pyseis: a high-performance, user-friendly Python package for GPU-accelerated seismic modeling and subsurface imaging

Stuart Farris, Guillaume Barnier, Ettore Biondi, and Robert Clapp, Stanford University*

SUMMARY

pyseis, a high-performance Python library, harnesses the power of low-level CUDA and C++ for computation, and provides a user-friendly Python interface for wave-equation modeling and inversion. It incorporates acoustic and elastic finite-difference solvers, adjoint state-based gradients, and seismic imaging and inversion support. Speed benchmarks, along with a sample FWI implementation, showcase the robustness of pyseis. Moreover, its capabilities are exemplified in tackling complex seismic problems, from elastic FWI and FWI by Model Extension to machine learning applications in seismic inversion. pyseis serves as a versatile and efficient tool for geophysicists, enabling them to perform intricate seismic computations with ease. The code is available at http://zapad.stanford.edu/sfarris/pyseis.

INTRODUCTION

Wave equation operators form the cornerstone of advanced geophysical studies and applications, serving as essential components in full waveform inverse problems, imaging algorithms, and the creation of machine learning datasets. The intricacy and multidimensional nature of such problems necessitate computational operators that can accurately model wave propagation within the Earth's subsurface. The evolution of computational geophysics, combined with the growing demand for more sophisticated and computationally intensive algorithms have increased the need for comprehensive software packages that possess the ability to handle large-scale geophysical problems.

As geophysical problems grow in size and complexity, so too does the demand for high-performance computing (HPC). Large-scale wave equation modeling (Biondi et al., 2023), extended waveform inversion algorithms (Barnier et al., 2023a,b), and the generation of sizeable machine learning datasets (Farris et al., 2022) all call for formidable computing power. The manipulation of massive datasets and the execution of complex operations present significant challenges, as they entail substantial memory bandwidth and operation throughput. Existing computational resources may struggle to meet these demands, resulting in an escalating need for innovative solutions that can leverage advanced computing technologies, such as graphical processing units (GPUs), to enhance performance and streamline computational tasks.

However, the pursuit of computational power should not eclipse usability. A user-friendly software package can substantially reduce the learning curve and development time, thereby maximizing the efficiency and productivity of researchers and professionals in the field. Such software would ideally offer a simplified interface that abstracts complex details, while still providing users with the flexibility and control necessary for executing advanced geophysical algorithms and computations. The rise of Python as a preferred language in the scientific community, due to its simplicity and versatility, further underlines the importance of developing software with usability at the forefront.

Presently, existing packages such as SPECFEM and Devito offer various capabilities for addressing wave-equation-based geophysics problems (Peter et al., 2011; Luporini et al., 2020; Louboutin et al., 2019). However, these solutions have their limitations. SPECFEM, while comprehensive, can be challenging to use and lacks the user-friendly nature of Python. Devito, on the other hand, offers a Python interface but lacks out-of-the-box multi-GPU acceleration needed for large-scale, high-performance computations. In both instances, these packages do not wholly satisfy the trifecta of high-performance computing, user-friendly interfaces, and full-waveform capabilities. Therefore, an emerging need is identifiable for a software package that can effectively bridge this gap, harmonizing the benefits of computational power, usability, and advanced wave-equation capabilities in a single, open-source platform. The introduction of pyseis is proposed as a response to this need, signifying an important stride towards addressing these unfulfilled requirements in the sphere of computational geophysics.

OVERVIEW

pyseis is a cutting-edge, open-source software package that seamlessly integrates the power of high-performance computing, the user-friendliness of Python, and the intricacy of wave-equation operations.

pyseis leverages the computational capabilities of GPUs through Cuda wave equation kernels, thereby enhancing operation throughput and memory bandwidth significantly. The software makes available two wave equations to the user: the acoustic, constant-density isotropic wave equation and the elastic, isotropic wave equation. These equations are implemented in both two and three dimensions and come equipped with their linearized forward and adjoint forms, which are indispensable for gradient-based inversion problems. Furthermore, pyseis can parallelize wavefield simulations across multiple GPUs, thereby augmenting its computational capabilities.

