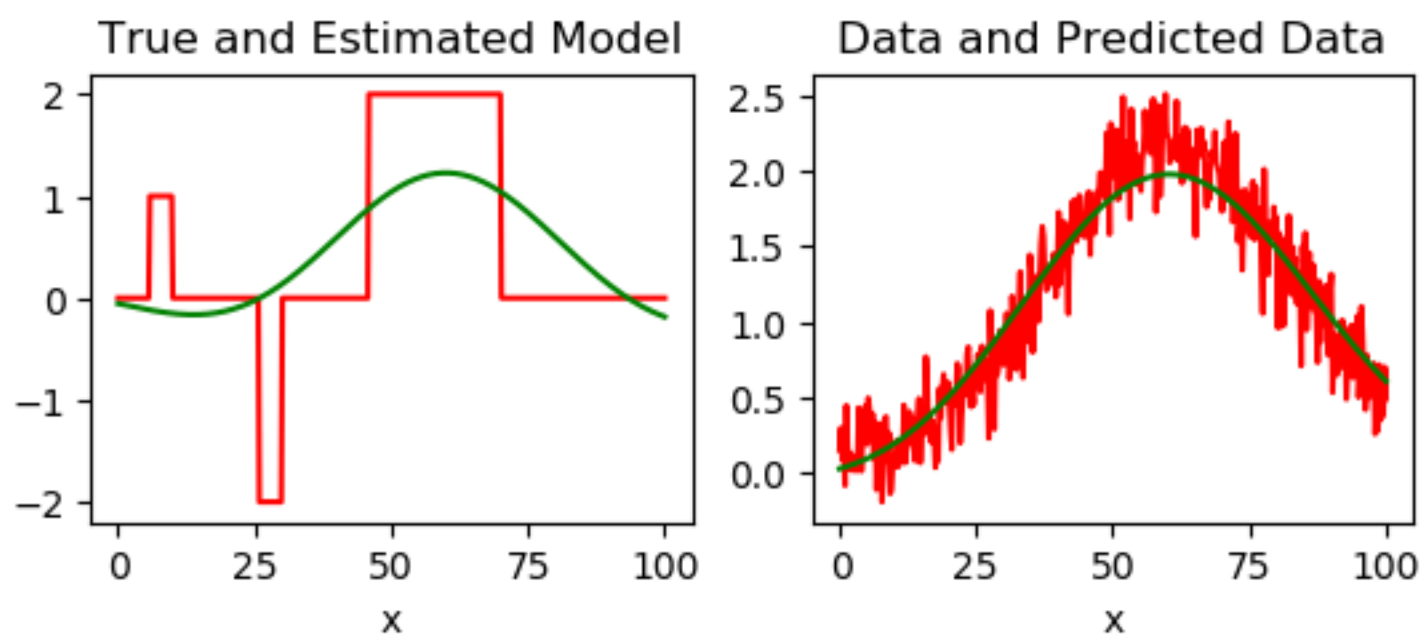
$$\frac{\partial^2 f(x)}{\partial x^2} \leftrightarrow -k^2 F(k)$$

Second Derivative is also a high-pass operator

Damped LS solution is when W=I

$$\mathbf{m}_{sol} = \underset{\mathbf{m}}{\operatorname{argmin}}\ J$$
$$= \underset{\mathbf{m}}{\operatorname{argmin}} \{ \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu \|\mathbf{m}\|_2^2 \}$$

$$\frac{\partial J}{\partial \mathbf{m}} = 0 \rightarrow \mathbf{m}_{sol} = (\mathbf{L}^T \mathbf{L} + \mu \mathbf{I})^{-1} \mathbf{L}^T \mathbf{d}$$



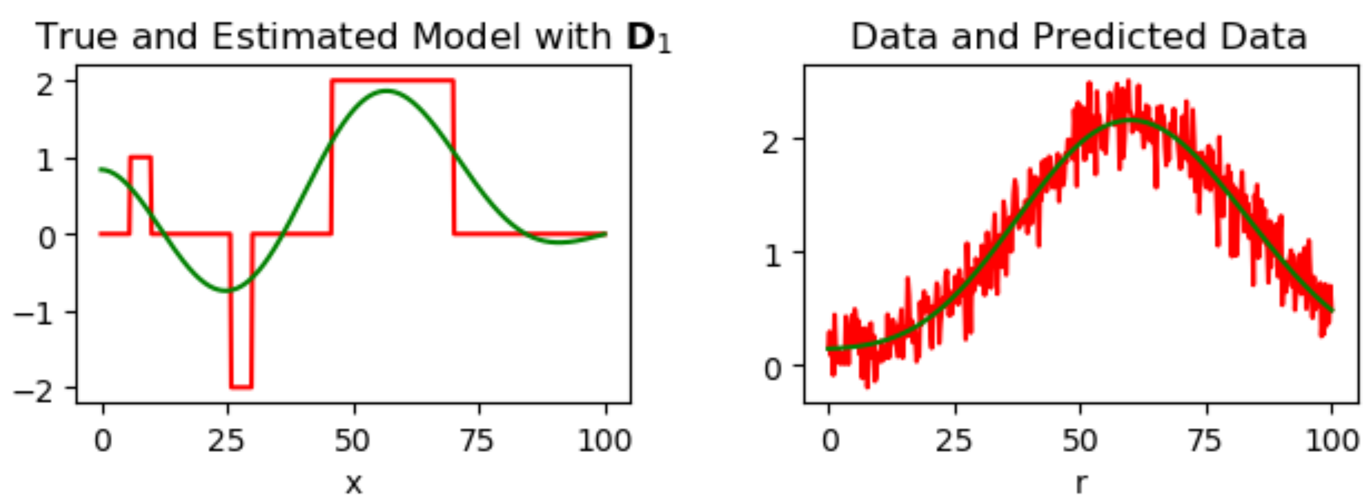
Red: True Model | Observed data
Green: Estimated Model | Predicted data

Example: Solution with smoothing

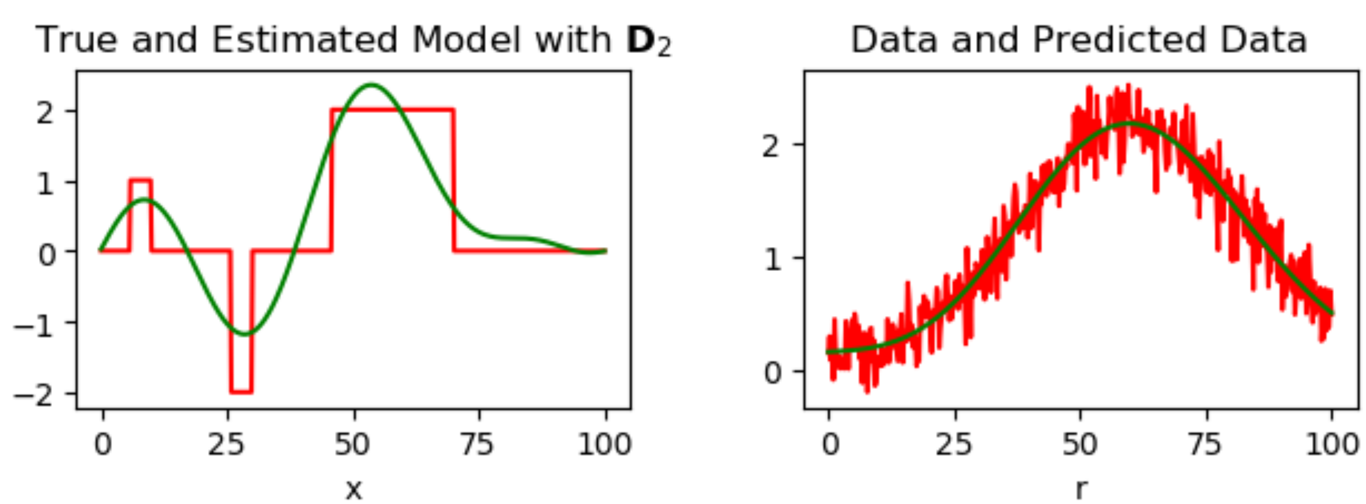
$$\mathbf{m}_{sol} = \underset{\mathbf{m}}{\operatorname{argmin}} \{ \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu \, \|\mathbf{W}\,\mathbf{m}\|_2^2 \}$$

$$\frac{\partial J}{\partial \mathbf{m}} = 0 \rightarrow \mathbf{m}_{sol} = (\mathbf{L}^T\,\mathbf{L} + \mu\,\mathbf{W}^T\,\mathbf{W})^{-1}\mathbf{L}^T\,\mathbf{d}$$

