Advancing Subsurface Modeling
with a Scalable Frequency-Domain Wave Solver

Allocation: "Efficient seismic simulation in challenging geological environments with a scalable frequency-domain solver"

Team:
Sergey Fomel (UT Austin, Bureau of Economic Geology; PI)
Andrey Bakulin (UT Austin, Bureau of Economic Geology)
Jacob Badger (UT Austin, Institute for Fusion Studies)
Lars Koesterke (TACC, Collaborator)

Frontera User Meeting

August 5-6, 2024

COI Disclosure

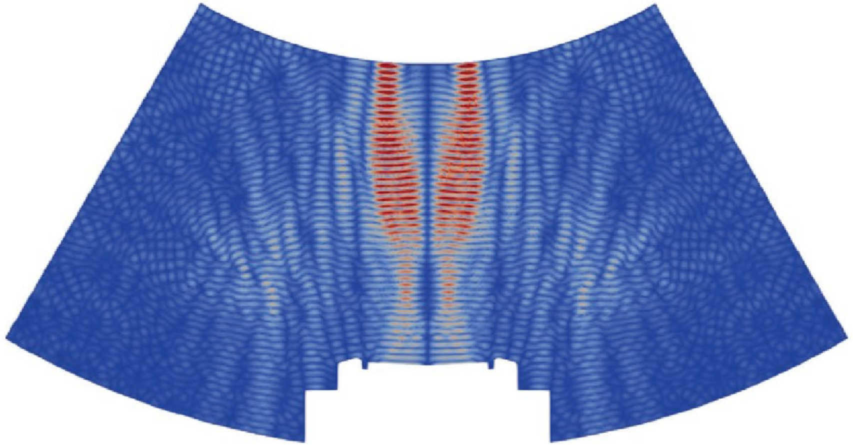
The presenter has a financial interest in Cillatory, LLC (defunct) and FrequenSol, LLC.

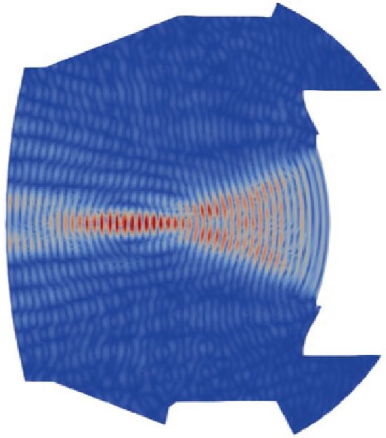
Motivation

Wave Applications

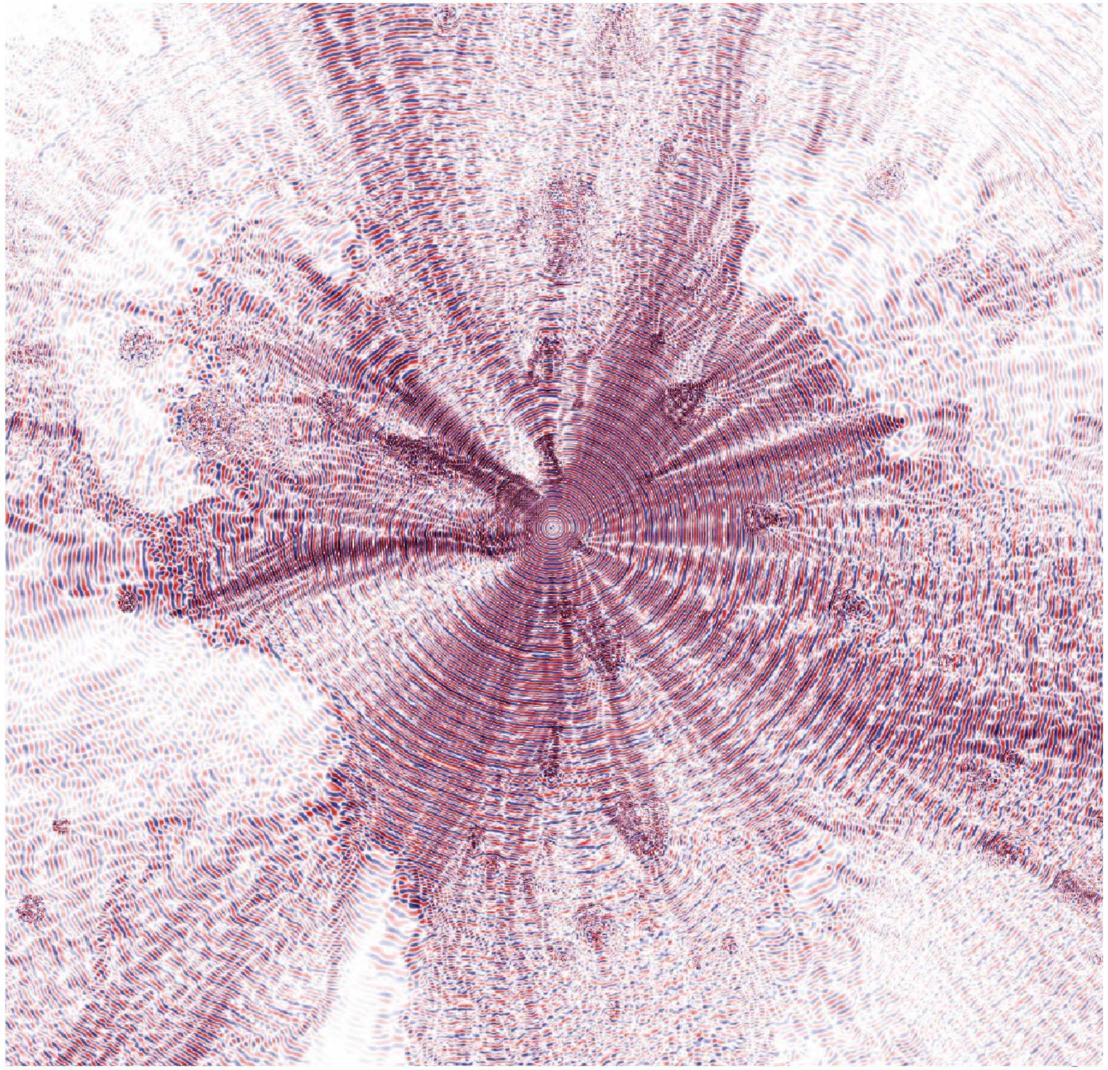
Medical ultrasound (acoustics)

Tokamak RF heating (electromagnetics)





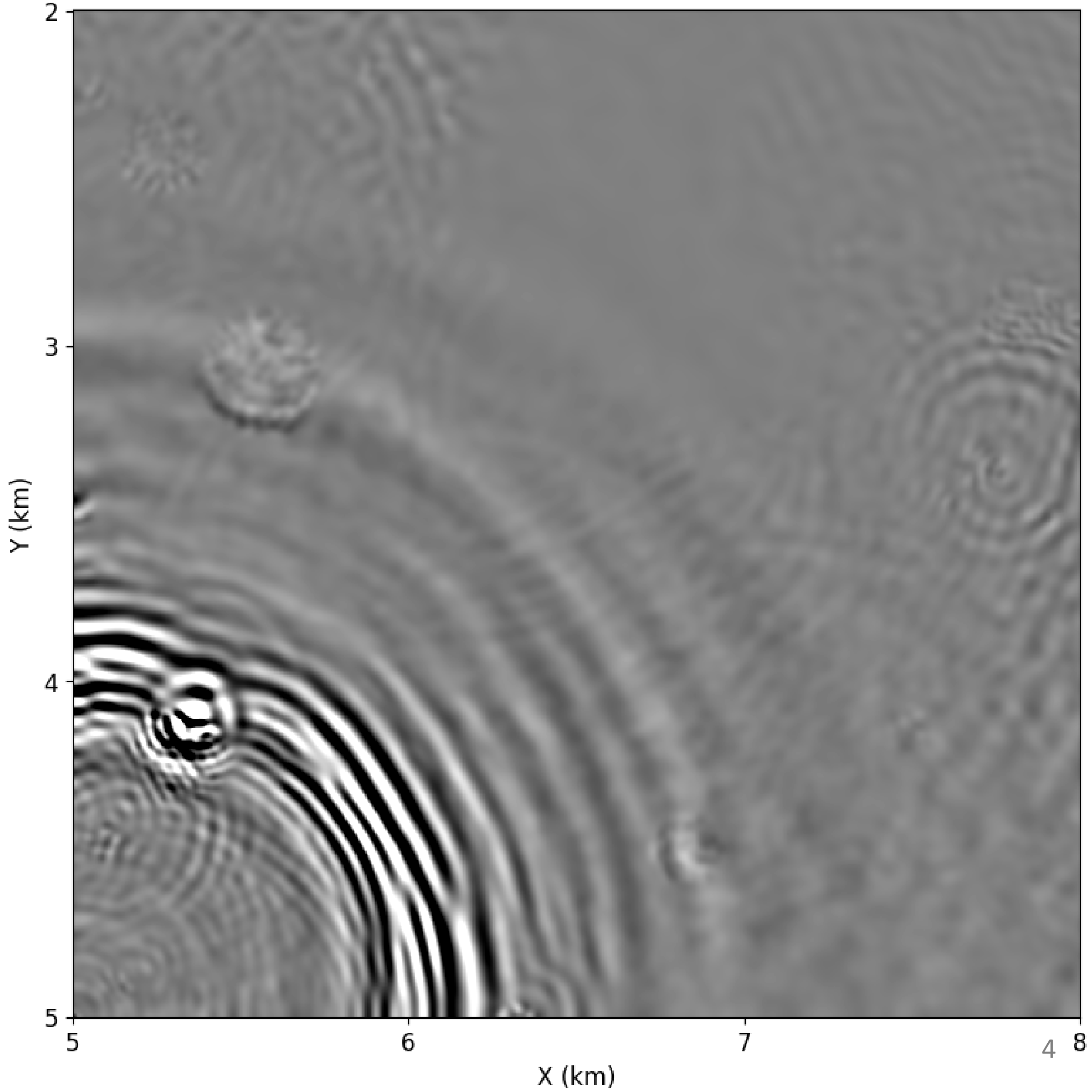
Seismic simulation (elasticity)



Wave Simulation (Time-Domain)

Time Domain Simulation

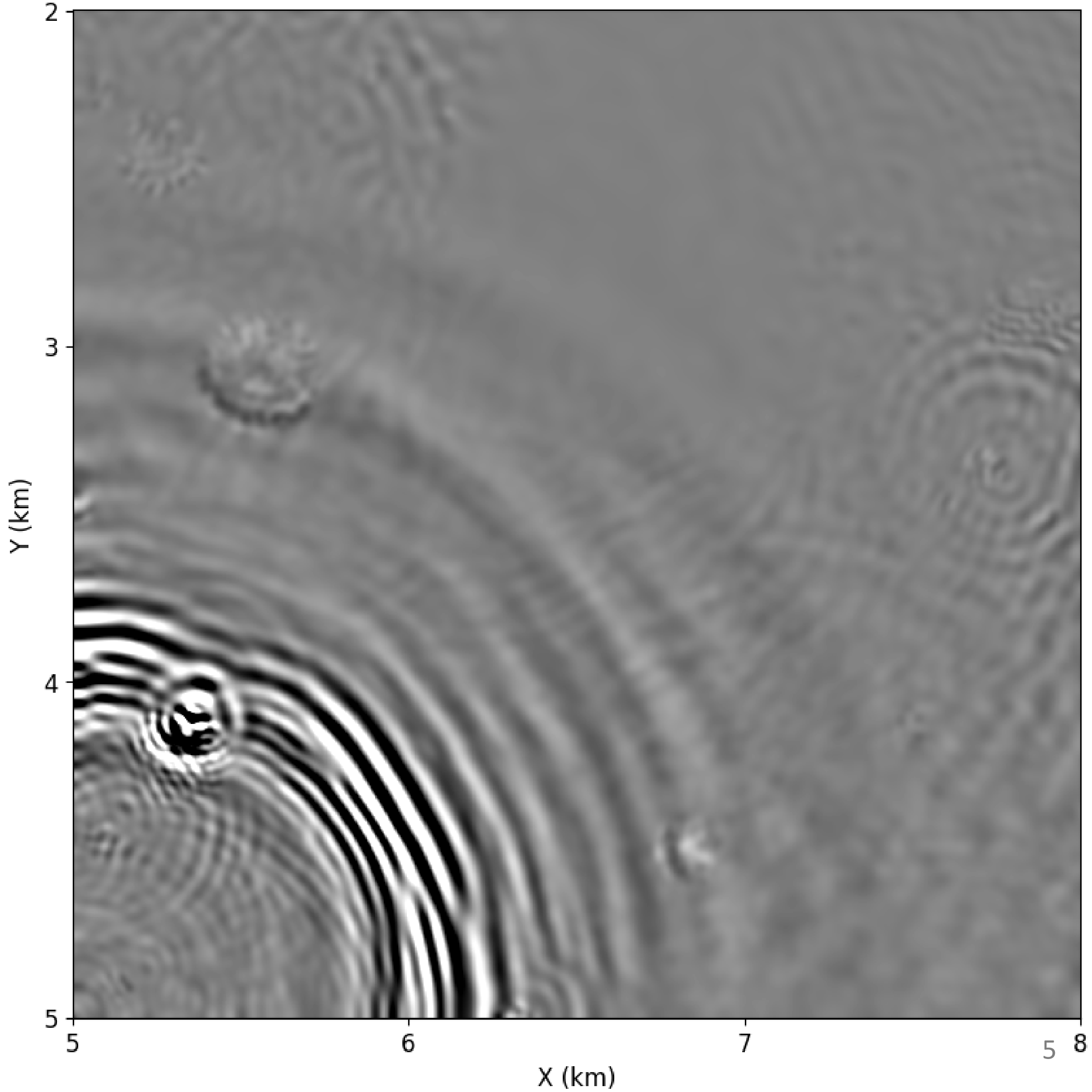
- Starts from known initial state



Wave Simulation (Time-Domain)