The software package is designed with an emphasis on usability, offering a Python API using pybind11 to wrap the Cuda kernels (Jakob et al., 2017). This feature allows users to interface exclusively with GPU-accelerated code through Python, thereby benefiting from the simplicity and prototyping capabilities of this versatile language. pyseis also simplifies experimentation by providing an object-oriented wave equation solver class, which abstracts complex details such as stability, dispersion, and boundary conditions.

As a holistic solution, pyseis also integrates with the Ocammypy inversion library, enabling it to employ linear and nonlinear optimization schemes for a variety of geophysical research problems (Biondi et al., 2021). This includes advanced algorithms such as full waveform inversion, least-squares reverse time migration, and wave equation migration velocity analysis.

WAVE EQUATION OPERATORS

The foundational elements of the pyseis package are the acoustic, constant-density isotropic wave equation and the elastic, isotropic wave equation. Both equations are crucial in exploration seismology and provide the basis for simulating seismic wave propagation in the Earth's subsurface.

The elastic isotropic wave equation is represented by:

$$\begin{aligned}\rho(\mathbf{x})\frac{\partial v_i(\mathbf{x},t)}{\partial t} &= \frac{\partial \sigma_{ik}(\mathbf{x},t)}{\partial x_k} + f_i(\mathbf{x},t),\\ \frac{\partial \sigma_{ij}(\mathbf{x},t)}{\partial t} &= \lambda(\mathbf{x})\frac{\partial v_k(\mathbf{x},t)}{\partial x_k}\delta_{ij}\\ &\quad + \mu(\mathbf{x})\left[\frac{\partial v_i(\mathbf{x},t)}{\partial x_j} + \frac{\partial v_j(\mathbf{x},t)}{\partial x_i}\right] + m_{ij}(\mathbf{x},t),\end{aligned} \tag{1}$$

where $\mathbf{x}$ and t denote spatial and temporal coordinates, δ_{ij} is the Kronecker delta, ρ is the density, λ is the first Lamé parameter, μ is the shear modulus, v_i is the particle velocity, and σ_{ij} is a component of the symmetric stress tensor (Aki and Richards, 2002). The source terms, f_i and m_{ij}, represent a volumetric force field and the moment tensor's time derivative, respectively. This equation assumes the Earth behaves as an elastic isotropic medium, thus enabling more realistic synthetic data generation albeit at the cost of greater computational demand.

Alternatively, the acoustic isotropic constant density wave equation simplifies the wave physics by treating the Earth as a homogeneous acoustic medium. This equation is expressed as:

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 p(\mathbf{x},t)}{\partial t^2} - \nabla^2 p(\mathbf{x},t) = s(\mathbf{x},t), \tag{2}$$

where v denotes the seismic wave velocity, p is the pressure, and s is the acoustic source function. Although this equation is more computationally efficient, it foregoes elastic effects, leading to a potentially less accurate portrayal of complex geological structures. A summary of finite difference details pertaining to each wave equation is presented in Table 1.

Both wave equations also come equipped with their linearized, adjoint operators, which pass the dot product test to double precision. Adjoint operators are essential for gradient-based inversion problems as they offer a computationally efficient method for calculating gradients of objective functions with respect to model parameters. The dot product test involves the computation of $\mathbf{y}'(\mathbf{F}\mathbf{x})$ and $(\mathbf{F}'\mathbf{y})'\mathbf{x}$ with randomly initialized vectors $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{F}$ and $\mathbf{F}'$ are the forward and adjoint operators, respectively. If the subroutines for $\mathbf{F}$ and $\mathbf{F}'$ are truly adjoint, these quantities should be computationally equal. Passing this test is a testament to the validity of the implemented adjoint operators in pyseis.

