Large-scale inverse problems in geoscience

Matteo Ravasi

Data Assimilation Summer School, Timisoara - 01/08/2019

Quiz

$$J = ||\mathbf{d} - \mathbf{G}\mathbf{m}||^2 + \lambda^2 ||\mathbf{D}\mathbf{m}||^2$$

How do you minimize this cost function *without* creating explicit matrices?

Goals

Goals

- Basic of linear algebra (hopefully already...)

Goals

- Basic of linear algebra (hopefully already...)
- From textbook linear algebra to real-life linear algebra

Goals

- Basic of linear algebra (hopefully already...)
- From textbook linear algebra to real-life linear algebra
- Linear algebra in software: **PyLops**

Goals

- Basic of linear algebra (hopefully already...)
- From textbook linear algebra to real-life linear algebra
- Linear algebra in software: **PyLops**

- Python is **Slow** if you write is as C, it is not if you leverage its advanced libraries (numpy, scipy, tensorflow, pytorch, pycuda, dask...)

Course outline

Course outline

- Motivation: inverse problems in geoscience, how large?

Course outline

- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)

Course outline

- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)
- Linear operators: philosophy and implementation (EX2)

Course outline

- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)
- Linear operators: philosophy and implementation (EX2)
- Least squares solvers (EX3)

Course outline

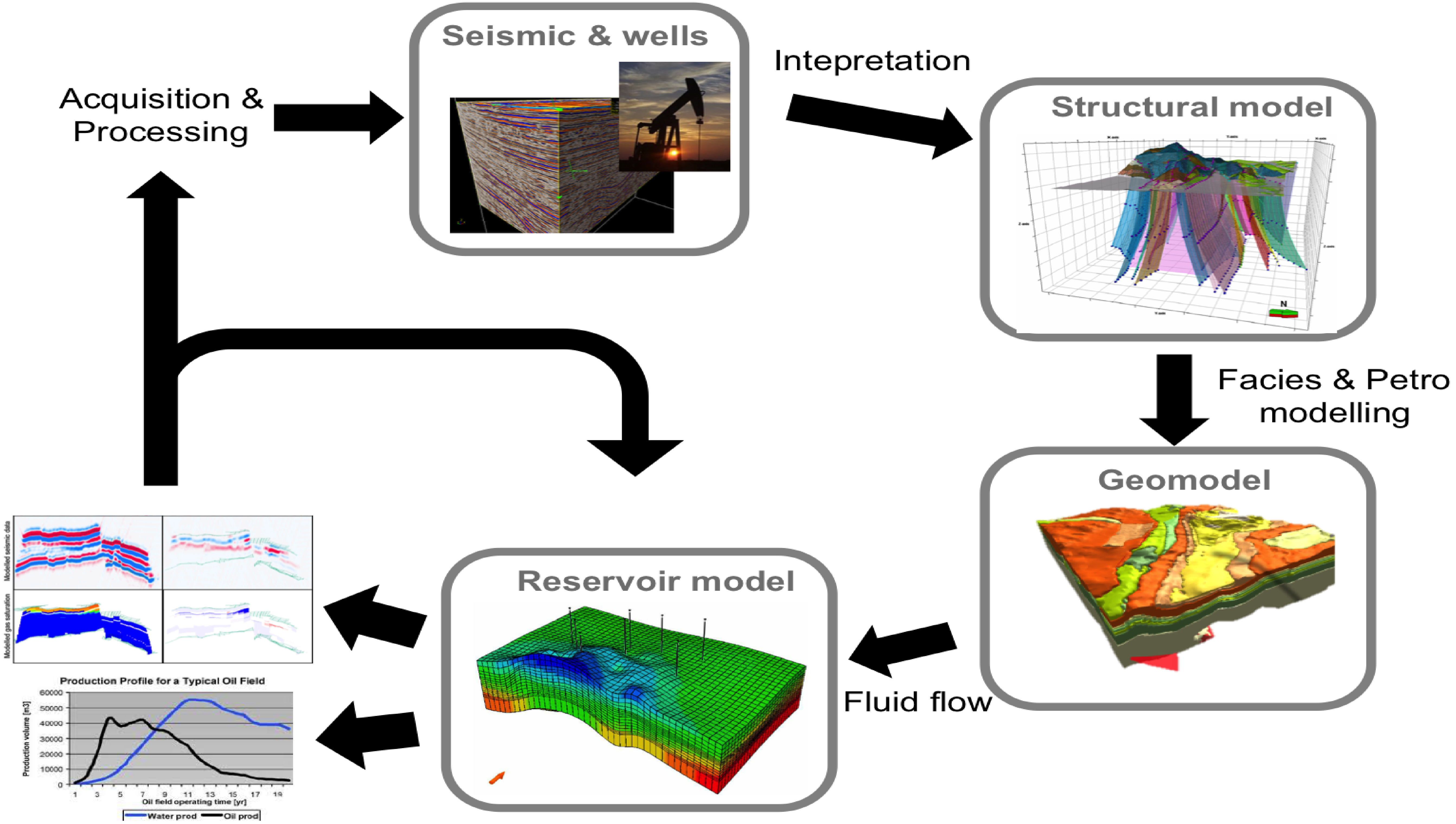
- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)
- Linear operators: philosophy and implementation (EX2)
- Least squares solvers (EX3)
- Sparse solvers (EX4)

