SBGf
SBGf Conference
18-20 NOV | Rio'25
Sustainable Geophysics at the Service of Society
In a world of energy diversification and social justice

Submission code: 409JXNWAZ7

See this and other abstracts on our website: https://home.sbgf.org.br/Pages/resumos.php

Solving Seismic Wave Equations with Astaroth library

Oscar Mojica (Supercomputing Center for Industrial Innovation; SENAI CIMATEC), Leonildes Soares de Melo Filho (Repsol Sinopec Brasil)

Solving Seismic Wave Equations with Astaroth library

Abstract Summary

Seismic modeling is fundamental to exploration seismology, supporting essential tasks like data acquisition, processing, and reservoir characterization. However, simulating wave propagation by solving the wave equation is computationally demanding, often slowing research progress when relying on traditional CPU-based implementations. Although GPU implementations provide a pathway to faster computations, GPU programming remains inherently complex and time-consuming. In this work, we explore Astaroth, a framework that streamlines this process through GPU-accelerated code generation, facilitating the development of high-performance applications for solving partial differential equations (PDEs) with optimized stencil code on GPUs. Through a case study, we illustrate how Astaroth supports the rapid development of a seismic forward modeling application.

Introduction

High-Performance Computing (HPC) plays a pivotal role in addressing the computational challenges of modern scientific simulations, particularly in exploration seismology, where accurate 3D wave modeling is essential for subsurface imaging and resource exploration. These simulations are computationally intensive, requiring significant processing power to model wave interactions over large domains and extended time periods. To meet these demands, writing high-performance GPU code is essential, yet it often increases development costs due to the complexity of GPU programming. One effective solution is automatic code generation, which allows geoscientists to focus on domain-specific problem descriptions while the underlying framework generates optimized low-level code. A leading example of such a framework is Devito (Louboutin et al., 2017), a Python-based domain-specific language (DSL) for finite difference computations, offering symbolic PDEs definitions, automated code generation, and optimizations for CPU and GPU platforms.

Building on this approach, we explore Astaroth (Pekkilä, 2019; Pekkilä et al., 2025), a library that uses the DSL paradigm to ease the efficient automated creation of high-performance GPU code. Astaroth offers an application programming interface (API) for accessing GPU resources, a DSL for creating stencil kernels, and a compiler that translates programs written in this DSL into highly optimized CUDA/HIP kernels. By automating the generation of efficient CUDA/HIP code, Astaroth simplifies GPU programming for geoscientists and provides stencil kernels that maximize GPU cache utilization. This makes it an attractive tool for geoscientists tackling large-scale 3D wave modeling, enabling them to achieve high performance without the burden of low-level optimization.

Astaroth's DSL

Astaroth's DSL simplifies writing efficient stencil operation kernels using a procedural style and a dataflow model that aligns with graphics pipelines, leveraging GPU parallel processing. It focuses on writing GPU programs, excluding external memory management, and uses a C/C++-like syntax with new keywords (i.e, `Kernel`, `Stencil`, etc) to translate high-level stencil pipelines into optimized CUDA/HIP kernels. Users define a `Stencil` explicitly, which is then converted into a function that can be utilized within kernels. By restricting read operations to these predefined stencils, Astaroth generates highly optimized code. A `Field` is stored in vertex buffers (dual "in" and "out" arrays), where values are read from "in" and written to "out" for performance reasons. The input and output buffers can be accessed using using `FIELD_IN` and `FIELD_OUT`. The DSL provide the current vertex's position with a built-in variable called `vertexIdx`, a type with three integers (`int3`). Arrays can be accessed without choosing an index; it then uses `vertexIdx`'s `x`, `y`, `z` values by default. For additional details on the DSL and API (not discussed here), we direct readers to Pekkilä (2019) and the Bitbucket repository (`https://bitbucket.org/jpekkila/astaroth/src/master/`).

