Wavefield migration by PSPI on HPC architectures

Andreas Hadjigeorgiou and Eric Verschuur
ENGAGE workshop on HPC and Data Science

The Cyprus Institute, 22 June 2023

Content

Wavefield migration

Phase Shift Plus Interpolation (PSPI)

Computational Complexity

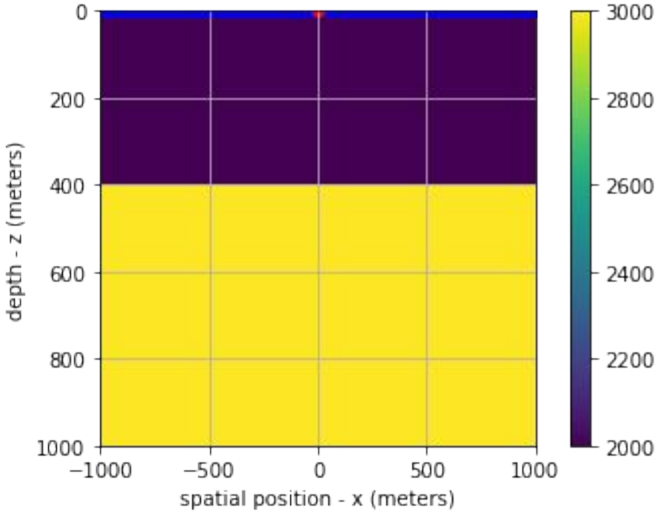
Optimization key-points

Conclusions

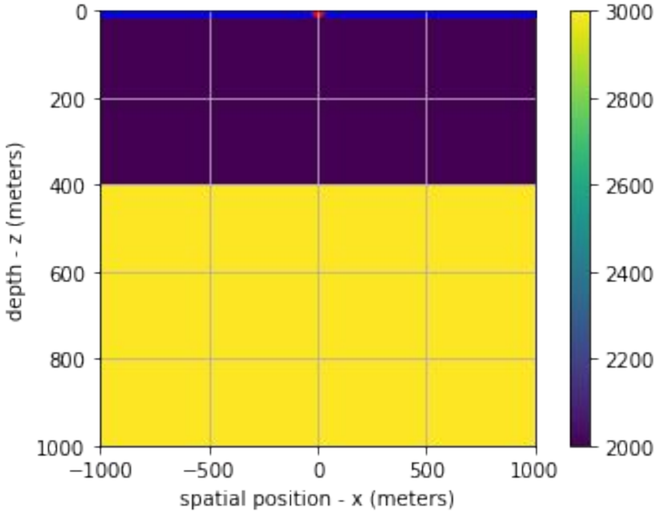
Wavefield Migration

The process of seismic migration involves the application of mathematical algorithms to **position the seismic data** accurately in **space** and **time**, so that the interpreted images align with the true subsurface features. It attempts to **reconstruct the subsurface structures and boundaries**, which can provide valuable insights into the presence and location of geological formations.