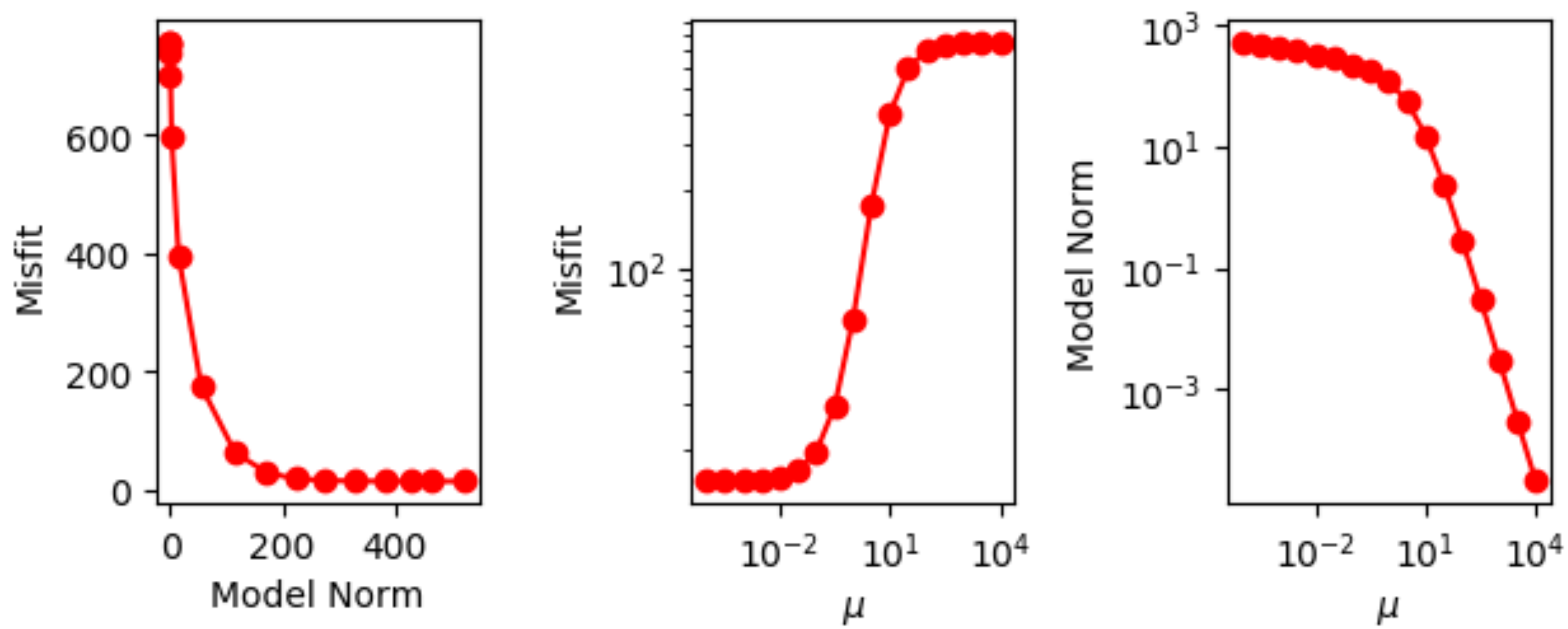
$\mathbf{W} = \mathbf{D}_1$



$\mathbf{W} = \mathbf{D}_2$



Trade-off curves (for Damped least-squares)



Underdetermined Problem with inaccurate data
Minimum Norm Solution

$$J = \textbf{Misfit} + \mu\, \textbf{Model Norm}$$

$$J = \|\mathbf{d} - \mathbf{Lm}\|_2^2 + \mu \|\mathbf{m}\|_2^2$$

Make residuals small = Find a model that fits the data

Make model parameters small = Stable solution that does not blow up

μ : Trade-off parameter of the inverse problem

Underdetermined Problem with inaccurate data
Minimum Norm Solution

- So we solve the problem

$$\mathbf{d} = \mathbf{L}\mathbf{m} + \mathbf{n}$$

by minimizing the cost $J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{m}\|_2^2$

- We already discussed how one can compute derivatives of the scalar function J with respect to a vector.

- Condition for minimum

$$\frac{dJ}{d\mathbf{m}} = 0 \rightarrow (\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})\,\mathbf{m} = \mathbf{L}^T\mathbf{d}$$

Underdetermined Problem with inaccurate data
Minimum Norm Solution

Cost $J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{m}\|_2^2$

- Solution $\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})^{-1}\mathbf{L}^T\mathbf{d}$
- Predicted data $\mathbf{d}_{pred} = \mathbf{L}\mathbf{m}_{sol} = \mathbf{L}(\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})^{-1}\mathbf{L}^T\mathbf{d}$
- Predicted residuals $\mathbf{r} = \mathbf{d}_{pred} - \mathbf{d}$
- You can also show the following important identity:

$$\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})^{-1}\mathbf{L}^T\mathbf{d} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T + \mu\mathbf{I})^{-1}\,\mathbf{d}$$

Underdetermined Problem with inaccurate data
Minimum Norm Solution

$$\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})^{-1}\mathbf{L}^T\mathbf{d} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T + \mu\mathbf{I})^{-1}\,\mathbf{d}$$

1 **2**

1 looks like the least-squares solution for overdetermined problems, so it is often called the regularized least-squares solution. Also called damped least-squares, ridge regression solution, least-squares solution with zero-order quadratic regularization, Tikhonov solution, etc.

2 looks like the minimum norm solution for exact data but now there is an additional constant diagonal term.

The method we described is also called Tikhonov regularization, so the solution given by either 1) or 2) is also the Tikhonov solution.

Underdetermined Problem with inaccurate data
Minimum Norm Solution

$$\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{I})^{-1}\mathbf{L}^T\mathbf{d} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T + \mu\mathbf{I})^{-1}\,\mathbf{d}$$

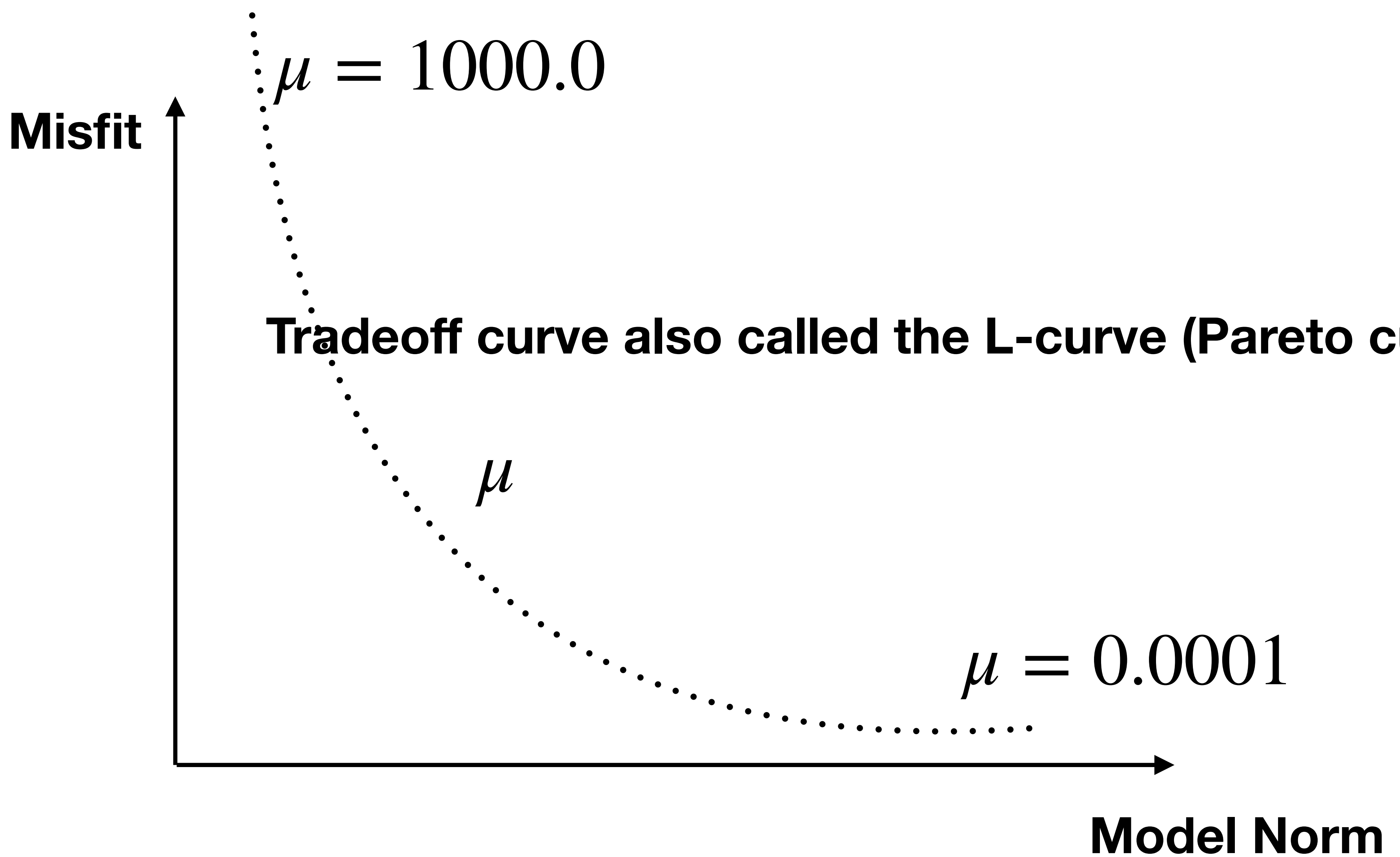
Where does the name damped least-squares solution come from? As one increases μ, more damping is introduced. In other words, the solution becomes smaller and less oscillatory!

The tradeoff parameter μ is sometimes also call the damping parameter.

The tradeoff parameter must be provided and it defines a family of solutions. I should have written $\mathbf{m}_{sol}(\mu)$ to indicate the dependency of the solution on the tradeoff parameter.

