Software acceleration of CARP, an iterative linear solver and preconditioner

Art Petrenko, Tristan van Leeuwen and Felix J. Herrmann
Seismic Laboratory for Imaging and Modeling, Department of Earth, Ocean and Atmospheric Sciences

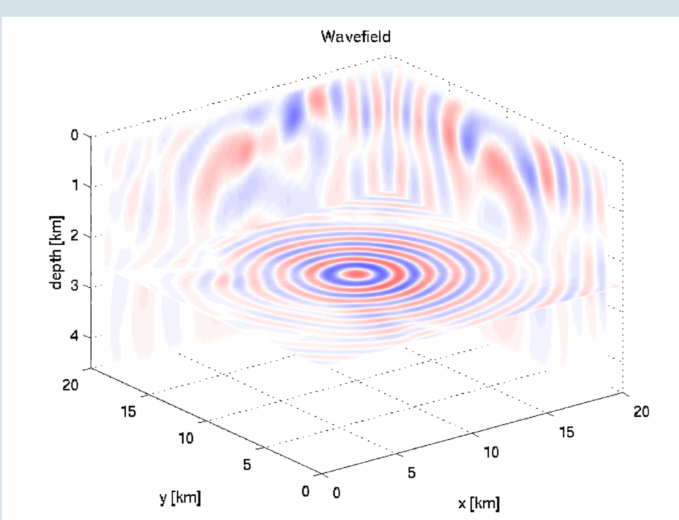
a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

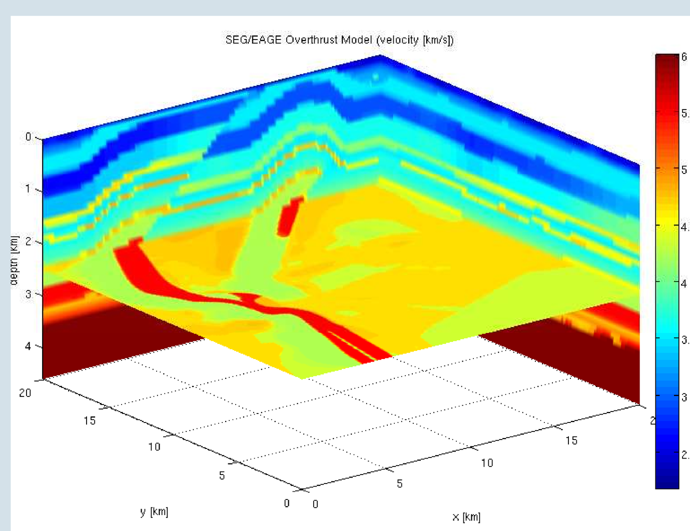
Introduction

Full-waveform inversion (FWI) is one important tool in seismic exploration (the search for hydrocarbons):

- Vibrational energy is sent into the earth.
- Observations of waves reflected and refracted back are made at the surface and in boreholes.
- The wave equation, parameterized with the subsurface quantity of interest (e.g.: slowness squared, $1/v^2$), is solved.
- The resultant wavefield is compared with the data, and based on this the parameter model is updated.

The process is repeated, refining the subsurface parameter model with each iteration. FWI is important because it puts geophysical inversion on a rigourous footing, eliminating the manual tuning of traditional algorithms (such as migration). Simulation of seismic wave propagation is the biggest computational burden of FWI.





The Wave Equation: $Hu = q$

- H: Wave physics operator; $N \times N$ Helmholtz matrix; large, sparse and has diagonal band structure.
- $\mathbf{m}$: N-dimensional; defines the model parameter throughout the $x \times y \times z = N$ physical domain .
- $\mathbf{u}$: Complex Fourier transform of the pressure with respect to time.
- $\mathbf{q}$: Amplitude of the source at angular frequency ω.
- Constant density isotropic acoustic wave equation formulated in the time harmonic (frequency) domain.
- Contains perfectly matched layer functions that eliminate numerical reflection artifacts from the boundaries.

References

[1] S. Kaczmarz, "Approximate solution of systems of linear equations," *International Journal of Control*, vol. 57, no. 6, pp. 1269–1271, 1993.

[2] T. van Leeuwen, D. Gordon, R. Gordon, and F. J. Herrmann, "Preconditioning the helmholtz equation via row-projections," in *EAGE technical program*, EAGE, 2012.