Wavefield Migration



Wavefield Migration

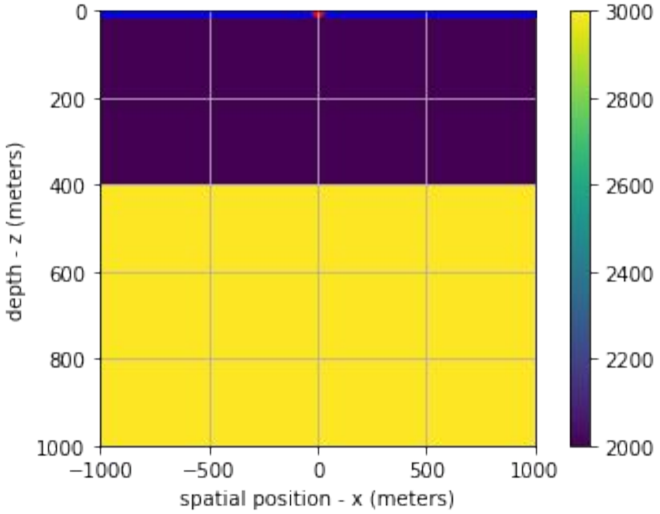


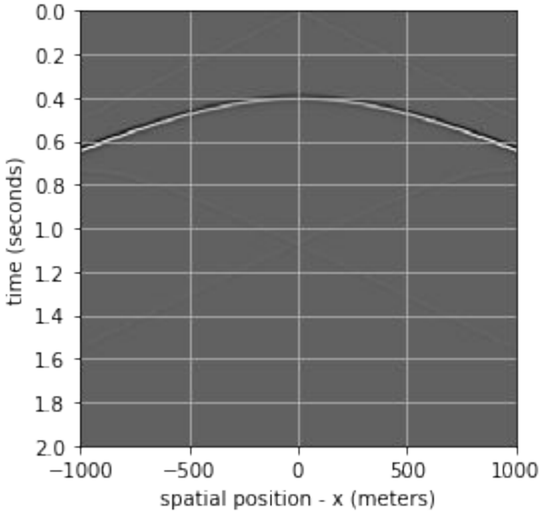
Let us assume we have the model on the left.

We place a source at the center x = 0 m, and place receivers on the surface level z = 0.

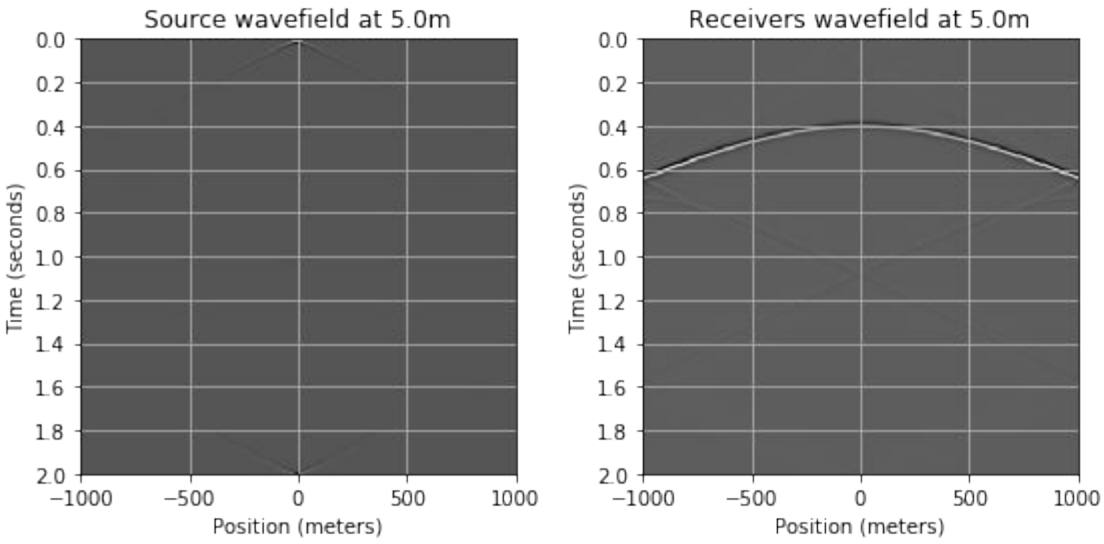
The medium presents a horizontal reflector at depth 400 m due to the velocity contrast from 2000 to 3000 m/s.