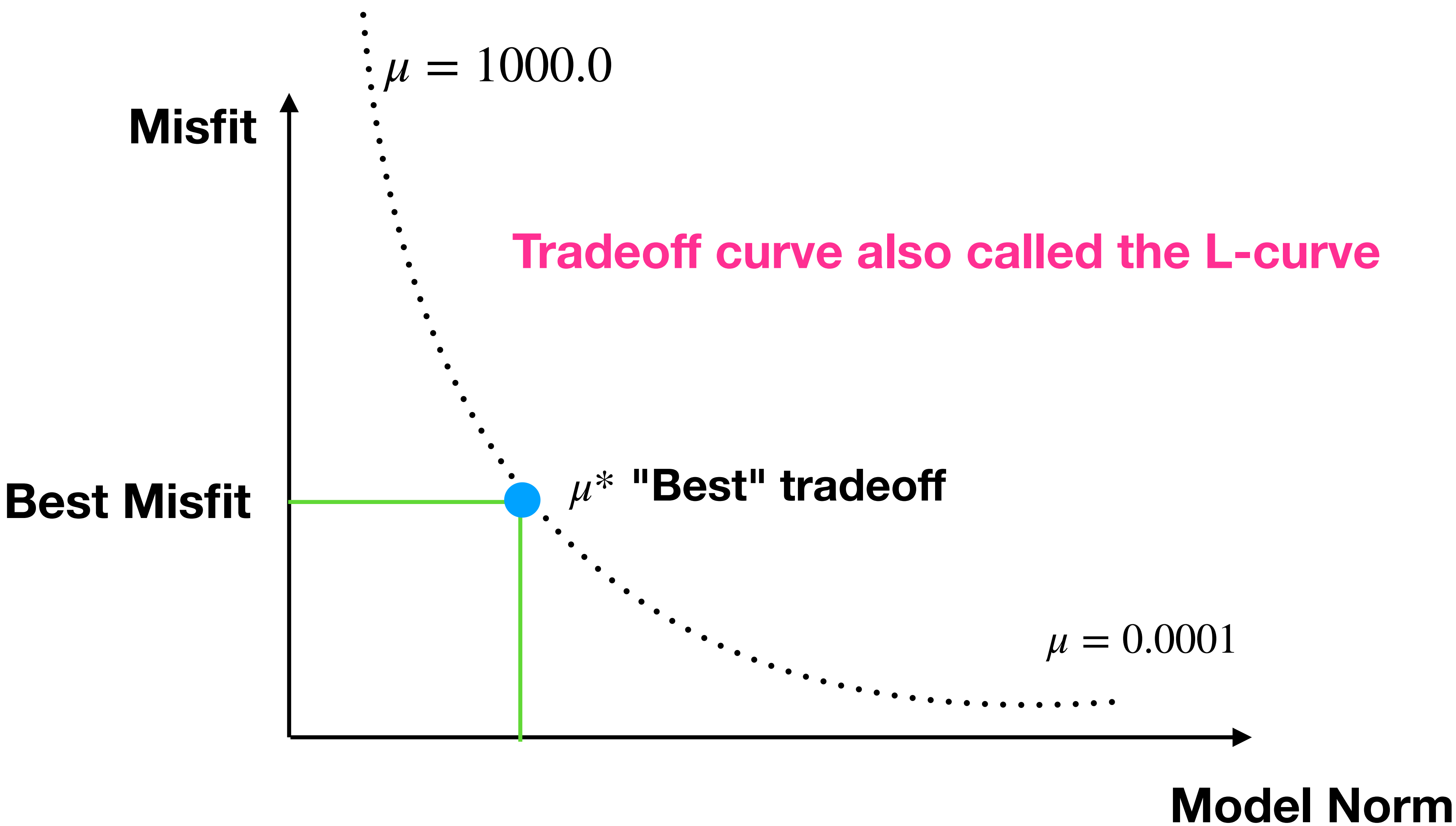
Underdetermined Problem with inaccurate data
Minimum Norm Solution

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{m}\|_2^2 = \textbf{Misfit } + \mu \textbf{ Model Norm}$$



Underdetermined Problem with inaccurate data
Minimum Norm Solution

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{m}\|_2^2 = \textbf{Misfit } + \mu \textbf{ Model Norm}$$

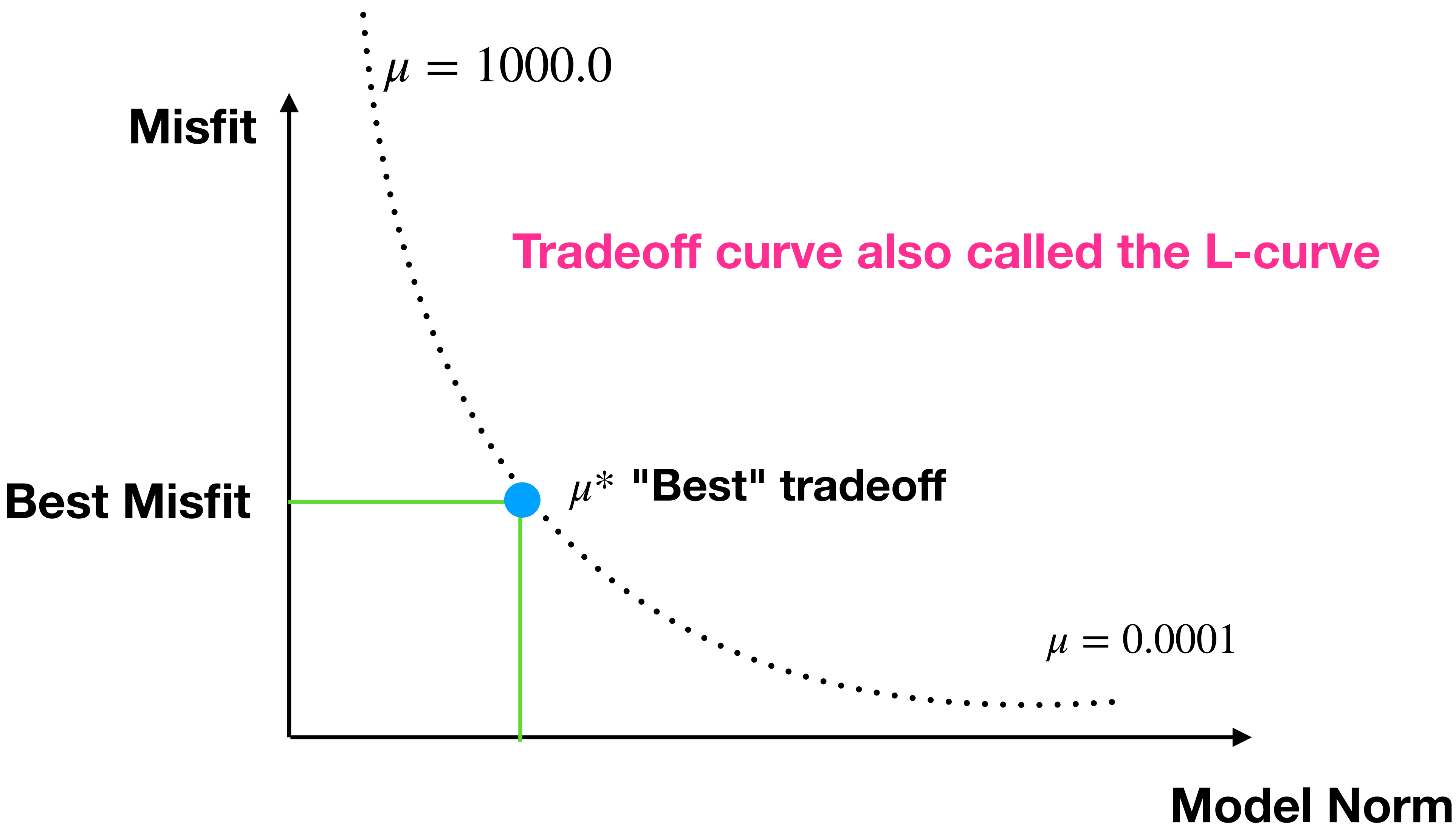


Underdetermined Problem with inaccurate data
Minimum Norm Solution

Recall original problem: $\mathbf{d} = \mathbf{L}\mathbf{m} + \mathbf{n}$.

Assume you know the variance of the noise: σ^2

Best misfit $= \|\mathbf{d} - \mathbf{L}\mathbf{m}_{sol}(\mu^*)\|_2^2 = N \times \sigma^2$

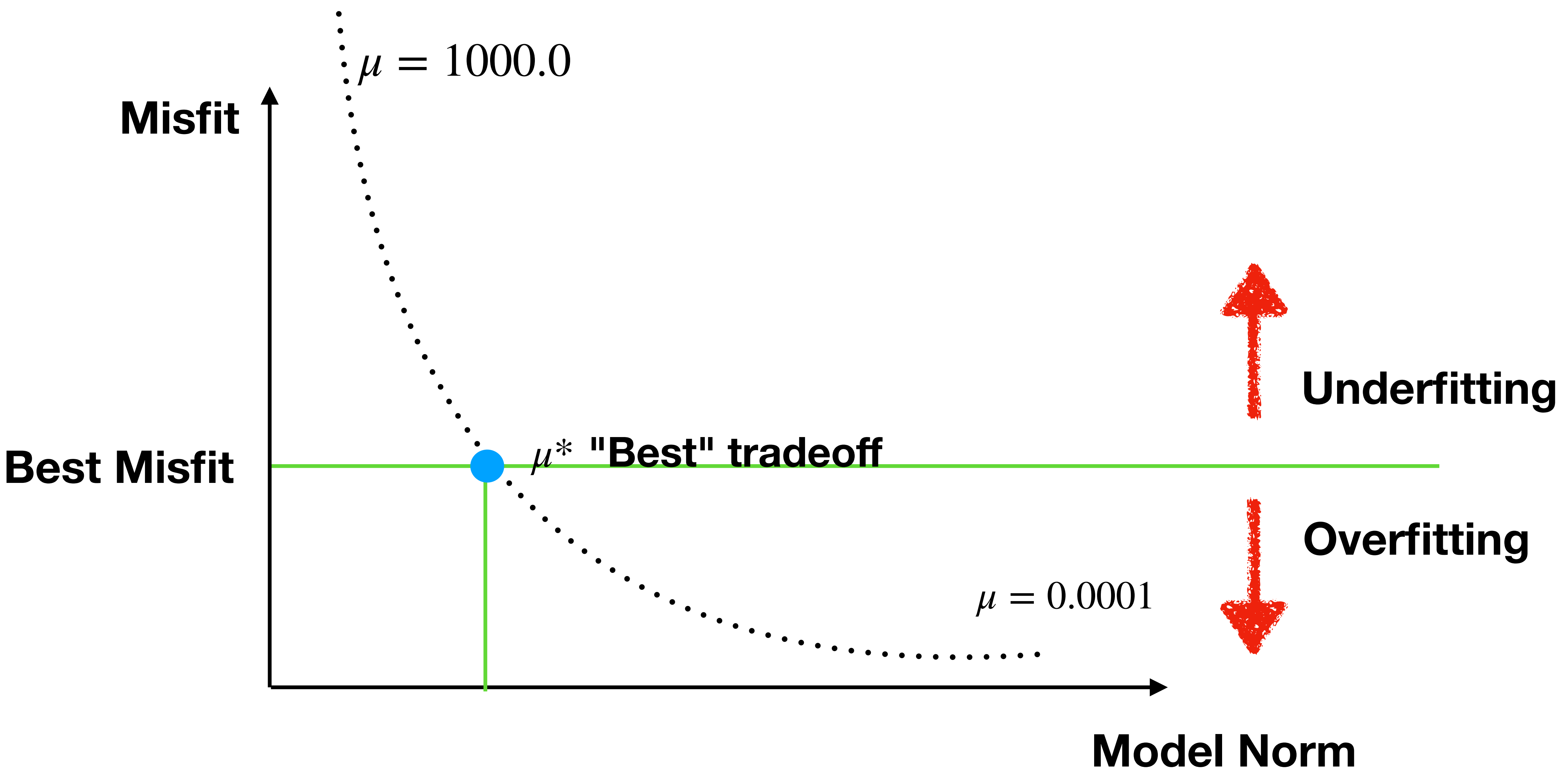


Underdetermined Problem with inaccurate data
Minimum Norm Solution

Recall original problem: $\mathbf{d} = \mathbf{L}\mathbf{m} + \mathbf{n}$.

