High-level python abstractions for data management in inversion problems

Checkpointing, done beautifully.

Navjot Kukreja
Imperial College London
London, UK

Jan Hückelheim
Imperial College London
London, UK

Michael Lange
Imperial College London
London, UK

Who Else
Imperial College London
London, UK

ABSTRACT

TODO The computation of adjoints in optimisation and inverse design problems requires storing intermediate data. The available memory size sets a limit to this, and necessitates recomputation in some instances. Revolve checkpointing offers an optimal schedule that trades computational cost for smaller memory footprints. Integrating Revolve into a modern python HPC code and combining it with code generation is not straightforward. We present an API that makes checkpointing accessible from a DSL-based code generation environment and present a benchmark study. TODO.

CCS CONCEPTS

• **Software and its engineering → Software design engineering**;

KEYWORDS

HPC, Code generation, API, Checkpointing, Adjoint, Inverse Problems

ACM Reference format:
Navjot Kukreja, Jan Hückelheim, Michael Lange, and Who Else. 1997. High-level python abstractions for data management in inversion problems. In *Proceedings of SuperComputing, Denver, Colorado, USA, November 2017 (SC17),* 2 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Seismic inversion, and more specifically Full Waveform Inversion (FWI) is a computationally heavy technique that uses data from seismic wave propagation experiments to calculate physical parameters of the earth's subsurface. In case of FWI, this additional information on the physical parameters drives the generation of better subsurface images (CITE). An FWI problem is setup as an optimization problem using a gradient-based optimization algorithm with the objective function being minimised is the misfit between the simulated data and the observed data received at the receivers. A finite-difference discretization of one of the forms of the wave equation acts as the constraint for the optimization. Since the gradient is calculated using the adjoint-state method, by the correlation between the fields in forward and adjoint (reverse) mode (CITE), the method requires that the forward and adjoint field be known for each time step in the simulation. Considering that the typical domain for such a setup is 3 dimensional with 1000 points in each dimension, the field requires $1000 \times 1000 \times 1000 \times 4 = 4 \times 10^9$ bytes (4 GB) of memory for a single timestep. Since the simulation is run for a typical 3000 time-steps, storing the entire forward field in memory would require $12TB$ of memory which is prohibitively high. We discuss this in section 2.

Previous work on similar inverse problems led to the *Revolve* algorithm [**?**] and the associated C++ tool which provides an optimal schedule at which to store checkpoints, i.e. states from which the forward simulation can be restored. This algorithm is further discussed in section 3.

In this paper we describe how the algorithm can be combined with code generation to make checkpointing, still considered an advanced technique by practitioners, much more accessible. The software that can enable this is described in section 4.

The performance impact of using the checkpointing scheme, as well as the effect of the size of memory available is studied in section 5.

Motivation: Why do we need to connect revolve and a python seismic inversion code? What is there to learn for others? How does our work help readers and the world?

2 SEISMIC INVERSION AND DEVITO

Some context on seismic imaging, the PDEs, the method. Why use adjoints. Why is this problem time-dependent. Why do we need a lot of data. Code generation, references to previous papers about Devito. Why not be brave and write this in Fortran.

3 REVOLVE

How does revolve work and save memory. How do others use it, why is this painful.

4 API: CONNECTING REVOLVE WITH DEVITO

This is the core of the paper.

SC17, November 2017, Denver, Colorado, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

5 TEST CASE, RESULTS

A nice test case that can be scaled up in size easily. Timings for different mesh sizes, with different amounts of memory set as the upper limit.

6 CONCLUSIONS

This work was highly successful, but more work could be done.

ACKNOWLEDGMENTS

The authors are very grateful.

REFERENCES