Universidade de São Paulo

Full Waveform Inversion with Finite Elements

Keith Roberts, Lucas Franceschini, João Anderson Isler, Alexandre Olender

Universidade de São Paulo
Escola Politécnica

Research Centre for Gas Innovation

Outline

1. **Purpose**
2. **Background**
   2.1. Problem statement
   2.2. Governing equations
3. **Methods**
   3.1. Spatio-temporal discretization
   3.2. Adjoint formulation
   3.3. Implementation of Full Waveform Inversion
4. **Results**
   4.4.1. Configuration
   4.4.2. Experimental results
5. **Next steps**

Purpose

Universidade de São Paulo

- Developing programs to solve inversion problems in seismologic domains using finite elements.
  - Using Firedrake: a Domain Specific Language (DSL) written in Python to solve variational problems.

- Workstreams 3 and 4 have been collectively developing:
  - ***Spyro***: Software for time domain FWI in Firedrake
    1. Automated mesh generation workflows for seismology in 2D and 3D for isotropic triangular and tetrahedral elements.
    2. Continuous- and Discontinous Galerkin wave propagators (acoustic and elastic) with arbitrary P order in 2D and 3D.
    3. SSPRK, Leapfrog, and Newmark time stepping schemes (support for up to 4th order accurate in time).
    4. Perfectly Matched Layer absorbing layers in 2D and 3D.
    5. Mesh-independent functional gradient for all discretizations using discrete adjoint method.

