Large-scale workflows for wave-equation based inversion in Julia

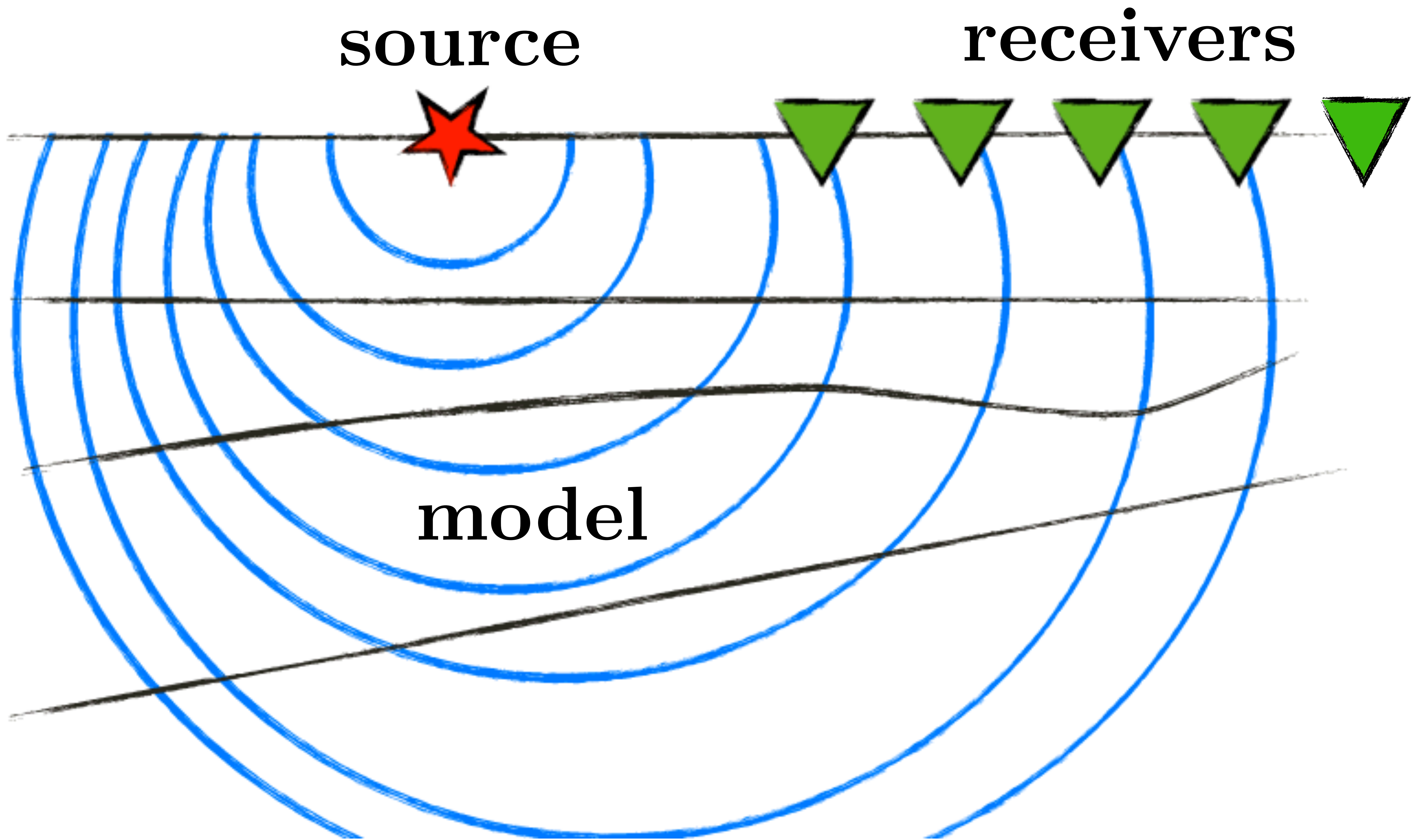
Philipp A. Witte, Mathias Louboutin and Felix J. Herrmann

SLIM
University of British Columbia

Motivation

Use Geophysics to understand the earth

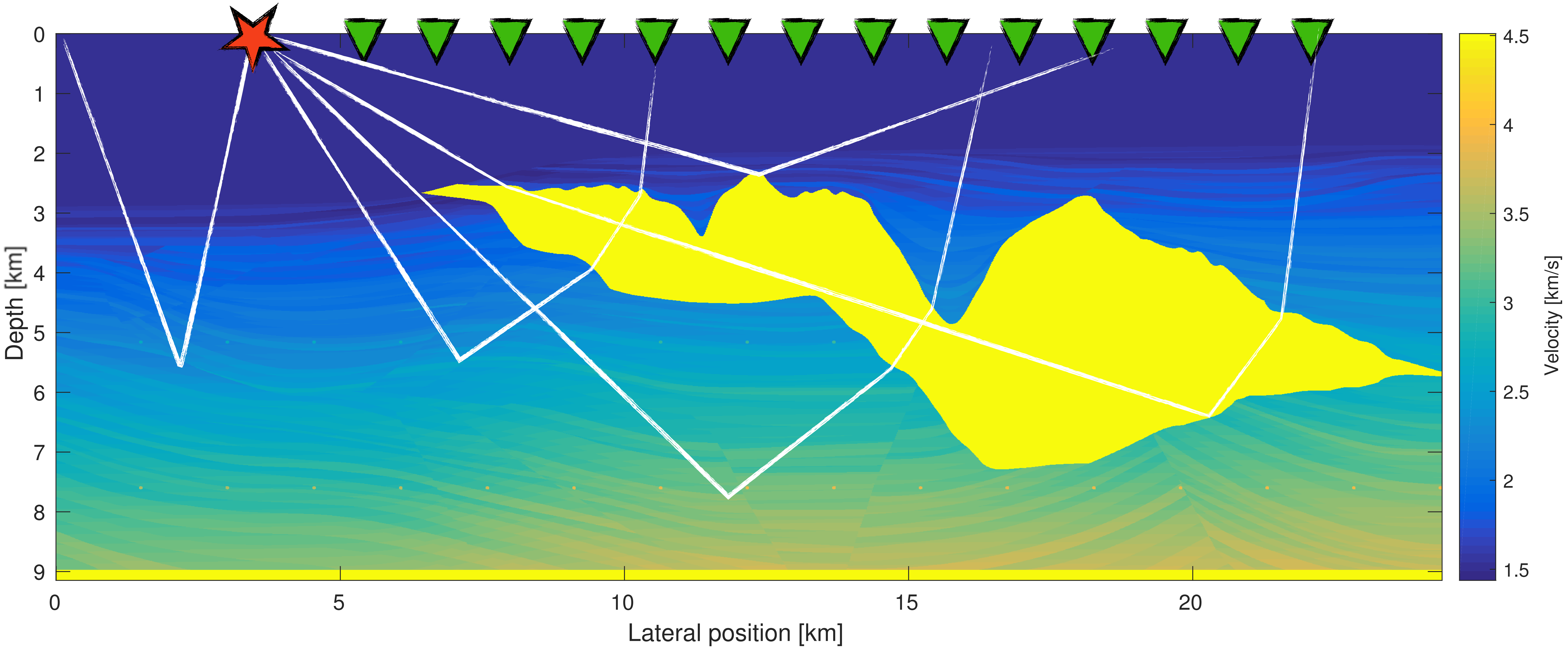
- obtain information about the earth from surface seismic experiments



Motivation

Use Geophysics to understand the earth

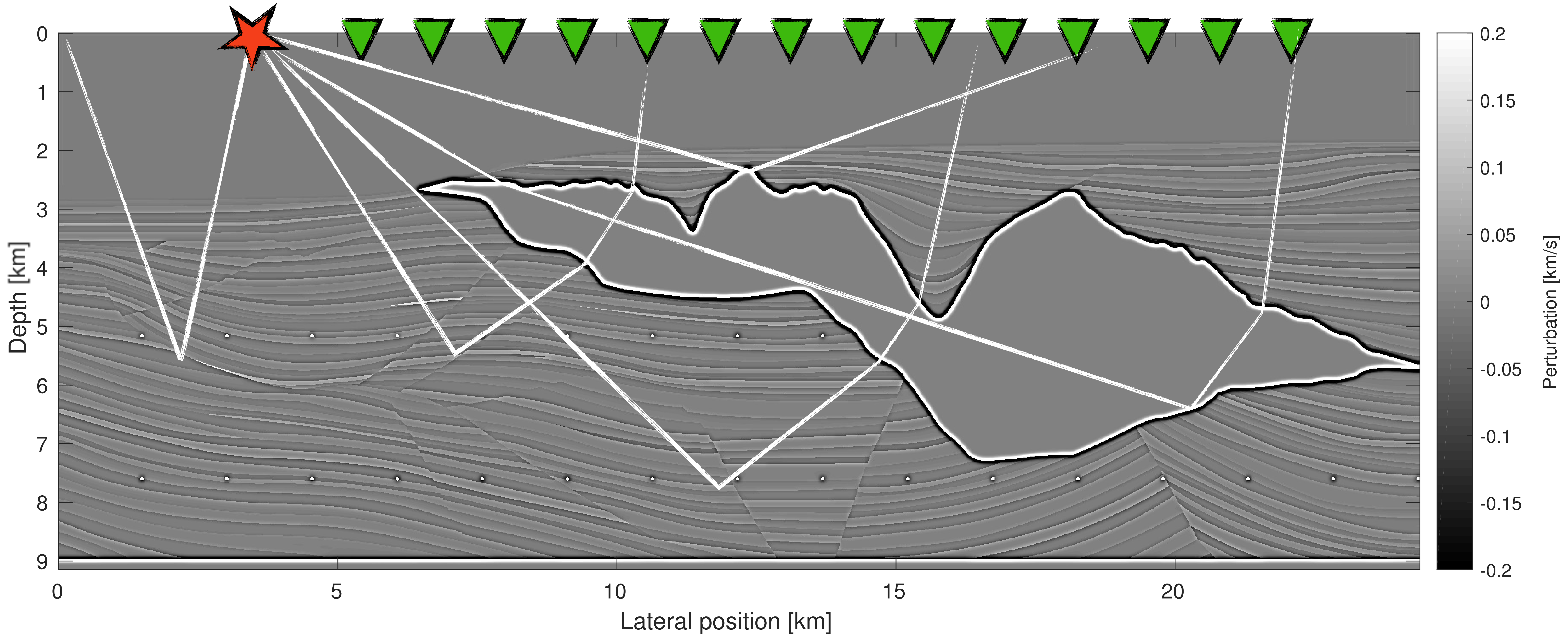
- invert for subsurface parameters, e.g. velocity, density, porosity