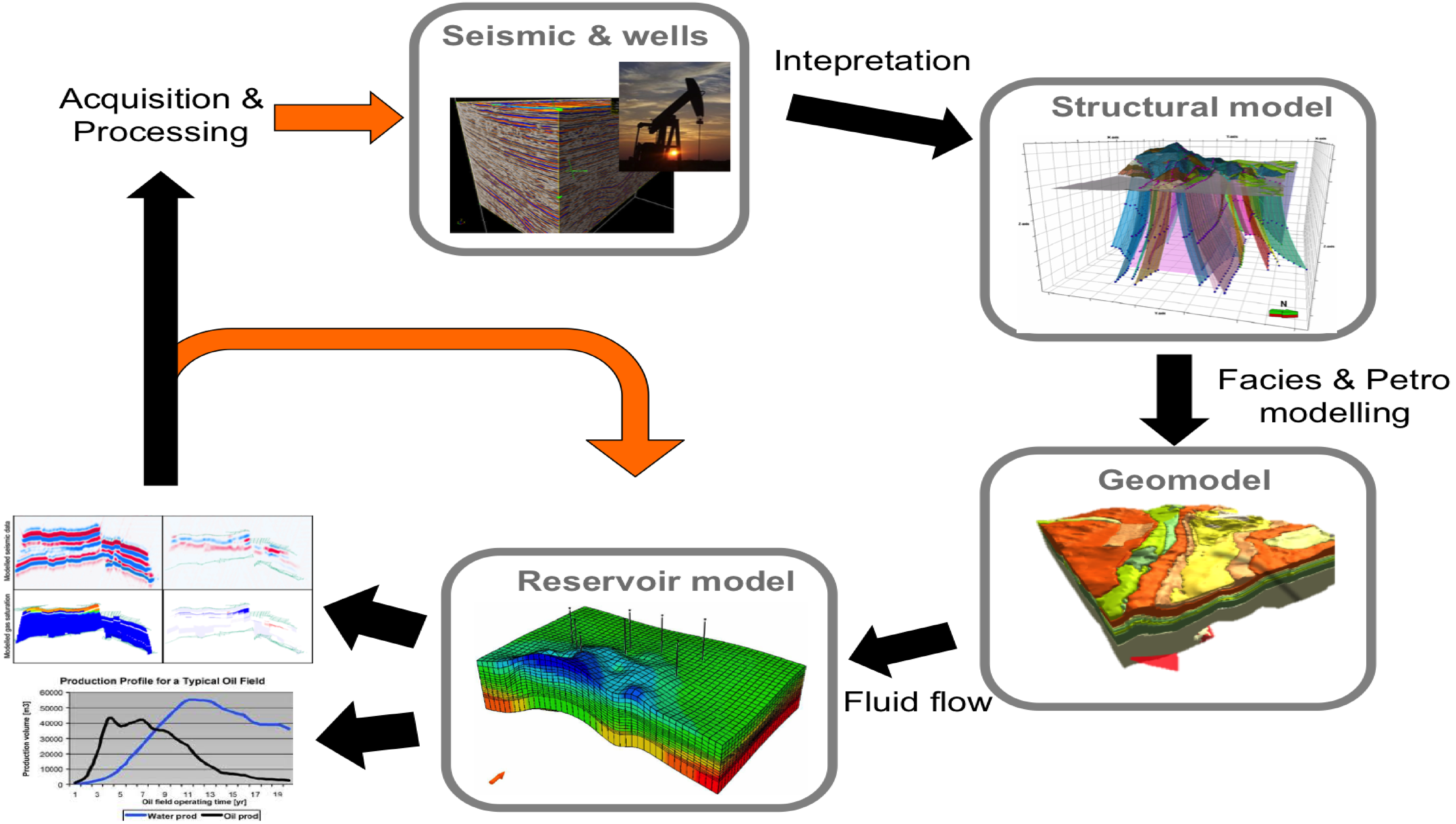
Course outline

- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)
- Linear operators: philosophy and implementation (EX2)
- Least squares solvers (EX3)
- Sparse solvers (EX4)
- Geophysical applications (EX5, EX6)

Course outline

- Motivation: inverse problems in geoscience, how large?
- Basics of inverse problems (EX1)
- Linear operators: philosophy and implementation (EX2)
- Least squares solvers (EX3)
- Sparse solvers (EX4)
- Geophysical applications (EX5, EX6)
- Beyond single-machine inverse problems: GPUs, distributed...


Seismic & wells
Intepretation
Structural model
Acquisition &
Processing
Facies & Petro
modelling
Geomodel
Reservoir model
Fluid flow
Production Profile for a Typical Oil Field
Production volume [m3]
Oil field operating time [yr]
Water prod
Oil prod


Seismic & wells
Intepretation
Structural model
Facies & Petro
modelling
Geomodel
Fluid flow
Reservoir model
Acquisition &
Processing
Production Profile for a Typical Oil Field
Production volume [m3]
Oil field operating time [yr]
Water prod
Oil prod

Large... how large?

- Seismic processing:

$$n_S = n_R = 10^3, n_t = n_{fft} = 2 \cdot 10^3 \quad (dt = 4ms, t_{ma}$$

$$\rightarrow \mathbf{G} : n_S \cdot n_R \cdot n_{fft}^2 = 4 \cdot 10^{12} * 32bit = 128T$$

- Seismic inversion:

$$n_x = n_y = 10^3, n_z = 800 \quad (dz = 5m, z_{max} = 400$$

$$\rightarrow \mathbf{G} : n_x \cdot n_y \cdot n_z^2 = 6.4 \cdot 10^{11} * 32bit \sim 20T$$

Inverse problems

Fundamental theory developed in the '60/'70

Inverse problems

Fundamental theory developed in the '60/'70

- Mathematicians: general theory, shared with other applications e.g., ML, data assimilation

Inverse problems

Fundamental theory developed in the '60/'70

- Mathematicians: general theory, shared with other applications e.g., ML, data assimilation
- Seismologists: inference of subsurface properties/behaviours from large set of data

Inverse problems

Fundamental theory developed in the '60/'70

- Mathematicians: general theory, shared with other applications e.g., ML, data assimilation
- Seismologists: inference of subsurface properties/behaviours from large set of data
- Atmosferic scientists: more concerned with data assimilation through time

Inverse problems

References:

A. Tarantola, Inverse Problem Theory;

S. Kay, Fundamentals of statistical signal processing;

J. Claerbout, Basic Earth Imaging;

...