Wavefield Migration

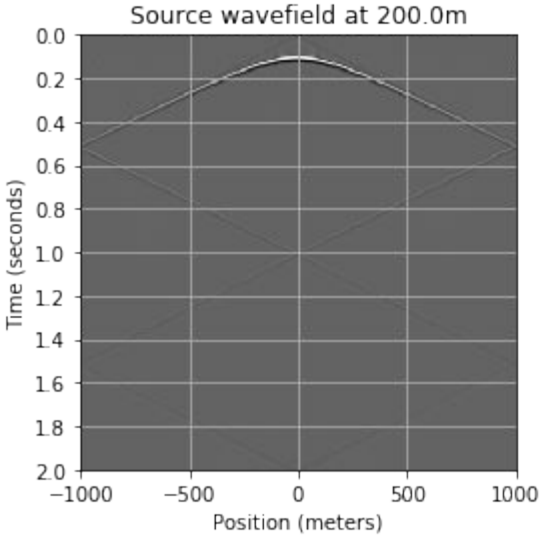


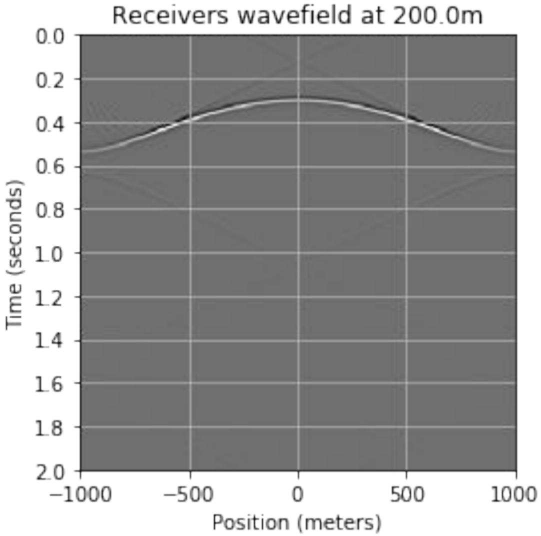


Wavefield Migration

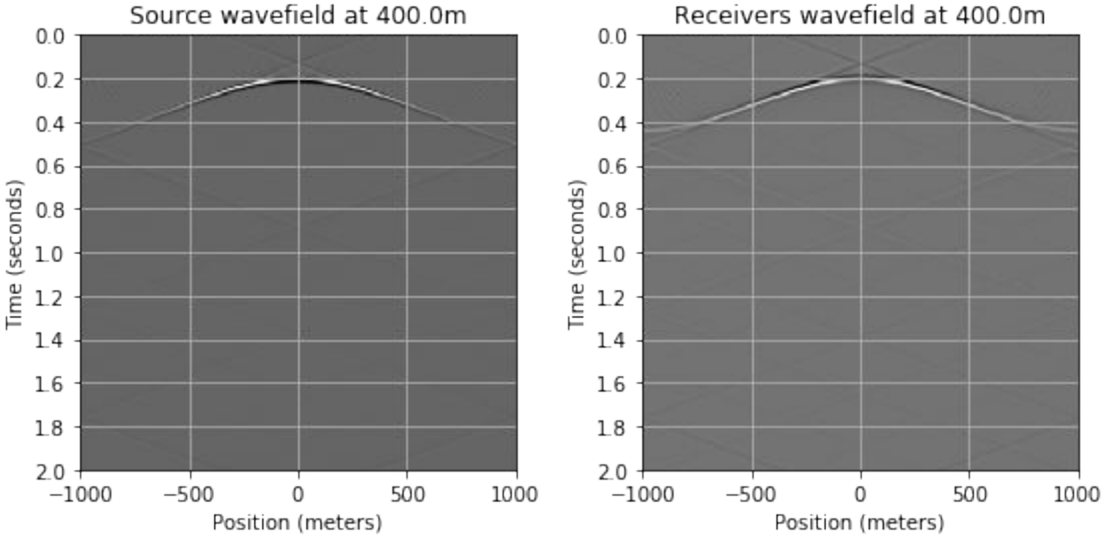


Wavefield Migration

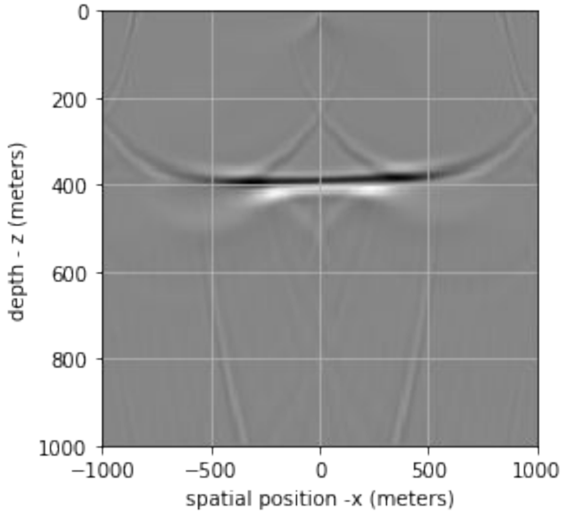




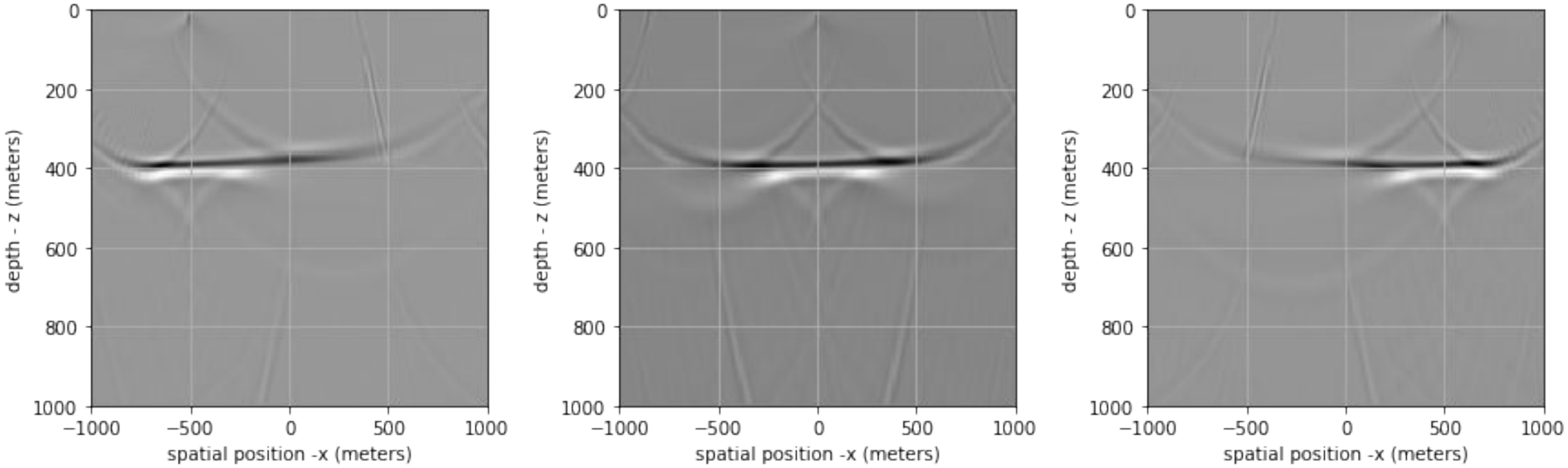
Wavefield Migration