Problem Statemen

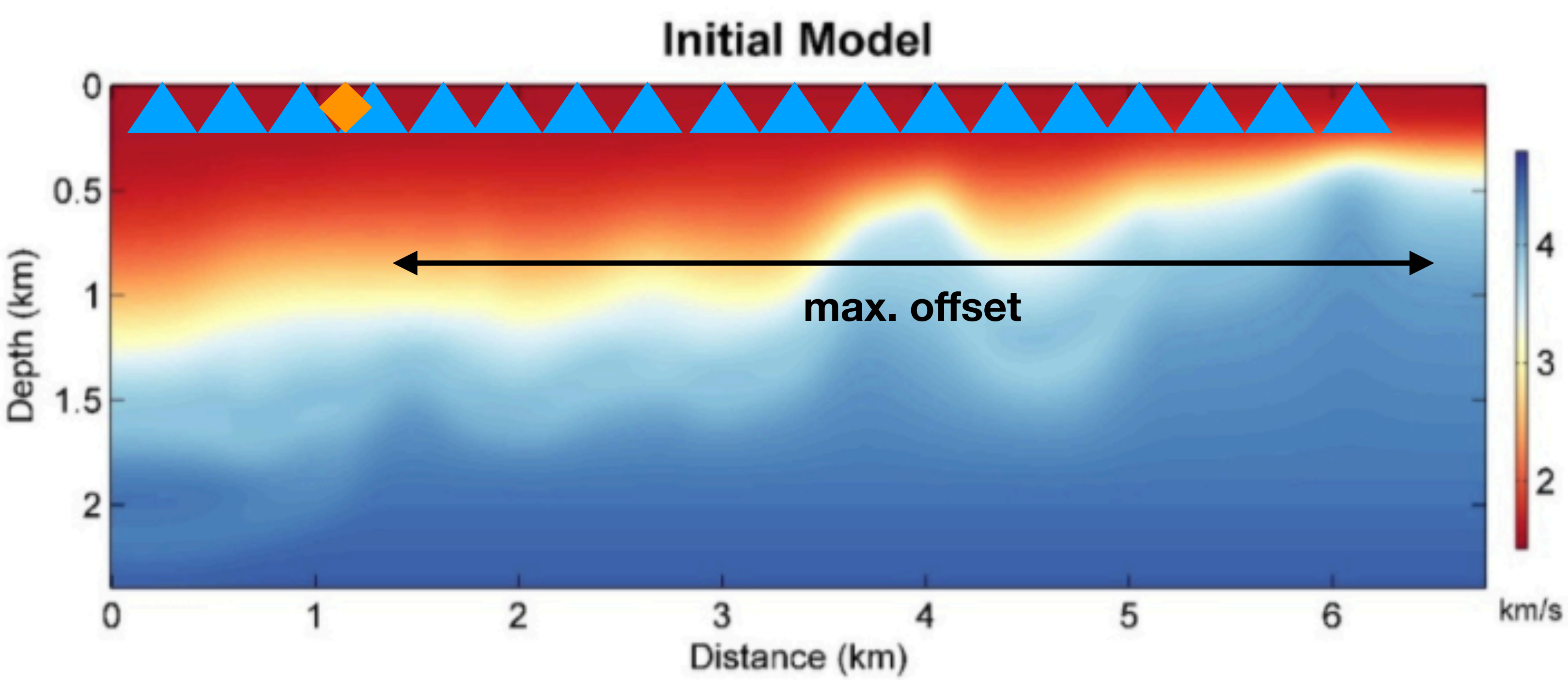
Universidade de São Paulo

- Full-Waveform Inversion (**FWI**) derives high-resolution velocity models by minimizing the difference between observed and modeled seismic waveforms (all the waveforms=full).
- Minimize the functional χ subject to constraints imposed by the acoustic wave equation.
  - Iteratively modify model parameter.

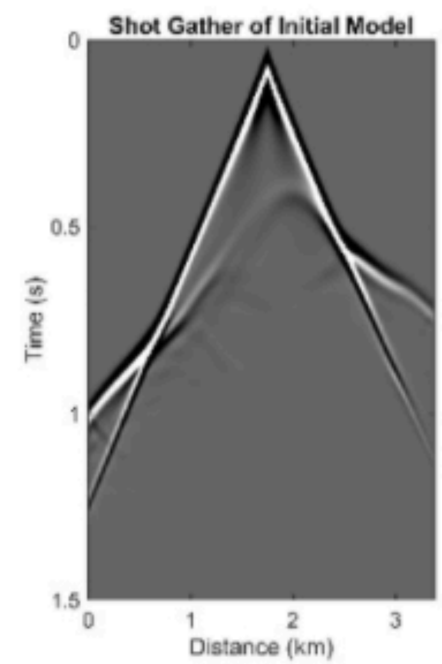
▲ **receiver**

◆ **source**

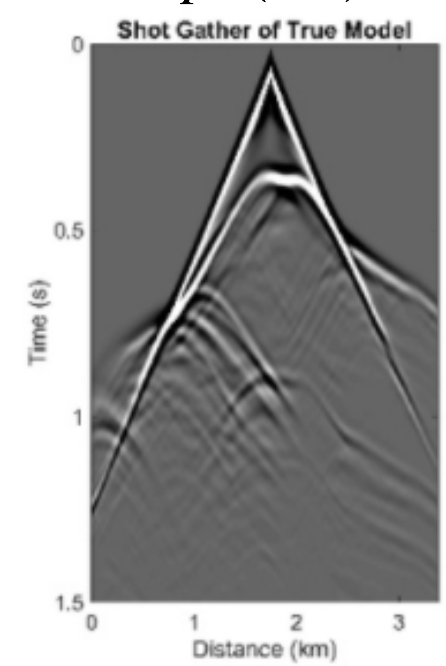
$$\chi(p) = \frac{1}{2}\int_T \sum_{k=1}^{N_p} [p(\boldsymbol{x}_k, t) - p^0(\boldsymbol{x}_k, t)]^2 dt$$



$p(x, t)$



$p^0(x, t)$



Governing equations

We consider the (second-order) scalar wave problem

$$\partial_{tt} p - \nabla \cdot (c^2 \nabla p) = f + c^2 \gamma \qquad \text{in } I \times \Omega, \tag{1}$$

$$\frac{\partial p}{\partial t} + c \nabla p \cdot \hat{n} = 0 \qquad \text{on } \Gamma_N, \tag{2}$$

$$p\big|_{t_0} = 0 \qquad \text{in } \Omega, \tag{3}$$

$$\partial_t p\big|_{t_0} = 0 \qquad \text{in } \Omega, \tag{4}$$

where p is the pressure, $c = \sqrt{\lambda / \rho}$ is the acoustic wave velocity, γ is a term representing the absorbing boundary condition, λ is the first Lamé parameter, ρ is the density, f is the source, $I = (0, T)$ is a finite interval, Ω is a bounded domain, Γ_N is a subset of the boundary.