Assume you know the variance of the noise: σ^2

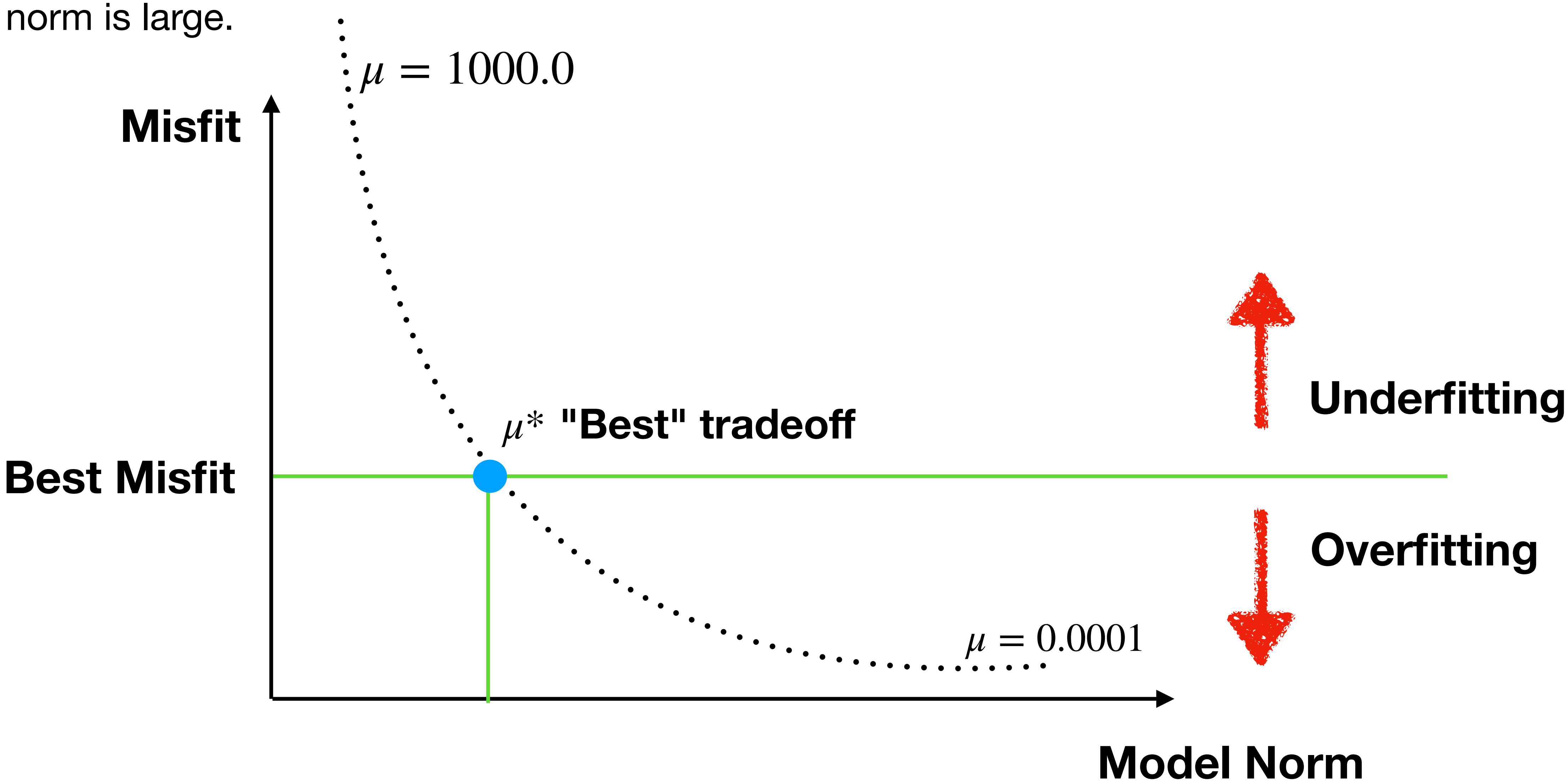
Best misfit $= \|\mathbf{d} - \mathbf{L}\mathbf{m}_{sol}(\mu^*)\|_2^2 = N \times \sigma^2$



Underdetermined Problem with inaccurate data
Minimum Norm Solution

Underfitting: Observations are not properly fit. You are overestimating the amount of noise in the data. Solution norm decreases. Residuals are too large. There is probably some structure in the residuals.

Overfitting: Residuals become too small and therefore your estimated model produces data that fits the noise. As a consequence of the latter, the model can become unstable and the model norm is large.



Underdetermined Problem with inaccurate data
Weighted Minimum Norm Solution

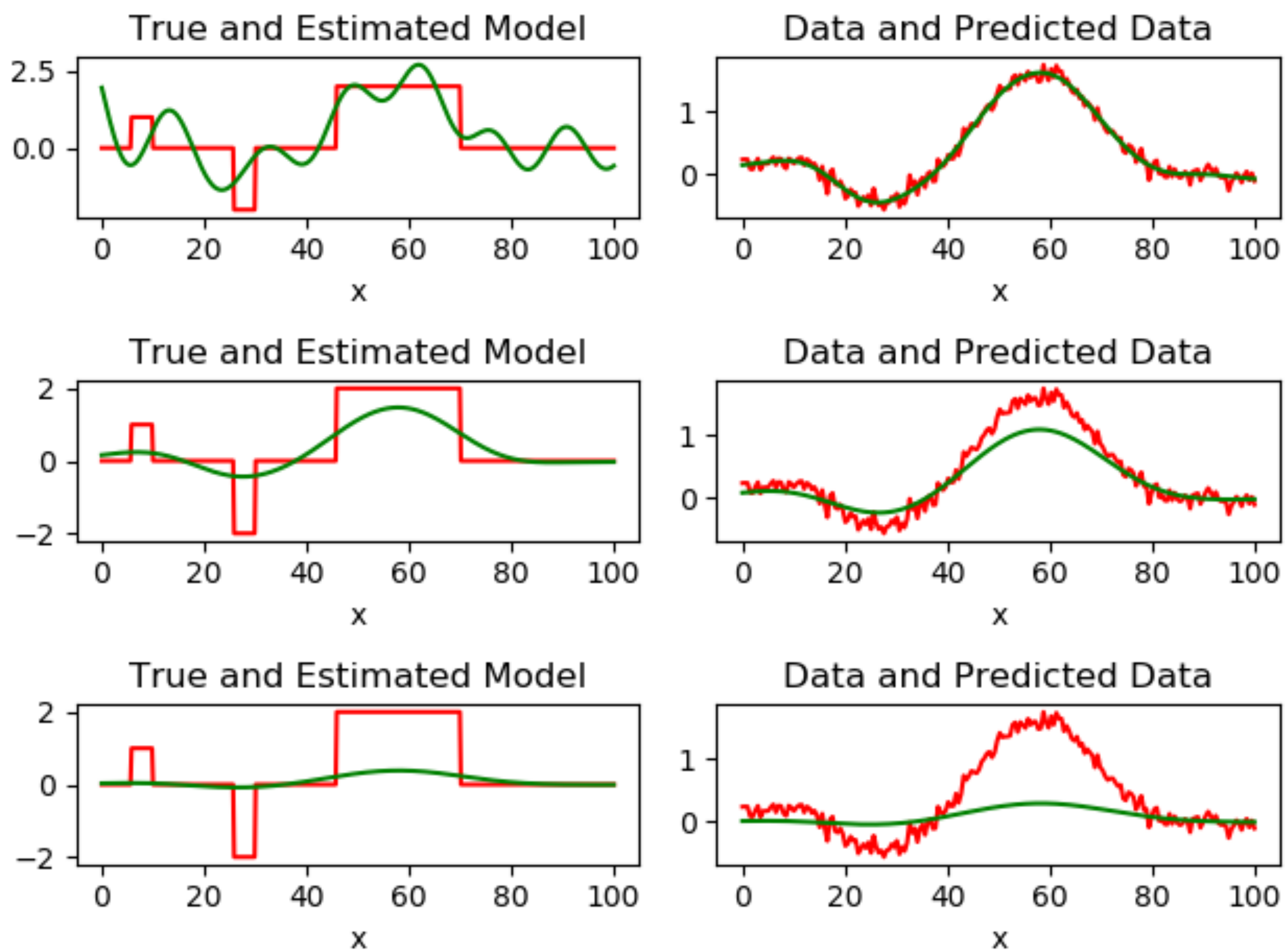
Cost $J = \|\mathbf{d}_{obs} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{W}\mathbf{m}\|_2^2$

- Solution $\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{W}^T\mathbf{W})^{-1}\mathbf{L}^T\mathbf{d}$
- Predicted data $\mathbf{d}_{pred} = \mathbf{L}\mathbf{m}_{sol} = \mathbf{L}(\mathbf{L}^T\mathbf{L} + \mu\mathbf{W}^T\mathbf{W})^{-1}\mathbf{L}^T\mathbf{d}_{obs}$
- Predicted residuals $\mathbf{r} = \mathbf{d}_{pred} - \mathbf{d}$
- You can also show the following important identity:

$$\mathbf{m}_{sol} = (\mathbf{L}^T\mathbf{L} + \mu\mathbf{Q})^{-1}\mathbf{L}^T\mathbf{d} = \mathbf{Q}^{-1}\mathbf{L}^T(\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^T + \mu\mathbf{I})^{-1}\,\mathbf{d}$$

$$\mathbf{Q} = \mathbf{W}^T\mathbf{W}$$

Trade-off curves (for Damped least-squares)



$\mu = 0.0001$

$\mu = 0.1$

$\mu = 1$

Changing the norm

Edge preserving regularization (EPR)