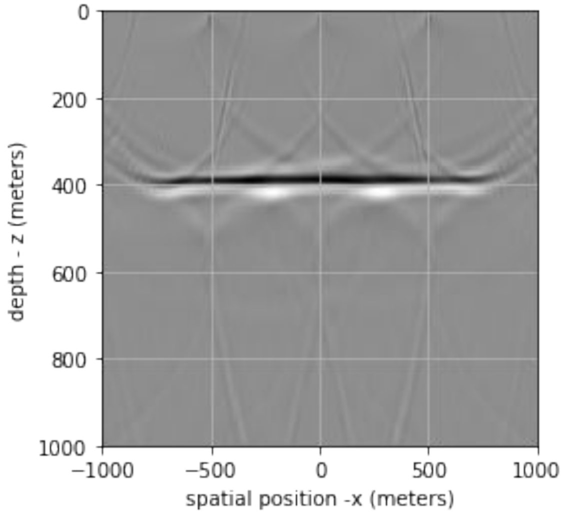
Wavefield Migration



Wavefield Migration



Wavefield Migration



Phase-Shift propagation

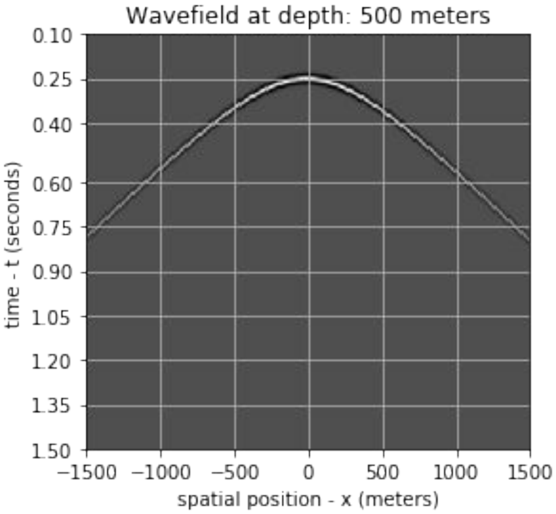
When the velocity model is homogeneous, we transform the wave-equation to the *frequency-wavenumber* domain with solutions of the form:

$$\tilde{P}_{z+\Delta z}(k_x;\omega) = e^{-ik_z \Delta z} P_z(k_x;\omega)$$

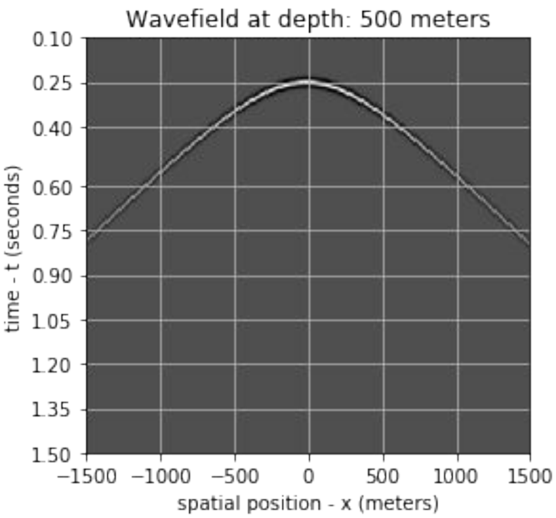
$$k_z = \sqrt{\frac{\omega^2}{c^2} - k_x^2} \quad , \quad \omega > 0$$

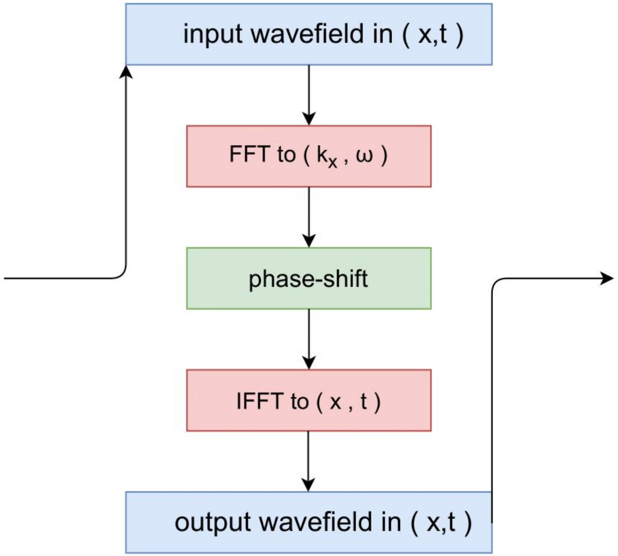
The solution above performs one-way wavefield propagation *forward-in-time*. We can do *backward-in-time* propagation by switching the sign of the exponential term.