Spatial Discretization

The weak formulation is obtained multiplying the acoustic wave equation by a test function, integrating over the domain and applying the Gauss divergence theorem. It is given by the following statement. Find $p \in V^C$ such that for all $q \in V^C$

$$\partial_{tt}(p, q)_\Omega + a^{(C)}(p, q) - < c^2 \nabla p \cdot \hat{n}, q >_{\partial\Omega} = (f, q)_\Omega + (c^2\gamma, q)_\Omega, \quad (5)$$

with

$$(p, q)_\Omega = \int_\Omega pq\mathrm{d}\mathbf{x},$$

$$a^{(C)}(p, q) = \int_\Omega c^2 \nabla p \cdot \nabla q \mathrm{d}\mathbf{x}$$

and

$$< \nabla p \cdot \hat{n}, q > = \int_{\partial\Omega} c^2 (\nabla p \cdot \hat{n}) q \mathrm{d}s = 0$$

Temporal discretization

The algebro-differential (Eq. 5) was temporally discretized such that $t_n = n\Delta t$ timesteps:

$$\underbrace{\int_\Omega \frac{p^{n+1} - 2p^n + p^{n-1}}{\Delta t^2} q \mathrm{d}\mathbf{x}}_{\text{mass matrix}} + \underbrace{\int_\Omega c^2 \frac{\partial q}{\partial x} \frac{\partial p^{n+1}}{\partial x} \mathrm{d}\mathbf{x}}_{\text{stiffness matrix}}$$

$$= \int_\Omega f^n q \mathrm{d}\mathbf{x} + \int_\Omega c^2 \gamma \mathrm{d}\mathbf{x} - \int_\Omega \frac{p^{n+1} - p^{n-1}}{2\Delta t} \mathrm{d}\mathbf{x}.$$

For the stiffness matrix, we choose to represent the ∇p variable at time $n+1$ for numerical stability.

Second-order system of ordinary differential equations

Thus, the discretization in space of the Continuous Galerking method leads to the linear second-order system of ordinary differential equations

$$\mathbf{M}\ddot{\mathbf{u}}_h(t) + \mathbf{A}\mathbf{u}_h(t) = \mathbf{f}_h(t), \quad t \in I \tag{7}$$

with initial conditions

$$\mathbf{u}_h(t_0) = \mathbf{u}_h(0) := \mathbf{0}, \quad \dot{\mathbf{u}}_h(t_0) = \dot{\mathbf{u}}_h(0) := \mathbf{0},$$

where **M** denotes the mass matrix and **A** the stiffness matrix.

Discrete adjoint

Universidade de São Paulo

- Theoretically simpler than continuous adjoint (transpose of system of equations).

<u>Why we didn't use automatic differentiation?</u>

- Automatic differentiation w/ Dolfin-adjoint does not support functionals defined at points (i.e., receivers).
- Much of the available gradient-based optimization methods are mesh-dependent.
- More flexibility in our implementation (e.g., support for both ensemble and spatial parallelism).

Discrete Adjoint – Mathematical Formulation

- Starting point: time-stepper for a variable ϕ:

$$\begin{cases} \phi_0 = \phi^0 & n = 0 \\ \mathbb{B}_1\phi_1 + \mathbb{B}_0\phi_0 + \mathbb{M}\dot{\phi}^0 + S_0 = 0 & n = 1 \\ \mathbb{A}_n\phi_n + \mathbb{A}_{n-1}\phi_{n-1} + \mathbb{A}_{n-2}\phi_{n-2} + S_{n-1} = 0 & n > 1 \end{cases}$$

- Matrices $\mathbb{B}_k, \mathbb{A}_k$ are spatial discrete matrices
- ϕ^0 and $\dot{\phi}^0$ are initial conditions (homogeneous)
- S_n are external source terms
- If PML is used, ϕ contains the pressure and other damping functions