[3] D. Gordon and R. Gordon, "Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems," *SIAM Journal on Scientific Computing*, vol. 27, no. 3, pp. 1092–1117, 2005.

[4] D. Gordon and R. Gordon, "CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs," *Parallel Computing*, vol. 36, no. 9, pp. 495–515, 2010.

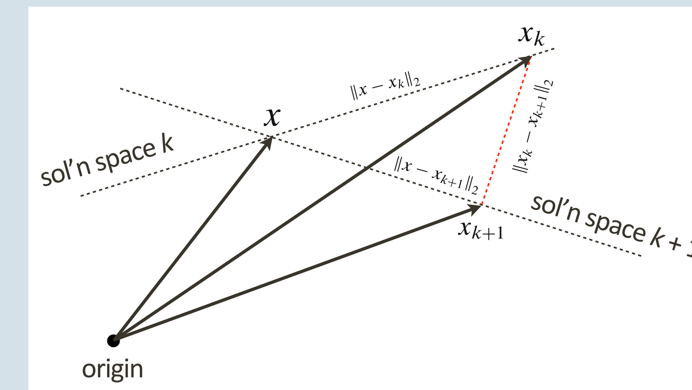
[5] S. Operto, J. Virieux, P. Amestoy, J.-Y. L'Excellent, L. Giraud, and H. B. H. Ali, "3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study," *Geophysics*, vol. 72, no. 5, pp. SM195–SM211, 2007.

[6] F. Aminzadeh, B. Jean, and T. Kunz, *3-D salt and overthrust models.* Society of Exploration Geophysicists, 1997.

[7] "Intel 64 and IA-32 architectures optimization reference manual." http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf, 2012.

Iterative Linear Solvers: Kaczmarz, CARP and CARP-CG

The Kaczmarz algorithm [1] solves a general linear system $A\mathbf{x} = \mathbf{b}$ by projecting the current iterate $\mathbf{x}_i$ onto the hyperplane orthogonal to a row $\mathbf{a}_k$ of the matrix, and setting the next iterate equal to the result (w is a relaxation parameter we set to 1.5, as in [2]):



$$\mathbf{x}_{i+1} = \mathbf{x}_i + w(b[k] - \langle \mathbf{a}_k, \mathbf{x}_i \rangle) \frac{\mathbf{a}_k^*}{\|\mathbf{a}_k\|^2}.$$

- CARP (Component Averaged Row Projections) [3] is a parallelization of the Kaczmarz algorithm that works on sets of rows simultaneously, averaging elements that sets have in common.
- Using CARP to cyclically project onto each row form first to last and back (a double sweep) gives an equivalent symmetric positive semi-definite system (which H is not), which can be solved with the method of conjugate gradients (CARP-CG) [4].

Implementation Details

- Diagonal storage format used for H, with rows contiguous in memory.
- Single-threaded Kaczmarz (CARP) sweeps.
- Programmed in C, reading matrix and vector (H, $\mathbf{u}_0$ and $\mathbf{q}$) inputs from disk.
- Complex numbers stored as arrays of structures.
- Compiled with `-O2` and `-xavx` switches to take advantage of vector CPU instructions.
- Simulations bound to one hardware thread on one core of an Intel Xeon E5-2670.
- Medium $\mathbf{m}$ was taken to be the SEG/EAGE overthrust velocity model, presented by [6], undersampled by various factors.
- Timing results are the average of 10 runs.