Motivation

Use Geophysics to understand the earth

- image geological interfaces (perturbations of earth parameters)



Motivation

Formulate inverse problems and use numerical optimization

- invert for velocity ➡ nonlinear least squares optimization problem

$$\underset{\mathbf{m}}{\text{minimize}} \quad \frac{1}{2}||\, \mathbf{A}(\mathbf{m})^{-1} \cdot \mathbf{q} - \mathbf{d}||_2^2$$

(Virieux and Operto, 2009)

- image the subsurface ➡ linear least squares optimization problem

$$\underset{\delta\mathbf{m}}{\text{minimize}} \quad \frac{1}{2}||\, \mathbf{J} \cdot \delta\mathbf{m} - \delta\mathbf{d}||_2^2$$

(Dong et al., 2012)

Need:

- access to objective function values and gradients
- matrices $\mathbf{A}(\mathbf{m})$, $\mathbf{J}$, or actions of these matrices on vectors

SLIM

Motivation

Challenges:

- problem sizes are huge:
  - seismic surveys consist of tens of thousands of individual experiments
  - model wave propagation over thousands of time steps in large domains
  - typical size of modeling matrix: $\mathbf{A}(\mathbf{m}) \in \mathbb{R}^{n \times n}, n = 1e16$
- model physical system ➡ data is irregular, has coordinates and meta data
- inverse problems are difficult to solve (ill-posed, non-convex, non-unique)
- software to solve seismic inverse problems:
  - needs to be fast and handle large amounts of data
  - often tailored towards a specific application
  - difficult to maintain and modify, often slow adaptation of new concepts
  - iterative imaging algorithms rely on using all the data in each iteration

Motivation

Our goal:

- flexible framework for solving seismic inverse problems
- abstract matrices and vectors to easily formulate algorithms
- data containers for irregular seismic data
- robust parallel framework that scales on HPC environments/the cloud
- interface Devito (finite difference DSL) to solve PDEs
- framework that can handle actual 3D industry-sized problems
    - lower computational costs using stochastic optimization

The forward problem

Solve (acoustic) wave equation for a given model

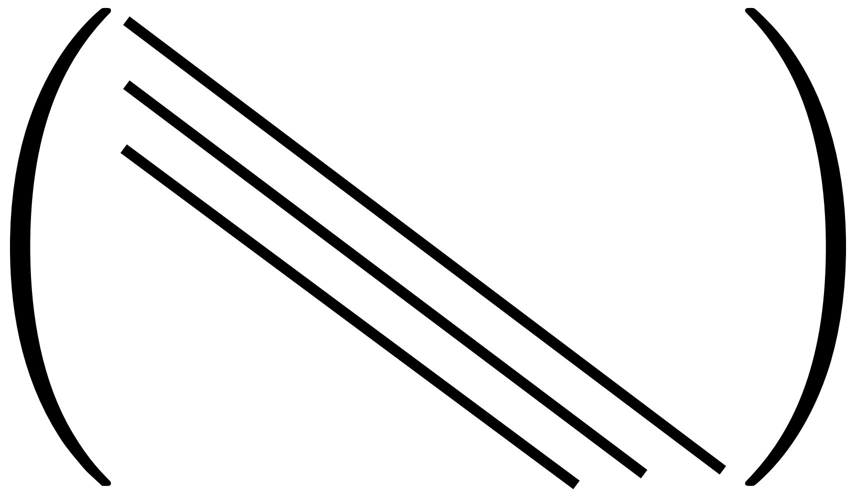
- continuous form

$$m \frac{\partial^2}{\partial t^2} u - \nabla^2 u = s$$

m: model parameters (slowness squared)
u : wave fields
s : source wave fields

- discretize and rewrite as linear system

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{s}$$



$\mathbf{A}$ is lower triangular, solve with forward elimination

The forward problem

Model surface recorded seismic data

$$\mathbf{d} = \mathcal{P}_r \cdot \mathbf{F} \cdot \mathcal{P}_s^\top \cdot \mathbf{q}$$

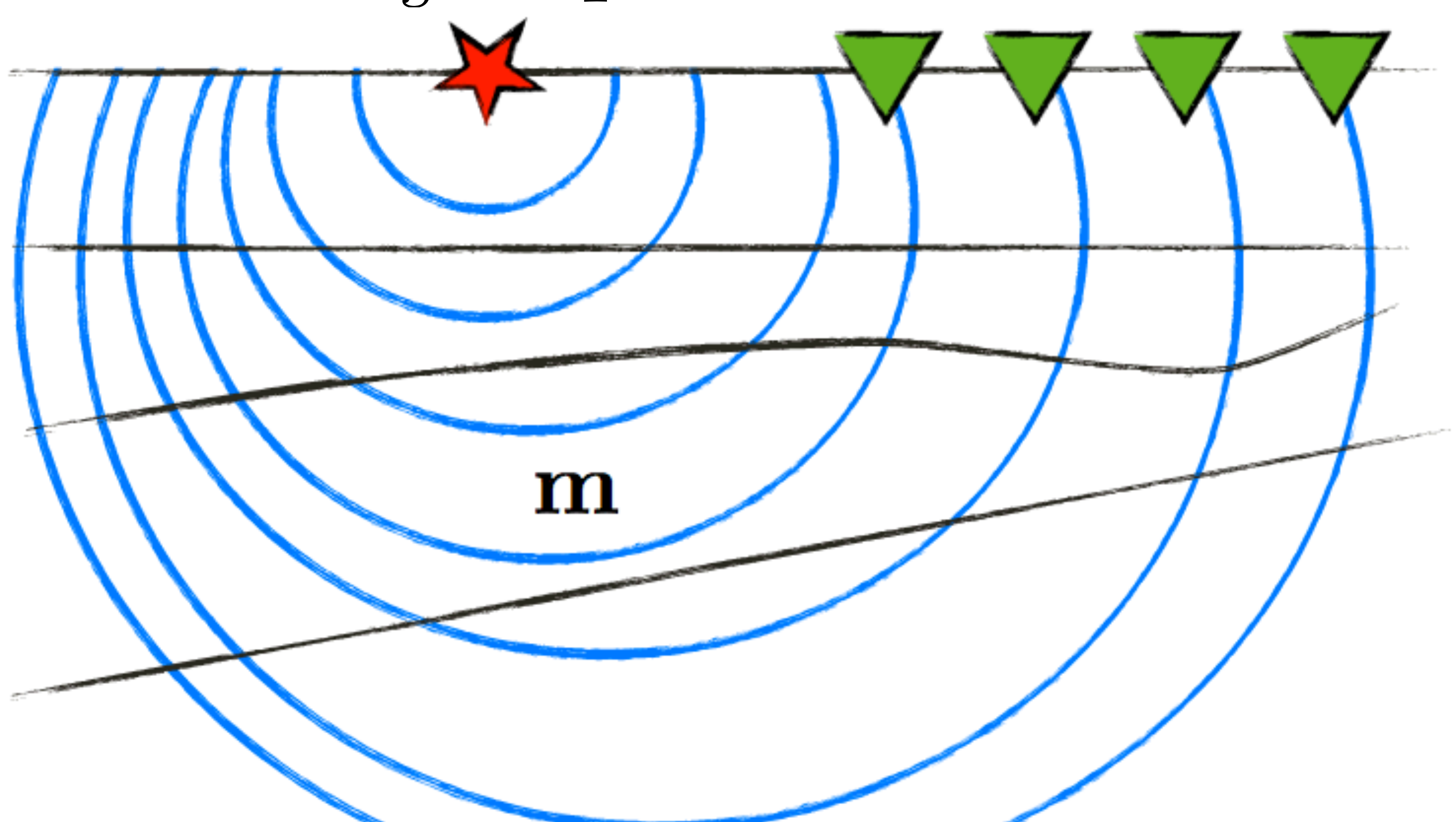
$\mathbf{F} := \mathbf{A}(\mathbf{m})^{-1}$