- Avoid smoothing to preserve edges
- We adopt the ell-1 norm of the first order derivative of model parameters
- Make the derivative of the model sparse

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{\textit{Error Norm}} + \mu \underbrace{\|\mathbf{D}_1\, \mathbf{m}\|_1}_{\textit{Model Norm}}$$

We ask $\mathbf{u} = \mathbf{D}_1 \mathbf{m}$ **to be sparse**

The L_1 and the L_2 norms

$$\mathbf{u} : N \times 1 \longrightarrow \|\mathbf{u}\|_2^2 = \mathbf{u}^H \mathbf{u} = \sum_{i=1}^{N} u_i u_i^* = \sum_{i=1}^{N} |u_i|^2$$

$$\mathbf{u} : N \times 1 \longrightarrow \|\mathbf{u}\|_1 = \sum_{i=1}^{N} |u_i|$$

Edge preserving regularization (EPR)

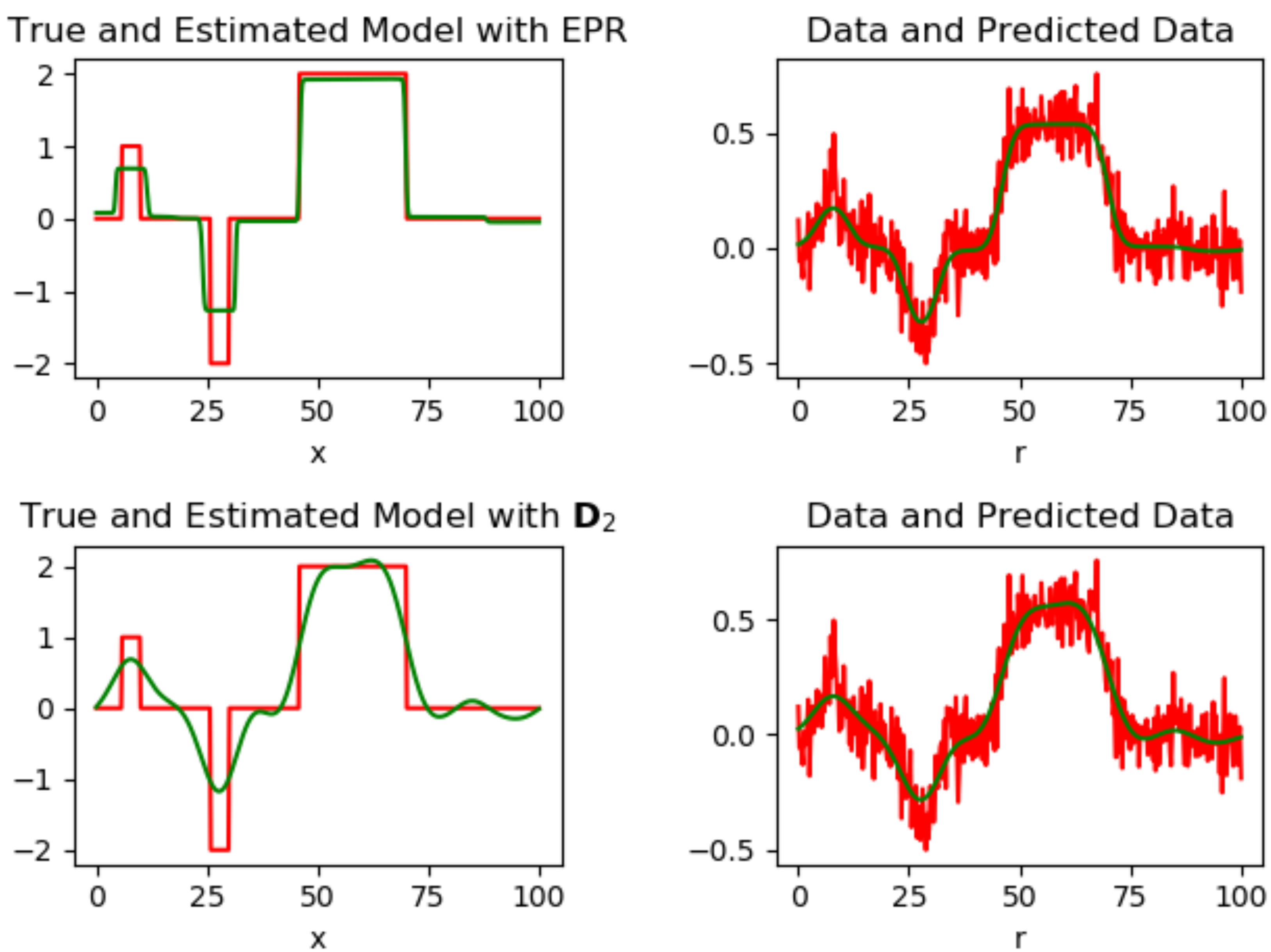
- Quadratic leads to close form solution (linear system of equations)

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{W}\,\mathbf{m}\|_2^2}_{Model\ Norm}$$

- EPR: Non-quadratic regularization leads to non-linear solution that tries to recover edges

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{D}_1\,\mathbf{m}\|_1}_{Model\ Norm}$$

Edge preserving regularization (EPR)



EPR
(non-quadratic)

Second Order Derivative Reg.
(quadratic)

Edge preserving regularization (EPR)

- Non-quadratic regularization leads to non-linear solution

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{D}_1\,\mathbf{m}\|_1}_{Model\ Norm}$$

- Solution

$$\nabla J = 0 \to (\mathbf{L}'\,\mathbf{L} + \mu\mathbf{D}_1'\,\mathbf{Q}\,\mathbf{D}_1)\mathbf{m} = \mathbf{L}'\mathbf{d}$$

- Where $\mathbf{v} = \mathbf{D}_1\,\mathbf{m}$ $\qquad Q_{ii} = \dfrac{1}{|\,v_i\,|}$

- To avoid division by zero $\qquad Q_{ii} = \dfrac{1}{\epsilon + |\,v_i\,|}$

Edge preserving regularization (EPR)

- Iterative re-weighted least-squares

$$\mathbf{m}^1 = \mathbf{m}_{initial}$$

For $k{=}1$ **until convergence**

$$\mathbf{v} = \mathbf{D}_1\,\mathbf{m}$$

$$Q_{ii}^k = \frac{1}{\epsilon + |\,v_i^k\,|}$$

$$\mathbf{m}^{k+1} = (\mathbf{L}'\,\mathbf{L} + \mu\mathbf{D}_1'\,\mathbf{Q}^k\,\mathbf{D}_1)^{-1}\mathbf{L}'\mathbf{d}$$

End

Sacchi, 1997, Re-weighting strategies for seismic deconvolution, GJI

Connection to sparsity

- This cost function generates a sparse solution

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{m}\|_1}_{Model\ Norm}$$

- Often used for Deconvolution

- Pre-stack data Reconstruction *(Liu and Sacchi, 2004, Geophysics; Hermann, 2010, Geophysics)*

- Radon Transfroms, etc etc etc *(Sacchi & Ulrych, 1995, Geophysics)*

- AVO Inversion (*Alemie and Sacchi, 2011, Geophysics*)

Connection to sparsity

- Make $\mathbf{m}$ sparse:

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{m}\|_1}_{Model\ Norm}$$

- Make the derivative of $\mathbf{m}$ sparse = Make $\mathbf{m}$ blocky

$$J = \underbrace{\|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2}_{Error\ Norm} + \mu \underbrace{\|\mathbf{D}_1\,\mathbf{m}\|_1}_{Model\ Norm}$$

Sparsity (more to follow…)

- **IRLS is the simplest solver one can imagine**
  - Many new solvers in recent years
    - ISTA, FISTA, SALSA, SPG-L1, ADMM, L1-Magic, etc etc etc.. show **Notes in Latex**
    - I usually use IRLS or FISTA
    - ISTA/FISTA: Only need to know how to apply $\mathbf{L}$ and $\mathbf{L}$' *(on-the-flight)*

PROBLEMS IN IMPLICIT FORM

Forward and Adjoint Operators

- **Two special operations (or operators?)**

$$\textbf{Forward} : \mathbf{d} = \mathbf{L}\mathbf{m}$$

$$\textbf{Conjugate transpose or adjoint} : \tilde{\mathbf{m}} = \mathbf{L}'\mathbf{d}$$

- $\mathbf{L}$ is a linear operator (Forward modelling operator).
- Why are them special?
  - Iterative solvers for large inverse problems only need to know how to evaluate $\mathbf{L}[\,\cdot\,]$ and $\mathbf{L}'[\,\cdot\,]$
  - I usually interpret operators as matrices. In reality, operators are codes applied *on the flight*
  - *We will see these operators everywhere today*

Forward and Adjoint Operators

- **For matrices**

Forward $: \mathbf{L}$

Conjugate transpose or adjoint $: \mathbf{L}' = \mathbf{L}^T$

- Examples of parts $(\mathbf{L}, \mathbf{L}')$
    - Demigration, Migration
    - Radon Modeling $\mathscr{R}$, Radon adjoint $\mathscr{R}'$
    - Sampling, Inserting zeros
    - Summation, Distribution or spraying
    - Fourier Synthesis $\mathscr{F}$, Fourier Analysis $\mathscr{F}^H$

Steepest descent method

- Compute the gradient and iterate downhill until convergence

- Let's see the special role of $\mathbf{L}$ and $\mathbf{L'}$ in steepest descent optimization (or in any iterative optimization algorithm that only requires $\mathbf{L}$ and $\mathbf{L'}$)