	Acoustic	**Elastic**
Temporal derivative	Second order	First order
Spatial derivative	Eighth order	Eighth order
Spatial Grid	Centered	Staggered
Boundary Condition	Absorbing	Absorbing
Free Surface Condition	in 2D & 3D	in 2D

Table 1: Overview of the finite difference implementation for the acoustic and elastic wave equations in pyseis.

IMPLEMENTATION

pyseis is engineered to combine the computational prowess of lower-level languages with the user-friendly nature of higher-level ones. This optimal blend enables the simultaneous achievement of high computational performance and a user-friendly interface.

The backbone of pyseis's computation is the implementation of CUDA finite difference kernels, which leverage the parallel processing capability of GPUs to solve the wave equations with high efficiency. These CUDA kernels are designed to optimize the GPU's memory hierarchy, which leads to an accelerated execution of the finite difference computations for wave propagation. The large problems, advanced algorithms, and sizable ML datasets, which would have otherwise been computationally heavy, can thus be tackled with relative ease.

To launch these CUDA kernels, C++ is used. Given the language's close-to-the-hardware nature and high performance, it is an ideal candidate to handle the orchestration of the CUDA kernels. Furthermore, pyseis uses an approach to parallelize the wavefield simulations over GPUs, specifically for shot-record calculations in seismic surveys. By handling multiple source simulations concurrently on separate GPUs, it is possible to drastically reduce the time required for extensive surveys.

However, the utilization of these low-level languages can pose challenges for users who are not familiar with them or for those who wish to use pyseis as a part of larger workflows often implemented in higher-level languages. To bridge this gap, pyseis leverages Pybind11, a lightweight header-only library that exposes C++ types in Python and vice versa. This allows pyseis's core functionalities, written in CUDA and C++, to be linked with Python.

Consequently, users can interact with pyseis through a Python interface. Python's ubiquity in scientific computing and its intuitive syntax make it an ideal interface for pyseis. It allows users to leverage pyseis's capabilities without the need for deep knowledge of CUDA or C++. This compatibility with Python further means that pyseis can be easily integrated into existing Python-based geophysics workflows, thus enhancing its usability.

SPEED BENCHMARK

To demonstrate the efficiency and robustness of `pyseis`, a series of speed benchmarks were performed on a single NVIDIA V100 GPU. In these tests, we kept the number of time steps constant while varying the size of the model domain, a realistic scenario where high-performance computing becomes critical as domain size increases.

The performance of the nonlinear forward wave propagation operators was quantified using the rate of processing, r_{proc}, which is defined as:

$$r_{proc} = \frac{n_m \times n_t}{\Delta t_{proc}} \tag{3}$$

where n_m is the number of voxels in the domain, n_t is the number of time steps, and Δt_{proc} is the total processing time. In 3D, n_m is given by $n_y \times n_x \times n_z$, and in 2D, $n_m = n_x \times n_z$. Throughout the speed benchmarks, we maintained $n_t = 2400$ to simulate a realistic seismic acquisition scenario.

The results of these tests are graphically presented in Figure 1. It is clear that while the acoustic wave equation is computationally more efficient due to its simplifications, the elastic wave equation exhibits commendable performance given its inclusion of more complex physics. Notably, even under these more demanding conditions of 3D wave propagation, `pyseis` maintains a high r_{proc}, underscoring its suitability for large-scale, high-resolution geophysical problems.