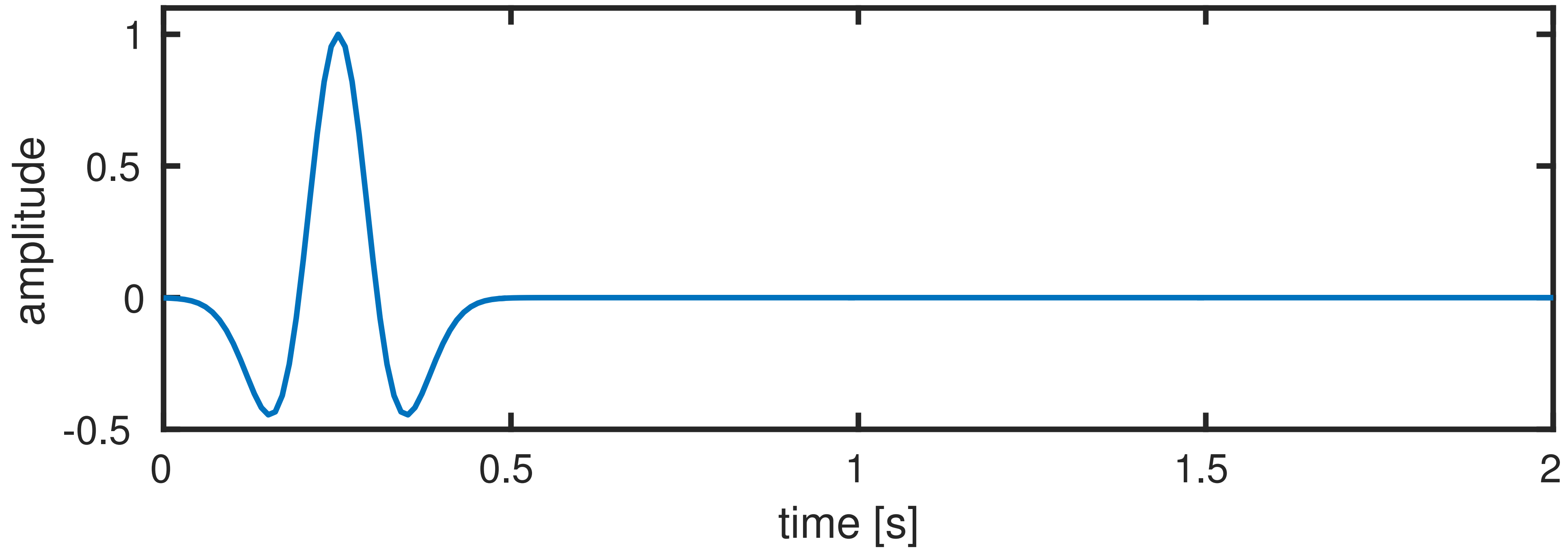
$\mathcal{P}_r$: receiver restriction

$\mathcal{P}_s^\top$: source injection

$\mathbf{q}$: source wavelet

$\mathcal{P}_s^\top \cdot \mathbf{q}$ $\mathcal{P}_r \cdot \mathbf{u}$





The forward problem

Model surface recorded seismic data (2 experiments)

$$\mathbf{d} = \mathcal{P}_r \cdot \mathbf{F} \cdot \mathcal{P}_s^\top \cdot \mathbf{q}$$

$$\begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} = \begin{bmatrix} \mathcal{P}_{r_1} & \\ & \mathcal{P}_{r_2} \end{bmatrix} * \begin{bmatrix} \mathbf{F}_1 & \\ & \mathbf{F}_2 \end{bmatrix} * \begin{bmatrix} \mathcal{P}_{s_1}^\top & \\ & \mathcal{P}_{s_2}^\top \end{bmatrix} * \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix}$$

- can barely store $\mathbf{d}$ $(1e11 \times 1)$
- cannot store $\mathcal{P}_s^\top \cdot \mathbf{q}$ $(1e16 \times 1)$
- cannot form $\mathbf{F}, \mathcal{P}_r, \mathcal{P}_s$ explicitly $(1e16 \times 1e16)$

The forward problem

Adapt idea of matrix-free linear operators

- SPOT - a linear operator toolbox in Matlab (van den Berg and Friedlander, 2012)
- operators look and behave like explicit matrices
- matrix "knows" how to apply itself to vectors
- forward, adjoint matrix-matrix, matrix-vector products etc.

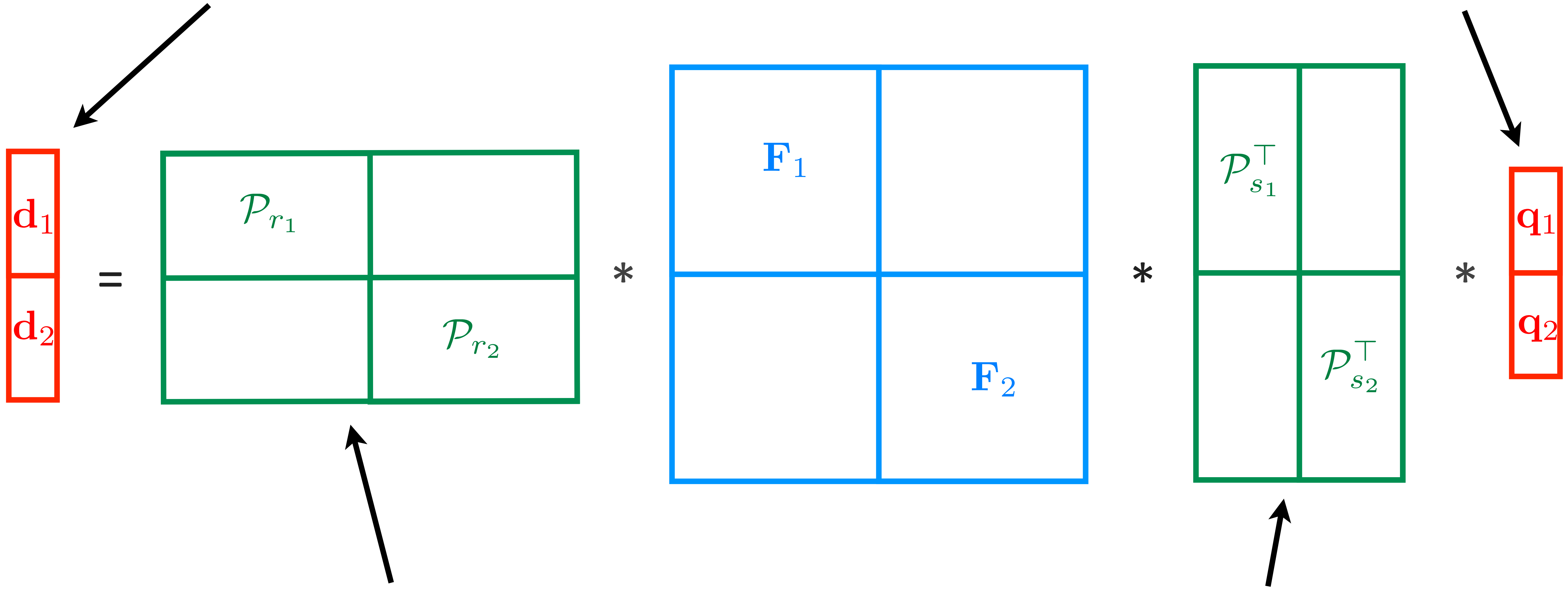
```
>> n = 1e3;
>> % Set up operator
>> F = opDFT(n)
F =
  Spot operator: DFT(1000,1000)
    rows:    1000    complex: yes
    cols:    1000    type:    DFT
>> % FFT
>> y = F*x;
>> % iFFT
>> z = F'*y;
```

The forward problem

Implement abstract linear operators and vectors in Julia

- JOLI - Julia Operator Library

```
type joData <: JoliVector
    m::Integer
    geometry::Geometry  # coordinates, sampleIntervals, numSamples
    data::Array{Any,1}  # Cell array w/ data in original dimensions
```




```
type joProjection <: JoliOperator
    m::Integer
    n::Integer
    info::Info            # numTimesteps, numGridpoints, numSources
    geometry::Geometry
```

The forward problem

Solve forward wave equation:

```
julia> d = Pr*F*Ps'*q
        From worker 5:  Nonlinear forward modeling (source no. 2)
        From worker 4:  Nonlinear forward modeling (source no. 3)
        From worker 3:  Nonlinear forward modeling (source no. 4)
        From worker 2:  Nonlinear forward modeling (source no. 1)
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/3443b8b1660866d91815a00df85b3a6cceed7ed9.c [0.20 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/83f4ea5a3ee3ebf1c51e1243816b457c431b50c7.c [0.22 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/415a7b8d0cf2663d3c552655b56bf924e343a30f.c [0.21 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/8ac99a871f7a68ef619460c7a5cdcfb1f2e58dd3.c [0.20 s]
(TimeModeling.joData{Float32},"Julia seismic data container",161555,1)
```

JIT compilation

Solve adjoint wave equation:

```
julia> q̂ = Ps*F'*Pr'*d
        From worker 4:  Nonlinear adjoint modeling (source no. 4)
        From worker 3:  Nonlinear adjoint modeling (source no. 3)
        From worker 2:  Nonlinear adjoint modeling (source no. 2)
        From worker 5:  Nonlinear adjoint modeling (source no. 1)
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/f43d668c3bca252a307acf2dbe0c51417a8a6f01.c [0.20 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/a0dbc0f7d088b2073dadeb15929a7347e244d417.c [0.21 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/87d901287ff2ce08855dde4c08fa9e825b5b1562.c [0.20 s]
CustomCompiler: compiled /tmp/pwittCT5MeP/devito-4080/3d1cb82e3ac510f999869dee3c5af806adedd5ab.c [0.21 s]
(TimeModeling.joData{Float32},"Julia seismic data container",1971,1)
```

Software design

What happens after executing the modeling command?