Forward model

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad / \quad \mathbf{d} = \mathbf{G}\mathbf{m}$$

Forward model

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad / \quad \mathbf{d} = \mathbf{G}\mathbf{m}$$

- **d**: observed data. Quantity that we can physicially measure - seismic, production, temperature, precipitation, MRI scan…

Forward model

DATA ASSIMILATION summer school

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad / \quad \mathbf{d} = \mathbf{G}\mathbf{m}$$

- **d**: observed data. Quantity that we can physicially measure - seismic, production, temperature, precipitation, MRI scan...
- **m**: model. Quantity that we are interested to know and we think affects the data - rock properties, pressure, saturation, human organs...

Forward model

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad / \quad \mathbf{d} = \mathbf{G}\mathbf{m}$$

- **d**: observed data. Quantity that we can physicially measure - seismic, production, temperature, precipitation, MRI scan...
- **m**: model. Quantity that we are interested to know and we think affects the data - rock properties, pressure, saturation, human organs...
- **G**: modelling operator. Set of equations that we think can explain the data by *nonlinear/linear combination* of model parameters - PDEs, seismic convolution model, camera blurring...

Forward model

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad / \quad \mathbf{d} = \mathbf{G}\mathbf{m}$$

Unique and *easy* to compute, perhaps time consuming...

Inverse model

$$\mathbf{m} = \mathbf{g}^{-1}(\mathbf{d}) \quad / \quad \mathbf{m} = \mathbf{G}^{-1}\mathbf{d}$$

Nonunique and hard to *compute*, both expensive and unstable...

- **square** (N = M): rarely the case in real life. Solution:

$$\mathbf{G}^{-1} \rightarrow \mathbf{m}_{est} = \mathbf{G}^{-1}\mathbf{d}$$

- **square** (N = M): rarely the case in real life. Solution:

$$\mathbf{G}^{-1} \rightarrow \mathbf{m}_{est} = \mathbf{G}^{-1}\mathbf{d}$$

- **overdetermined** (N > M): most common case, robust to noise as more data poin than model parameters. Least-squares solution:

$$J = ||\mathbf{d} - \mathbf{G}\mathbf{m}||^2 \rightarrow \mathbf{m}_{est} = (\mathbf{G}^{\mathbf{H}}\mathbf{G})^{-1}\mathbf{G}^{\mathbf{H}}\mathbf{d}$$

- **square** (N = M): rarely the case in real life. Solution:

$$\mathbf{G}^{-1} \rightarrow \mathbf{m}_{est} = \mathbf{G}^{-1}\mathbf{d}$$

- **overdetermined** (N > M): most common case, robust to noise as more data poin

  than model parameters. Least-squares solution:

$$J = ||\mathbf{d} - \mathbf{Gm}||^2 \rightarrow \mathbf{m}_{est} = (\mathbf{G^H G})^{-1}\mathbf{G^H d}$$

- **underdetermined** (N < M): not ideal, but sometimes only option - e.g., MRI scar

  Least-squares solution:

$$J = ||\mathbf{m}||^2 \quad s.t \quad \mathbf{d} = \mathbf{Gm} \rightarrow \mathbf{m}_{est} = \mathbf{G^H}(\mathbf{GG^H})^{-}$$

  Sparse solution:

$$J = ||\mathbf{m}|| \quad s.t \quad \mathbf{d} = \mathbf{Gm}$$

Inversion in practice

Now that we know the analytical expressions, how do we find

$$\mathbf{G}^{-1} \quad (\mathbf{G}^{\mathbf{H}}\mathbf{G})^{-1} \quad (\mathbf{G}\mathbf{G}^{\mathbf{H}})^{-1}$$

Solutions studied in the fields of *Numerical analysis and Optimization*.

Two families of solvers:

Inversion in practice

Now that we know the analytical expressions, how do we find

$$\mathbf{G}^{-1} \quad (\mathbf{G}^{\mathbf{H}}\mathbf{G})^{-1} \quad (\mathbf{G}\mathbf{G}^{\mathbf{H}})^{-1}$$

Solutions studied in the fields of *Numerical analysis and Optimization*.

Two families of solvers:

- **direct**: LU, QR, SVD...