Time Domain Simulation

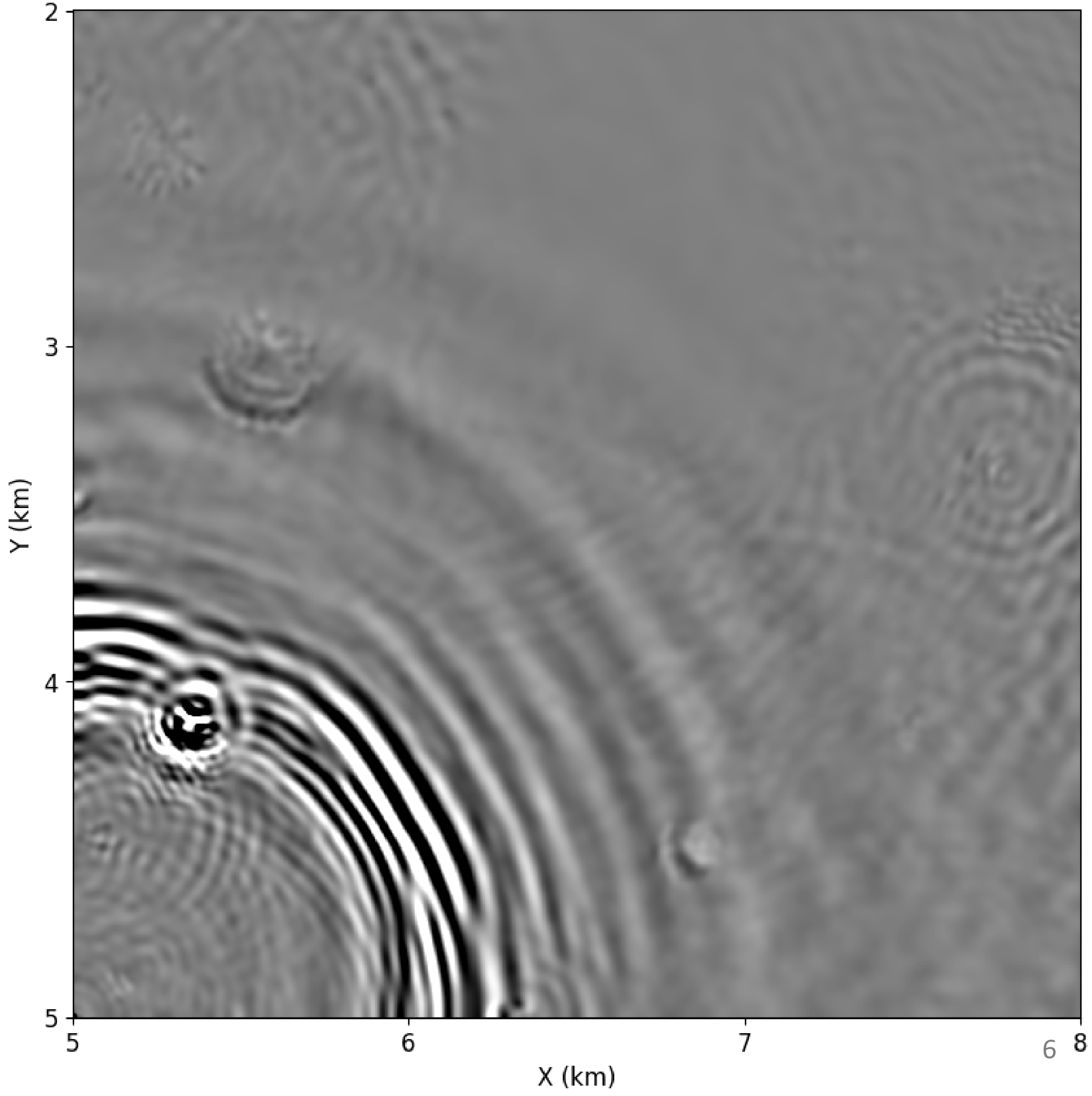
- Starts from known initial state
- "Steps" forward by small time increments



Wave Simulation (Time-Domain)

Time Domain Simulation

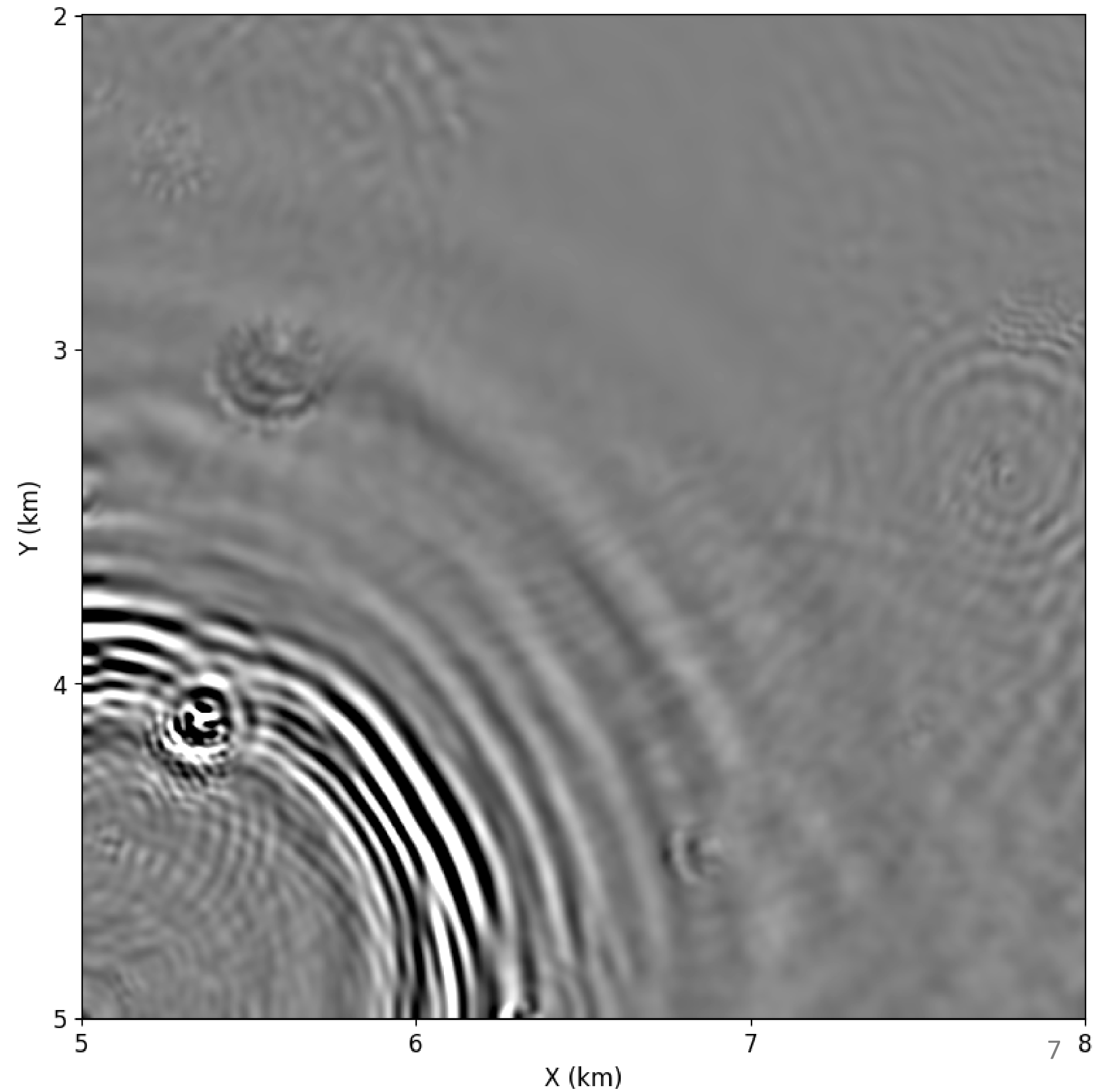
- Starts from known initial state
- “Steps” forward by small time increments
- Stepping via:
  - Matrix multiplication (explicit)
  - Elliptic linear solve (implicit)



Wave Simulation (Time-Domain)

Time Domain Simulation

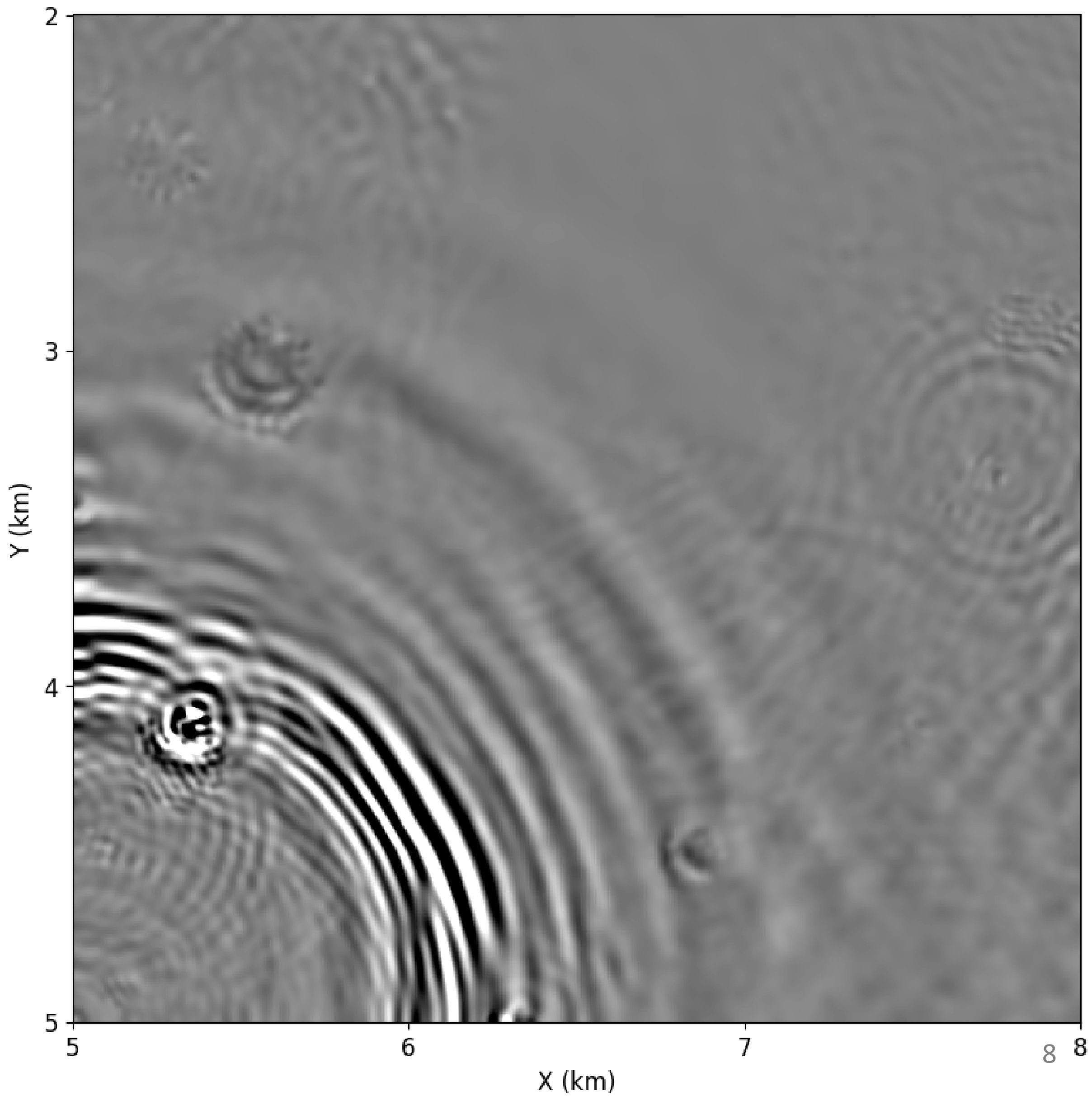
- Starts from known initial state
- “Steps” forward by small time increments
- Stepping via:
    - Matrix multiplication (explicit)
    - Elliptic linear solve (implicit)



Wave Simulation (Time-Domain)