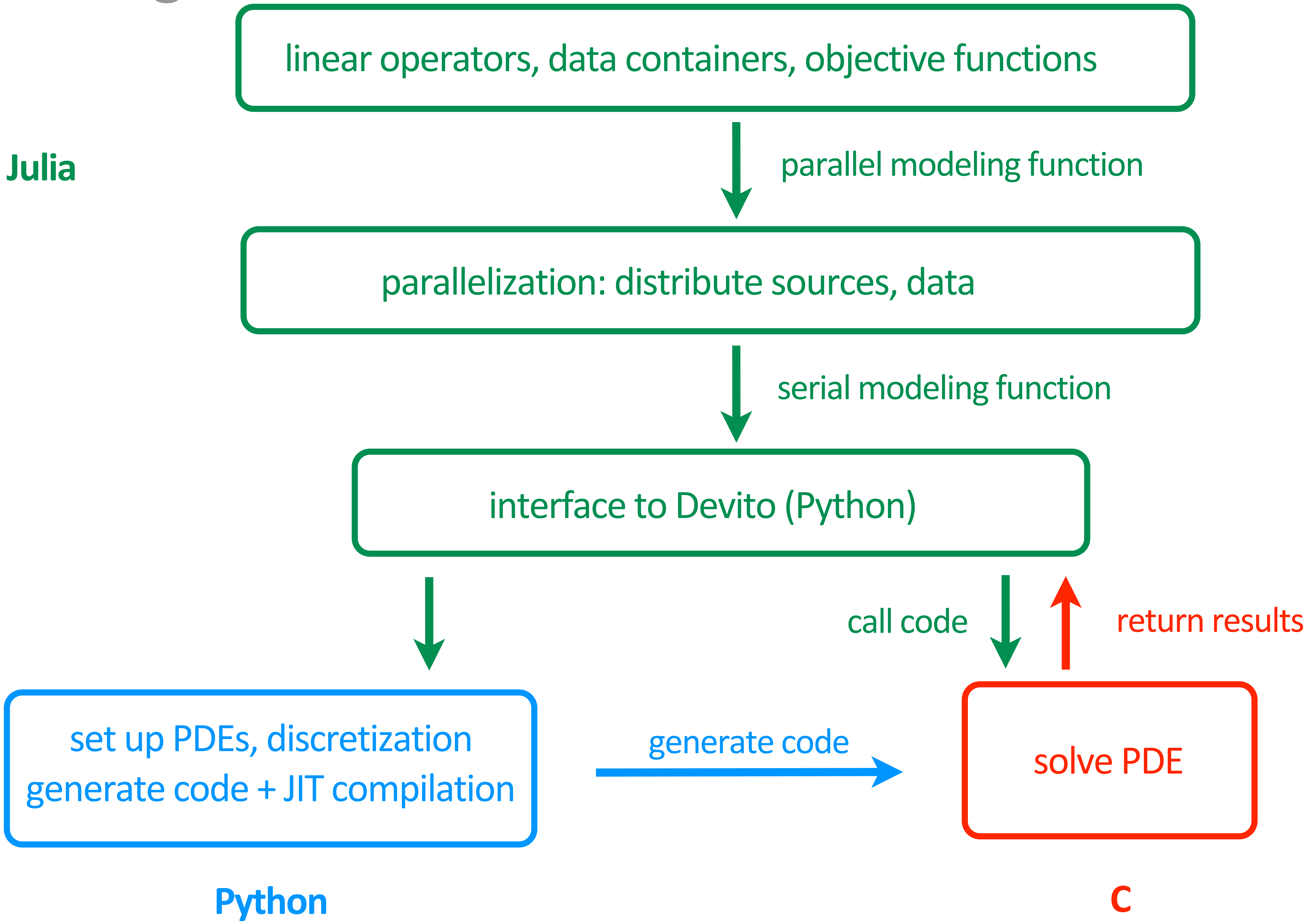
```
julia> d = Pr*F*Ps'*q
```

- check that dimensions are correct
- check that geometries match

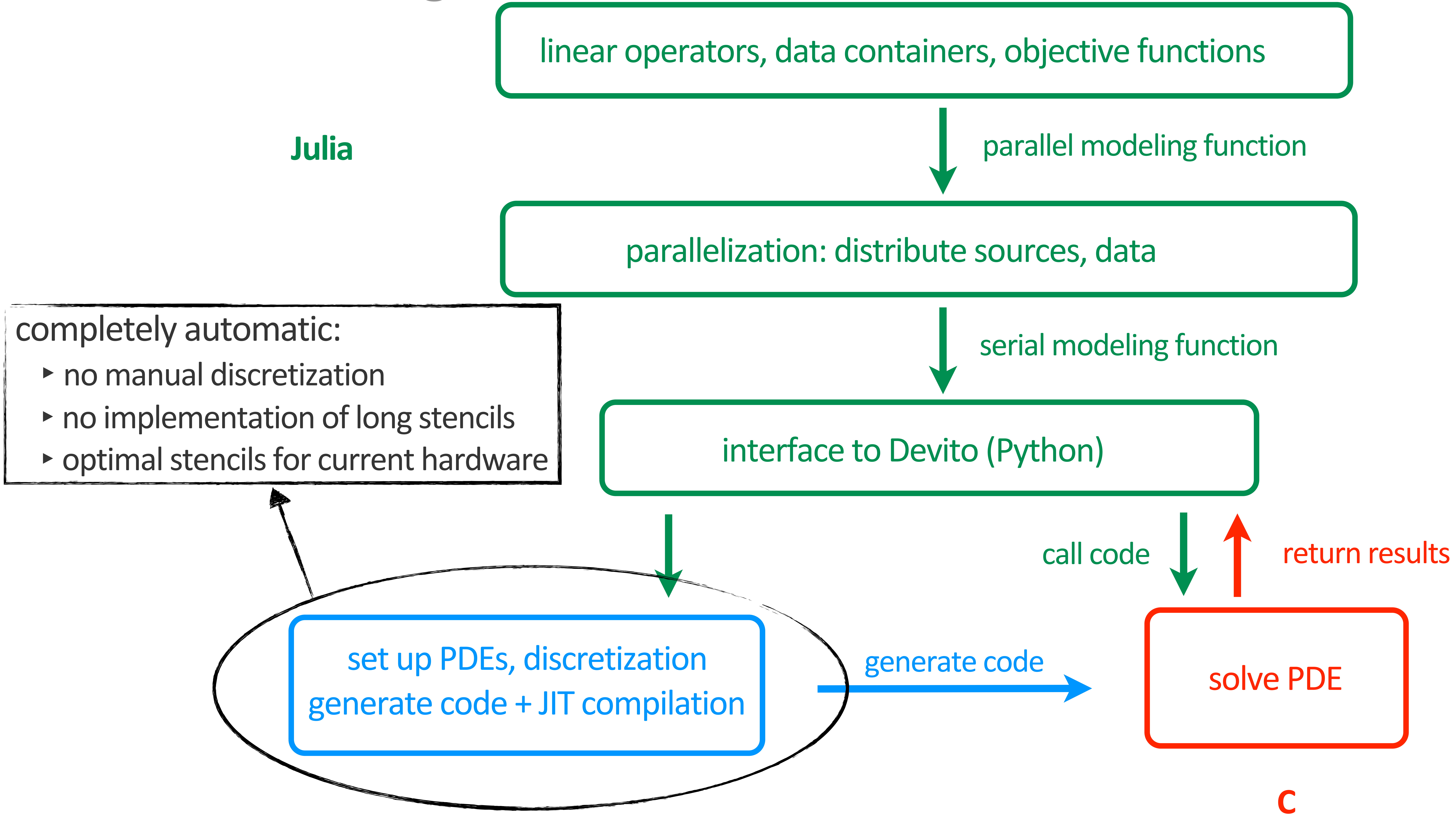
```
julia> time_modeling(model,srcGeometry,srcData,recGeometry,recData,'F',1)
```