Hypothesis

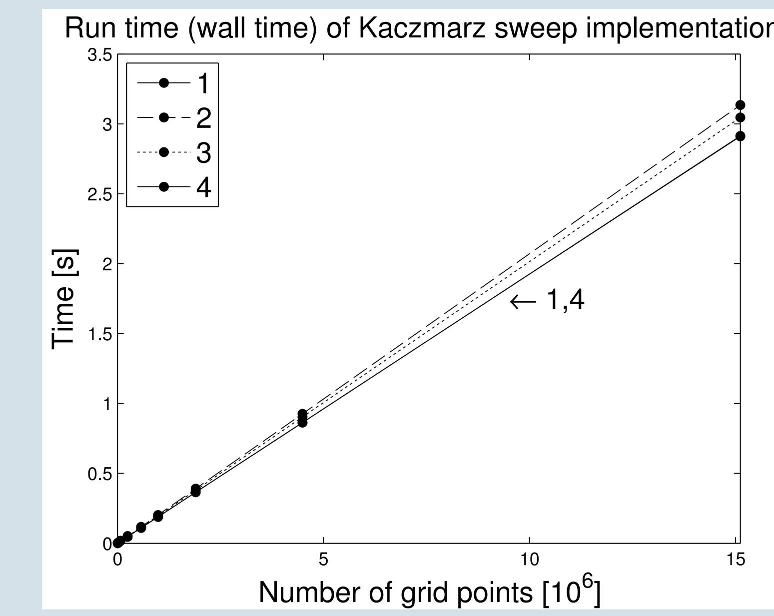
Since the double CARP sweep is the most time-consuming part of CARP-CG it was the object of optimization. We noticed that the forward Kaczmarz sweeps were consistently faster than the backward sweeps; our hypothesis was that this was due to different patterns of memory access:

- Forward sweeps access matrix H elements in the order in which they are stored: both row and diagonal indices increasing.
- Backward sweeps jump from the end of one row to the start of the previous row: row index decreasing, diagonal index increasing.

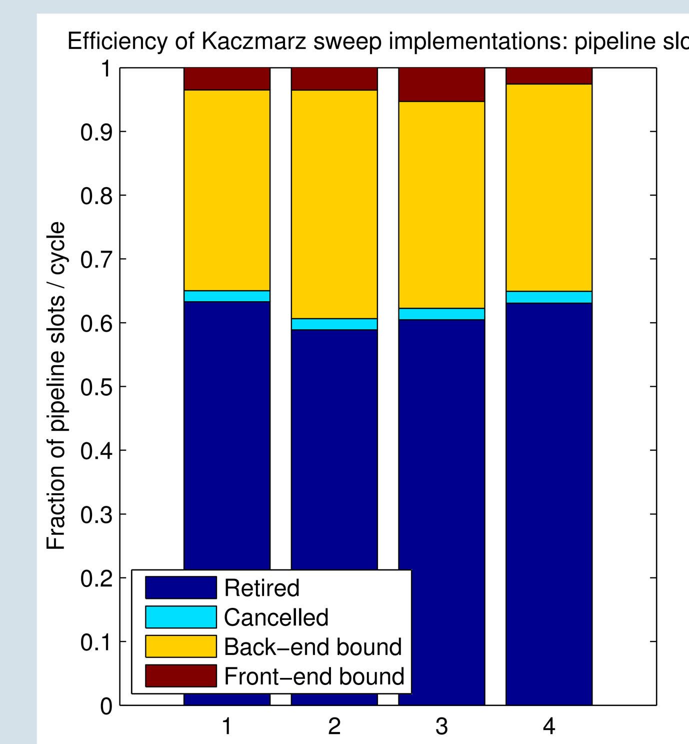
Results

The four memory access patterns tested and their execution times.

ID	Sweep	Diagonal index
1	forward	increment
2	backward	increment
3	forward	decrement
4	backward	decrement



- Note that the curves for implementations 1 and 4 overlap.
- For the last set of points the speed-up between implementations 4 and 2 is 7%.
- For comparison, the speed-up we measured between sweeps programmed in MATLAB (not shown) and any of the C-sweeps is approximately 80-fold.
- For the $94 \times 401 \times 401 \approx 15 \cdot 10^6$ model, efficiency was measured with processor pipeline slots, which are the basic unit of instruction level parallelism. They can be in one of four states [7]:



- ○ retired: have successfully finished their calculation
- ○ cancelled: due to mis-predicted execution branches
- ○ back-end bound: waiting on computational or memory access resources
- ○ front-end bound: unable to timely deliver the next instruction in the pipeline

Conclusions

- The execution times presented above are a demonstration of the well known rule that memory is best addressed in order.
- Improvements in efficiency metrics such as the percentage of time the pipeline is waiting for memory access (back-end bound pipeline slots), do not translate directly into improvements in execution time.

Acknowledgements

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, CGG, Chevron, ConocoPhillips, ION GXT, Petrobras, PGS, Total SA, WesternGeco, and Woodside.

apetrenko@eos.ubc.ca