Time Domain Simulation

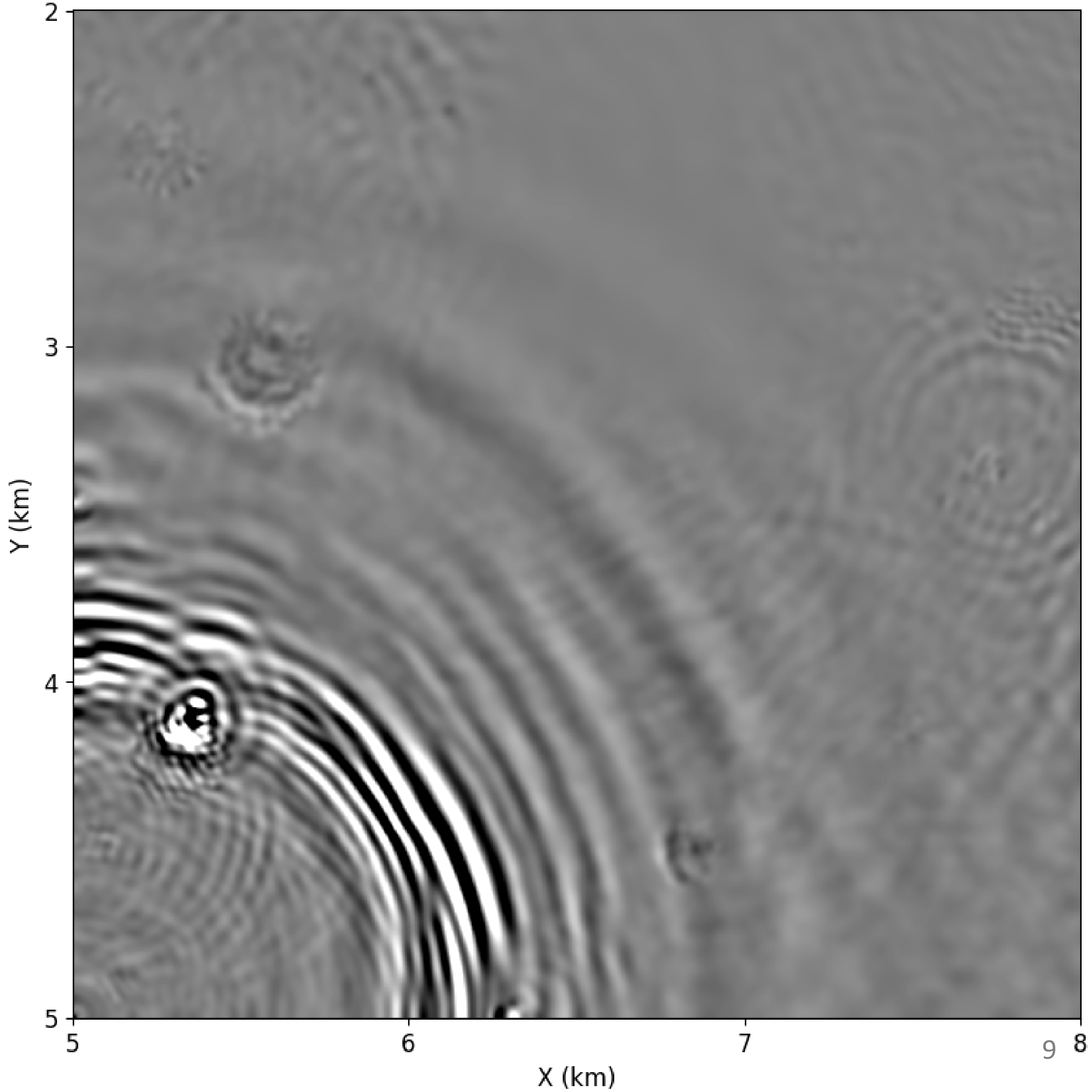
- Starts from known initial state
- “Steps” forward by small time increments
- Stepping via:
  - Matrix multiplication (explicit)
  - Elliptic linear solve (implicit)



Wave Simulation (Time-Domain)

Time Domain Simulation

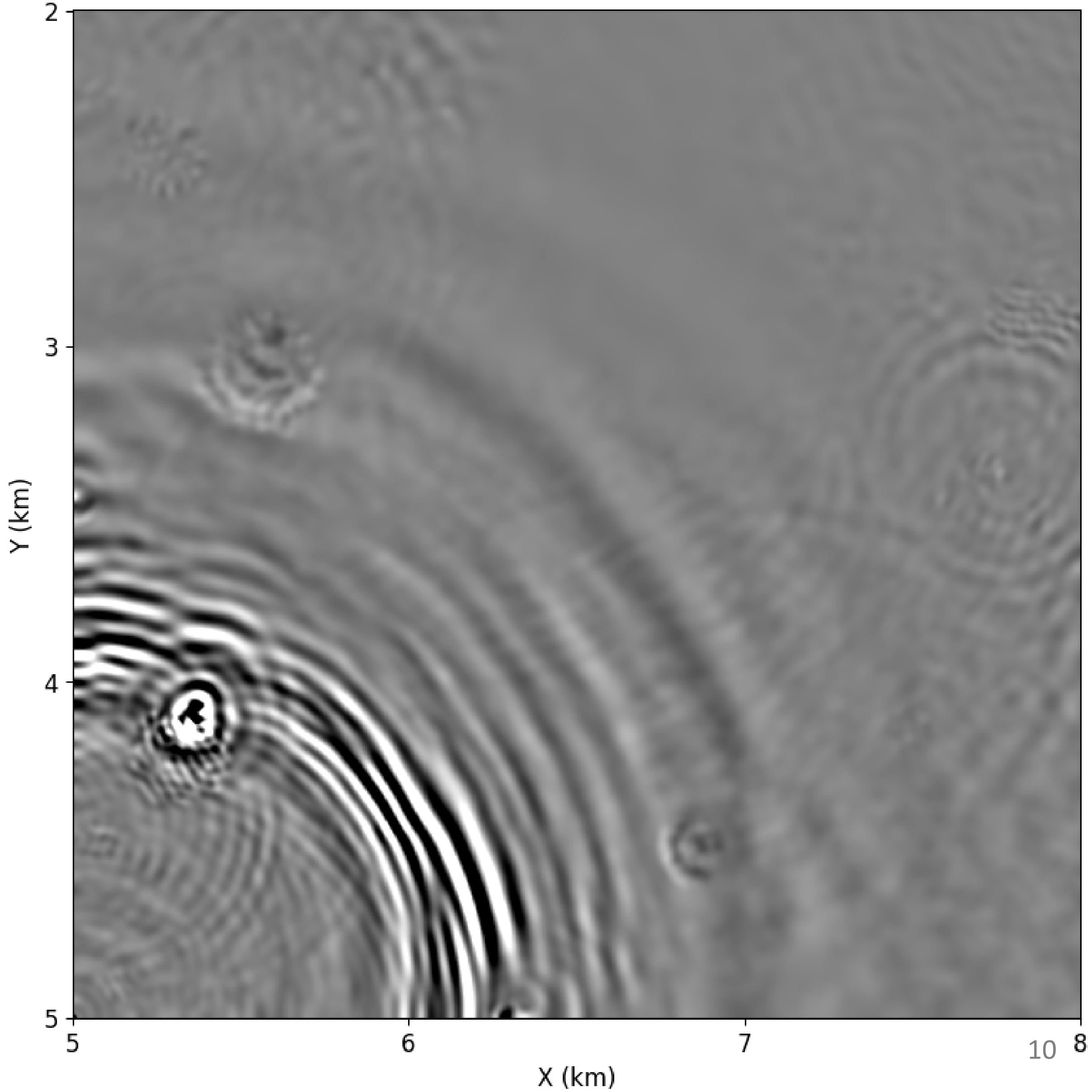
- Starts from known initial state
- "Steps" forward by small time increments
- Stepping via:
  - Matrix multiplication (explicit)
  - Elliptic linear solve (implicit)



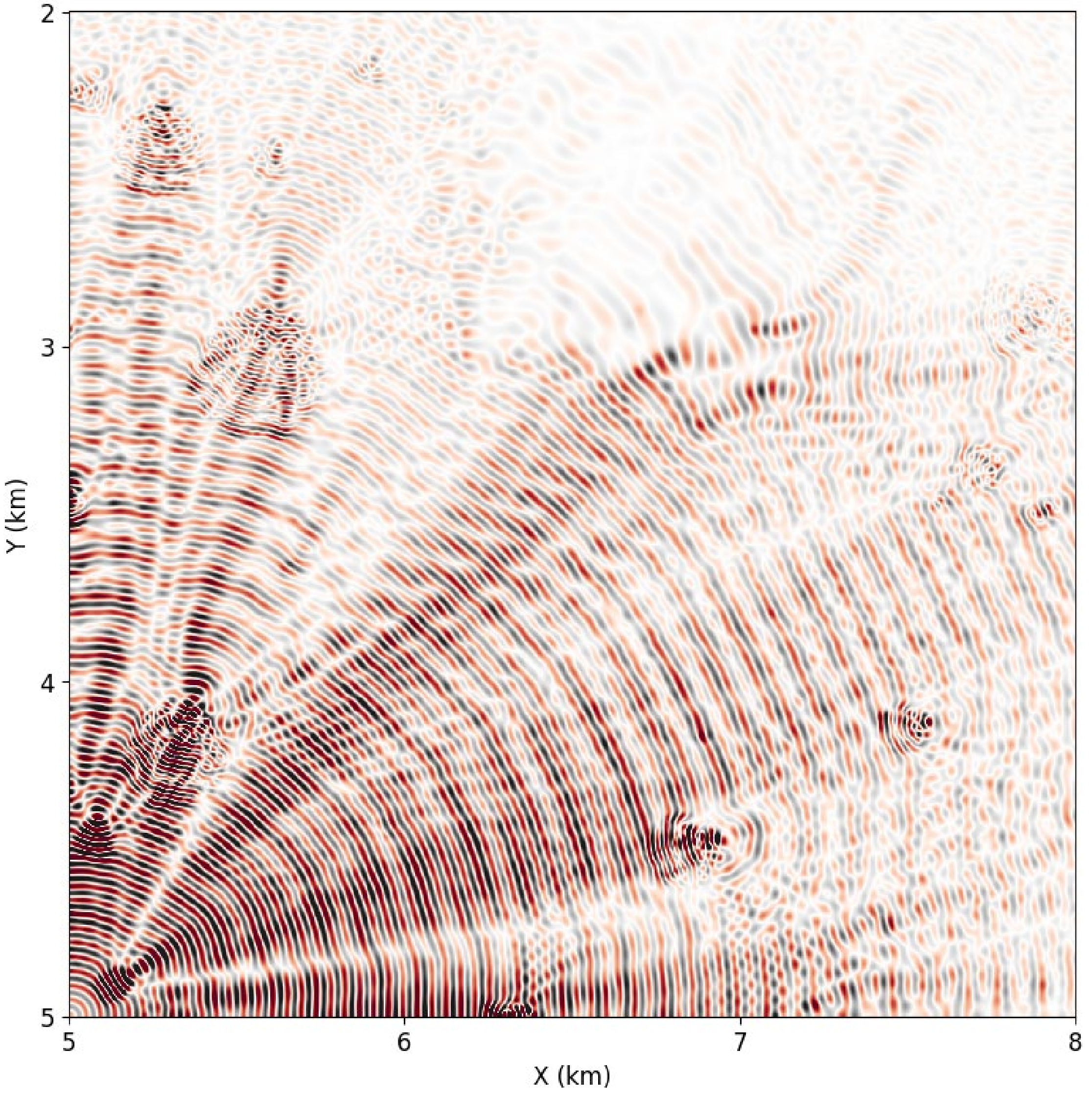
Wave Simulation (Time-Domain)

Time Domain Simulation

- Starts from known initial state
- "Steps" forward by small time increments
- Stepping via:
  - Matrix multiplication (explicit)
  - Elliptic linear solve (implicit)



Wave Simulation (Frequency-Domain)



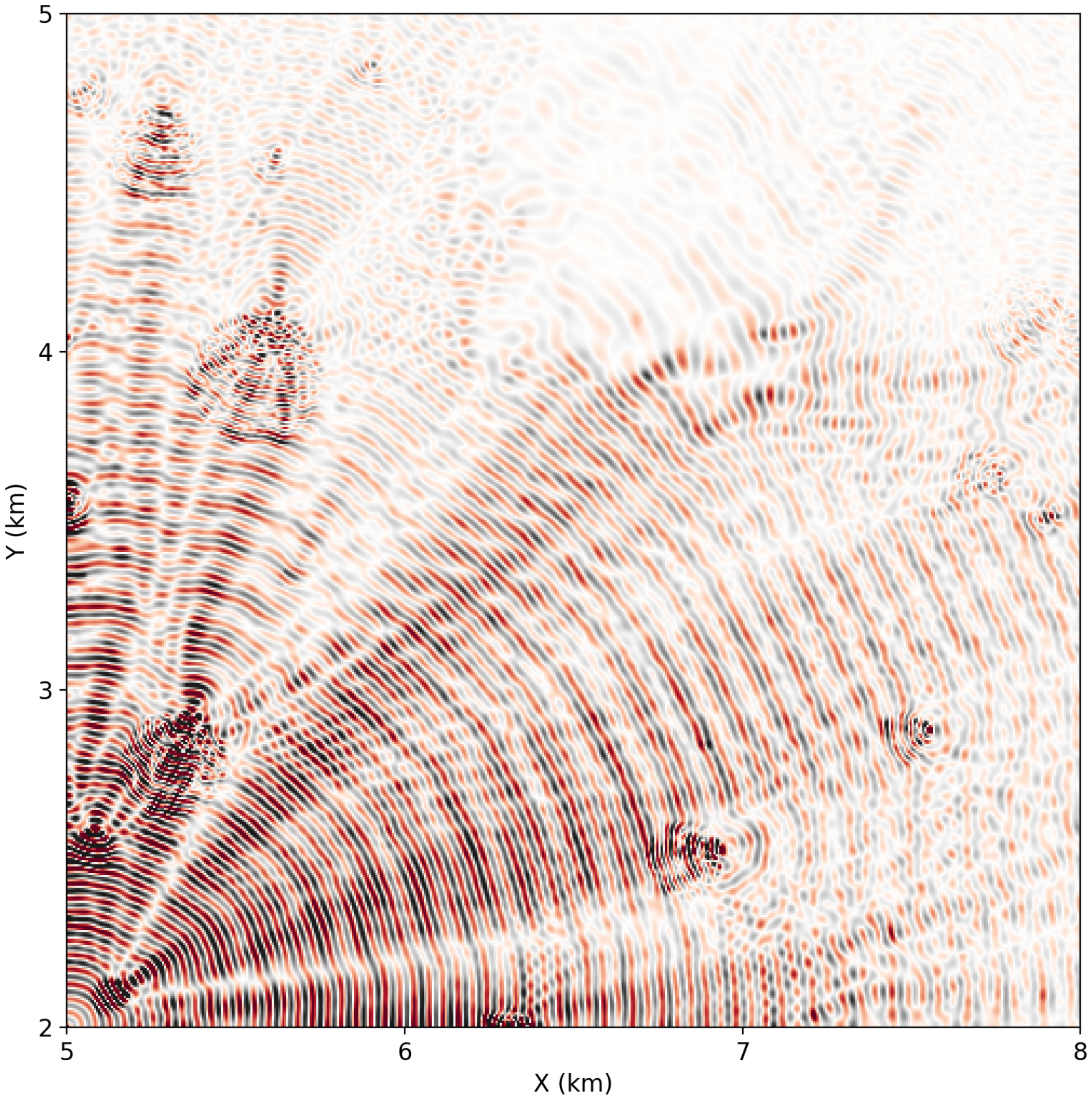
Frequency Domain Simulation

- Separates problem into independent frequencies (ω)

$$\mathrm{u}_\omega(\mathbf{x}, t) = \Re\{\widetilde{\mathrm{u}}_\omega(\mathbf{x}) e^{i\omega t}\} \quad (1)$$