Inversion in practice

Now that we know the analytical expressions, how do we find

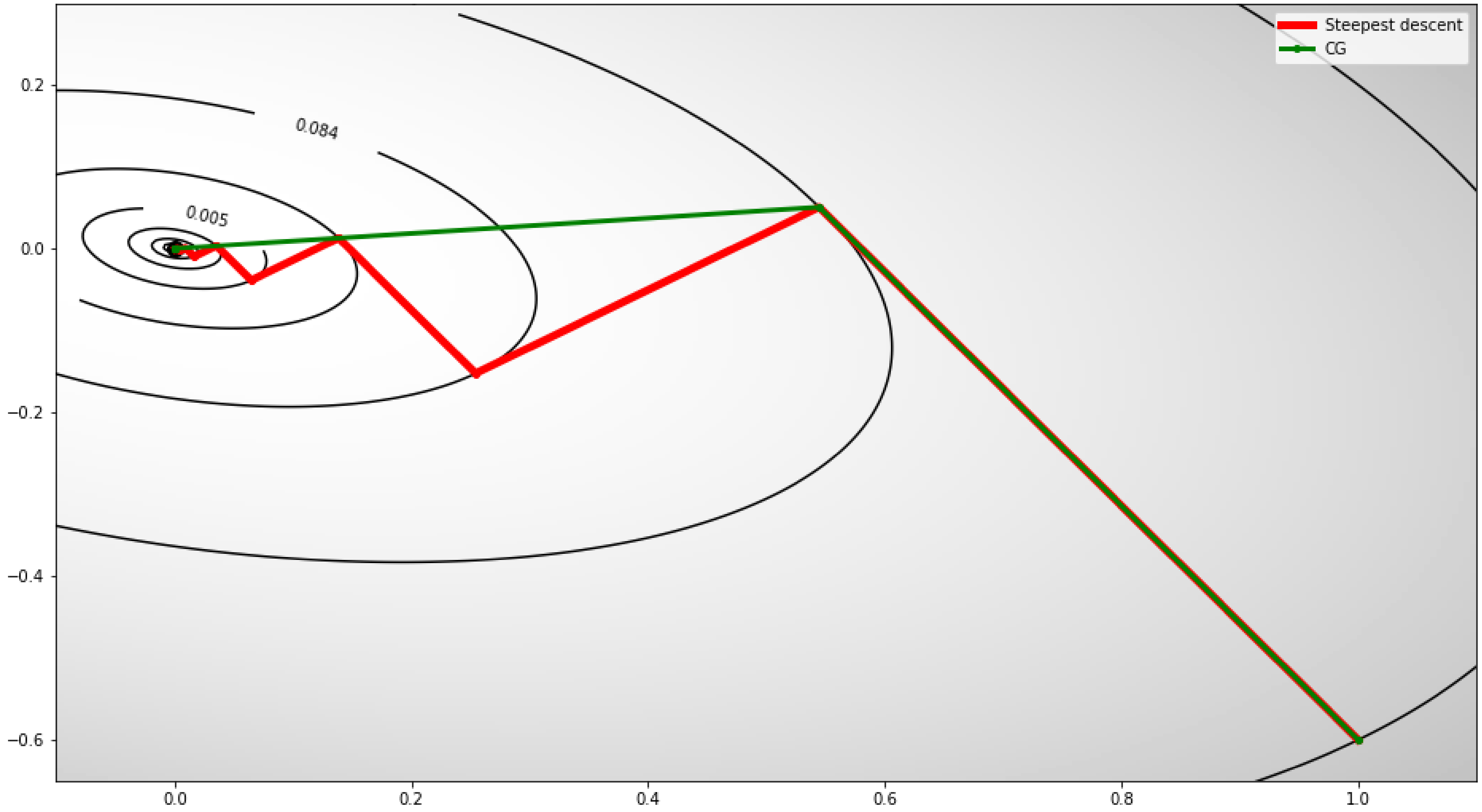
$$\mathbf{G}^{-1} \quad (\mathbf{G}^{\mathbf{H}}\mathbf{G})^{-1} \quad (\mathbf{G}\mathbf{G}^{\mathbf{H}})^{-1}$$

Solutions studied in the fields of *Numerical analysis and Optimization*.

Two families of solvers:

- **direct**: LU, QR, SVD...
- **iterative**: gradient based methods - CG, LSQR, GMRES...

Inversion in practice



Let's practice EX1.

Why iterative?

$\mathbf{G}$ is too large to be inverted (or solved explicitely)

$\mathbf{G}$ is too large to be stored in memory

Take on message:

$$\mathbf{m}_{est} = \sum_{i=0}^{N_{iter}} f(\mathbf{G}, \mathbf{d}, \mathbf{m}_0) \qquad \mathbf{Gm}, \mathbf{G}^H\mathbf{d}, (\mathbf{m}^H\mathbf{m}, \mathbf{d}^H$$

Linear Operators

A piece of computer code that can perform *forward* and *adjoint* operations without the need to store an *explicit matrix*:

$$\mathbf{Gm}, \mathbf{G}^H\mathbf{d}$$

but how do we make sure that forward and adjoint are correctly implemented? --> **Dot-Test**

Ex: Diagonal

$$
\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & d_N \end{bmatrix}, \mathbf{D}^H = \begin{bmatrix} d_1 & 0 & \dots \\ 0 & d_2 & \dots \\ \dots & & \\ 0 & 0 & \dots \end{bmatrix}
$$

```
# forward
def _matvec(x)
    return self.diag * x
# adjoint
def _matvec(x)
    return self.diag * x
```

Ex: Diagonal

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & d_N \end{bmatrix}, \mathbf{D}^H = \begin{bmatrix} d_1 & 0 & \dots \\ 0 & d_2 & \dots \\ \dots & & \\ 0 & 0 & \dots \end{bmatrix}$$

```
# forward
def _matvec(x)
    return self.diag * x
# adjoint
def _matvec(x)
    return self.diag * x
```

Ex: Diagonal

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & d_N \end{bmatrix}, \mathbf{D}^H = \begin{bmatrix} d_1 & 0 & \dots \\ 0 & d_2 & \dots \\ \dots & & \\ 0 & 0 & \dots \end{bmatrix}$$

```
# forward
def _matvec(x)
    return self.diag * x
# adjoint
def _matvec(x)
    return self.diag * x
```

Ex: Diagonal

Dot-Test: a correct implementation of forward and adjoint for a linear operator should verify the the following *equality* within a numerical tolerance:

$$(\mathbf{G} \cdot \mathbf{u})^H \mathbf{v} = \mathbf{u}^H (\mathbf{G}^H \cdot \mathbf{v})$$

Ex: First derivative

DATA ASSIMILATION summer school

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & \dots & 0 & 0 \\ -0.5 & 0 & 0.5 & \dots & 0 \\ \dots & & & & \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

```
def _matvec(x):
x, y = x.squeeze(), np.zeros(self.N, self.dtype)
y[1:-1] = (0.5 * x[2:] - 0.5 * x[0:-2]) / self.sampling
# edges
y[0] = (x[1] - x[0]) / self.sampling
y[-1] = (x[-1] - x[-2]) / self.sampling
```