Experiments

While Astaroth was originally designed to solve the standard set of PDEs for magnetohydrodynamics (MHD) simulations, we demonstrate here that, with certain tweaks and adaptations, it can be utilized to efficiently compute solutions for the second-order acoustic wave equation:

$$m\partial_{tt}u - \Delta u = f, \tag{1}$$

where ∂_{tt} means the second partial derivative with respect to the time, $m(\mathbf{x}) = 1/v^2(\mathbf{x})$ at $\mathbf{x} = (x, y, z)$, u is the pressure field, Δ is the Laplacian operator and f is a source of acoustic energy. We solve Eq. 1 using an homogeneous model (v=1.5 km/s), which is discretized into 256$\times$256$\times$256 grid points with a 10 m grid interval. We use a Ricker wavelet (10 Hz peak) as the source and place one shot at the model's center. Listing 1 show how the wave equation may be solved with the DSL.

```
int AC_step_number
int3 AC_src_pos
real AC_two
hostdefine STENCIL_ORDER (8) // max order used here
Field u,sq_vel,src

Stencil laplace3d_ord4 {
  [0][0][0] = 1
  ,[0][0][-2] = 1
  ,[0][0][-1] = 1
  ,[0][0][1] = 1
  ,[0][0][2] = 1
  ,[0][-2][0] = 1
  ,[0][-1][0] = 1
  ,[0][1][0] = 1
  ,[0][2][0] = 1
  ,[-2][0][0] = 1
  ,[-1][0][0] = 1
  ,[1][0][0] = 1
  ,[2][0][0] = 1
}
Kernel solve3d_ord4() {
  force = real(0)

  if vertexIdx.x == AC_src_pos.x && vertexIdx.y == AC_src_pos.y && vertexIdx.z == AC_src_pos.z {
        force = FIELD_IN[src][AC_step_number]
  }

  write(u,(AC_two*FIELD_IN[u][IDX(vertexIdx.x,vertexIdx.y,vertexIdx.z)] +
           FIELD_IN[sq_vel][IDX(vertexIdx.x,vertexIdx.y,vertexIdx.z)]*laplace3d_ord4(u) -
           previous(u)) + force)
}
```

Listing 1: Example of Astaroth DSL Code for solving the scalar wave equation using 4th order FD. Stencil coefficients are set at runtime.

Within the DSL, the functions `previous` and `write` are used to manage field updates. The `previous` function accesses elements in the "out" buffer, enabling retrieval of a field's previous state, while the `write` function stores the computed result in the "out" buffer. In the current version of the DSL, handling 1D arrays directly isn't fully supported. As a workaround, we use standard 3D fields accessed via `FIELD_IN`. Since v^2 (`sq_vel`) is only read, its "out" buffer remains unused, leading to more memory usage than necessary. Similarly, mapping 1D arrays as the source f into a 3D field results in additional memory consumption. This is illustrated in Fig. 1-a, where the gray color represents the unused extra memory.

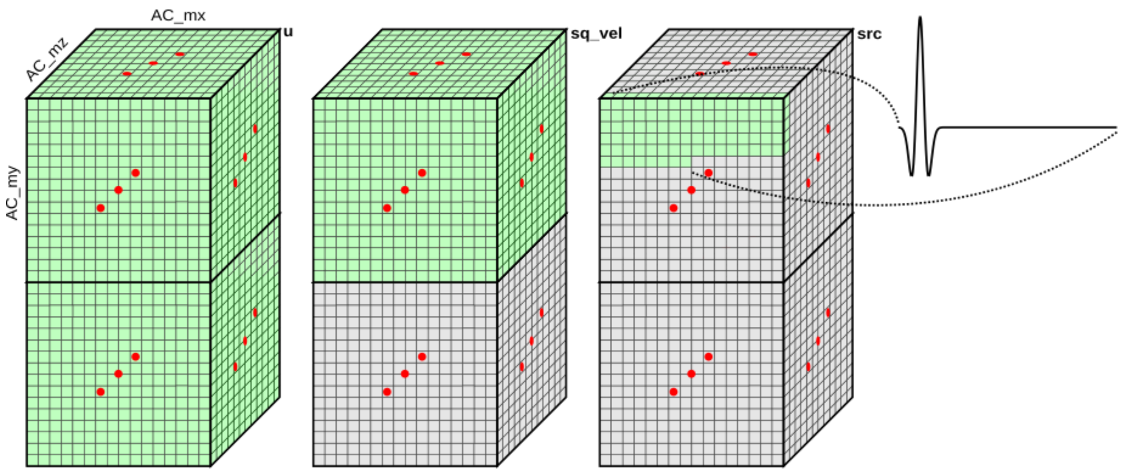

Figure 1: a) Illustrative scheme of the `Fields` used in wave modeling (see Listing 1). The labels `AC_mx`, `AC_my`, and `AC_mz` denote the total number of cells in the computational domain along the x-, y-, and z-axes, respectively, including halo cells.