- call serial/parallel modeling function
- set function arguments with parameters from linear operators

Software design



Software design



The inverse problem

Inverse problem

- invert model for given data
- how does a perturbation in the model relate to a perturbation in the wave field?

$$\mathbf{u} = \mathbf{A}(\mathbf{m})^{-1} \cdot \mathbf{q}$$

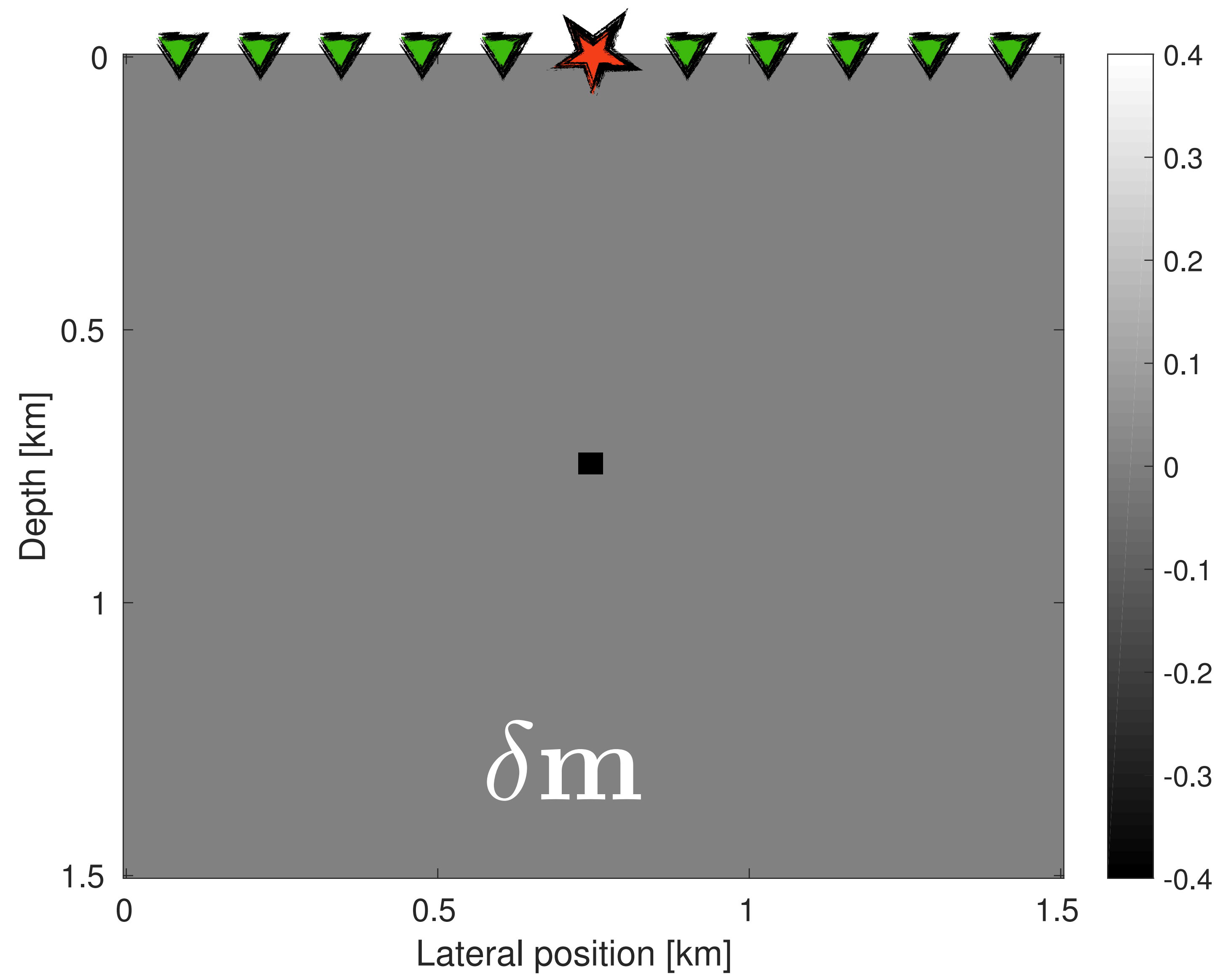
$$\frac{\partial \mathbf{u}}{\partial \mathbf{m}} = \frac{\partial}{\partial \mathbf{m}} \left(\mathbf{A}(\mathbf{m})^{-1} \cdot \mathbf{q} \right) := \mathbf{J} \qquad \text{(Jacobian)}$$

(Virieux and Operto, 2009)

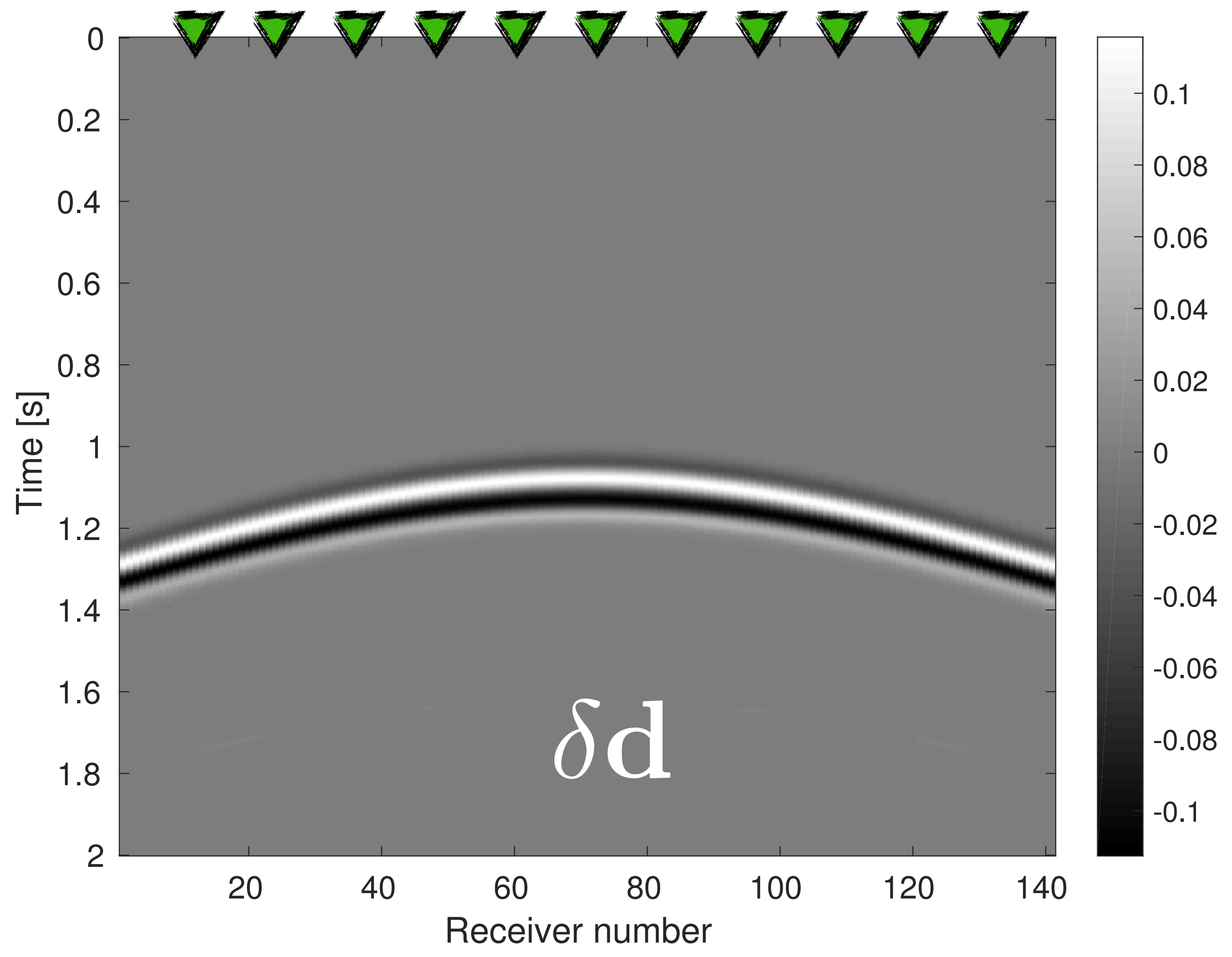
- including source/receiver projections:

$$\mathbf{J} = -\mathcal{P}_r \cdot \mathbf{A}(\mathbf{m})^{-1} \cdot \frac{\partial \mathbf{A}(\mathbf{m})}{\partial \mathbf{m}} \cdot \mathbf{A}(\mathbf{m})^{-1} \cdot \mathcal{P}_s^{\top} \cdot \mathbf{q}$$