- Each frequency requires an *indefinite* linear solve
- Produces a complex-valued field ($\tilde{u}_\omega$) for each frequency

Wave Simulation (Frequency-Domain)



Frequency Domain Simulation

- Separates problem into independent frequencies (ω)

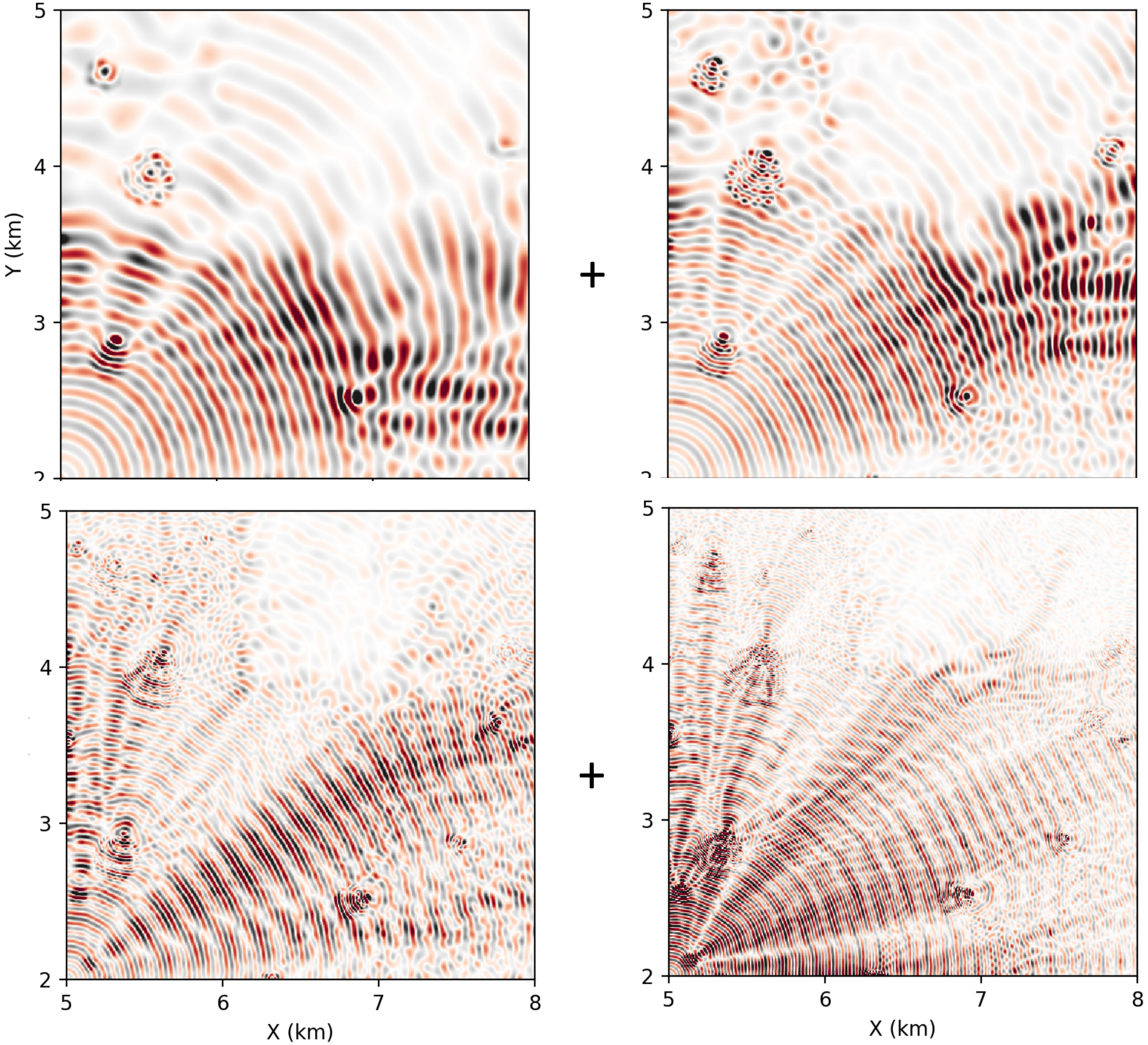
$$\mathrm{u}_\omega(\mathbf{x}, t) = \Re\{\widetilde{\mathrm{u}}_\omega(\mathbf{x})e^{i\omega t}\} \tag{1}$$

- Each frequency requires an *indefinite* linear solve
- Produces a complex-valued field ($\tilde{u}_\omega$) for each frequency
- Time-harmonic field recovered via (1)

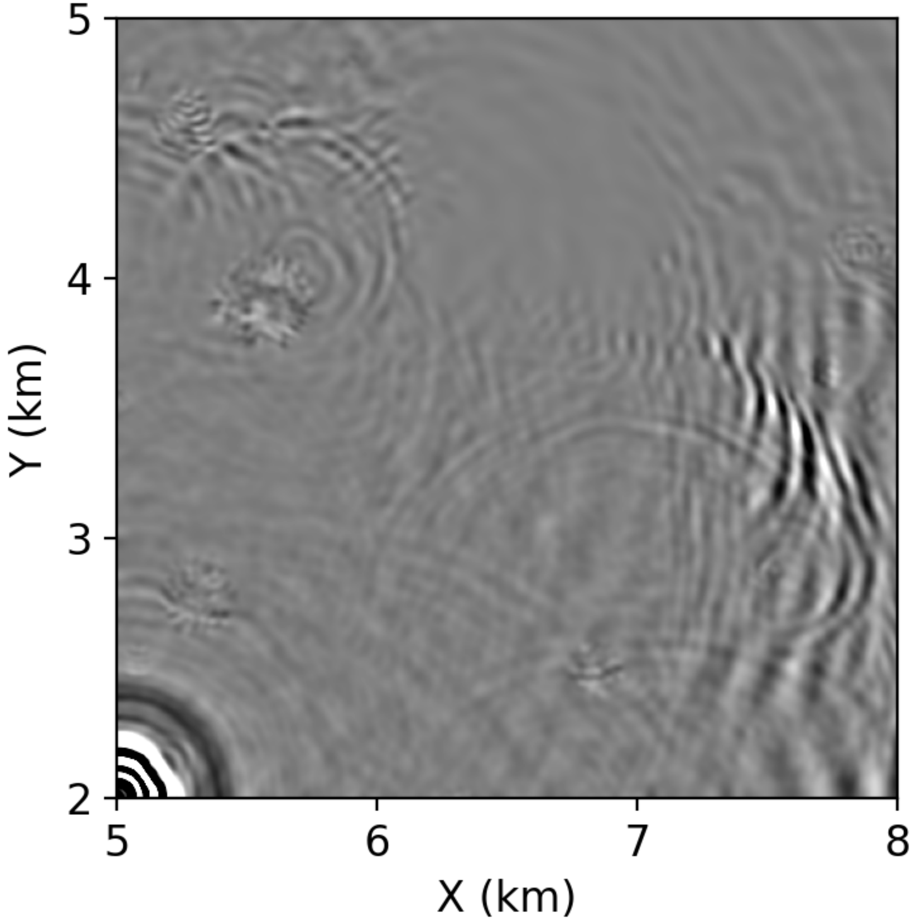
Wave Simulation (Frequency-Domain)

Time-domain solution can be reconstructed by summing over frequencies:

$$\mathbf{u}(\mathbf{x},t) = \Re\left\{\sum_{\omega} c_\omega \widetilde{\mathbf{u}}_\omega(\mathbf{x}) e^{i\omega t}\right\}$$



+ … =



Challenges

Leading methods for all approaches have same $O(\omega^4)$ complexity (in 3D)

Time domain (explicit):

- Stability
  - Timestep depends on wavespeed contrast, minimum element size (grid spacing), etc.
  - Often requires many timesteps, especially for high-contrast problems

Time domain (implicit):

- Efficiency
  - Elliptic solve per timestep
  - Often relies on (pre-computed) sparse matrix factorizations

Frequency domain:

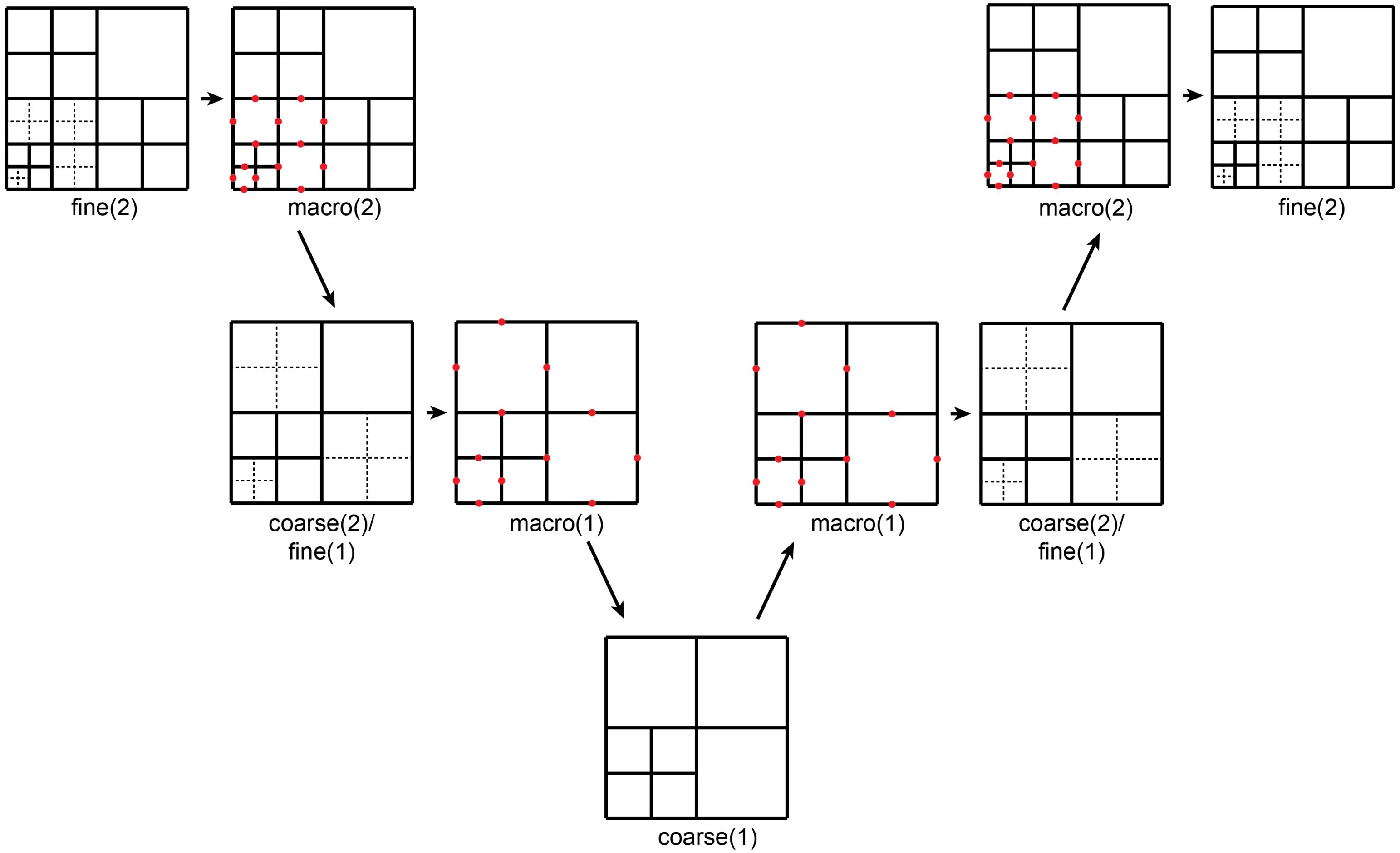
- Scalability
  - Direct solvers: factored in $O(\omega^6)$, applied in $O(\omega^4)$
  - Leading scalable iterative solvers achieve $O(\omega^4)$ complexity
  - Previously limited to ~1 billion degrees of freedom

Fast Frequency-Domain Solver

Fast Frequency-Domain Solver[1]

Key Points:

- Iterative solver (PCG iteration)
- Multigrid preconditioner
- Based on discontinuous Petrov-Galerkin (DPG) finite element method of Demkowicz and Gopalakrishnan[2]



[1] J. Badger. Scalable DPG multigrid solver with applications in high-frequency wave propagation. Ph.D. Thesis, The University of Texas at Austin, 2024.
[2] Leszek Demkowicz and Jay Gopalakrishnan. A class of discontinuous Petrov–Galerkin methods. Part I: the transport equation. Comput. Methods Appl. Mech. Enrg., 199(23-24):1558–1572, 2010.