Discrete Adjoint – Mathematical Formulation

- Under the above constraints, we look for minimizing some error $J(\phi_n)$:

$$J(\phi_n) = \frac{1}{2}\sum_{n=0}^{N} (\mathbb{H}\phi_n - r_n)^T(\mathbb{H}\phi_n - r_n)$$

- $\mathbb{H}$ is the measure operator, r_n are the shot records
- We resort to the Lagrangian formalism:

$$\mathrm{L}(\phi_n, \phi_n^a, c) = J(\phi_n) + \phi_0^{a,T}(\phi_0 - \phi^0) + \phi_1^{a,T}\left(\mathbb{B}_1\phi_1 + \mathbb{B}_0\phi_0 + \mathbb{M}\dot{\phi}^0 + S_0\right)$$

$$+\sum_{n>1}^{N} \phi_n^{a,T}(\mathbb{A}_n\phi_n + \mathbb{A}_{n-1}\phi_{n-1} + \mathbb{A}_{n-2}\phi_{n-2} + S_{n-1})$$

Discrete Adjoint – Mathematical Formulation

- Taking its variation with respect to the direct variables ϕ_n leads to:

$$\begin{cases} \mathbb{A}_n^T \phi_n^a + \mathbb{A}_{n-1}^T \phi_{n+1}^a + \mathbb{A}_{n-2}^T \phi_{n+2}^a = -\mathbb{H}^T\left(\mathbb{H}\phi_n - r_n\right) & n < N-1 \\ \mathbb{A}_n^T \phi_n^a + \mathbb{A}_{n-1}^T \phi_{n+1}^a = -\mathbb{H}^T\left(\mathbb{H}\phi_n - r_n\right) & n = N-1 \\ \mathbb{A}_n^T \phi_n^a = -\mathbb{H}^T\left(\mathbb{H}\phi_n - r_n\right) & n = N \end{cases}$$

- Taking its variation with respect to the control c leads (Leap-Frog, no PML) to:
- Note: M is on RHS of gradient calculation!!

$$\mathbb{M}\nabla_c J = 2\sum_{n=0}^{N}\int c\nabla\phi_n \cdot \nabla\phi_n^a \delta c \, d\mathbf{x}$$

Implementation of FWI

Universidade de São Paulo

Algorithm 1: Multiscale time-domain full waveform inversion.

Result: Optimized velocity model $c(X)$ over a range of source frequencies $freq$.

$c^0 \leftarrow$ *initial velocity model*;
$k \leftarrow 0$;
for $freq \leftarrow freq_{min}$ **to** $freq_{max}$ **do**
  Assign source frequency $freq$;
  while $\nabla J > 0$ & $J > 0 \; \| k \leq (iter_{max} - 1)$ **do**
    for $iter \leftarrow 0$ **to** $(iter_{max} - 1)$ **do**
      Compute all forward simulations for all n shots;
      Calculate J_n at receivers;
      Compute local gradient ∇J_n via discrete adjoint;
      Sum J_n onto master;
      Sum ∇J_n onto master;
      if *rank is master* **then**
        Given ∇J and J using L-BFGS produce Δc^k;
        $c^{k+1} \mathrel{+}= \Delta c^k$;
      end if
      Broadcast c^{k+1} from master.
    end for
  end while
end for

Implementation of FWI

- A new point evaluation function.
  - Interpolating point data at arbitrary P-order quickly.

- Support for high-order spectral elements in tetrahedral cells.
  - Current tetrahedral Firedrake implementation only uses equispaced elements and doesn't have optmizations, such as sum-factorization, increasing operations needed for matrix assembly and matrix-free calculations.

Mesh Developmen

- 2D/3D serial and distributed memory parallel triangular meshing for a slab of Earth in Python using signed distance functions.

- https://github.com/krober10nd/SeismicMesh

SEG-Y file —> simulation ready mesh

- Python and C++ bound together using pybind11.