The inverse problem



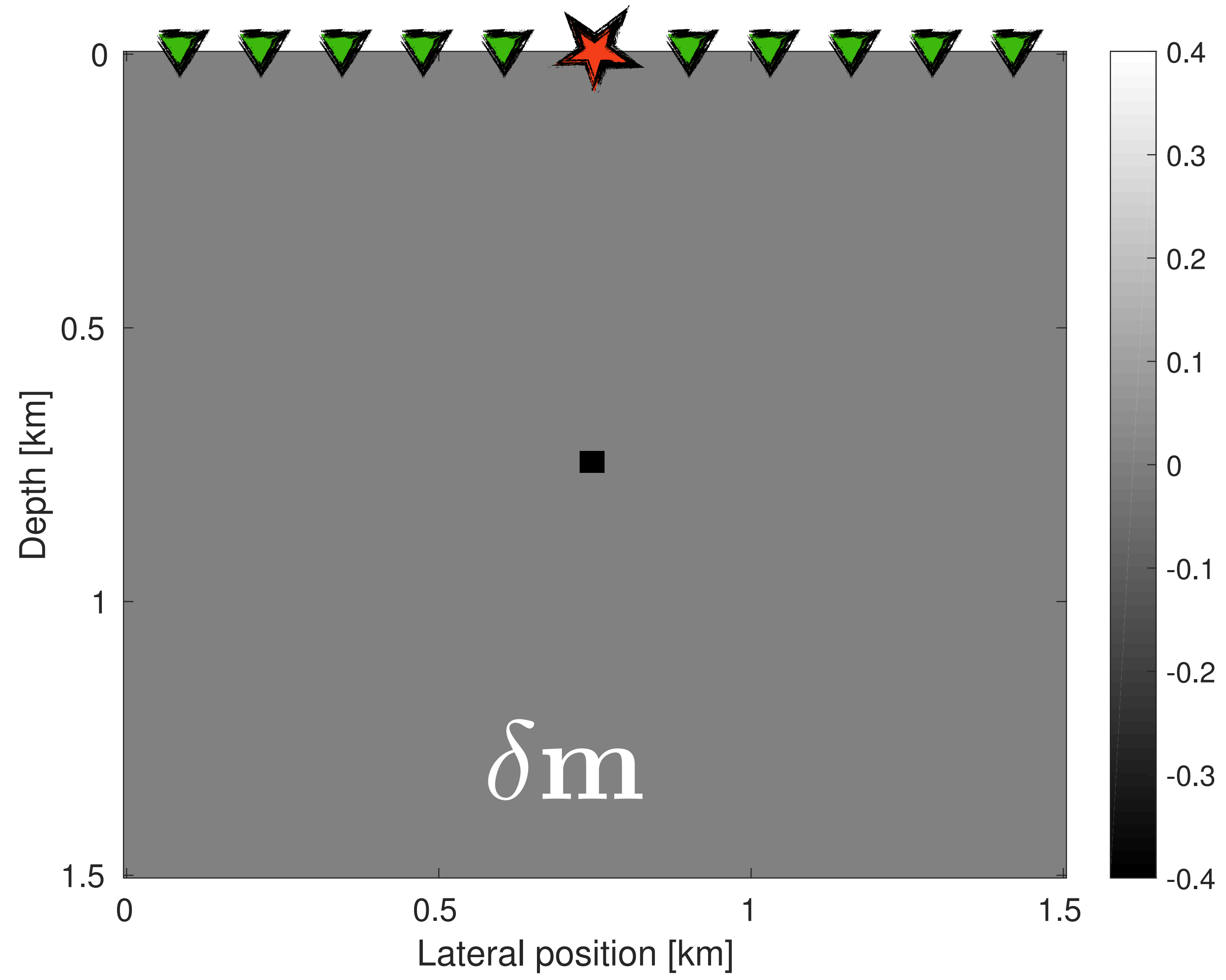
$$\mathbf{J} \cdot \delta \mathbf{m}$$



Perturbation w.r.t background model

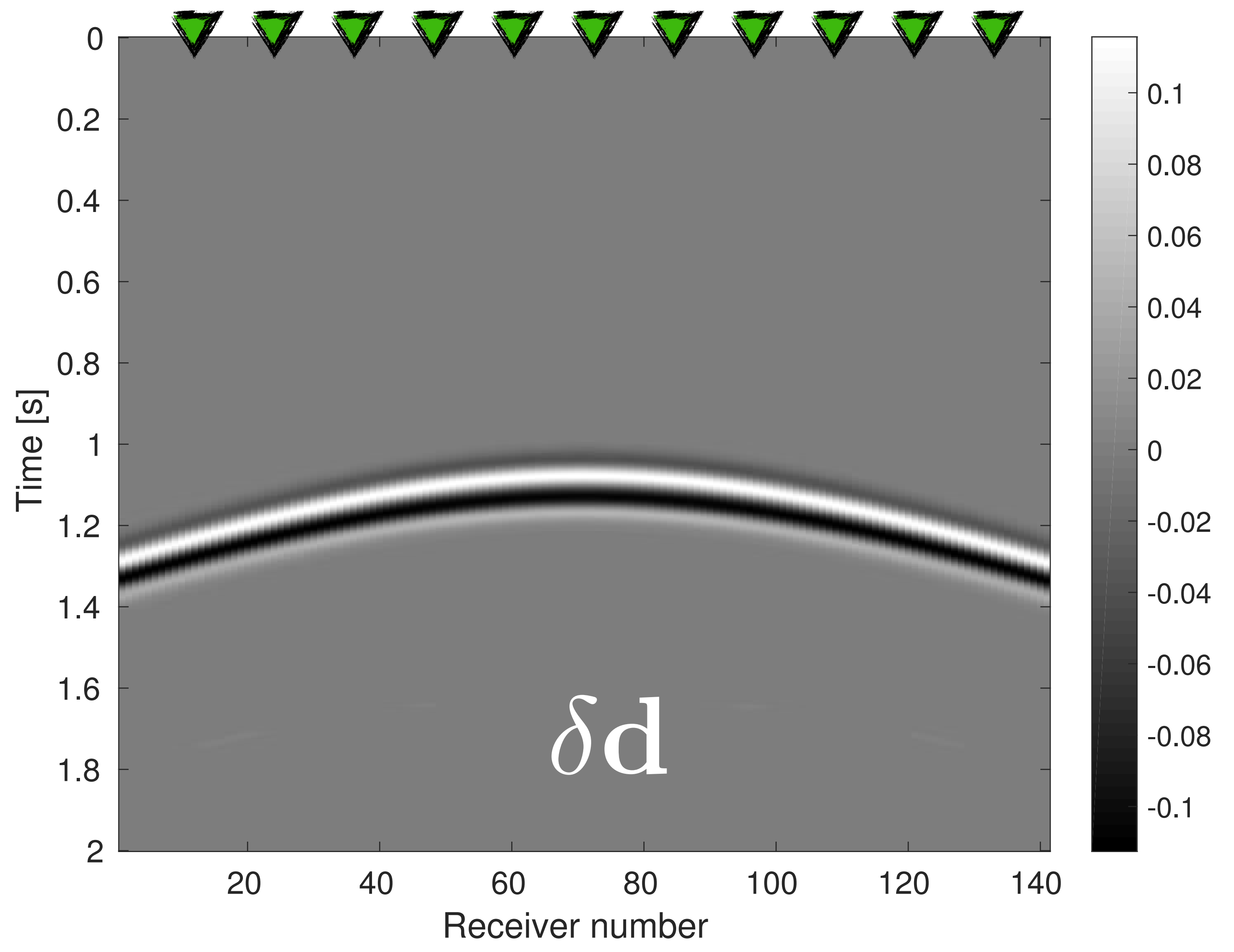
Surface recorded seismic data

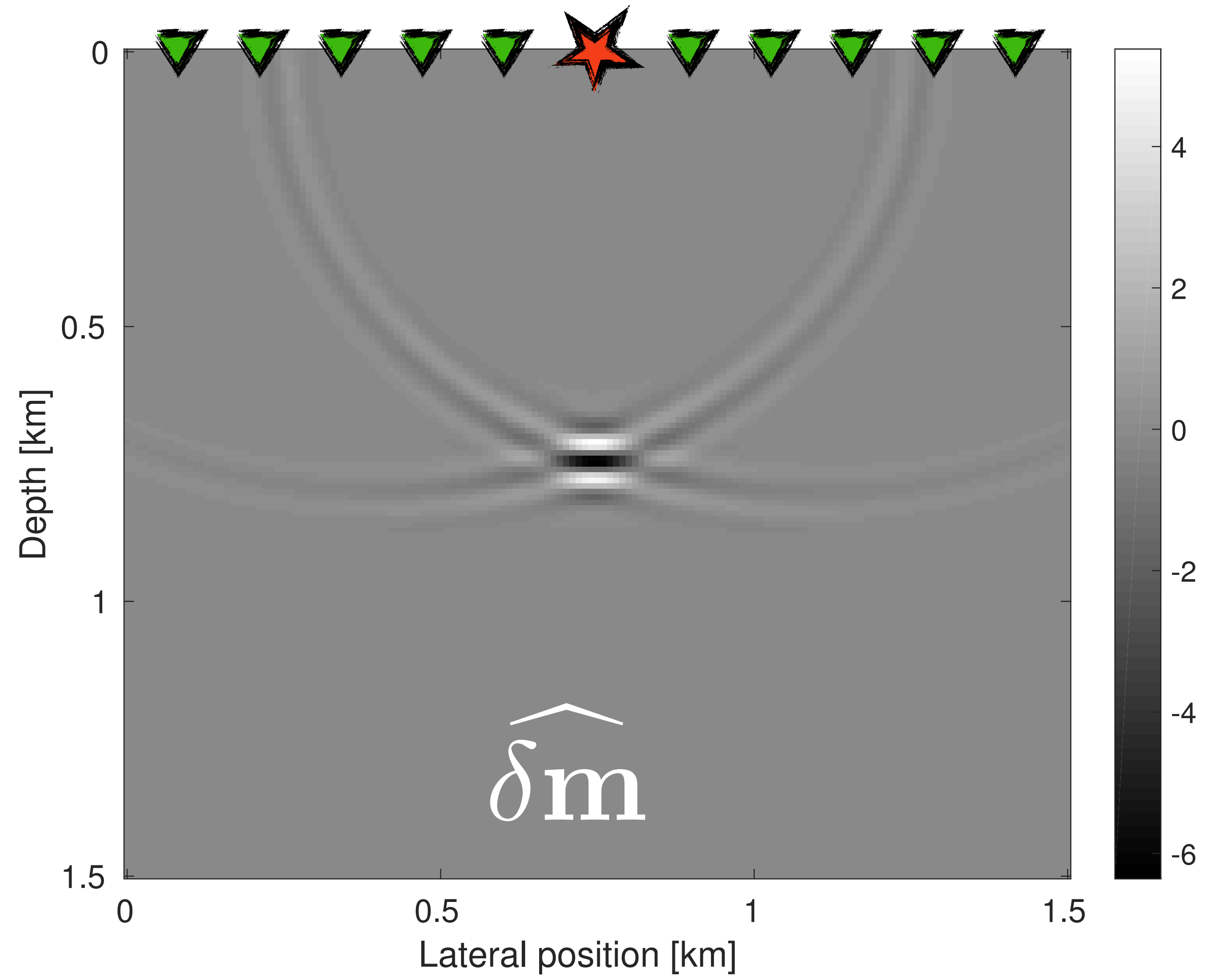
The inverse problem



$\mathbf{J} \cdot \delta \mathbf{m}$

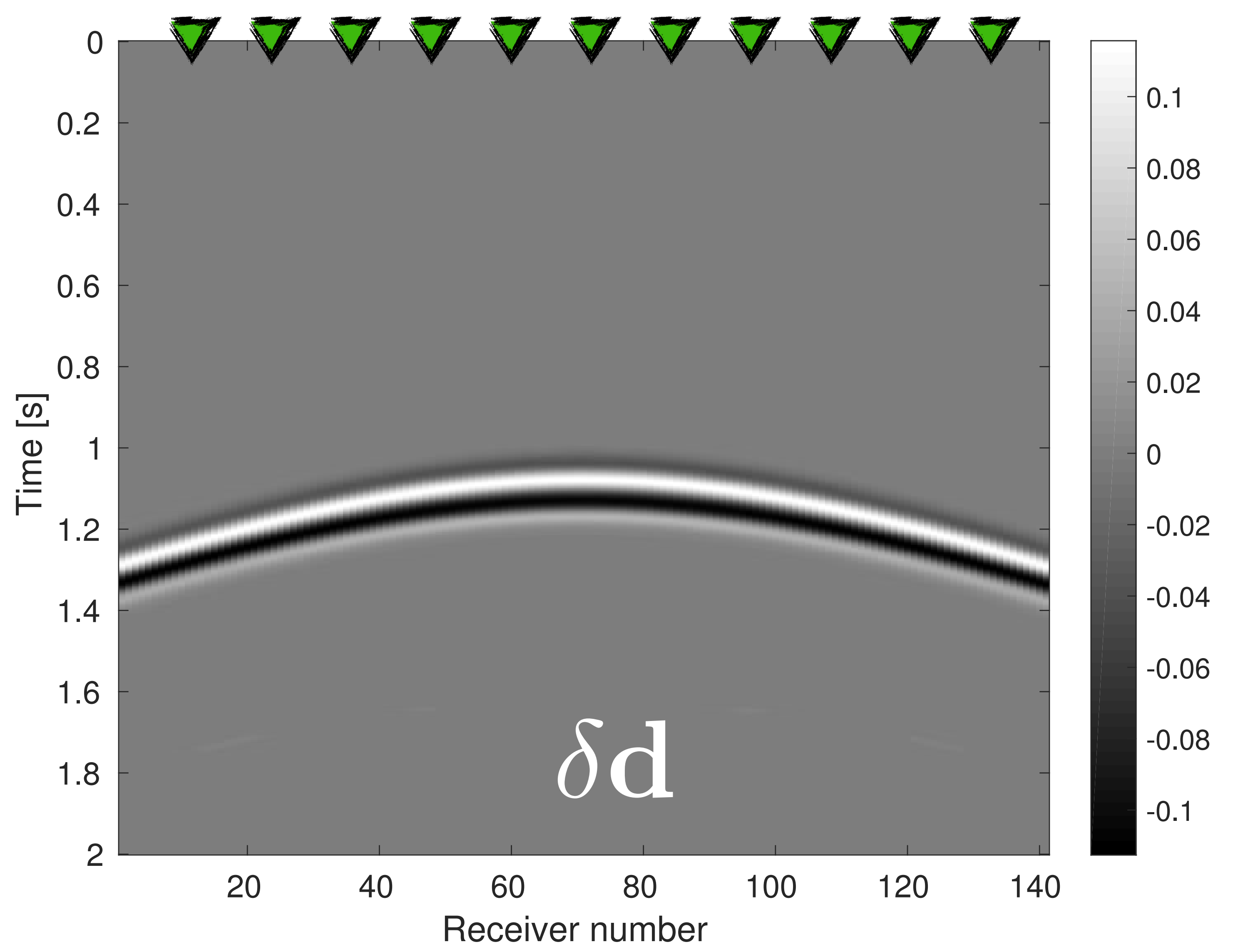
"modeling"





$\mathbf{J}^{\top} \cdot \delta \mathbf{d}$

"imaging"



Linear inverse problems: seismic imaging

Seismic imaging as a linear least squares problem (Dong et al., 2012)

- objective function:

$$\underset{\delta \mathbf{m}}{\text{minimize}} \quad \frac{1}{2}|| \mathbf{J} \cdot \delta \mathbf{m} - \delta \mathbf{d}||_2^2 \qquad (\mathbf{J} \in \mathbb{R}^{m \times n}, m > 1e11, n > 1e9)$$

- easy to implement a solver using JOLI operators:

```
# initialize solution
dm = zeros(n)

for j=1:k
    # residual and gradient
    r = J*dm - d
    g = J'*r

    # step size
    t = norm(r)^2/norm(g)^2

    # update x
    dm -= t*g
end
```

Seismic least squares imaging

Seismic imaging as a linear least squares problem

- $\mathbf{J}$ is ill-conditioned and very large, can only afford 10s of iterations, need to save $\ddot{\mathbf{u}}[\mathbf{t}]$ (~TB)
- reduce computational cost using techniques from stochastic optimization
- linearized Bregman method (Yin, 2010)

$$\underset{\delta\mathbf{m}}{\text{minimize}} \quad \lambda||\mathbf{x}||_1 + \frac{1}{2}||\mathbf{x}||_2^2$$
$$\text{subject to: } ||\,\mathbf{A}\cdot\mathbf{x} - \mathbf{b}||_2 \leq \sigma$$

λ : thresholding parameter

σ : noise level

- designed for compressive sensing problems with $\mathbf{A} \in \mathbb{R}^{m\times n}, m << n$
- related to sparse (block-) Kaczmarz solver for problems w/ any $\mathbf{A} \in \mathbb{R}^{m\times n}$ (Lorenz et al. , 2014)

Seismic least squares imaging

Linearized Bregman method for least squares imaging:

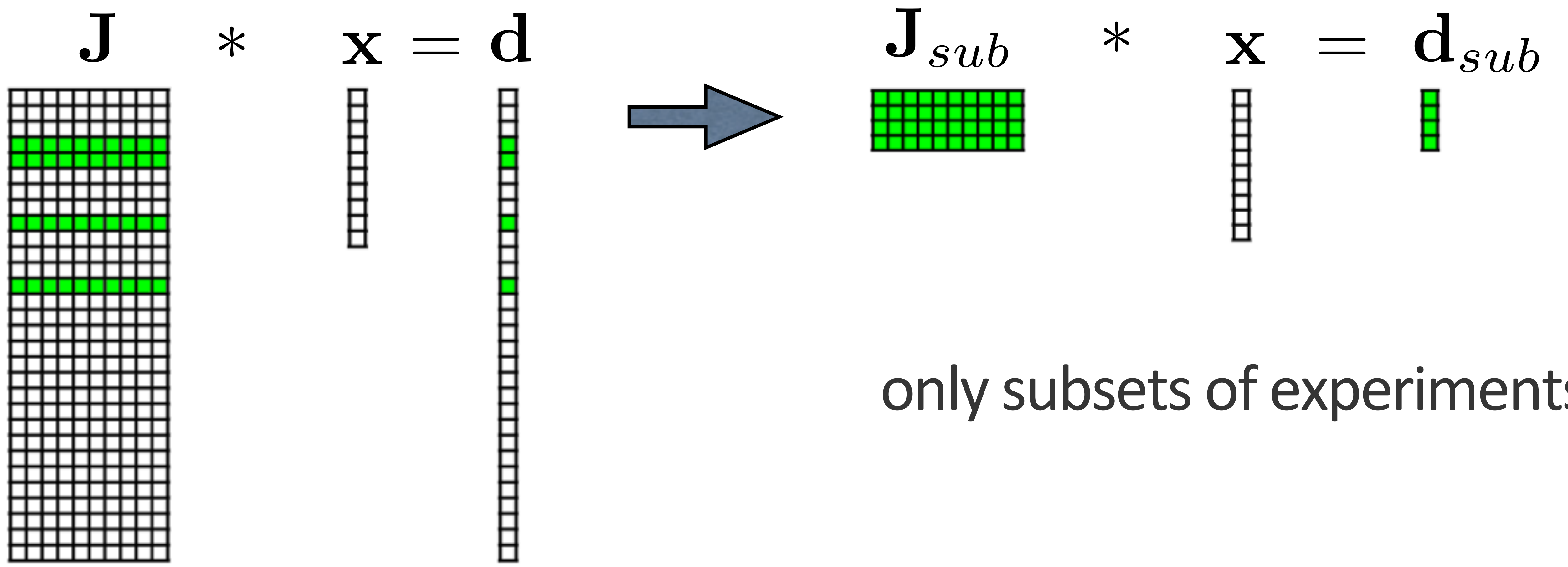
$$\underset{\delta \mathbf{m}}{\text{minimize}} \quad \lambda ||\mathbf{C} \cdot \delta \mathbf{m}||_1 + \frac{1}{2} ||\mathbf{C} \cdot \delta \mathbf{m}||_2^2$$
$$\text{subject to: } ||\ \mathbf{J} \cdot \delta \mathbf{m} - \delta \mathbf{d}||_2 \leq \sigma$$

$\mathbf{C}$: Curvelet transform

λ : thresholding parameter

σ : noise level

- in each iteration possible to work w/ subset of rows of $\mathbf{J}, \delta \mathbf{d}$



only subsets of experiments in each iteration

Seismic least squares imaging

Linearized Bregman method for least squares imaging:

- Algorithm (Lorenz et al. , 2014)

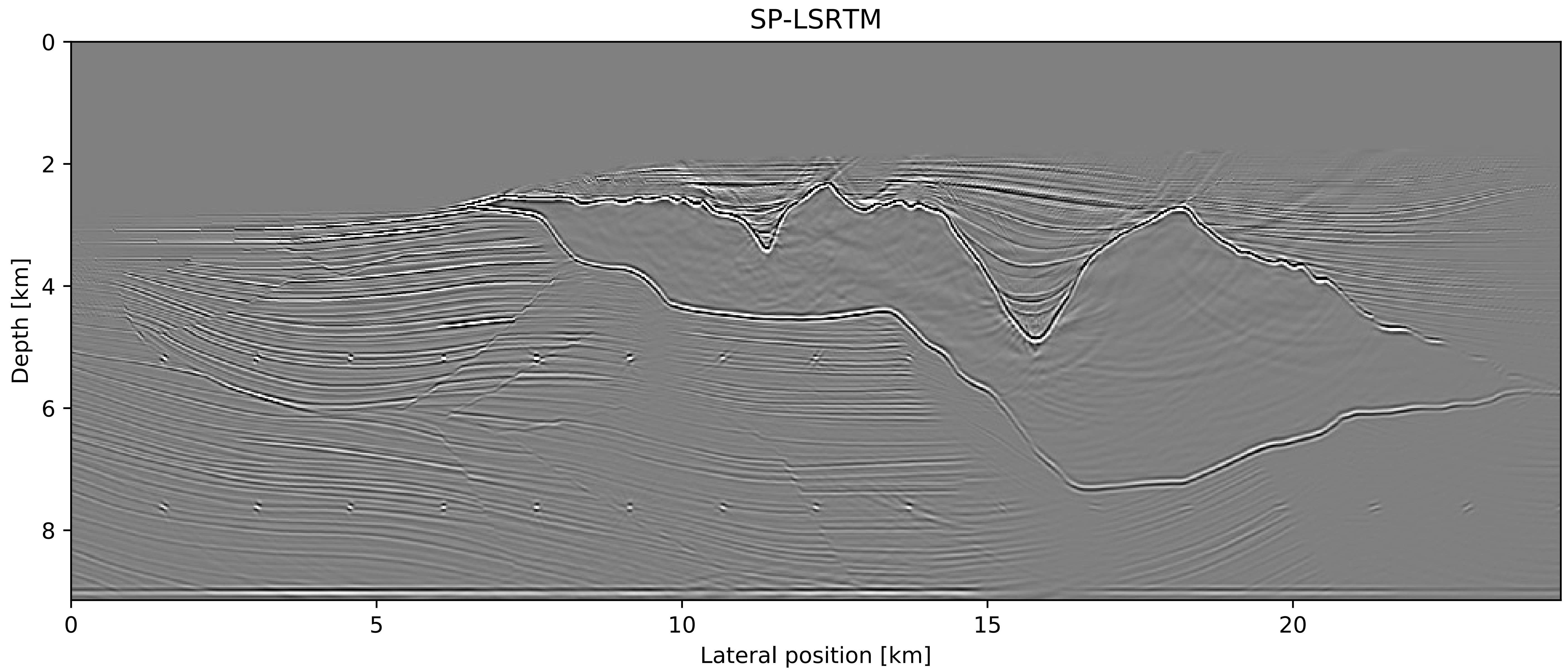
$$
\begin{array}{ll}
1. & \textbf{for } k = 0, 1, \cdots \\
2. & \quad \mathbf{z}_{k+1} = \mathbf{z}_k - t_k \mathbf{J}_{r(k)}^{\top} (\mathbf{J}_{r(k)} \mathbf{x}_k - \delta \mathbf{d}_{r(k)}) \cdot \max(0, 1 - \frac{\sigma}{||\hat{\mathbf{J}}_{r(k)} \mathbf{x}_k - \mathbf{b}_{r(k)}||_2}) \\
3. & \quad \mathbf{x}_{k+1} = \mathbf{C}^* S_\lambda (\mathbf{C} \mathbf{z}_{k+1}) \\
4. & \textbf{end for}
\end{array}
$$

- $r(k)$ is sequence of subsampled experiments (cyclic or random)
- instead of “touching” full data set in each iteration, only touch every data sample one or twice
- calculate gradients on a few large nodes (TB RAM) rather than many small ones

Seismic least squares imaging

Example with model from the introduction:

- large 2D model ($4e6$ grid points)
- 1000 surface seismic experiments ($2.5e9$ data points)
- 20 iterations w/ 100 experiments per iteration (4000 PDE solves)



Nonlinear inverse problems

Objective functions for nonlinear inverse problems

- functions that spin off function values and gradients
- can be passed to black-box optimization libraries (e.g. NLopt, JuMP)

Example: invert for velocity model (full waveform inversion)

- nonlinear least squares problem

$$\underset{\mathbf{m}}{\text{minimize}} \quad \frac{1}{2}|| \mathcal{P}_r \cdot \mathbf{A}(\mathbf{m})^{-1} \cdot \mathcal{P}_s^\top \cdot \mathbf{q} - \mathbf{d}||_2^2$$

- gradient given by

$$\mathbf{g} = \mathbf{J}^\top \cdot \left(\mathcal{P}_r \cdot \mathbf{A}(\mathbf{m})^{-1} \cdot \mathcal{P}_s^\top \cdot \mathbf{q} - \mathbf{d}\right) \longrightarrow$$

```
# fwi function value and gradient
f,g = fwi_objective(m,dobs,q)
```

Nonlinear inverse problems

Some final words about scaling:

- so far only large 2D example ($4e6$ model parameters, $2.5e9$ data points)
- large 3D problems have $2e8$ model parameters, $1e12$ data points (or more)

For large-scale 3D inverse problems:

- bounded by RAM, need to store the forward wavefields for all experiments
- with stochastic optimization methods (linearized Bregman etc.), reduce number of experiments per iteration
- 3D becomes feasible, if working with few, but “large” nodes (RAM > 1 TB)

Conclusions

Julia framework for seismic modeling and inversion

- modular software structure
- matrix-free linear operators and seismic data containers
- efficient and fast PDE solves through Devito
- scales to very large and realistic problem sizes
- parallel framework, resilience to hardware failures
- easy to formulate algorithms, objective functions + gradients, etc.
- easy to interface optimization libraries

Outlook

In the future we plan to:

- test our framework in the cloud
- add IO functions for seismic data (automatic set up of data containers + geometry)
- application to 3D seismic field data sets (requires nodes with ~ TB RAM)

Acknowledgements

This research was carried out as part of the SINBAD project with the support of the member organizations of the SINBAD Consortium.

SINBAD