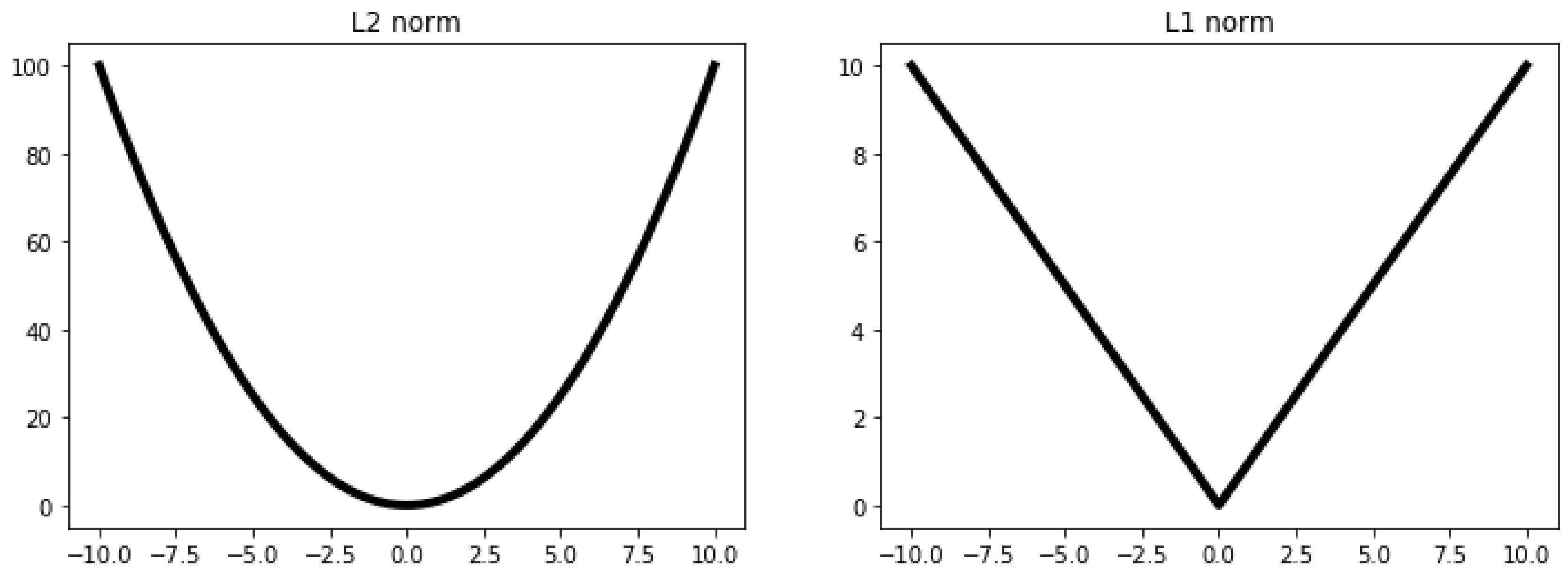
Ex: First derivative

$$\mathbf{D}^H = \begin{bmatrix} -1 & -0.5 & \dots & 0 & 0 \\ 1 & 0 & -0.5 & \dots & 0 \\ \dots & & & & \\ 0 & 0 & \dots & -0.5 & 1 \end{bmatrix}$$

```
def _rmatvec(x):
x, y = x.squeeze(), np.zeros(self.N, self.dtype)
y[0:-2] -= (0.5 * x[1:-1]) / self.sampling
y[2:] += (0.5 * x[1:-1]) / self.sampling
# edges
y[0] -= x[0] / self.sampling
y[1] += x[0] / self.sampling
y[-2] -= x[-1] / self.sampling
y[-1] += x[-1] / self.sampling
```

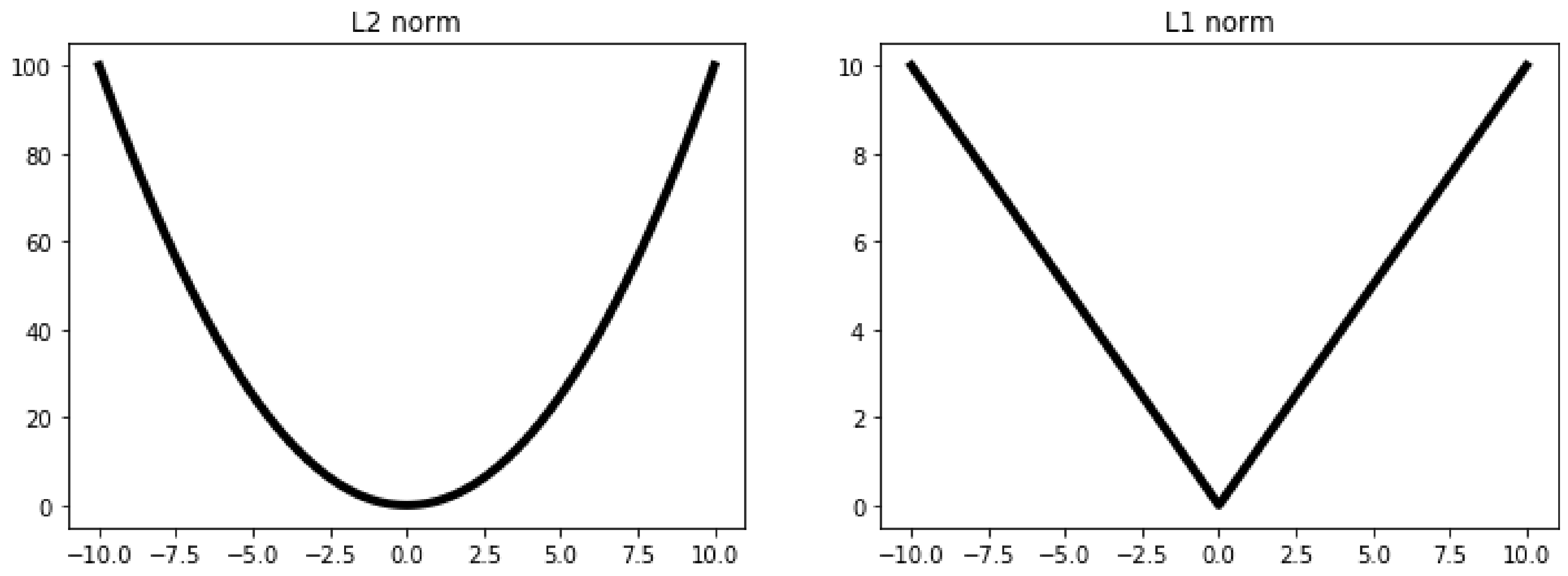
Let's practice EX2.