DPG (Linear Acoustics)

DPG:

$$\mathfrak{u} = (p, \mathbf{u}) \in L^2(\Omega_h) \times \boldsymbol{L}^2(\Omega_h)$$

$$\hat{\mathfrak{u}} = (\hat{p}, \hat{u}_n) \in H^{1/2}_{p_0,\Gamma_1}(\Gamma_h) \times H^{-1/2}_{u_0,\Gamma_2}(\Gamma_h)$$

$$\mathfrak{v} = (q, \mathbf{v}) \in H^1(\Omega_h) \times H(\mathrm{div}, \Omega_h)$$

$$\|\mathfrak{v}\|_V^2 = \alpha\|q\|^2 + \alpha\|\mathbf{v}\|^2 + \|i\omega\bar{c}^{-2}q + \mathrm{div}_h\,\mathbf{v}\|^2 + \|i\omega\,\bar{\mathbf{I}}\,\mathbf{v} + \nabla_h q\|^2$$

$$b(\mathfrak{u}, \mathfrak{v}) = i\omega(c^{-2}p, q) - (\mathbf{u}, \nabla_h q) + i\omega(\mathbf{I}\,\mathbf{u}, \mathbf{v}) - (p, \mathrm{div}_h\,\mathbf{v})$$

$$\hat{b}(\hat{\mathfrak{u}}, \mathfrak{v}) = \langle \hat{u}_n, q\rangle_{\Gamma_h} + \langle \hat{p}, v_n\rangle_{\Gamma_h}$$

$$l(\mathfrak{v}) = (\mathrm{f}_p, q) + (\mathbf{f}_\mathbf{u}, \mathbf{v})$$

Discrete System:

$$\begin{bmatrix} \mathsf{G}_{qq} & \mathsf{G}_{qv} & \mathsf{B}_{qp} & \mathsf{B}_{qu} & & \hat{\mathsf{B}}_{q\hat{u}_n} \\ \mathsf{G}_{vq} & \mathsf{G}_{vv} & \mathsf{B}_{vp} & \mathsf{B}_{vu} & \hat{\mathsf{B}}_{v\hat{p}} & \\ \mathsf{B}^*_{qp} & \mathsf{B}^*_{vp} & & & & \\ \mathsf{B}^*_{qu} & \mathsf{B}^*_{vu} & & & & \\ & \hat{\mathsf{B}}^*_{v\hat{p}} & & & & \\ \hat{\mathsf{B}}^*_{q\hat{u}_n} & & & & & \end{bmatrix} \begin{bmatrix} \mathsf{q} \\ \mathsf{v} \\ \mathsf{p} \\ \mathsf{u} \\ \hat{\mathsf{p}} \\ \hat{\mathsf{u}}_n \end{bmatrix} = \begin{bmatrix} \mathsf{f}_p \\ \mathsf{f}_u \\ \\ \\ \\ \\ \end{bmatrix}$$

Galerkin:

$$\mathfrak{u} = p \in H^1_0(\Omega_h)$$

$$\mathfrak{v} = q \in H^1_0(\Omega_h)$$

$$b(\mathfrak{u}, \mathfrak{v}) = (\nabla p, \nabla q) - \omega^2 c^{-2}(p, q)$$

$$l(\mathfrak{v}) = (\mathrm{f}_p, q)$$

Discrete System:

$$[\mathsf{B}][\mathsf{p}] = [\mathsf{f}]$$

- DPG element matrices have ~10x as many DOFs (as Galerkin)
- Can be condensed onto ‘trace’ DOFs on element level:
  - Requires assembling and inverting large dense matrices
  - Condensed system 2x has many unknowns (4x as many non-zeros)
  - But resulting system is Hermitian positive definite

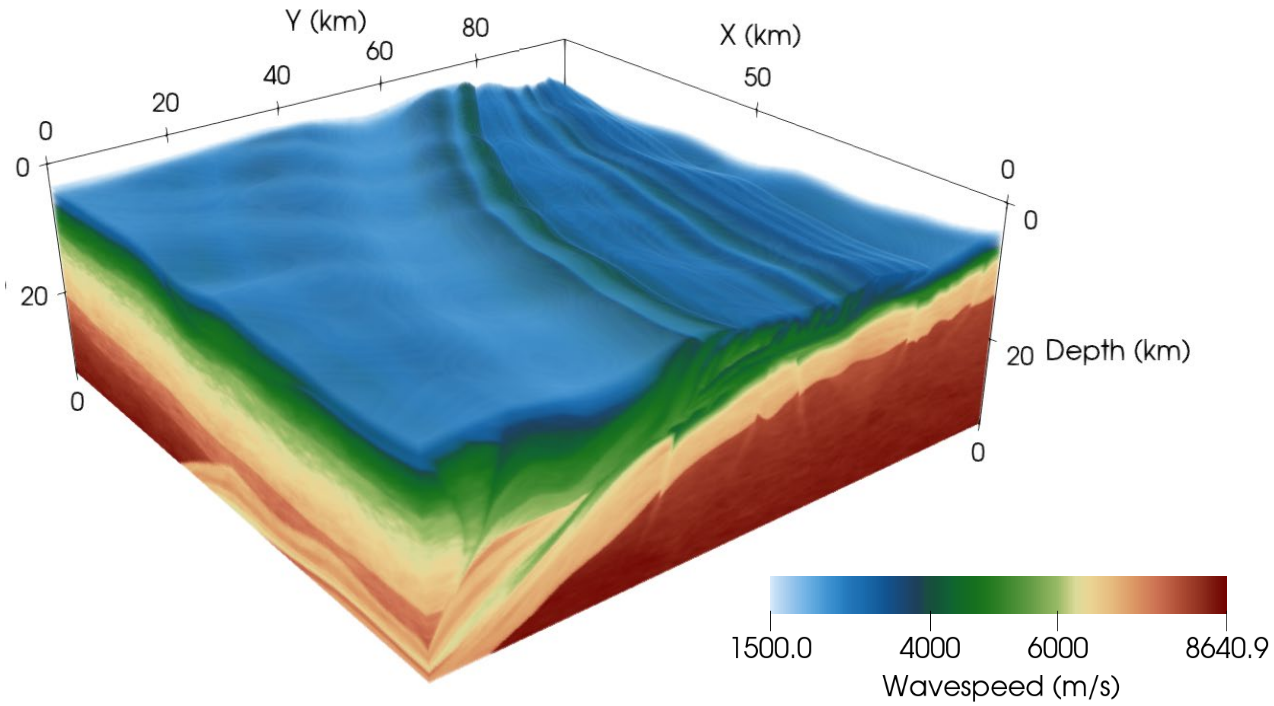
Example: 4th order elasticity

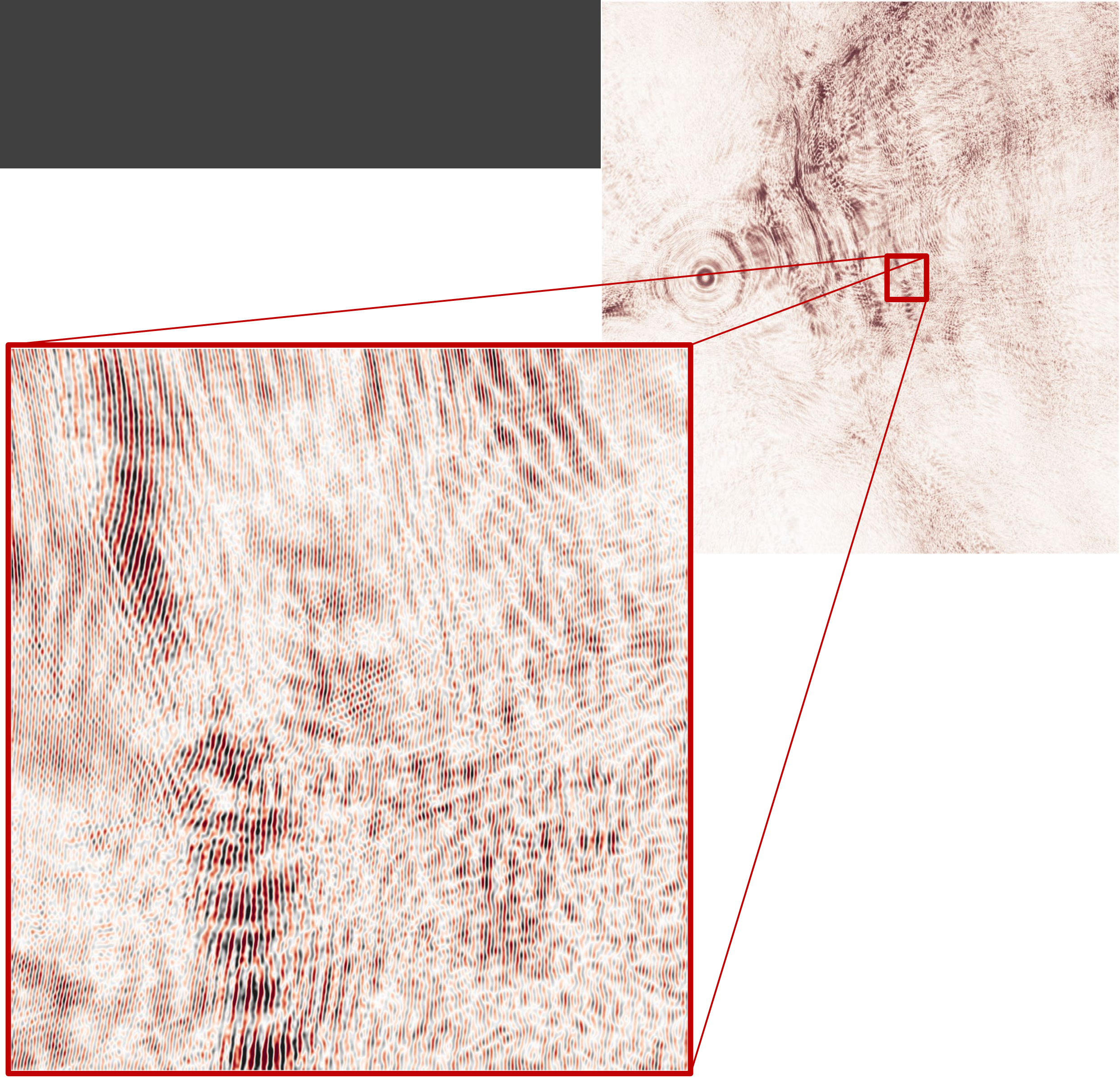
- Assembling blocks of full DPG system: ~0.2s
- Condensing onto trace DOFs: ~0.6s

Assembling a single 4th order element can take ~1s (and scales as p^9)

Frontera-Enabled Outcomes

Scale (GO_3D_OBS Model[1])

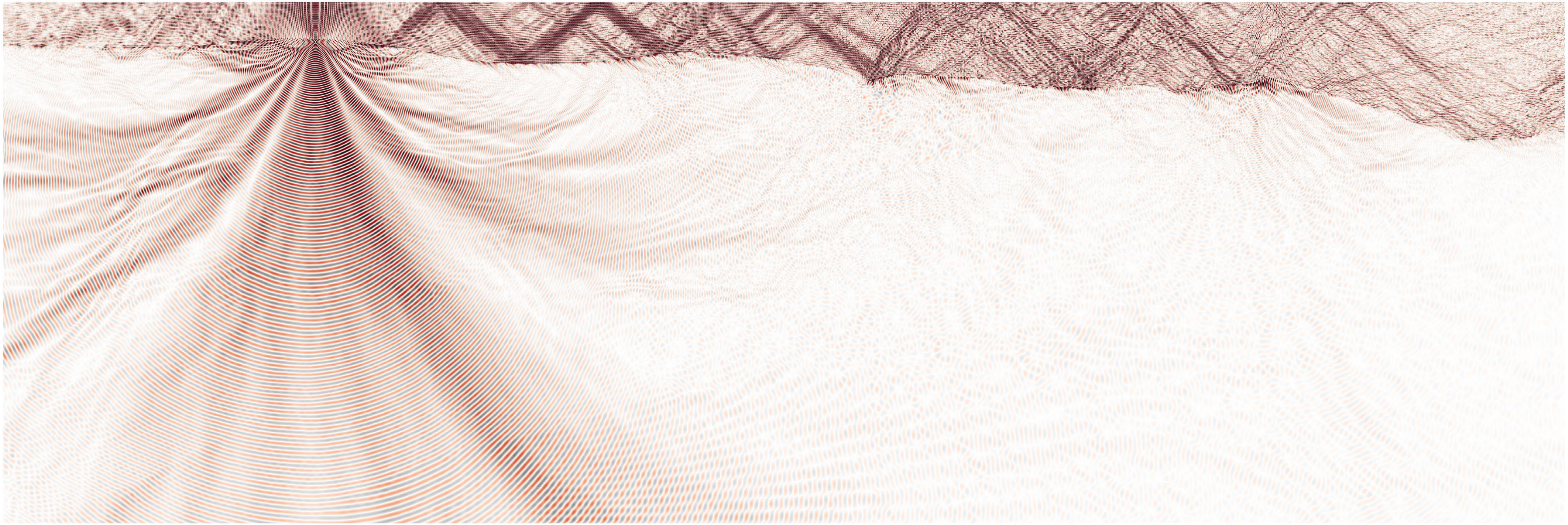




- Visco-acoustic simulation (~6x contrast)
- 15 Hz (1,000 wavelengths)
- 800 billion DOFs (4^{th} order)

[1] Andrzej Górszczyk and Stéphane Operto. GO_3D_OBS: the multi-parameter benchmark geomodel for seismic imaging method assessment and next-generation 3D survey design (version 1.0). *Geosci. Model Dev.,* 14(3):1773–1799, 2021.