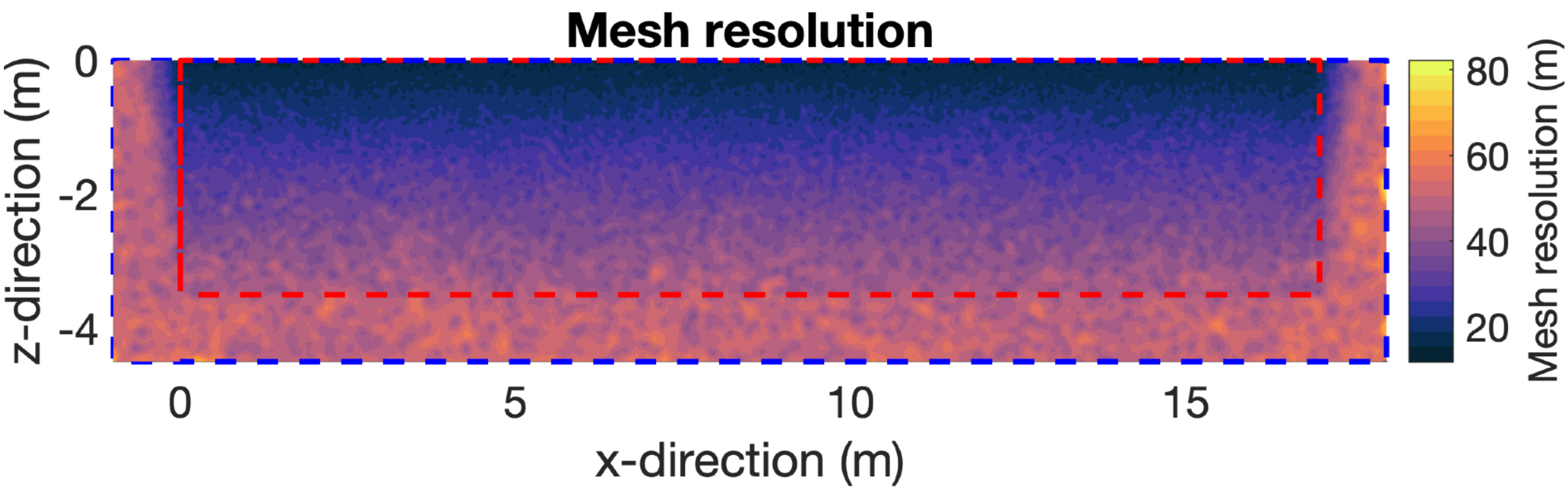
- Modifications to *DistMesh* [2] algorithm.

  - Computational Graphic Algorithms Library (CGAL) and Boost are used for all “expensive” geometrical operations.

- Pre- and post processing (e.g., input file creation, mesh size function class, boundary condition applier, etc.).

Mesh Development

- Minimum P wave speed , maximum source frequency, and spatial order determine minimum resolution.
- $h(X) = \dfrac{v_p}{f_{max} * \alpha_{wl}}, \alpha = f(p)$
- $Cr(h) < CFL, h_{min} \leq h \leq h_{max}, \nabla h \leq g$
    - 39,346 vertices and 77,649 elements