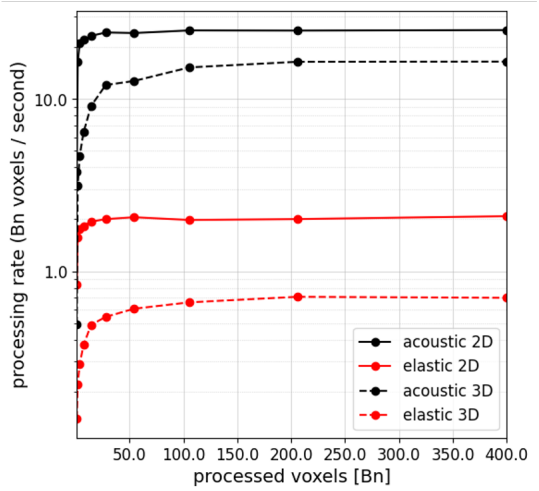

Figure 1: The benchmark processing rates, r_{proc}, for the 2D and 3D wave equations of `pyseis`, with results for both the acoustic and elastic wave equations. The number of processed voxels, $n_m \times n_t$ is increased by increasing the size of the spatial domain, n_m, while and holding n_t at 2400 samples.

FWI EXAMPLE

The FWI example encapsulates the simplicity and effectiveness of the Python interface in executing an advanced inversion operation. Figure 2a provides a snippet of a Python script, demonstrating the ease of performing a 2D acoustic FWI using the Marmousi model with `pyseis` (Martin et al., 2006). The script covers a series of steps, including the initialization of an acoustic isotropic 2D wave equation solver, defining necessary parameters such as model properties, spatial and time sampling rates, source and receiver locations, and GPUs to use. The final steps outline the FWI setup and execution, showcasing the straightforward application of `pyseis` for such complex geophysical computations. The evolution of the inversion process can be tracked by monitoring the normalized L2 loss across iterations as presented in Figure 2b, indicating the progression and effectiveness of the FWI operation. Overall, this FWI example reinforces the usability of `pyseis`, making advanced seismic inversion accessible to users without the need for extensive knowledge of the underlying complex computations.

a)

```python
from pyseis.wavelets.acoustic import Acoustic2D
from pyseis import inversion

# make 2d acoustic wave equation solver
acoustic_2d_solver = acoustic_isotropic.AcousticIsotropic2D(
  model=marm_model,             # 2d numpy array velocity model
  model_sampling=(10.0,10.0),   # spatial sampling rate
  wavelet=ricker,               # 1d numpy array time series
  d_t=0.004,                    # time sampling rate
  src_locations=srcs,           # numpy array of src locations
  rec_locations=recs,           # numpy array of rec locations
  gpus=[0,1,2,3])               # list of gpus to use

# model observed data
data_obs = acoustic_2d_solver.forward(true_model)

# setup fwi problem
fwi_prob = inversion.Fwi(acoustic_2d_solver,
                         data_obs,
                         starting_model,
                         num_iter=500)

# run fwi
history = fwi_prob.run()
```

b)

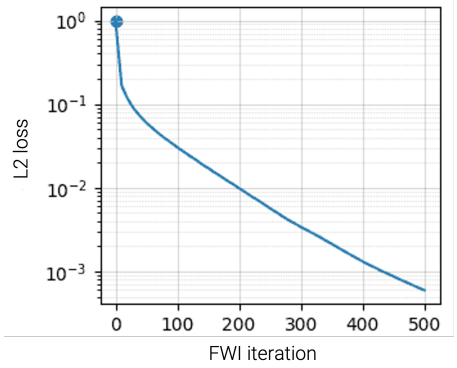

Figure 2: a) A code snippet that demonstrates a simple 2D acoustic full-waveform inversion (FWI) implementation using the Marmousi model with the `pyseis` package. b) The normalized L2 loss across the 500 iterations of FWI.

10.1190/image2023-3916155.1

RECENT APPLICATIONS

Three recent geophysical applications have successfully used the `pyseis` library, demonstrating the versatility and effectiveness of the library in tackling complex seismic problems.

The first application, termed "Target-oriented elastic full-waveform inversion through acoustic extended image-space redatuming," focused on reducing the computational cost associated with elastic FWI (Biondi et al., 2023). The methodology leveraged the extended image space to redatum the surface data to a new survey geometry placed in proximity to an identified target, greatly improving the speed and efficiency of the elastic FWI estimation. The resulting reconstructed datasets were then compared with the surface datasets, with the proposed method proving successful in retrieving the elastic parameters of a potential subsurface prospect (See Figure 3).