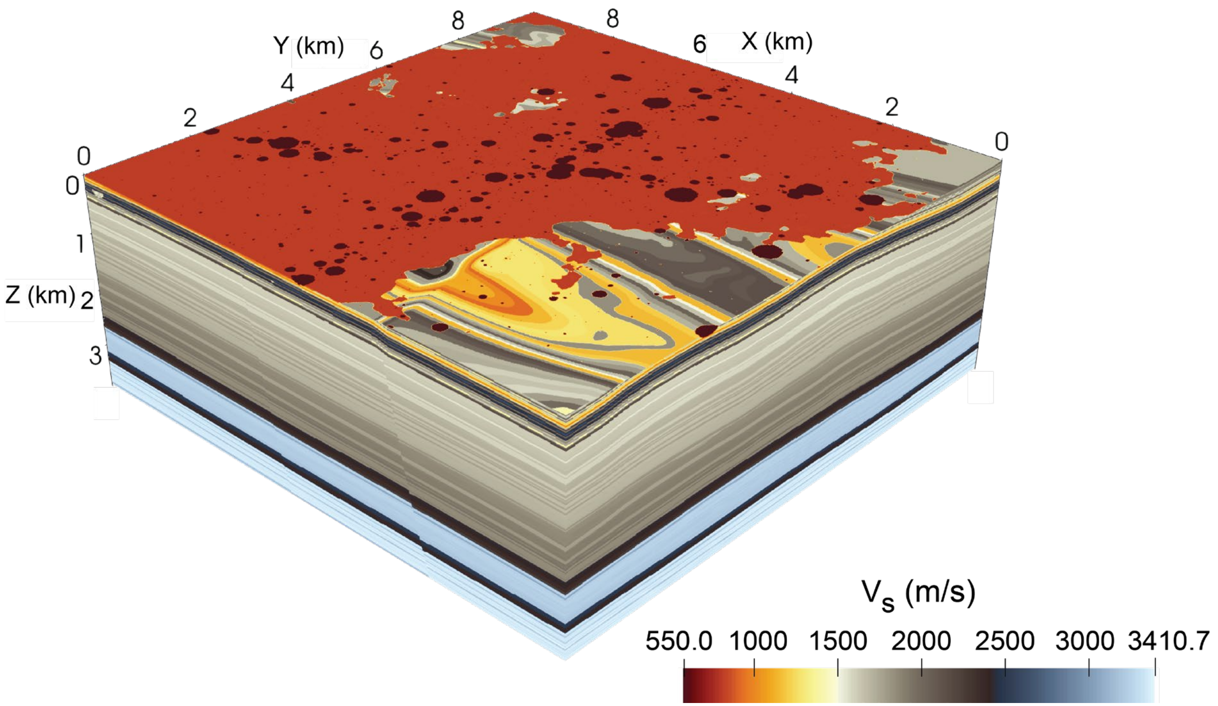
Scale (GO_3D_OBS Model[1])

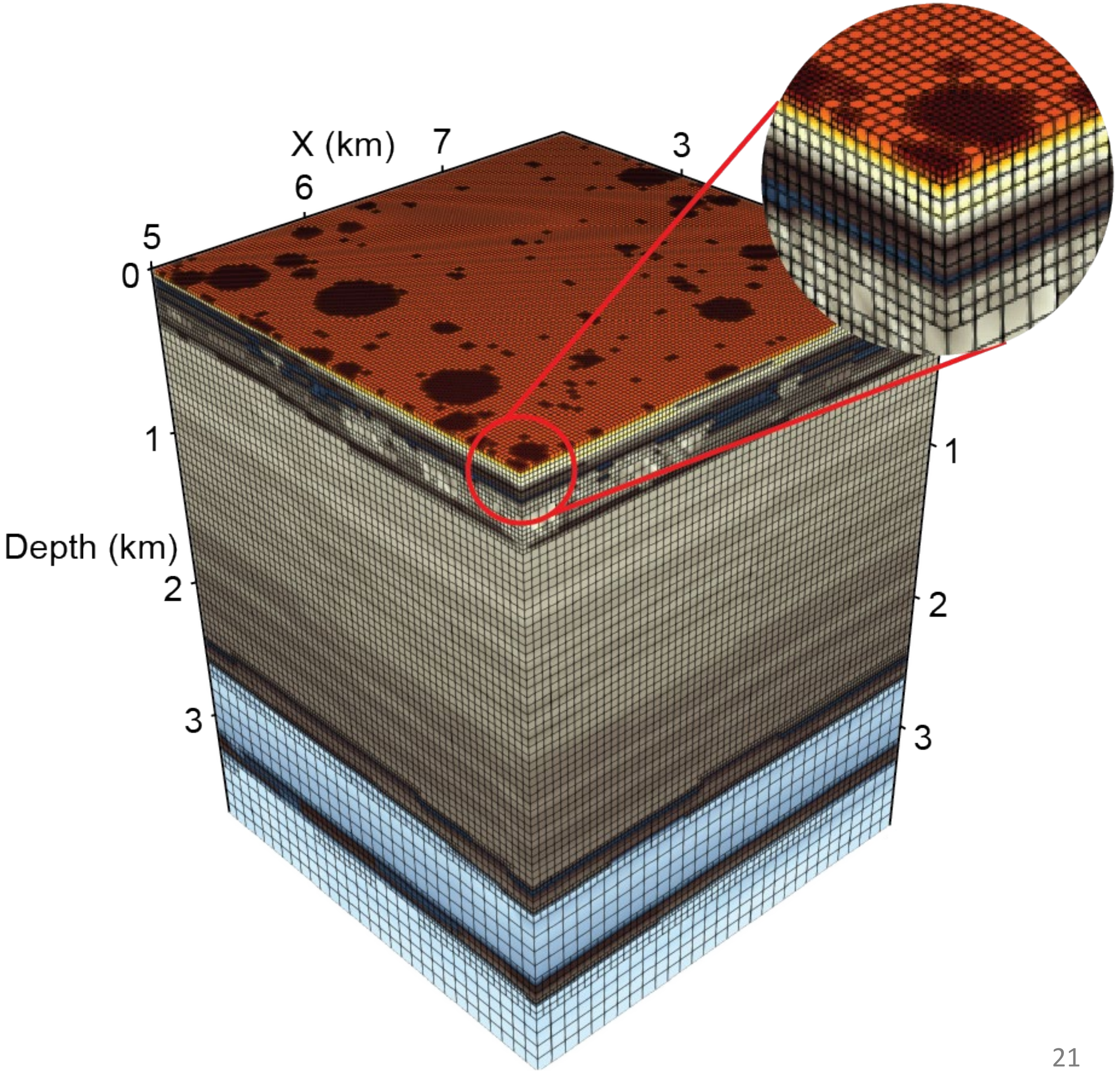


- 460,000 cores (8,192 Frontera nodes)
- ~500x larger than any previous result

[1] Andrzej Górszczyk and Stéphane Operto. GO_3D_OBS: the multi-parameter benchmark geomodel for seismic imaging method assessment and next-generation 3D survey design (version 1.0). *Geosci. Model Dev.,* 14(3):1773–1799, 2021.

Scale (SEAM Arid Model[1])



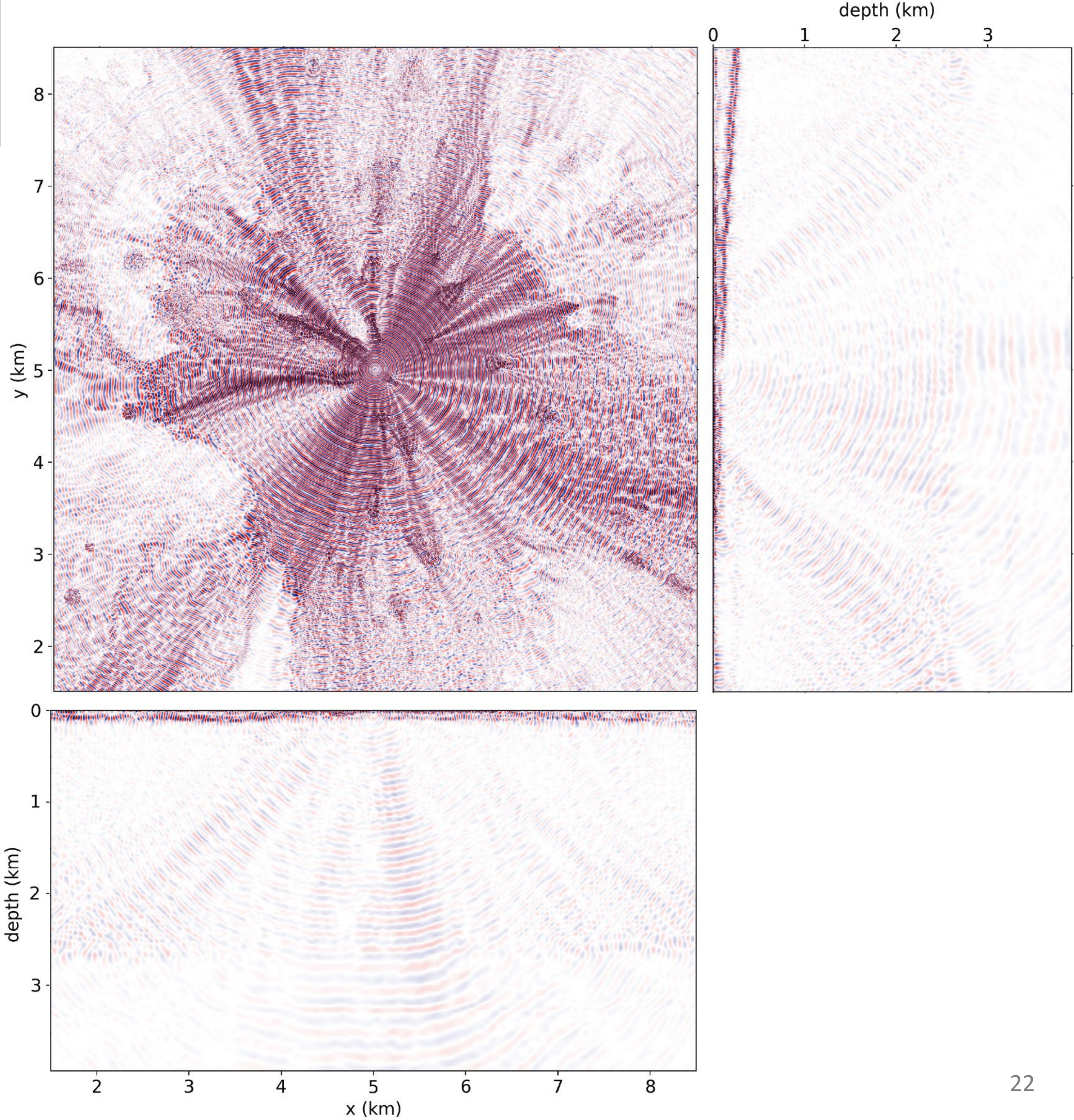


- HTI visco-elastic model (10x contrast)
- 25 Hz (~300 shear wavelengths)
- 7.4 Billion DOFs (~20x reduction vs. uniform mesh)

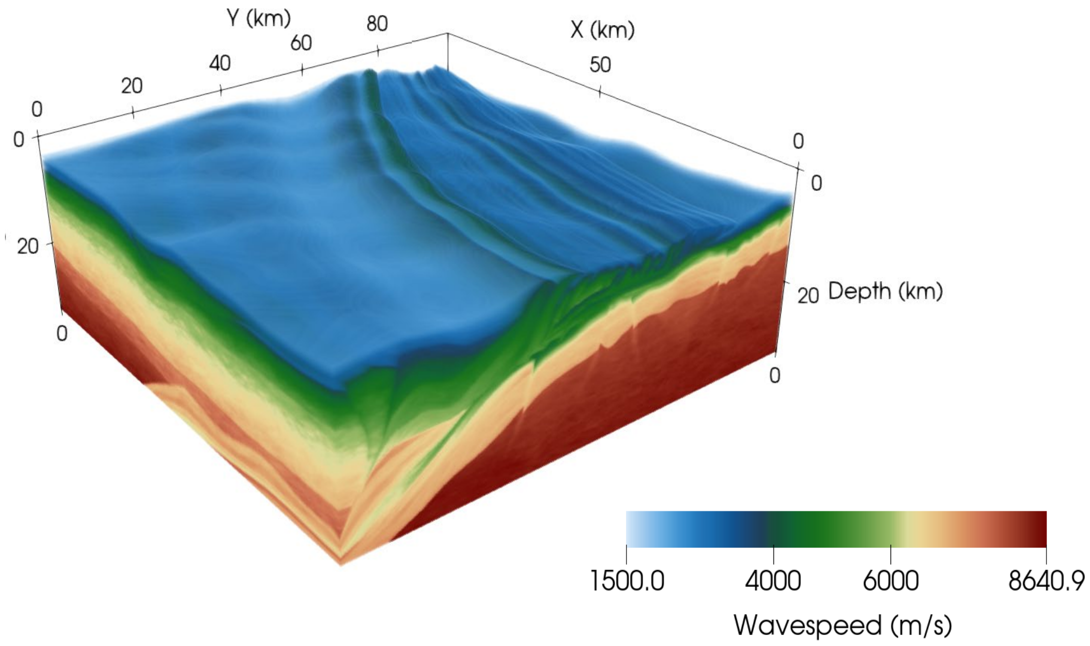
[1] C. Regone, J. Stefani, P. Wang, C. Gerea, G. Gonzalez, and M. Oristaglio. Geologic model building in SEAM Phase II — Land seismic challenges. *The Leading Edge*, 36(9), 738–749, 2017.