Mesh Development

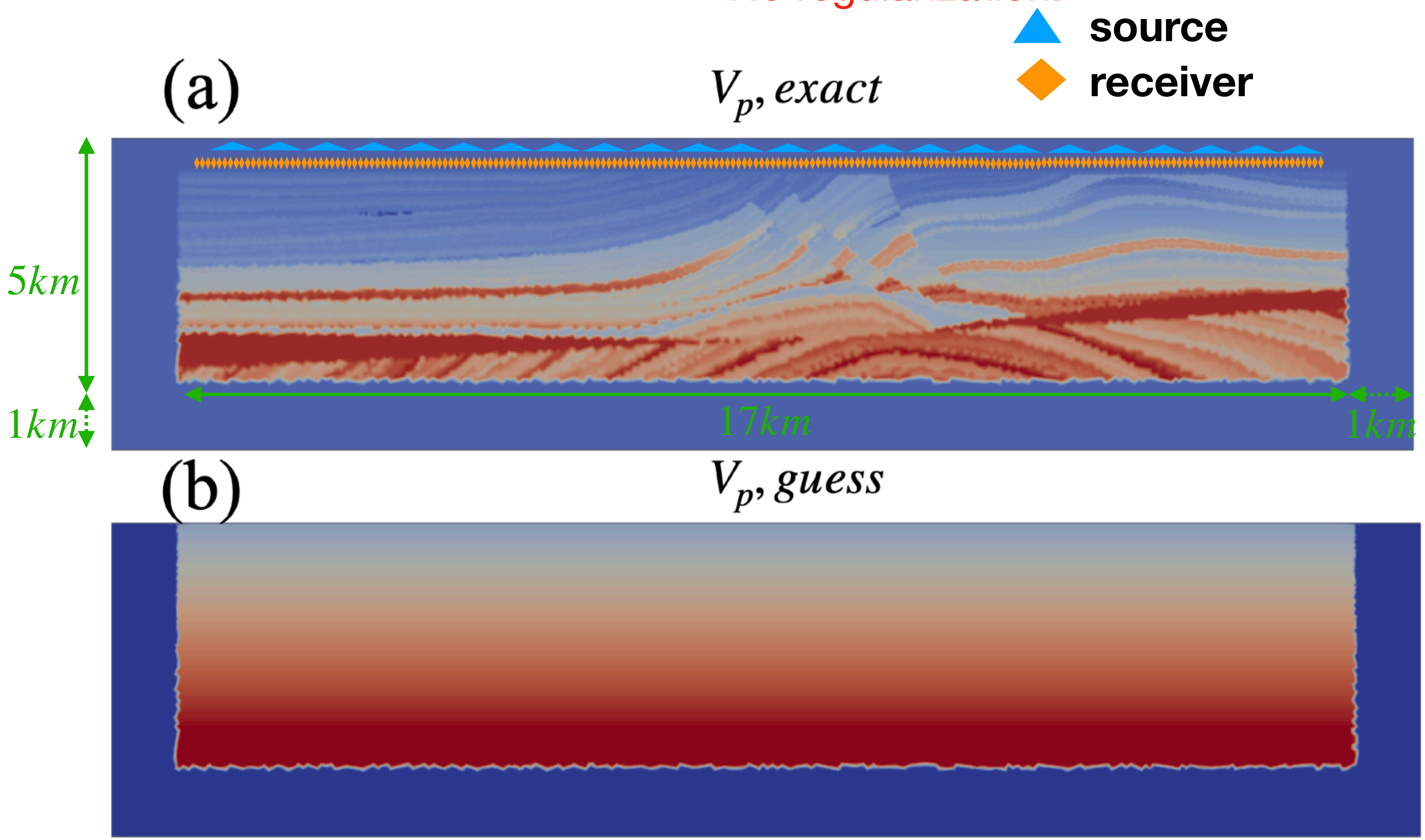
Universidade de São Paulo

```python
import meshio
import numpy as np

import SeismicMesh


def example_2D():
    # Name of SEG-Y file containg velocity model.
    fname = "velocity_models/vel_z6.25m_x12.5m_exact.segy"
    bbox = (-12e3, 0, 0, 67e3)

    # Construct mesh sizing object from velocity model
    ef = SeismicMesh.MeshSizeFunction(
        bbox=bbox,
        model=fname,
        domain_ext=1e3,
        dt=0.001,
        grade=0.15,
        freq=5,
        wl=5,
        hmax=1e3,
        hmin=50.0,
    )

    # Build mesh size function
    ef = ef.build()

    ef.WriteVelocityModel("BP2004")

    # Visualize mesh size function
    ef.plot()

    # Construct mesh generator
    mshgen = SeismicMesh.MeshGenerator(
        ef, method="cgal"
    )  # if you have cgal installed, you can use method="cgal"

    # Build the mesh (note the seed makes the result deterministic)
    points, facets = mshgen.build(max_iter=50, nscreen=1, seed=0)
```