The second application showcased "Full-waveform inversion by model extension," a method which leverages the robustness of Waveform Elastic Migration Velocity Analysis with the high-resolution and accuracy of FWI (Barnier et al., 2023a). The FWI by Model Extension (FWIME) technique was applied to realistic synthetic tests, demonstrating its capacity to retrieve excellent solutions under challenging conditions. One of the most significant benefits of FWIME is its simplicity and effectiveness in inverting any type of waves without the need for coherent low-frequency signal or accurate initial guesses (See Figure 4).

Lastly, the third application presented a study on "Bridging the gap: deep learning on seismic field data with synthetic training for building Gulf of Mexico velocity models," using Convolutional Neural Networks (CNNs) to predict low-wavenumber velocity models from field seismic data (Farris and Clapp, 2023). Trained on synthetic data, the CNN was successfully applied to field data, thus bridging the 'domain gap' between synthetic and field data. This study highlighted the potential of machine learning in seismic velocity model building, indicating that more nuanced synthetic data generation can yield more accurate and generalizable results (See Figure 5).

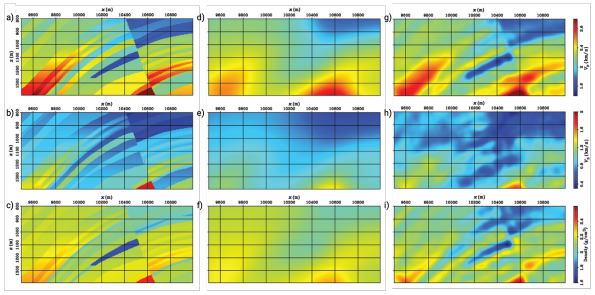

Figure 3: Comparison of true, initial, and inverted elastic parameters in the target area of the elastic Marmousi model, as presented in Biondi et al. (2023). This representative result is achieved using the `pyseis` library. Columns (a-c), (d-f), and (g-i) present true, initial, and inverted P-wave, S-wave, and density parameters, respectively, ordered from top to bottom.

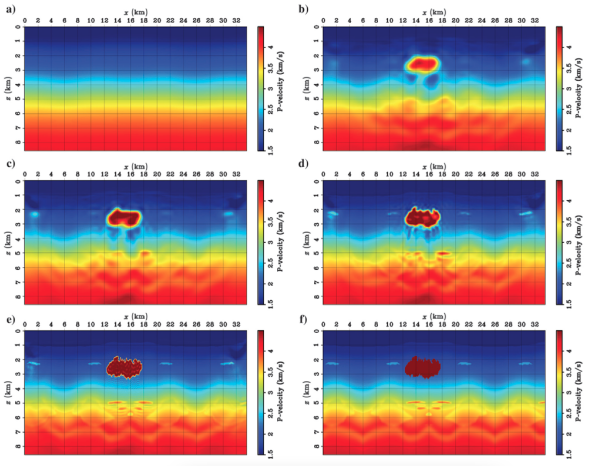

Figure 4: Sequence of 2D velocity models inverted at various stages in the FWIME workflow, implemented using the `pyseis` library (Barnier et al., 2023a). Panels (a-d) show initial and intermediate models, while (e) presents the final model after 250 BFGS iterations. The true model is in (f).