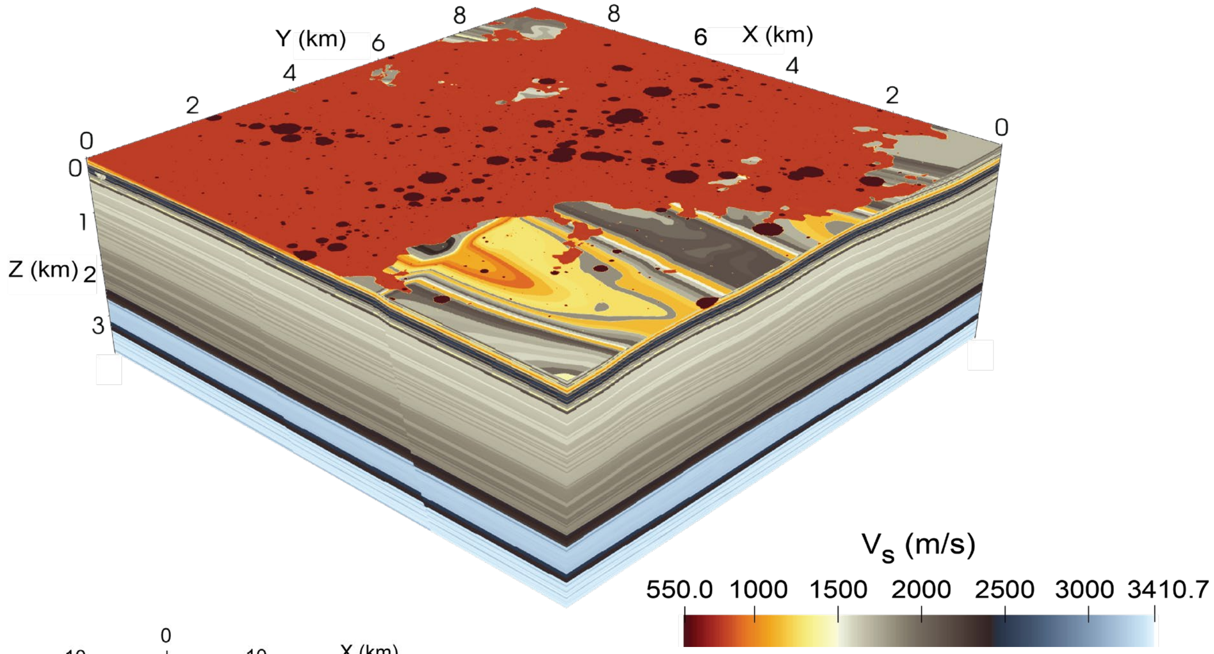
Scale (SEAM Arid Model)

- 14,336 cores (256 Frontera nodes)
- ~20x larger than any previous result
  (on basis of DOFs, for 3D frequency-domain elasticity)



Acknowledgement

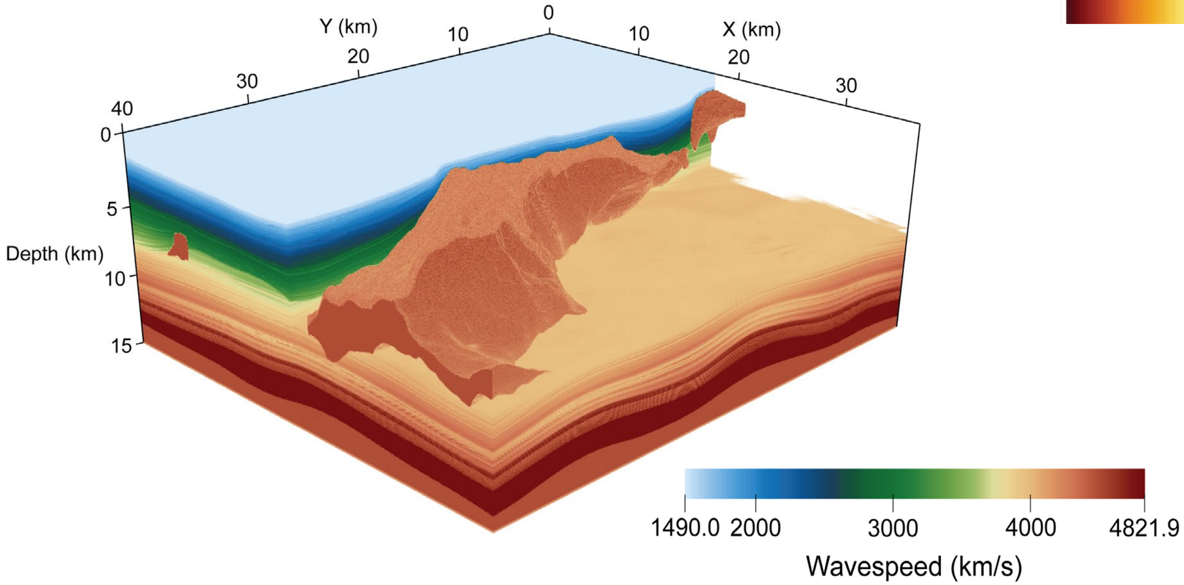




Special thanks to

Franchesca Samsel (TACC),

fantastic colormaps make a huge difference in visualization



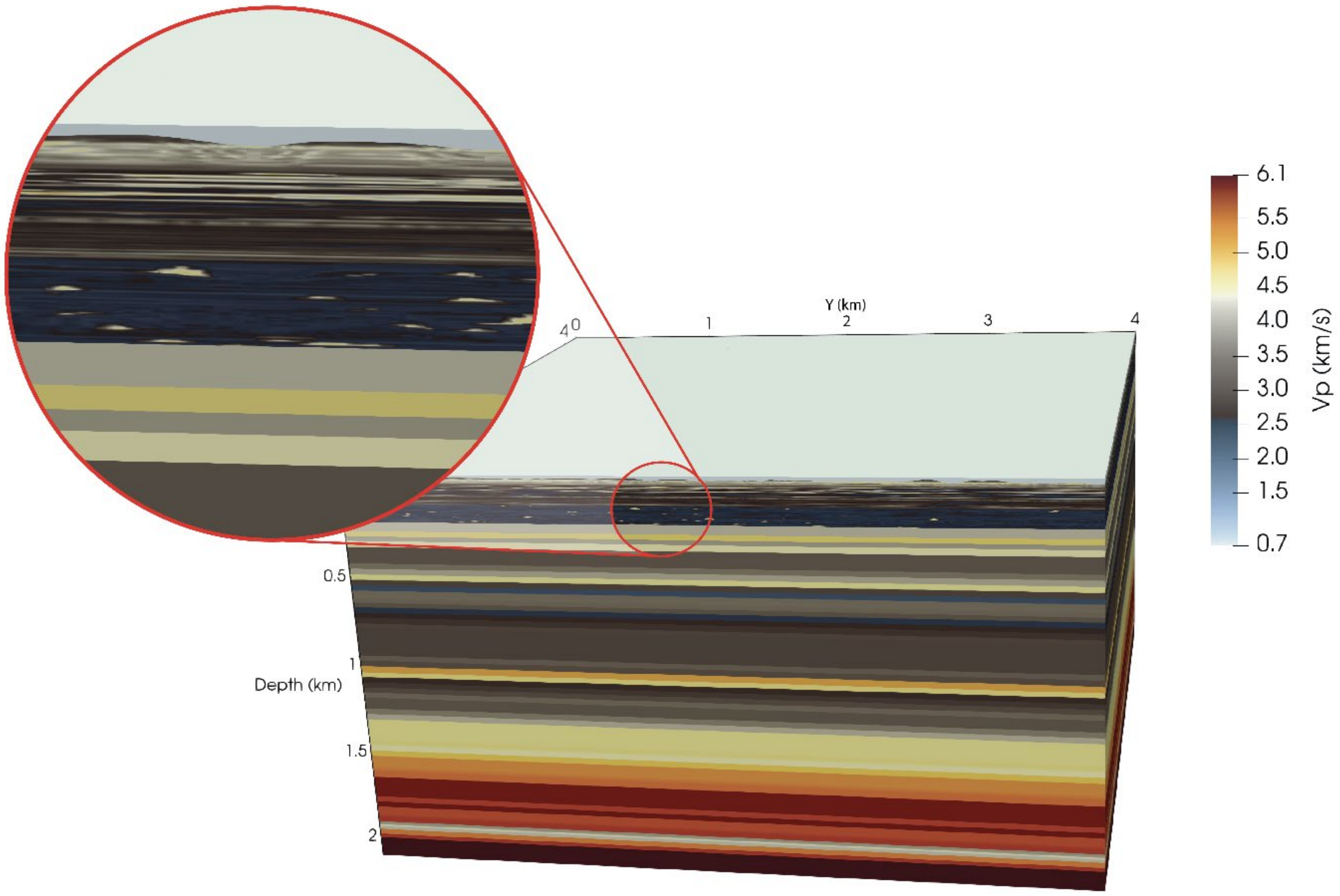
LRAC Allocation

“Efficient seismic simulation in challenging geological environments with a scalable frequency-domain solver”

Directions

1. Efficient & accurate simulation of topography/small-scale heterogeneity
2. Design of distributed acoustic sensing (DAS) installations in complex geological settings
3. In-situ fracture imaging (applications in CCS, engineered geothermal, etc.)

Desert Environment with Buried Topography



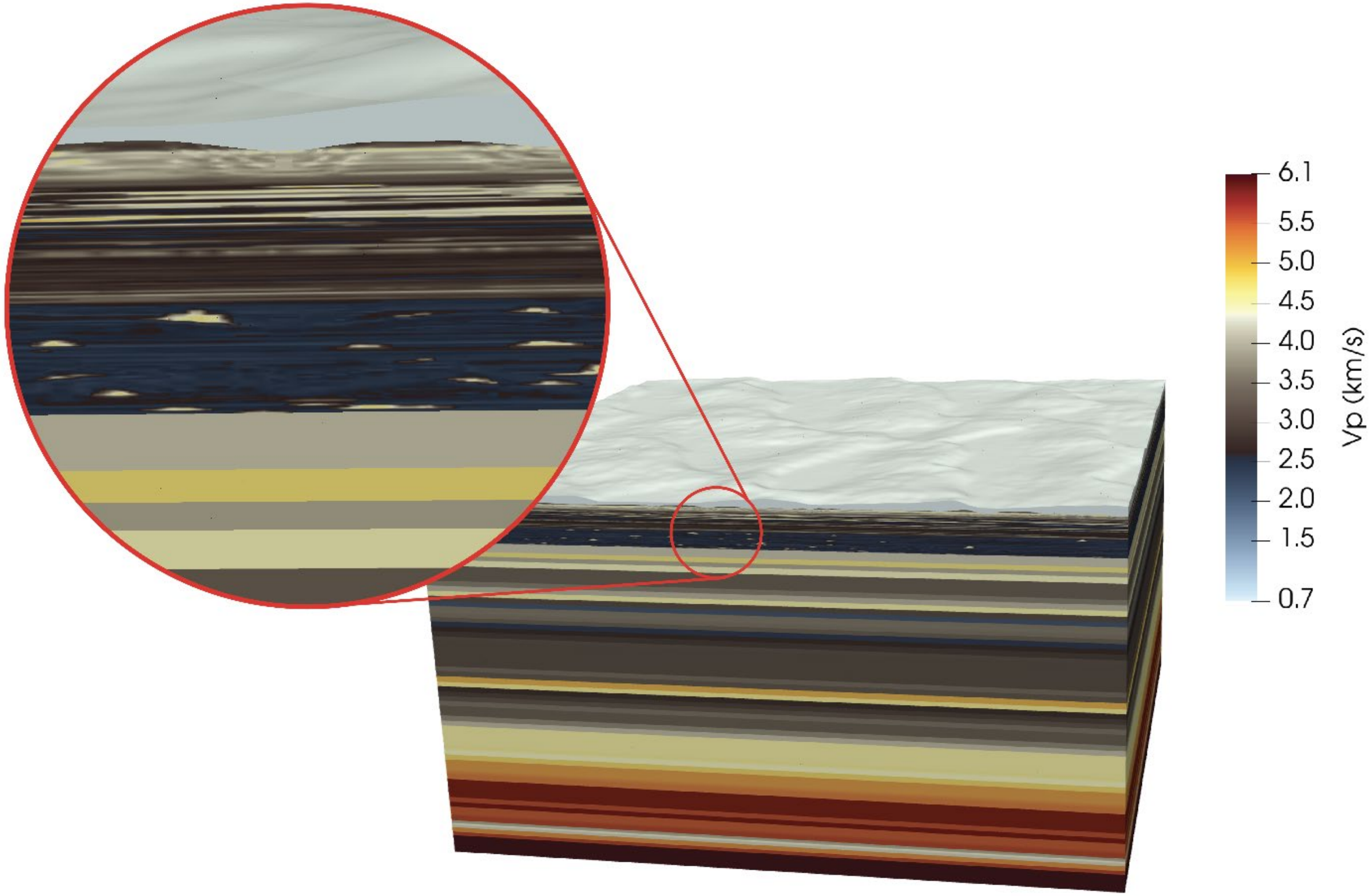
- Visco-acoustic model
- High contrast (ca. 10x)
- Buried topography (flat surface)
- Attenuating near surface (Q_p=40)
- Frequencies 2-32 Hz simulated, then Fourier transformed
  (0.133 Hz spacing, 225 frequencies in total)

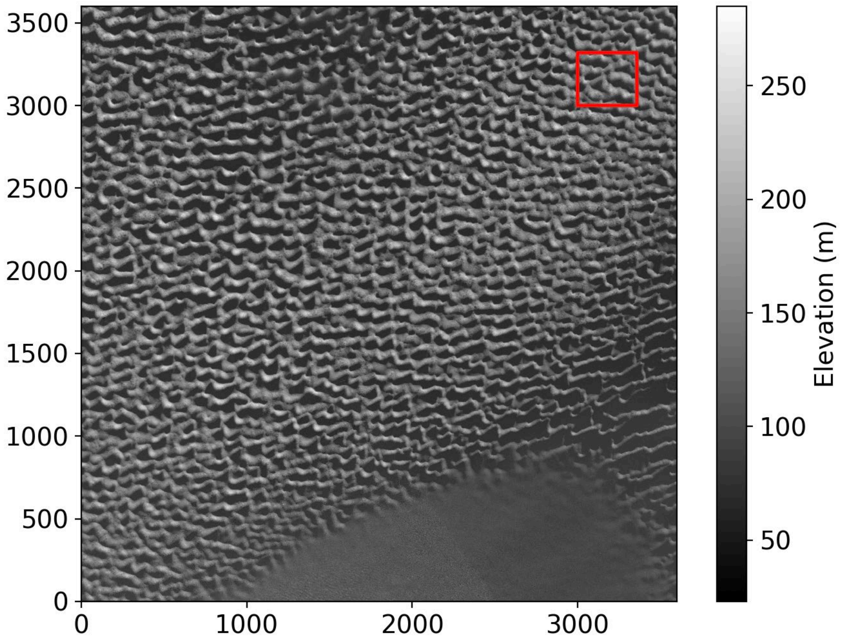
Model Courtesy of Ahmad Ramdani, Saudi Aramco