Solvers


L2 norm
L1 norm
100
80
60
40
20
0
10
8
6
4
2
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

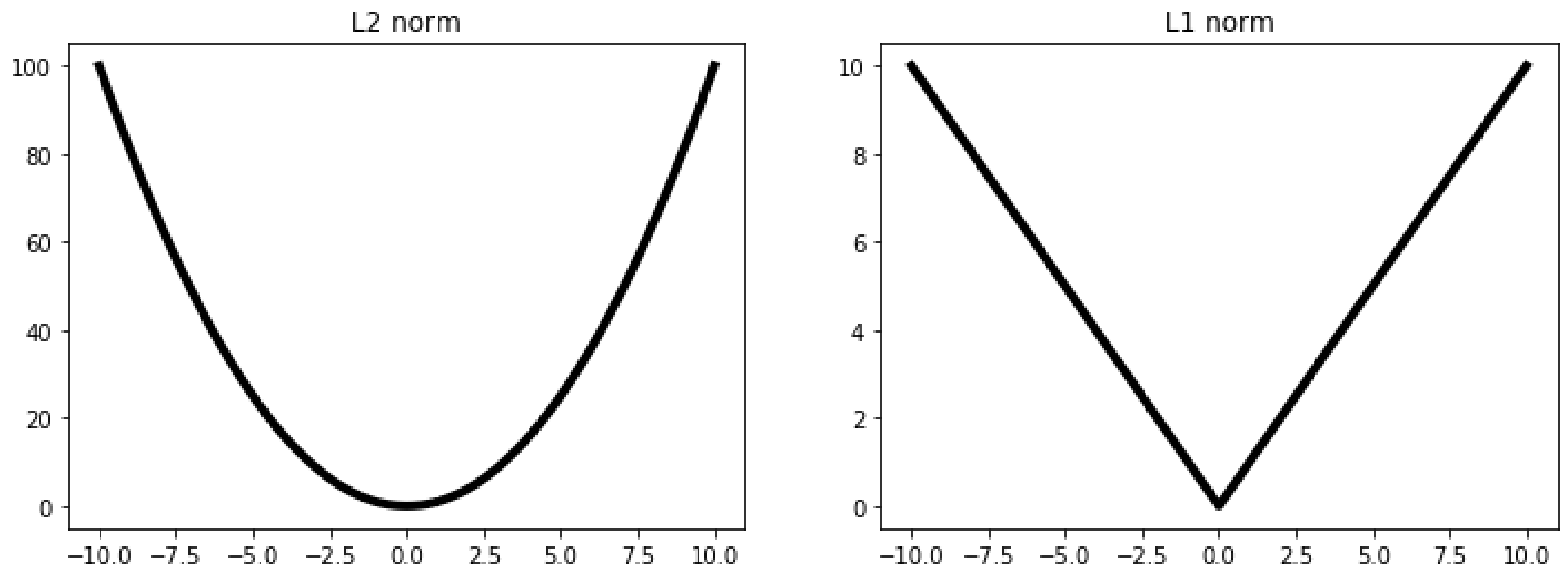
Solvers

- **Least-squares**: regularized, preconditioned, bayesian



Solvers

- **Least-squares**: regularized, preconditioned, bayesian
- **L1**: sparsity promoting, blockiness promoting



Least-squares - Regularized inversion

Add information to the inverse problem --> mitigate *ill-posedness*

$$J = ||\mathbf{d} - \mathbf{G}\mathbf{m}||^2_{\mathbf{W}_d} + \sum_i \epsilon^2_{R_i} ||\mathbf{d}_{R_i} - \mathbf{R}_i \mathbf{m}||^2_{\mathbf{W}_{R_i}}$$

Least-squares - Regularized inversion

Add information to the inverse problem --> mitigate *ill-posedness*

```
def RegularizedInversion(G, Reg, d, dreg, epsR):
# operator
Gtot = VStack([G, epsR * Reg])
# data
dtot = np.hstack((d, epsR*dreg))
# solver
minv = lsqr(Gtot, dtot)[0]
```

Least-squares - Bayesian inversion

Add prior information to the inverse problem --> mitigate *ill-posedness*

$$J = ||\mathbf{d} - \mathbf{G}\mathbf{m}||^2_{\mathbf{C}_d^{-1}} + ||\mathbf{m}_0 - \mathbf{m}||^2_{\mathbf{C}_m^{-1}}$$

$$\begin{bmatrix} \mathbf{C}_d^{-1/2}\mathbf{G} \\ \mathbf{C}_m^{-1/2} \end{bmatrix} \mathbf{m} = \begin{bmatrix} \mathbf{C}_d^{-1/2}\mathbf{d} \\ \mathbf{C}_m^{-1/2}\mathbf{m}_0 \end{bmatrix} \rightarrow$$

$$\mathbf{m} = (\mathbf{G}^H\mathbf{C}_d^{-1}\mathbf{G} + \mathbf{C}_m^{-1})^{-1}(\mathbf{G}^H\mathbf{C}_d^{-1}\mathbf{d} + \mathbf{C}_m^{-1}\mathbf{m_0})$$

Least-squares - Bayesian inversion

Add prior information to the inverse problem --> mitigate *ill-posedness*

$$J = ||\mathbf{d} - \mathbf{G}\mathbf{m}||^2_{\mathbf{C}_d^{-1}} + ||\mathbf{m}_0 - \mathbf{m}||^2_{\mathbf{C}_m^{-1}}$$

$$\mathbf{m} = \mathbf{m_0} + \mathbf{C}_m\mathbf{R}^H(\mathbf{R}\mathbf{C}_m\mathbf{R}^H + \mathbf{C}_d)^{-1}(\mathbf{d} - \mathbf{R}\mathbf{m_0})$$

Least-squares - Bayesian inversion

Add prior information to the inverse problem --> mitigate *ill-posedness*

```
def BayesianInversion(G, d, Cm, Cd):
# operator
Gbayes = G * Cm * G.H + Cd
# data
dbayes = d - G * m0
# solver
minv = m0 + Cm * G.H * lsqr(Gbayes, dbayes)[0]
```