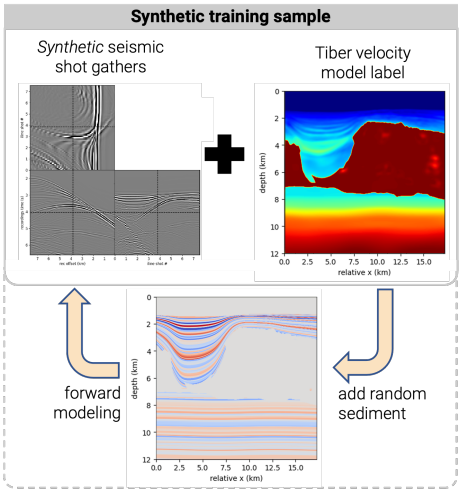

Figure 5: Illustration of a sample feature/label pair from the synthetic training dataset, created using the `pyseis` library (Farris and Clapp, 2023). The synthetic examples were generated by incorporating random sediment layers into the legacy velocity model and simulating the seismic experiment via `pyseis`.

Together, these applications underscore the efficacy and adaptability of the `pyseis` library in handling diverse seismic inversion problems, from targeted elastic FWI methods to deep learning applications.

10.1190/image2023-3916155.1

REFERENCES

Aki, K., and P. G. Richards, 2002, Quantitative seismology, 2nd ed.: University Science Books.

Barnier, G., E. Biondi, R. G. Clapp, and B. Biondi, 2023a, Full waveform inversion by model extension: Practical applications: Geophysics, **88**, no. 5, R609–R643, doi: https://doi.org/10.1190/geo2022-0382.1.

Barnier, G., E. Biondi, R. G. Clapp, and B. Biondi, 2023b, Full waveform inversion by model extension: Theory, design and optimization: Geophysics, **88**, no. 5, R579–R607, doi: https://doi.org/10.1190/geo2022-0350.1.

Biondi, E., G. Barnier, B. Biondi, and R. G. Clapp, 2023, Target-oriented elastic full-waveform inversion through acoustic extended image-space redatuming: Geophysics, **88**, no. 3, R269–R296, doi: https://doi.org/10.1190/geo2022-0404.1.

Biondi, E., G. Barnier, R. G. Clapp, F. Picetti, and S. Farris, 2021, An object-oriented optimization framework for large-scale inverse problems: Computers Geosciences, **154**, 104790, doi: https://doi.org/10.1016/j.cageo.2021.104790.

Farris, S., G. Barnier, and R. Clapp, 2022, Deep learning velocity model building using an ensemble regression approach: Second International Meeting for Applied Geoscience & Energy, SEG/AAPG, Expanded Abstracts, 3637–3641, doi: https://doi.org/10.1190/image2022-w7-01.1.

Farris, S., and R. Clapp, 2023, Bridging the gap: Deep learning on seismic field data with synthetic training for building Gulf of Mexico velocity models: Third International Meeting for Applied Geoscience & Energy, SEG/AAPG, Expanded Abstracts, 945–949, doi: https://doi.org/10.1190/image2023-3905650.1.

Jakob, W., J. Rhinelander, and D. Moldovan, 2017, py-bind11 — seamless operability between c++11 and python, https://github.com/pybind/pybind11.

Louboutin, M., M. Lange, F. Luporini, N. Kukreja, P. A. Witte, F. J. Herrmann, P. Velesko, and G. J. Gorman, 2019, De-vito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration: Geoscientific Model Development, **12**, 1165–1187, doi: https://doi.org/10.5194/gmd-12-1165-2019.

Luporini, F., M. Louboutin, M. Lange, N. Kukreja, P. Witte, J. Hückelheim, C. Yount, P. H. J. Kelly, F. J. Herrmann, and G. J. Gorman, 2020, Architecture and performance of devito, a system for automated stencil computation: ACM Transactions on Mathematical Software, **46**, 1–28, doi: https://doi.org/10.1145/3374916

Martin, G. S., R. Wiley, and K. J. Marfurt, 2006, Marmousi2: An elastic upgrade for Marmousi: The Leading Edge, **25**, 156–166, doi: https://doi.org/10.1190/1.2172306.

Peter, D., D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz, T. Nissen-Meyer, P. Basini, and J. Tromp, 2011, Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes: Geophysical Journal International, **186**, 721–739, doi: https://doi.org/10.1111/j.1365-246X.2011.05044.x.