Experimental configuration

Universidade de São Paulo

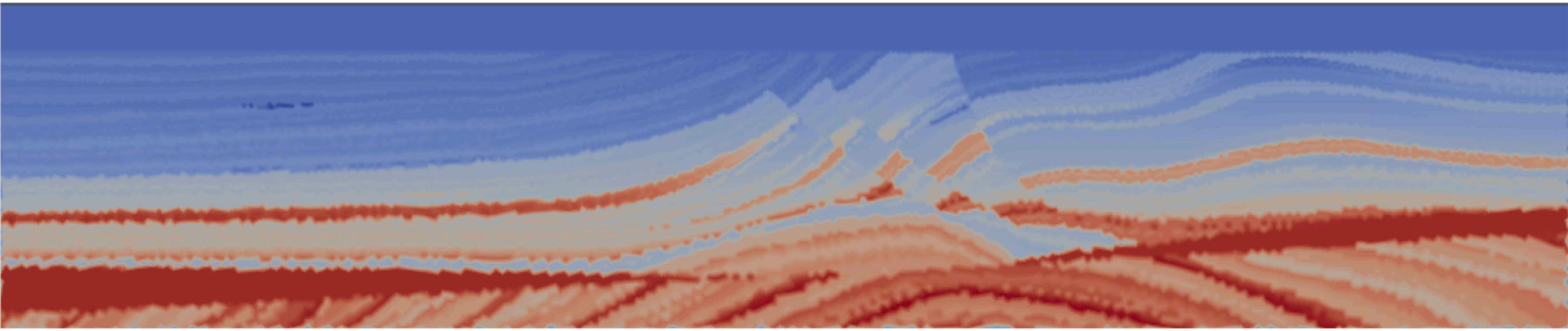
- 24 shots 50-m below the surface.
- 300 receivers 100-m below the surface
- Single-band $3hz$ source frequency.
- Simulation $T = 5, \Delta t = 0.005$ seconds
- 1 km domain extension with ABC.
- Observed shot record generated with different mesh
- Forward simulation kept in RAM
- No regularization.



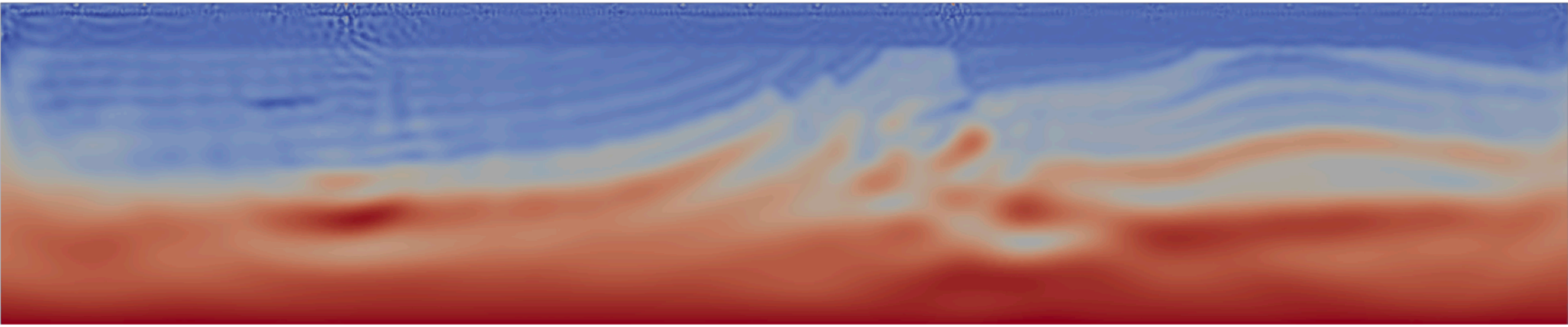
Results

300 iterations, ~3 hours.
24 processors on AWS cluster

(a) V_p, *exact*



(b) $V_p^{k=300}$, *guess*



Next steps

- Repeating using PML implementation and with the Gato do Mato velocity model.
- Using a time-domain multiscale approach (i.e., progressively increasing source frequency).
- Using “observed” shot record created from another model (e.g. elastic “observed” shot record) for acoustic FWI.
- Checkpointing schemes!

References

[1] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. Mcrae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw.*, 43(3):24:1–24:27, 2016. URL: http://arxiv.org/abs/1501.01809, arXiv:1501.01809, doi:10.1145/2998441.

[2] Per Olof Persson and Gilbert Strang. A Simple Mesh Generator in MATLAB. SIAM Review, 46:2004, 2004

Thanks for listening!

Research Centre for Gas Innovation

USP Universidade de São Paulo