- Simplify problem by considering minimization of quadratic cost J

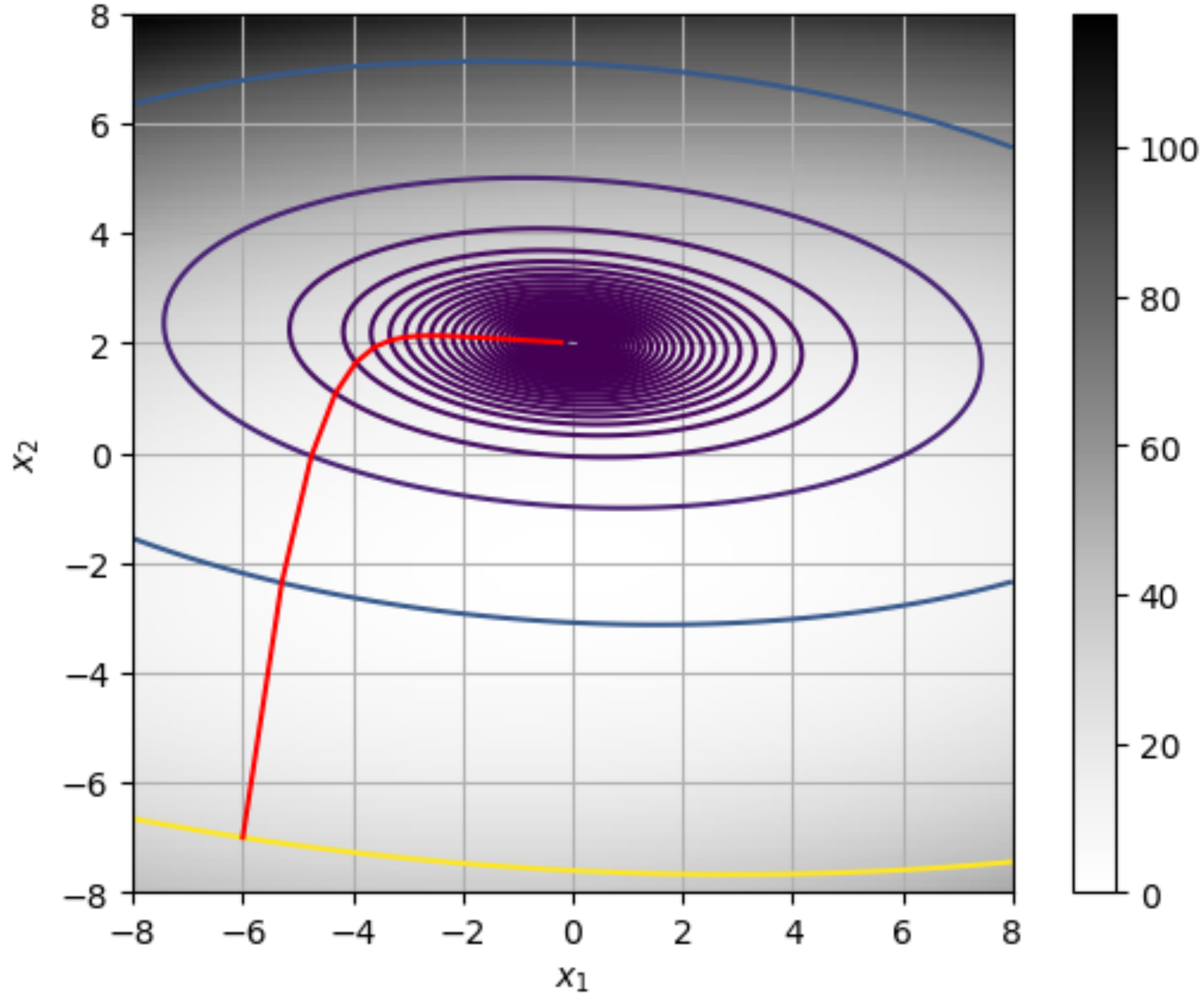
$$J = \|\mathbf{d} - \mathbf{Lm}\|_2^2$$

Steepest descent method

- Cost $J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2$
- Gradient $\mathbf{g} = \nabla J = \mathbf{L}'(\mathbf{L}\mathbf{m} - \mathbf{d})$
- Update $\mathbf{m}^{k+1} = \mathbf{m}^k - \alpha \mathbf{g}^k$
- Interesting, one can minimize J with a simple rule that does not involve inverting matrices. This is great because we can replace matrices by linear operators!!!!
- Aren't you excited ?

Steepest descent (SD) method

- Minimization of a quadratic cost by SD



x_init = [-6.0,-7.9]
x_final=[-0.009,2.005]
x_true=[0.0,2.0]

K = 50 iterations at fixed step size

Steepest descent method

SD iterations:

Do until convergence

$$\mathbf{m}^{k+1} = \mathbf{m}^k - \alpha(\,\mathbf{L}'(\mathbf{L}\mathbf{m}^k - \mathbf{d}))$$

End Do

The above code can also be written as follows

Do until convergence

$$\mathbf{r}^k = (\mathbf{L}\mathbf{m}^k - \mathbf{d})$$

$$\mathbf{m}^{k+1} = \mathbf{m}^k - \alpha\,\mathbf{L}'(\mathbf{r}^k)$$

End Do

Steepest descent method: Matrices are replaced by Linear Operators packed into Functions or Subroutines

Do until convergence

$$\mathbf{r}^k = (\mathbf{L}\mathbf{m}^k - \mathbf{d})$$

$$\mathbf{m}^{k+1} = \mathbf{m}^k - \alpha\,\mathbf{L}'(\mathbf{r}^k)$$

End Do

Do until convergence

$$\mathbf{r}^k = \textbf{Do\_It}\,[\mathbf{m}^k, flag = f] - \mathbf{d}$$

$$\mathbf{m}^{k+1} = \mathbf{m}^k - \alpha\,\textbf{Do\_It}[\mathbf{r}^k, flag = a]$$