Flat Sand Model

Desert Environment with Dunes

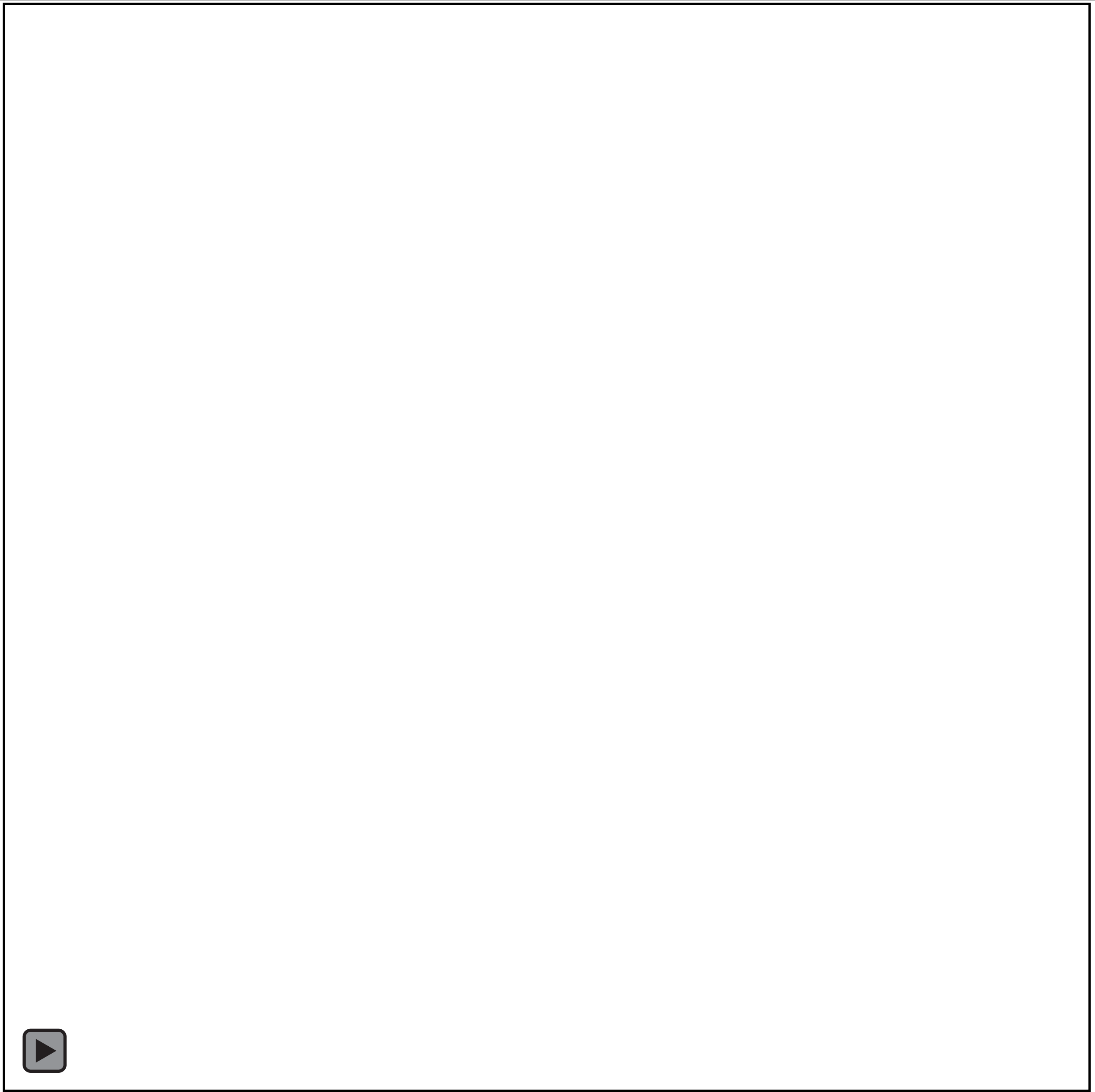
- Dune topography added to sand layer
  (from NASA shuttle radar topography mission)
- Remainder of model unchanged

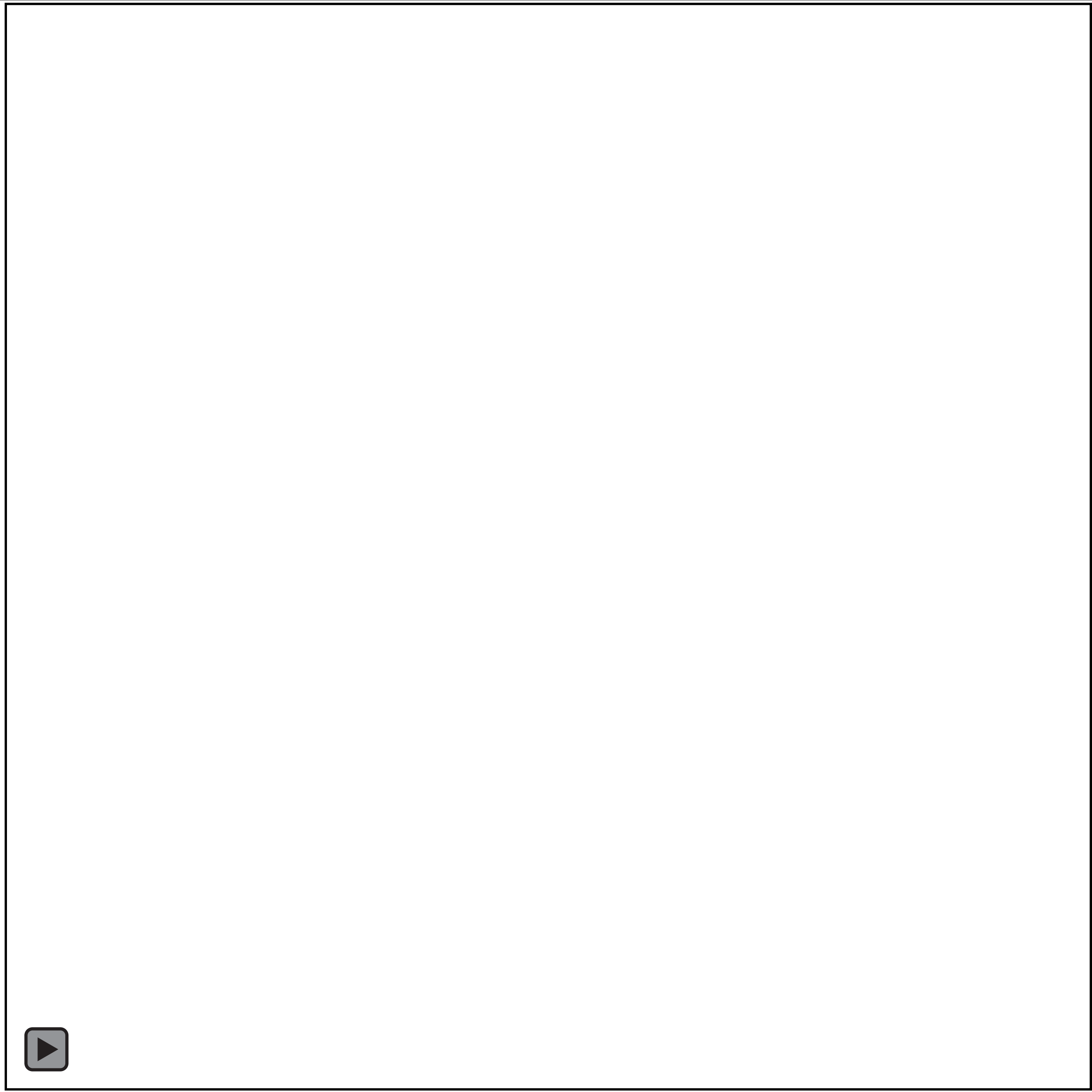




NASA radar tomography mission N22E054.hgt,
selected region boxed

Dunes





Insights

Insights

- Other promising frequency domain solvers include:
  - Domain decomposition (e.g. ORAS)
  - Complex-shifted Laplacian + multigrid
  - Many others
- Leading scalable solvers all have the same $O(\omega^4)$ complexity
- The rest is a game of constants (and DPG starts with a 4x disadvantage)

Insights

- Other promising frequency domain solvers include:
  - Domain decomposition (e.g. ORAS)
  - Complex-shifted Laplacian + multigrid
  - Many others
- Leading scalable solvers all have the same $O(\omega^4)$ complexity
- The rest is a game of constants (and DPG starts with a 4x disadvantage)

Points of Differentiation:

- Dense matrix operations (no sparse matrices)
- Adaptive mesh refinement
- Scale (Frontera + fully distributed data structures)

Sparse is Slow

Example: 4th order DPG acoustics (~200 x 200 matrix blocks)

- **Method 1:** MKL spBLAS
  - 64-bit indices
  - Complex single-precision
- **Method 2:** Element-wise block (dense) operations
  - Complex single-precision

```
for group in elem_groups:

    # gather local block arrays from global array
    x_exp = group.expand(x)

    for block in group:
        off = block.offset()
        n = block.size()

        # Apply block
        Ax_loc[off:off+n] = block.apply(x_loc[off:off+n])

    # Collapse (reduce) onto global array
    Ax = group.collapse(Ap_loc)
```

Method 2: Sparse matrix multiplication via element-wise block operations

Sparse is Slow

Example: 4^{th} order DPG acoustics (~200 x 200 matrix blocks)

- **Method 1:** MKL spBLAS
  - 64-bit indices
  - Complex single-precision
- **Method 2:** Element-wise block (dense) operations
  - Complex single-precision

Results (Frontera CLX node)	
1 core	1.4x
28 cores (OpenMP)	9.1x
28 cores + load batching (16 loads)	21x

Speed up of block-wise matrix multiplication relative to sparse

Reasons:

- Dense blocks are *Hermitian* (Stored in packed or RFP format, then unpacked)
- Sparse indices further double bandwidth per entry (for 64-bit indices w/ complex single precision)
- Less vectorizable (dot vs. outer product)
- Often higher cache miss rate

Sparse is Slow

Sparse Alternatives:

- Matrix-free (partial assembly; CEED, MFEM, etc.)
  - Memory efficient
  - Amenable to modern accelerators
  - Restrictive (tensor-product elements, uniform *p*, etc.)
  - Does ~2-5x as many operations
  - Requires point smoothers

- Storing unassembled dense element matrices (single RHS)
  - More general (hybrid & p-adaptive meshes, etc.)
  - Block smoothers
  - Memory inefficient
  - Fewer operations but bandwidth limited (GEMV)

Sparse is Slow

Sparse Alternatives:

- Matrix-free (partial assembly; CEED, MFEM, etc.)
    - Memory efficient
    - Amenable to modern accelerators
    - Restrictive (tensor-product elements, uniform *p*, etc.)
    - Does ~2-5x as many operations
    - Requires point smoothers

- Storing unassembled dense element matrices (batched RHS)
    - More general (hybrid & p-adaptive meshes, etc.)
    - Block smoothers
    - Fewer operations (GEMM)
    - System memory amortized over RHS's
        - Flexible PCG iteration requires at least 6 arrays per RHS
        - GMRES often requires many more arrays per RHS

- (Variable sized) block sparse matrices
    - In context of finite elements, blocks via nodal interactions

Future Directions & Summary

Summary

- **Scalable solver** + **Frontera** enabled simulation of frequency domain at unprecedented scale
  (~500x larger than any previous result we are aware of)

- **Adaptive meshing** + **performant implementation** can make the frequency-domain approach competitive
  (even in time domain data where hundreds of frequencies required)

- LRAC allocation enabling high-impact research on contemporary challenges:
  - 4D seismic noise and repeatability in on-shore monitoring for carbon capture and storage (CCS)
  - Design of distributed acoustic sensing (DAS) installations
  - In-situ fracture imaging (applications in CCS, engineered geothermal)

Open-source release of dissertation code intended soon

(disclosure completed in April 2024, awaiting signatures & decisions)

Final Thoughts

- **Frequency-domain** approach will be hard to beat in **frequency-sparse** contexts
  - Narrow bandwidth vibratory seismic sources
  - Plasma RF heating
  - Design of micro-optical filters, etc.
  - (and contexts with attenuation)

- Alternative discretizations may further improve efficiency of frequency-domain approach
  - DPG enables use of PCG iteration but is expensive (>100x slower assembly, 4x more non-zero entries vs. continuous Galerkin)
  - Block-wise dense operations and adaptivity largely responsible for surprising performance (neither are limited to DPG)
  - May require re-thinking the role of sparse matrices/factorizations in existing solvers

- **Implicit** time-domain methods + adaptivity may have potential in challenging contexts
  - Current practice dominated by explicit finite difference & spectral element methods:
    1. Less amenable to adaptivity
    2. Time-step limited by smallest element size
  - May require re-thinking the role of sparse matrices/factorizations

- Performant code requires work that is often less-incentivized by academic model
  - Implementation and optimization work can be difficult to publish; often happens in industry/at national labs
  - TACC has been an invaluable resource in bridging this gap