We use a hardware-managed caching (HWC) strategy included in Astaroth for data reuse (Pekkilä et al., 2025). We benchmarked the time to compute the wave equation using second- to eighth-order finite differences (FD), i.e., radius 1 to 4 stencils, with PyTorch and Astaroth. Adapting wave modeling to use tensors, a fundamental data structure in machine learning, has become a common practice due to the availability of optimized machine learning libraries that leverage GPU acceleration for tensor operations. This led us to choose PyTorch for performance comparisons. In the benchmarks, we measured the median running time of 100 iterations. Fig. 2-left displays the wavefield after 100 time steps of wave propagation simulation at at xy plane (z=1.28 Km), while Fig. 2-right shows the time per step for computing the wave equation with Astaroth (double and single precision) and Pytorch (single precision).

Conclusions

We introduced the Astaroth framework as an effective tool for developing 3D FD seismic wave propagation codes that leverage the high-performance computing capabilities of GPUs. Astaroth employs multiple optimization strategies to enhance the performance of stencil computations on GPUs. Numerical experiments using a synthetic 3D model confirm that Astaroth is both feasible and practical for solving the wave equation efficiently.

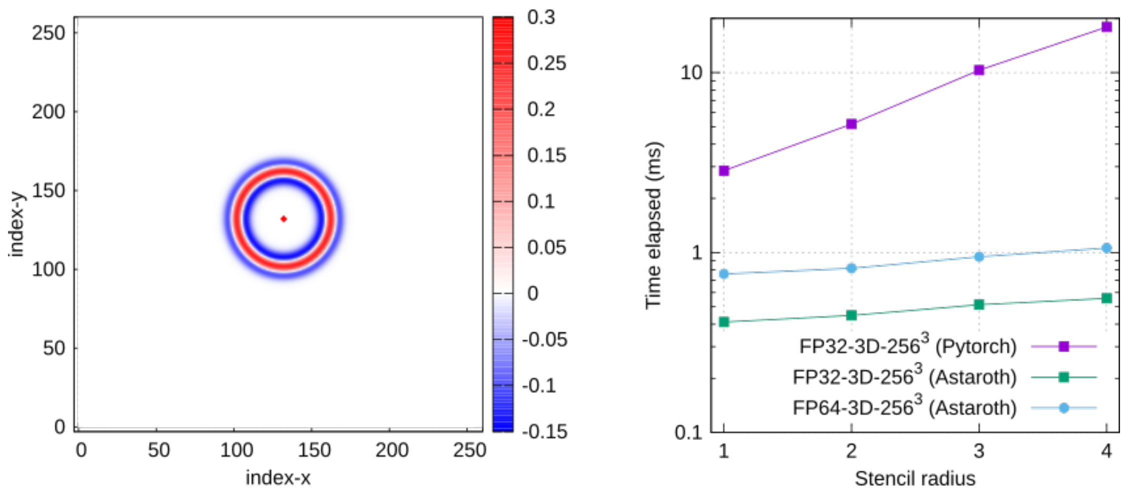

Figure 2: (Left) Snapshot of the wavefield at the final time step on the xy plane (z=1.28 Km). (Right) Time per step for solving the wave equation using Astaroth (single and double precision) and PyTorch (single precision) on a single NVIDIA V100 GPU. Lower is better.

Acknowledgements

The authors would like to thank Repsol Sinopec Brasil for supporting this research under the “Siren Project” agreement, in accordance with the regulations of the National Agency of Petroleum, Natural Gas, and Biofuels (ANP) RD&I levy fund.

References

Louboutin, M., P. Witte, M. Lange, N. Kukreja, F. Luporini, G. Gorman, and F. J. Herrmann, 2017, Full-waveform inversion, part 1: Forward modeling: The Leading Edge, **36**, 1033–1036.

Pekkilä, J., 2019, Astaroth: A library for stencil computations on graphics processing units: Master’s thesis, Aalto University School of Science.

Pekkilä, J., O. Lappi, F. Robertsén, and M. J. Korpi-Lagg, 2025, Stencil computations on amd and nvidia graphics processors: Performance and tuning strategies: Concurrency and Computation: Practice and Experience, **37**, e70129.