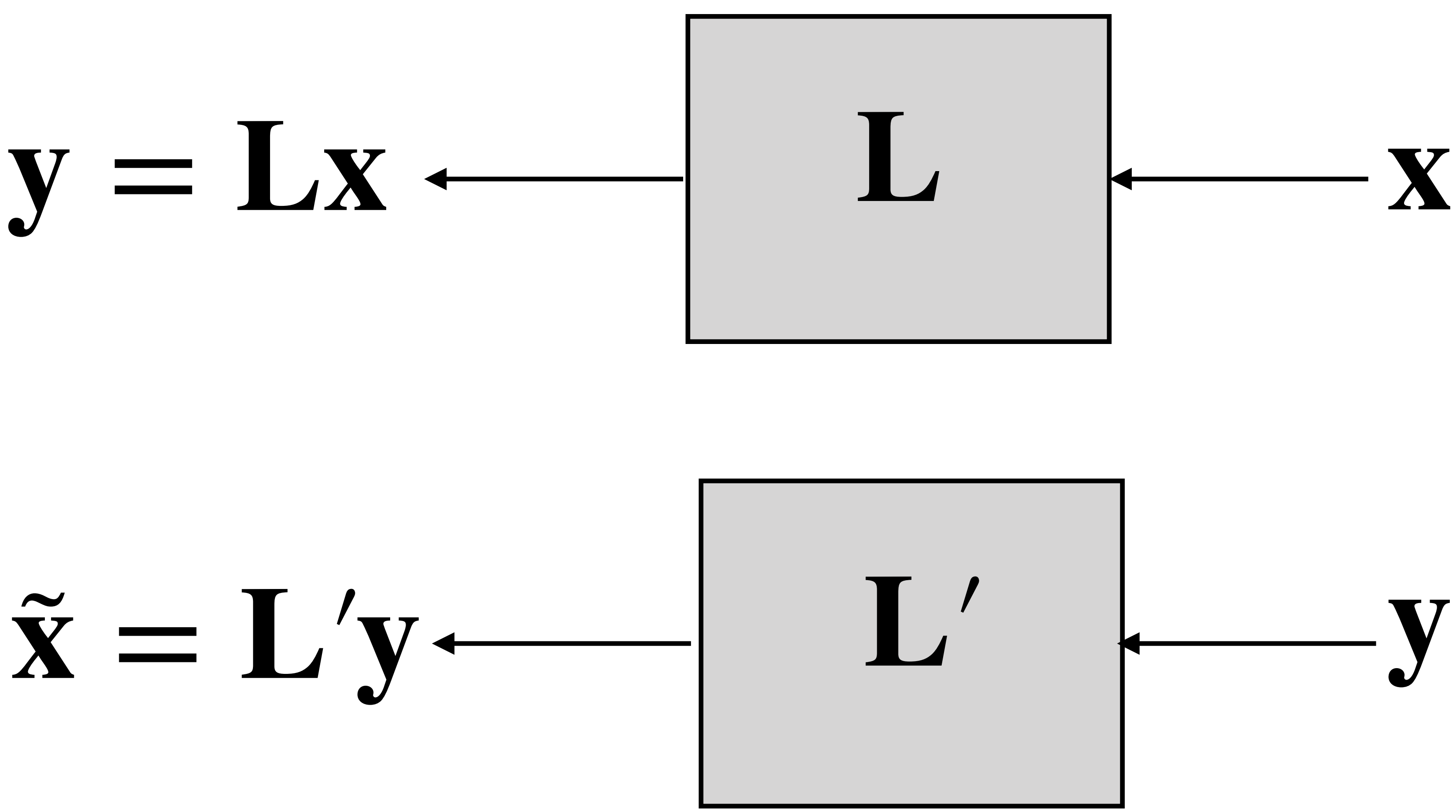
End Do

f: *Forward*

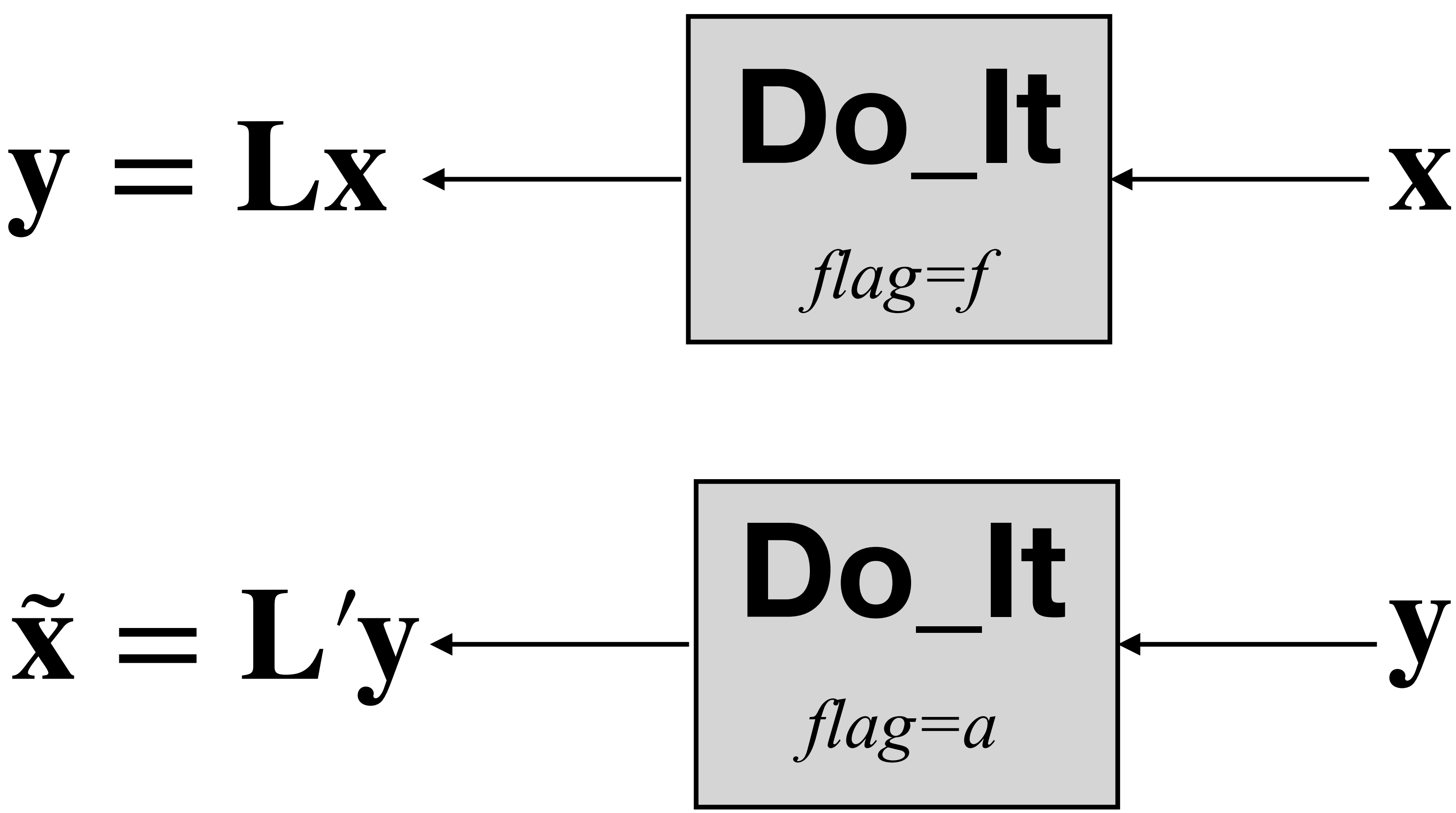
a: *Adjoint or transpose*

Operators *on the flight*

(Matrices are replaced by implicit form linear operators!!)



Operators *on the flight*



f: *Forward*
a: *Adjoint or transpose*

Why this is important?

- Migration and de-migration operators cannot be written as matrices. Least-squares migration requires the **Do_It** approach.

- In addition, time-domain Radon transforms can not be written in explicit form. For Radon problems I also use the **Do_It** approach.

- Reconstruction problems often entail using operators that cannot be written via explicit matrices (FFTs, Curvelet Frames, etc). In this case, we also adopt the **Do_It** approach

- In general, multidimensional problems where $\mathbf{m}$ and $\mathbf{d}$ are not vectors can still be analyzed via linear algebra tools by providing simple rules that make linear operators behave like matrices

Dot-product test

- How do you guarantee that the codes for the forward and adjoint operators behave like $\mathbf{L}$ and $\mathbf{L}'$?

- Let's $\mathbf{L}$ be the forward operator and let's call $\mathbf{B}$ the tentative adjoint or transpose operator

$$\mathbf{y}_1 = \mathbf{L}\,\mathbf{x}_1\,, \quad \mathbf{x}_2 = \mathbf{B}\,\mathbf{y}_2$$

- Form the two inner products

$$\mathbf{y}_2^T\mathbf{y}_1 = \mathbf{y}_2\mathbf{L}\mathbf{x}_1$$
$$\mathbf{x}_1^T\mathbf{x}_2 = \mathbf{x}_1\mathbf{B}\mathbf{y}_2$$

$$< \mathbf{y}_2, \mathbf{y}_1 > = < \mathbf{y}_2, \mathbf{L}\mathbf{x}_1 >$$
$$< \mathbf{x}_1, \mathbf{x}_2 > = < \mathbf{x}_1\mathbf{B}\mathbf{y}_2 >$$

More general notation

Dot-product test

- Notice, the definition of the inner product can be extended to multidimensional array

- For vectors $$\mathbf{y}_2^T \mathbf{y}_1 = \textbf{sum}(y_2.*y_1)$$

- For ND-arrays $$< \mathbf{y}_2, \mathbf{y}_1 > = \textbf{sum}(y_2[:]\,.*\,y_1[:])$$

Dot-product test

The two inner products are equal

$$\mathbf{y}_2^T\mathbf{y}_1 = \mathbf{y}_2\mathbf{L}\mathbf{x}_1 \qquad\qquad \mathbf{x}_1^T\mathbf{x}_2 = \mathbf{x}_1\mathbf{B}\mathbf{y}_2$$

If and only if $\mathbf{B} = \mathbf{L}'$

Therefore, one can write a code for $\mathbf{L}$ and a code for $\mathbf{L}$**’**, do the dot-product test and if the two product are equal then one can say that $\mathbf{L}$ and $\mathbf{L}$**’** (packed by the function **Do_It**) behave like a matrix and transpose multiplications, respectively. Then you can safely use all you know about linear algebra to solve an inverse problem!

Dot-product test in practice

```
using LinearAlgebra, FFTW


# Dot product test example using operators rather than matrices
# Fourier DFT matrices and its Hermitian Transpose are replaced by
# on-the-flight FFTs
M = 512
x1 = randn(M)
y1 = fft(x1)
y2 = randn(M)
x2 = M*ifft(y2)            => Why I have multiplied by M?
dot_x = x1'*x2
dot_y = y1'*y2
println(dot_x)
println(dot_y)
```

Conjugate Gradients

- It is more efficient to use the Conjugate Gradient (CG) method than SD.
- I will not discuss CG but it basically amounts to also applying on the flight the operators $\mathbf{L}$ and $\mathbf{L'}$ in each iteration (step)
- The Conjugate Gradient algorithm is quite popular so you can find one in C,Fortran, Matlab, Python, etc by just doing a google search.
- *CG minimizes the general quadratic cost function:*

$$J = \|\mathbf{d} - \mathbf{Lm}\|_2^2$$

Concatenation of operators

- Forward

$$\mathbf{L} = \mathbf{ABC}$$

- Adjoint

$$\mathbf{L}' = \mathbf{C}'\mathbf{B}'\mathbf{A}'$$

- Dot product test must work for individual operators (codes) and then it will work for $\mathbf{L}$ and $\mathbf{L}'$

Preconditioning

- When using iterative methods (SD or CG), I prefer not to worry about matrices of weights

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{W}\mathbf{m}\|_2^2$$

- Therefore, I like to use the following change of variables

$$\mathbf{W}\mathbf{m} = \mathbf{u} \rightarrow \mathbf{P}\mathbf{u} = \mathbf{m}$$

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{P}\mathbf{u}\|_2^2 + \mu\|\mathbf{u}\|_2^2$$

Preconditioning

- Minimize with iterative solver

$$J = \|\mathbf{d} - \mathbf{LPu}\|_2^2 + \mu\|\mathbf{u}\|_2^2$$

- With **Do_It** operators:

$$\textbf{Forward} : \mathbf{LP}$$

$$\textbf{Adjoint} : \mathbf{P}'\mathbf{L}'$$

Preconditioning

- In this example, I use SD to minimize

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{P}\mathbf{u}\|_2^2 + \mu\|\mathbf{u}\|_2^2$$

Do until convergence

$$\mathbf{r}^k = (\mathbf{L}\mathbf{P}\mathbf{u}^k - \mathbf{d})$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \alpha(\,\mathbf{P}'\mathbf{L}'(\mathbf{r}^k) + \mu\,\mathbf{u}^k)$$

End

$$\mathbf{m}_{sol} = \mathbf{P}\mathbf{u}$$

Preconditioning

- **W** and **P** should behave like the inverse of each other

$$\mathbf{Wm} = \mathbf{u} \rightarrow \mathbf{Pu} = \mathbf{m}$$

Then

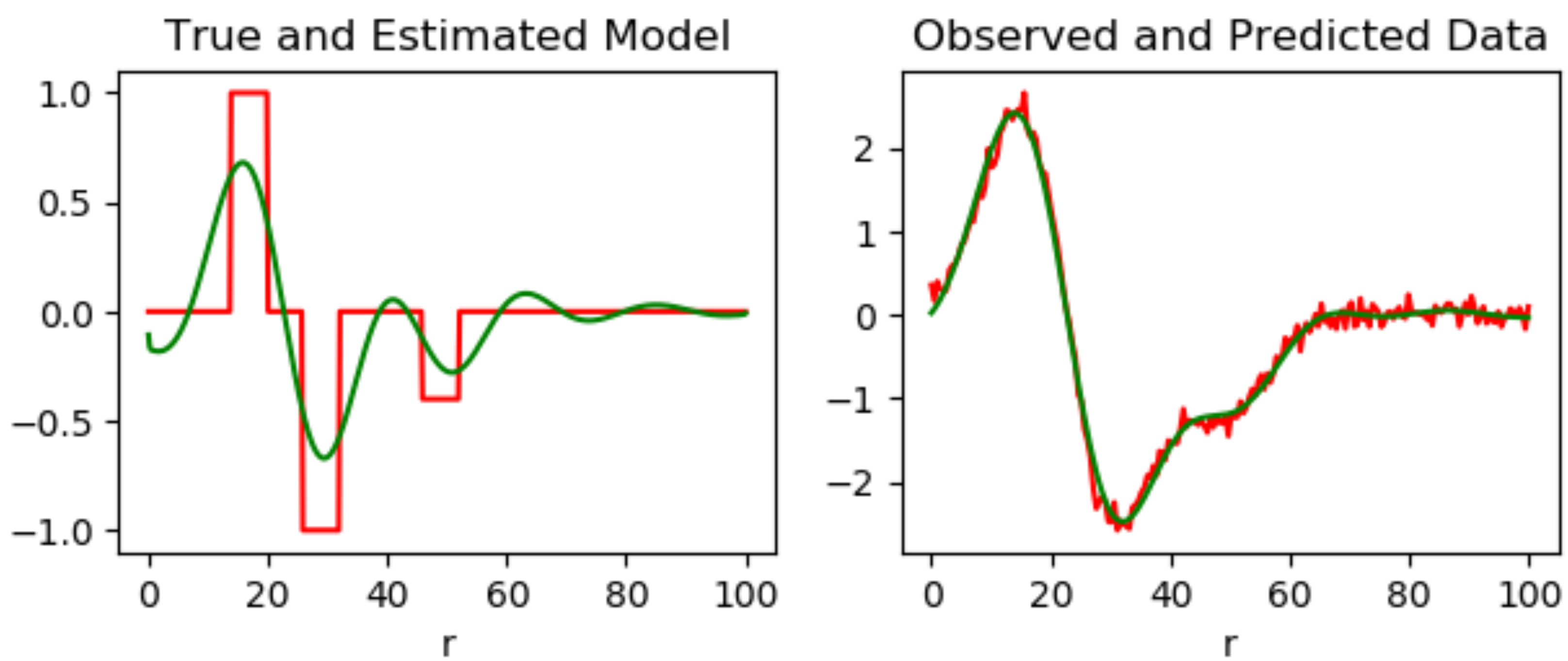
W is a high-pass operator (applies roughening)