Phase-Shift propagation

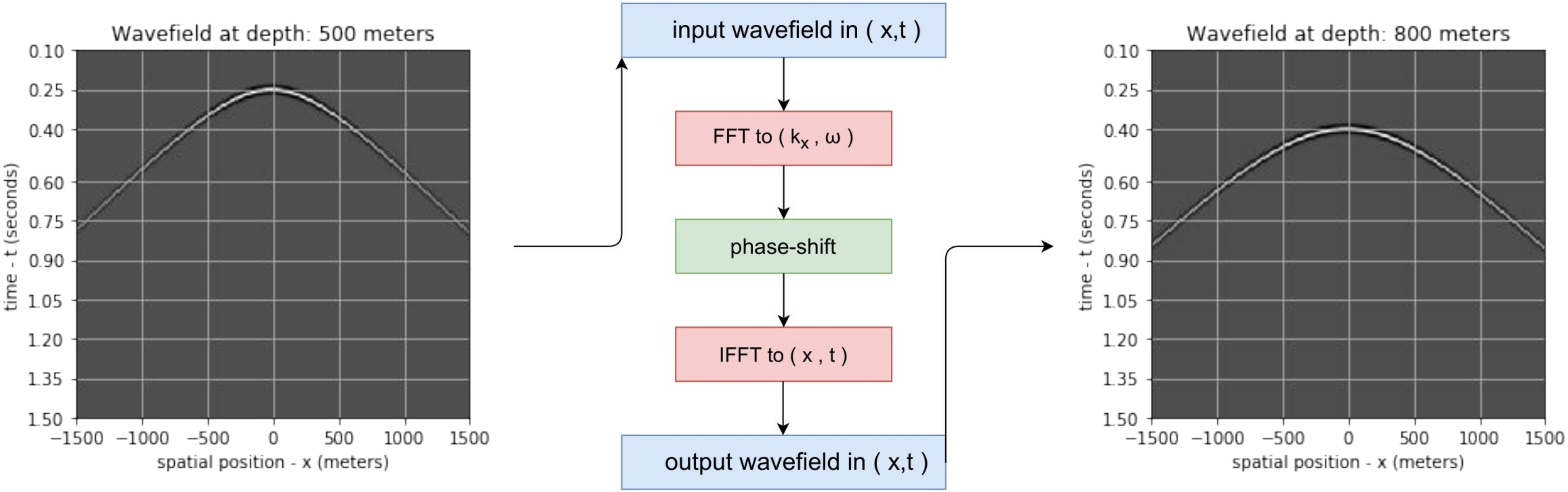


Phase-Shift propagation

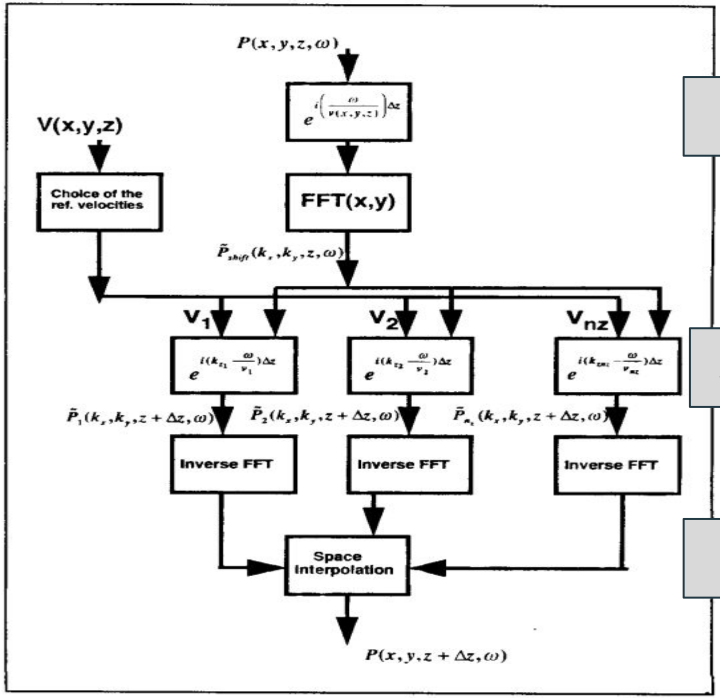




Phase-Shift propagation



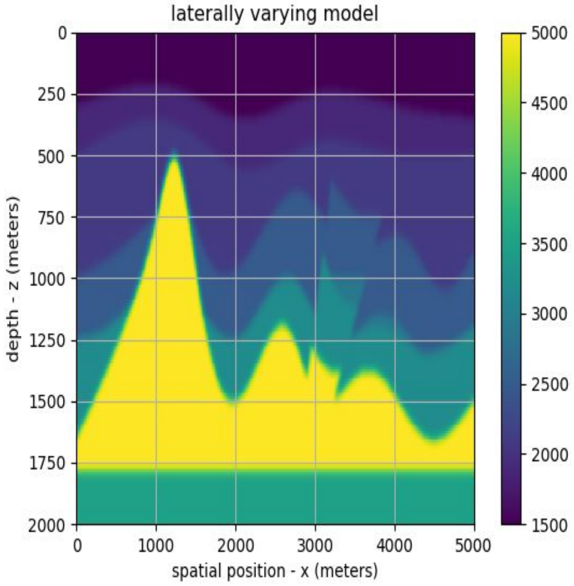
Phase-Shift Plus Interpolation



frequency-space

frequency-wavenumber

frequency-space



Computational complexity

PS_vertical: $N\omega * Nx$
PS_horizontal: $N\mathtt{ref} * N\omega * Nx$
FFT_x: $N\mathtt{ref} * N\omega * Nx * \mathrm{Log2}(Nx)$
Interpolation: $N\omega * Nx$
Selection: Nx

where,
Nx is the number of spatial points in X direction
$N\omega$ is the number of frequencies
$N\mathtt{ref}$ is the number of reference velocities

	6	Bagaini	8	10	12	14	16
FFTs	1.22	1.35	1.59	1.94	2.29	2.66	3.01
PS_h	0.21	0.25	0.27	0.35	0.45	0.54	0.66
Interp.	0.05	0.05	0.05	0.05	0.05	0.05	0.05
PS_v	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Sele.[1]	0.01	0.04	0.01	0.01	0.01	0.01	0.01
Total	1.52	1.73	1.96	2.39	2.84	3.30	3.77

Run-time of the individual kernels for different number of reference velocities. “Bagaini” denotes a statistical approach to the selection of velocities that aims to pick denser close to velocity distribution maxima (<6.7>).