Least-squares - Preconditioned inversion

Limit the range of plausible models --> mitigate *ill-posedness*

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

$$\mathbf{m} = \mathbf{Pp}$$

Least-squares - Preconditioned inversion

Limit the range of plausible models --> mitigate *ill-posedness*

```
def PreconditionedInversion(G, P, d):
# operator
Gtot = G * P
# solver
minv = lsqr(Gtot, d)[0]
```

Sparsity

Introduce L1 norms to cost function

Sparsity

Introduce L1 norms to cost function

Sparsity

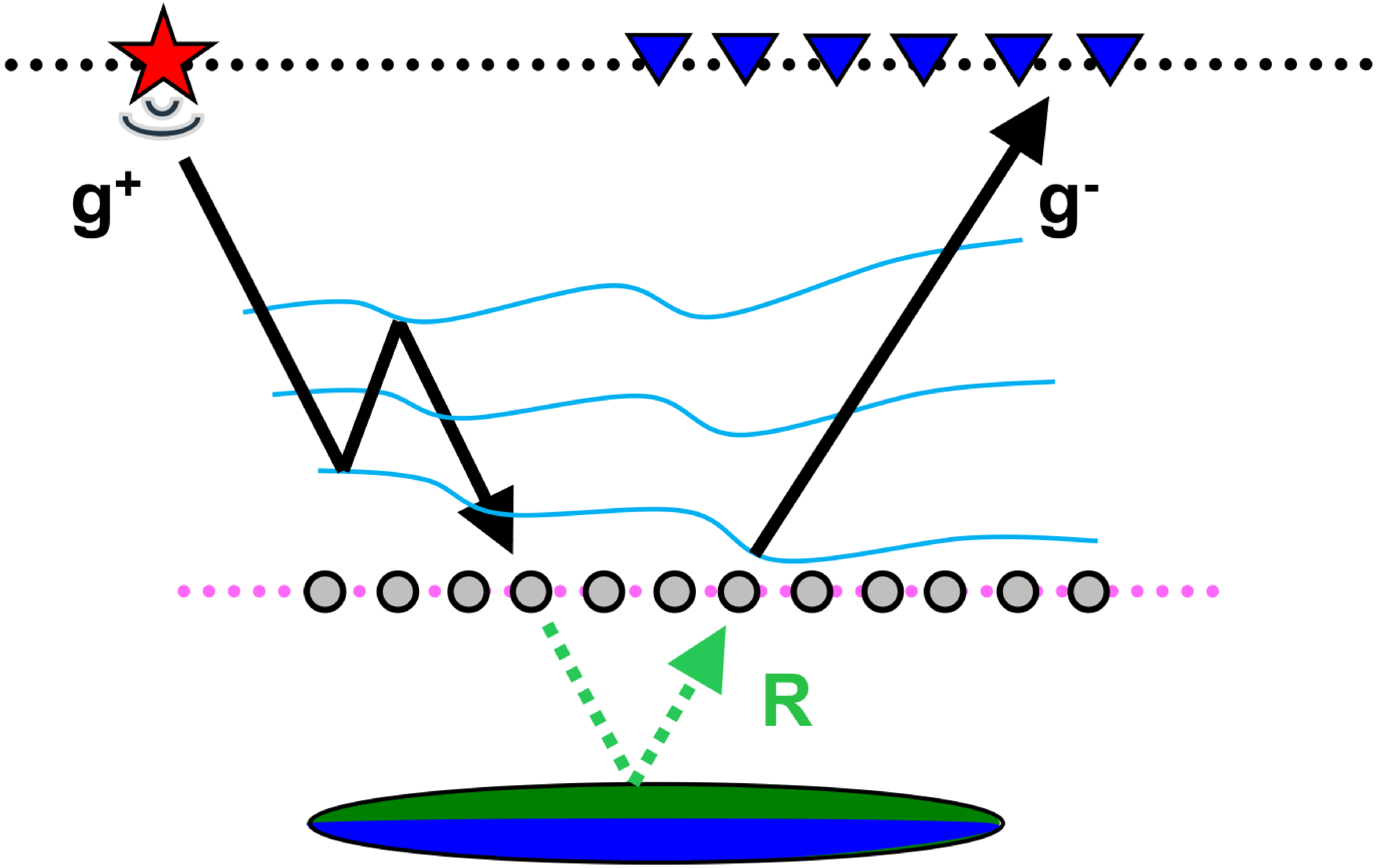
Introduce L1 norms to cost function

Sparsity

Introduce L1 norms to cost function

Let's practice EX3-EX4.

Seismic redatuming



Seismic redatuming

Integral relation:

$$g^{-}(t, x_s, x_v) = \mathscr{F}^{-1}\left(\int_S g^{+}(f, x_s, x_r)\mathscr{F}(R(t, x_r, x_v)\right.$$

Discretized relation:

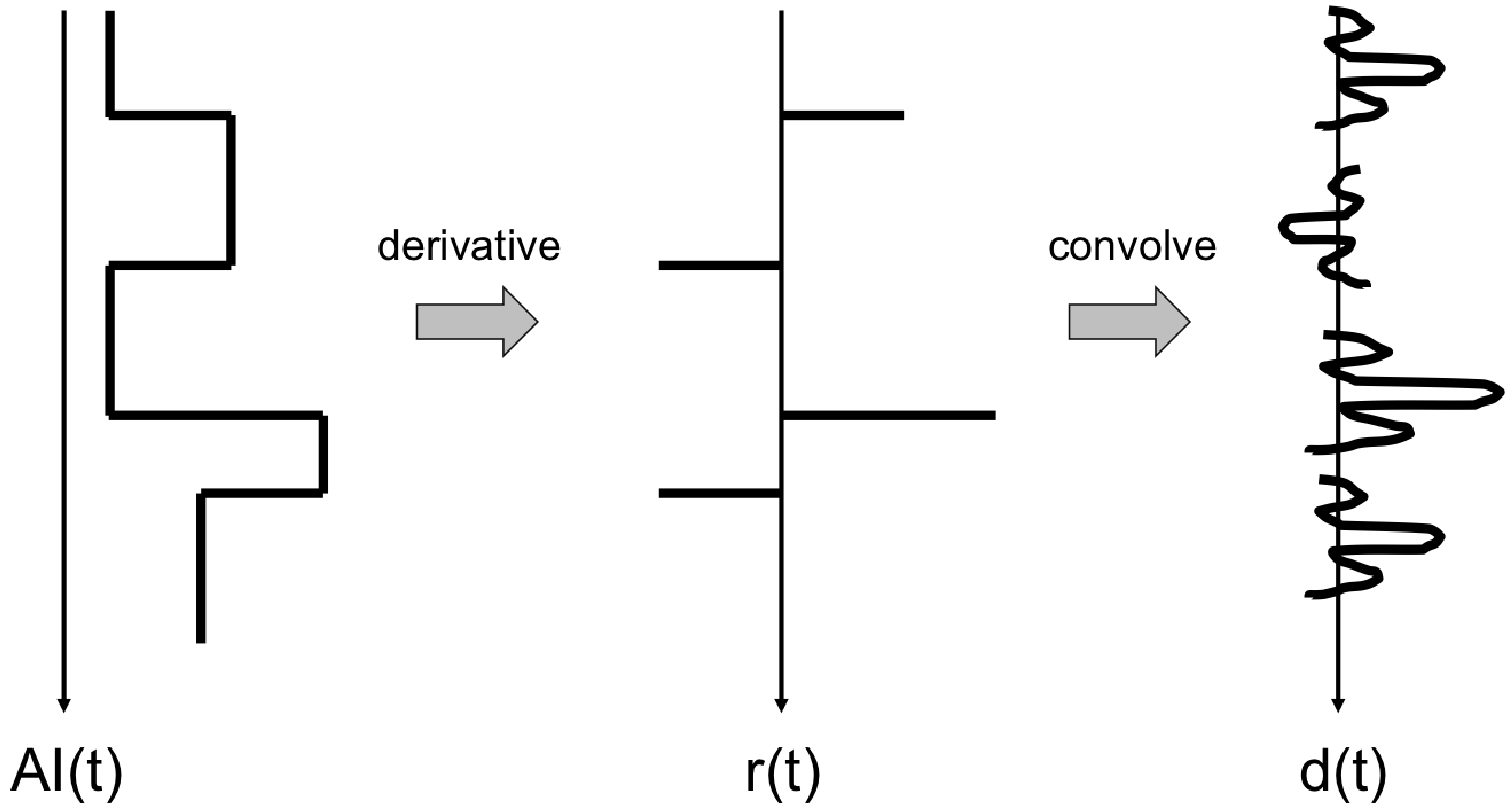
$$\mathbf{G}^{-} = \hat{\mathbf{G}}^{+}\mathbf{R}$$

where:

$$\hat{\mathbf{G}}^{+} = \mathbf{F}^{H}\mathbf{G}^{+}\mathbf{F}$$

Let's practice EX5.

Seismic inversion



Seismic inversion

Integral relation:

$$d(t) = w(t) * \frac{d(ln(AI(t)))}{dt}$$

Discretized relation:

$$\mathbf{d} = \mathbf{WDai}$$

where **D** is a derivative operator and **W** is a convolution operator.

Let's practice EX6.

Some words about implementation...

Some words about implementation…

- Solving large-scale inverse problems can be daunting --> *Divide and conquer* paradigm

Some words about implementation...

- Solving large-scale inverse problems can be daunting --> *Divide and conquer* paradigm
- Focus on fast operators as well as on advanced solvers

Some words about implementation...

- Solving large-scale inverse problems can be daunting --> *Divide and conquer* paradigm
- Focus on fast operators as well as on advanced solvers
- Various paradigms (deterministic, bayesian..) can share same frameworks

Beyond laptop usage - GPUs

Beyond laptop usage - GPUs

- Computational cost of PyLops: forward and adjoint passes (dot products...)

Beyond laptop usage - GPUs

- Computational cost of PyLops: forward and adjoint passes (dot products...)
- Several operators are convolution filters in disguise --> leverage ML libraries

Beyond laptop usage - GPUs

- Computational cost of PyLops: forward and adjoint passes (dot products...)
- Several operators are convolution filters in disguise --> leverage ML libraries
- Seismic inversion example: **d** = **W D m**, **W**: convolution with w, **D**: derivative = convolution with [-1, 1]

Beyond laptop usage - GPUs

- Computational cost of PyLops: forward and adjoint passes (dot products...)
- Several operators are convolution filters in disguise --> leverage ML libraries
- Seismic inversion example: **d** = **W D m**, **W**: convolution with w, **D**: derivative = convolution with [-1, 1]
    - *TensorFlow*, **PyTorch**, Cupy, PyCuda...

Beyond laptop usage - Distributed computing

Beyond laptop usage - Distributed computing

DATA ASSIMILATION summer school

- Data and model can outreach RAM size --> distribute

Beyond laptop usage - Distributed computing

DATA ASSIMILATION
summer school

- Data and model can outreach RAM size --> distribute
- Several operators can be easily parallelized

Beyond laptop usage - Distributed computing

- Data and model can outreach RAM size --> distribute
- Several operators can be easily parallelized
- Solvers can be partially parallelized

Beyond laptop usage - Distributed computing

DATA ASSIMILATION summer school

- Data and model can outreach RAM size --> distribute
- Several operators can be easily parallelized
- Solvers can be partially parallelized
- MDC example: *G+* and *G-* can reach 100++ GB for 3D seismic acquisition, **G+** is a batched matrix-matrix multiplication

Beyond laptop usage - Distributed computing

- Data and model can outreach RAM size --> distribute
- Several operators can be easily parallelized
- Solvers can be partially parallelized
- MDC example: *G+* and *G-* can reach 100++ GB for 3D seismic acquisition, **G+** is a batched matrix-matrix multiplication
    - Joblib, mpi4py, **Dask**...