P is low-pass operator (applies smoothing)

Oldie but goodie: S. Ronen, D. Nichols, R. Bale, and R. Ferber, “Dealiasing DMO: Good-pass, bad-pass, and unconstrained”, SEG Technical Program Expanded Abstracts 1995. January 1995, 743-746

Preconditioning (SD)

$$\mathbf{P} = \frac{1}{2}\begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$



My favourite CG (CGLS)

- I prefer to use a CG algorithm that minimizes the so called Standard Form

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{m}\|_2^2$$

```
function ConjugateGradients(d,operators,parameters;Niter=10,mu=0)
   r = copy(d)
   g = LinearOperator(r,operators,parameters,adj=true)
   m = zero(g)
   s = copy(g)
   gamma = InnerProduct(g,g)

   for iter = 1 : Niter
      t = LinearOperator(s,operators,parameters,adj=false)
      delta = InnerProduct(t,t) + mu*InnerProduct(s,s)
      alpha = gamma/delta
      m = m + alpha*s
      r = r - alpha*t
      g = LinearOperator(r,operators,parameters,adj=true)
      g = g - mu*m
      gamma0 = copy(gamma)
      gamma = InnerProduct(g,g)
       beta = gamma/gamma0
      s = beta*s + g
    end

   return m
end
```

$$\mathbf{g} = \mathbf{L}'\mathbf{r}$$

$$\mathbf{t} = \mathbf{L}\mathbf{s}$$

$$\mathbf{g} = \mathbf{L}'\mathbf{r}$$

https://github.com/SeismicJulia/SeisReconstruction.jl/blob/master/src/Tools/ConjugateGradients.jl

Modified from: https://web.stanford.edu/group/SOL/software/cgls/

Advantages of CGLS to minimize the standard form

You can turn a general problem with quadratic regularization into its standard form and use the previous page CGLS algorithm

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 + \mu\|\mathbf{W}\mathbf{m}\|_2^2$$

$$\Downarrow$$

$$J = \|\mathbf{d} - \mathbf{L}\mathbf{P}\mathbf{u}\|_2^2 + \mu\|\mathbf{u}\|_2^2, \ \textbf{with}, \mathbf{u} = \mathbf{P}\mathbf{m}$$

$$\mathbf{P} = \mathbf{W}^{-1} \ \leftarrow \ \textbf{to discuss}$$

Advantages of CGLS to minimize the standard form

You can also concatenate linear operators

$$J = \|\mathbf{W}_d[\mathbf{d} - \mathbf{A}\mathbf{B}\mathbf{m}]\|_2^2 + \mu\|\mathbf{W}_m\mathbf{m}\|_2^2$$

$$J = \|\mathbf{d}_{w_d} - \mathbf{W}_d\mathbf{A}\,\mathbf{B}\,\mathbf{P}_m\mathbf{u}\|_2^2 + \mu\|\mathbf{u}\|_2^2, \ \textbf{with}, \mathbf{u} = \mathbf{P}_m\mathbf{m}, \ \mathbf{d}_{w_d} = \mathbf{W}_d\,\mathbf{d}$$

operators = $[\mathbf{W}_d\,, \mathbf{A}\,, \mathbf{B}\,, \mathbf{P}_m]$

LinearOperator(Input,operators,parameters,adj=false) **Output** $= \mathbf{W}_d\,\mathbf{A}\,\mathbf{B}\,\mathbf{P}_m$ **Input**

LinearOperator(Input,operators,parameters,adj=true) **Output** $= \mathbf{P}_m'\,\mathbf{B}'\,\mathbf{A}'\,\mathbf{W}_d'$ **Input**

Example: Time-domain Parabolic Radon Transform

- Notebook: `ConjugateGradient_in_SeisReconstruction.ipynbin`

- Compute reflections with parabolic moveout and then use CG to retrieve the Radon domain coefficient that model the reflections.

- Radon adjoint and forward operators:

$$m'(\tau, p) = \sum_h d(t = \tau + pg(h), h) \rightarrow \mathbf{m}' = \mathbf{L}'\mathbf{d}$$

$$d(t, h) = \sum_p m(\tau = t - pg(h)) \rightarrow \mathbf{d} = \mathbf{L}\mathbf{m}$$

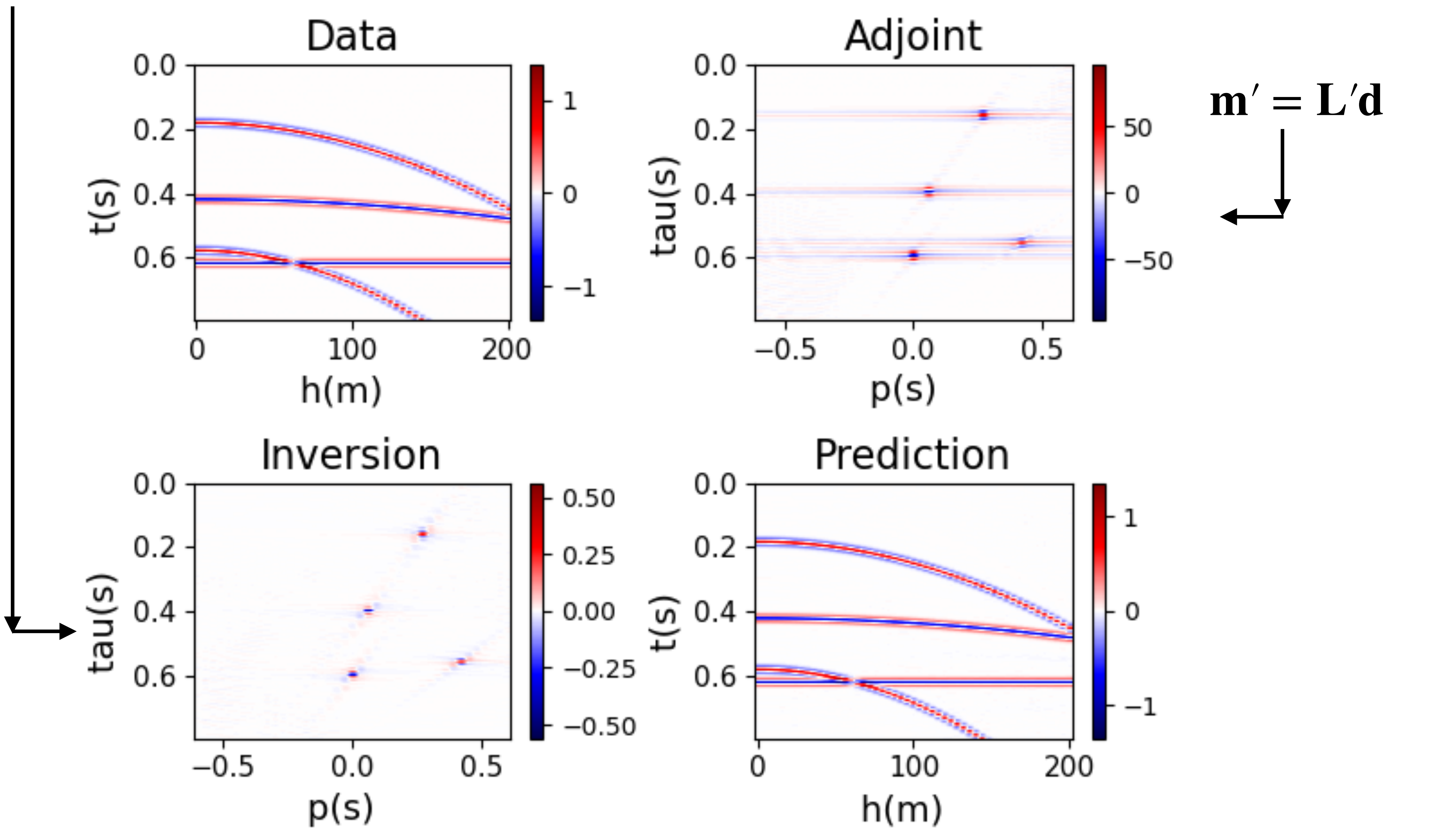
- **Linear RT** $g(h) = h, p$: **Slope**

- **Parabolic RT** $g(h) = (h/h_{max})^2, p$: **Residual moveout at far offset**

Example: Time-domain Parabolic Radon Transform

$\mathbf{m}_{sol} = \textbf{argmin}_{\mathbf{m}} \|\mathbf{Lm} - \mathbf{d}\|_2^2 + \mu\|\mathbf{m}\|_2^2$ **Via CGLS**

$\mathbf{m}' = \mathbf{L}'\mathbf{d}$



`CGLS_versus_SD_example.ipynb`

Comparison of SD and CGLS path for a simple problem with two unknown where one can visualize the cost function