Optimization key-points

- Selection of reference velocities using the statistical entropy of velocity distribution
- Batched FFTs
- Table-driven approach
- Predefine optimal data-layout

Batched FFTs

State-of-the-art HPC libraries for FFTs work based on plans:

1. The plan is created based on the **type** of FFT, the **precision** of the data, the **layout** of data in memory, etc.
2. The plan is executed (actual computation)
3. The plan is destroyed

High-level languages and programming tools usually hide the implementation details; unknown what the underlying implementation does.

We want to have an explicit control over this for performance reasons!

Batched FFTs: Overhead of create/destroy

- **Batched FFTs using cuFFT library**

Algorithm 1:
loop over N
- create plan(1d FFT length NX)
- execute plan
- destroy plan

Algorithm 2:
- plan N x (1d FFTs of length NX)
- execute plan
- destroy plan

- NVIDIA recommends to use batched FFTs for higher performance, so it is expected Algorithm 1 to be slower : **Alg. 1 -> 48s, Alg. 2 -> 8.2s** !!!

- create/destroy plan -> GPU memory allocation/free -> synchronization

Table-driven approach

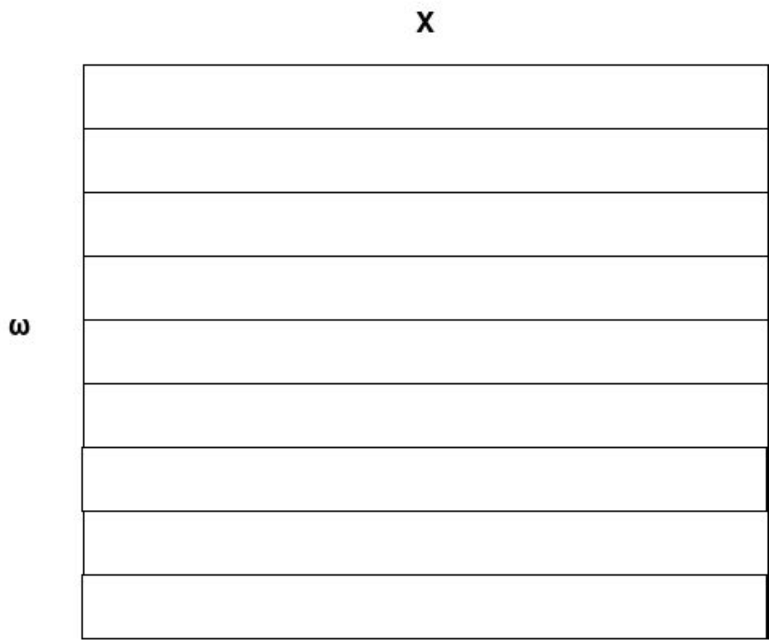
The table-driven approach aims at improving run-time by reducing the number of computations associated with the phase-shift horizontal kernel, which is the second most expensive part of the PSPI algorithm.

$$\tilde{P}_r(x,\omega) = \tilde{P}(x,\omega)\ e^{\pm i\left(k_{z_r} - \frac{\omega}{u_r}\right)\Delta z}$$

$$k_{z_r} = \sqrt{\frac{\omega^2}{u_r^2} - k_x^2}$$

We compute a set of operators that correspond to values $k_r = \frac{\omega}{u_r}$ from 0 to a *kmax*, and during propagation the most suitable is retrieved and applied directly. *--> Up to 40 % speed-up!*

Data-layout of wavefield



Data are:

1. contiguous across X
2. strided by *Nx* points across ω

The layout enables:

1. Batched FFTs using state of the art libraries
2. Develop custom implementations for the other routines optimally

Optimization in the context of HPC starts with defining a suitable data-layout!

Migration by PSPI

For each depth **Z**:

For each source **S**:

- Propagate by PSPI the source wavefield forward in time
- Propagate by PSPI the receivers wavefield backward in time
- Apply imaging condition (IC)

Cross-correlation IC: $$I_{z,s}(x) = Re\left\{ \sum_{\omega} \tilde{P}^{+}(x,\omega)\ \tilde{P}^{-}(x,\omega)^{*} \right\}$$

Assignment

1. Implement the **cross-correlation imaging condition** in:
   a. C-OpenMP,
   b. C-CUDA

2. Remove the direct response from the seismic data
3. Perform imaging using more sources

https://github.com/ahadji05/ENGAGE_workshop_seismic_imaging