Accelerating an iterative Helmholtz solver with FPGAs

Art Petrenko, Felix J. Herrmann: SLIM
Diego Oriato, Simon Tilbury: Maxeler Technologies
Tristan van Leeuwen: Centrum Wiskunde & Informatica
November 14, 2013

SLIM
University of British Columbia

Oh by the way: I have a stutter.

Big Picture

Goal: Accelerate solution of the seismic forward problem.

Method: Implement time-harmonic wave equation solver on reconfigurable hardware (FPGAs).

Why FPGAs?

Execution time directly proportional to size of problem.

Algorithm **complexity irrelevant.**

Success of time-domain wave simulation on FPGAs. [Pell, 2013]

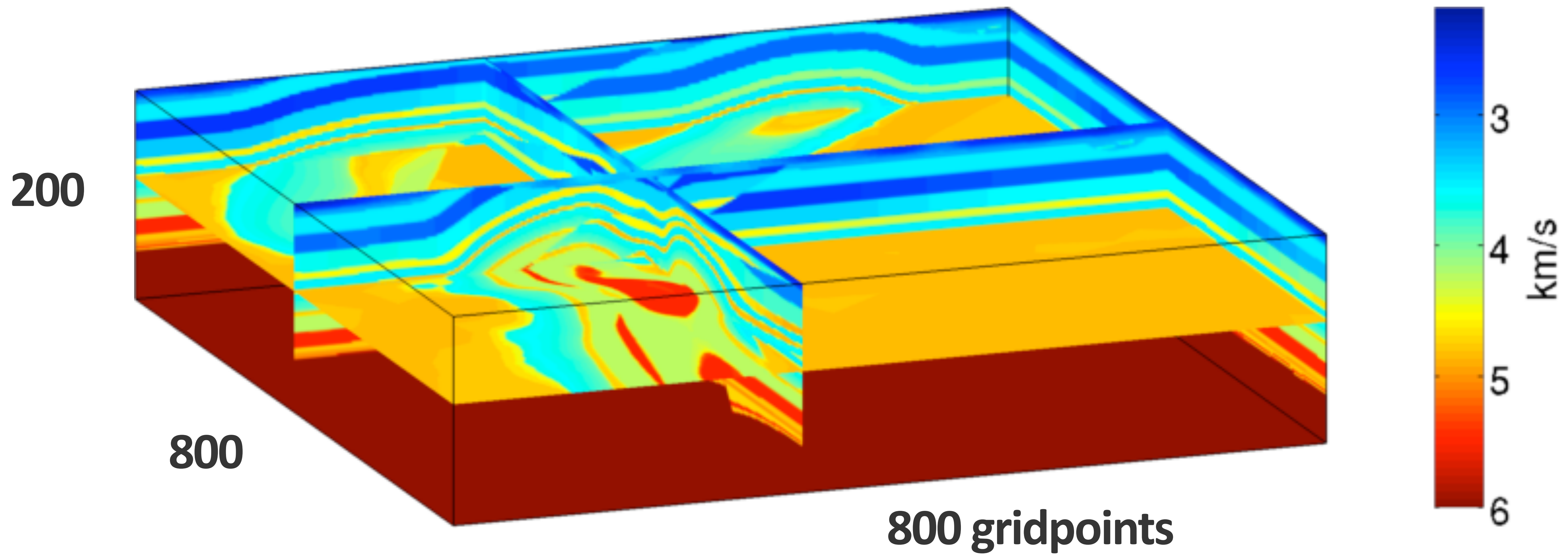
Projected Results

Speed-up of time-harmonic seismic forward problem:

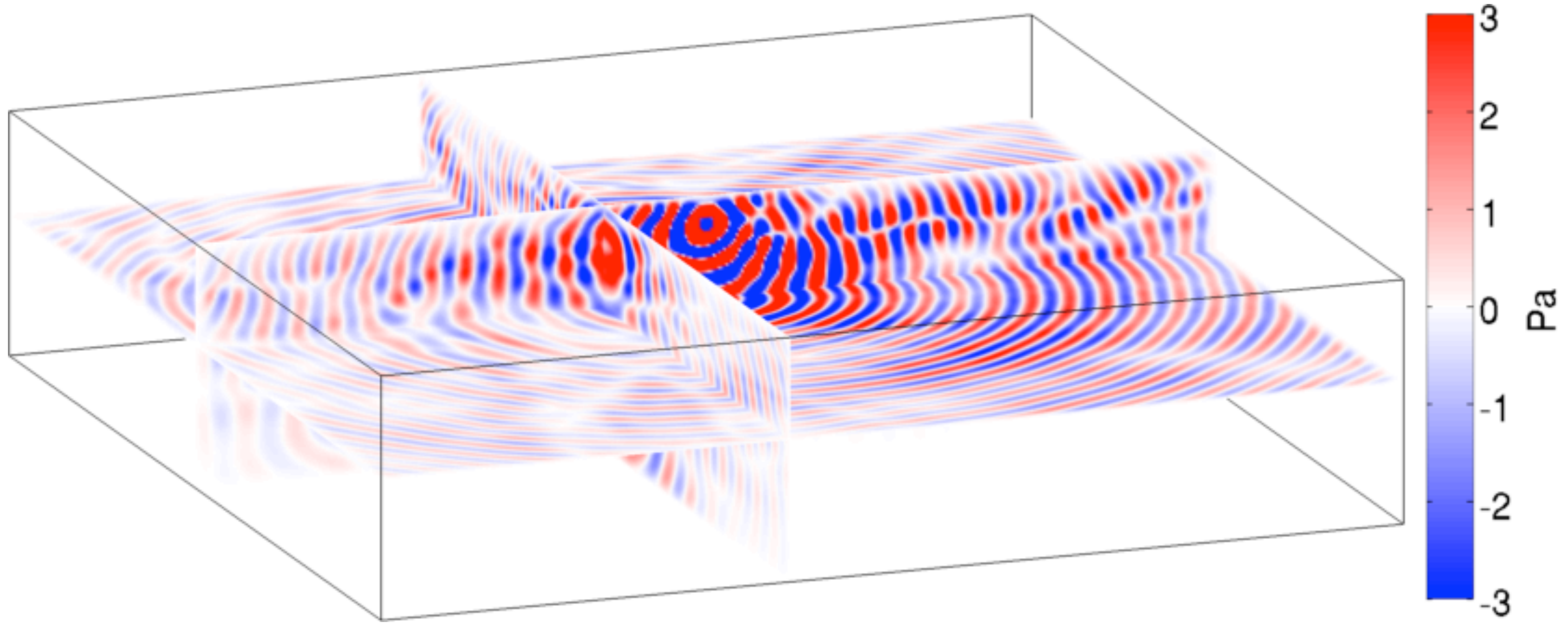
1 FPGA = **1** unit of time

1 Intel Xeon core (MATLAB/C): **32** units of time

Earth model $\mathbf{v}$ ($\mathbf{m} = 1/\mathbf{v}^2$)



$$A(\mathbf{m}, \omega)\mathbf{u} = \mathbf{q}$$



Fourier transform of pressure wavefield $\mathbf{u}$

The Helmholtz System

wave equation
frequency domain
acoustic
constant density
isotropic
3 x 3 x 3 cube stencil
[Operto, 2007]

Solving the Helmholtz System

Use iterative solver: the method of **conjugate gradients (CG)**.

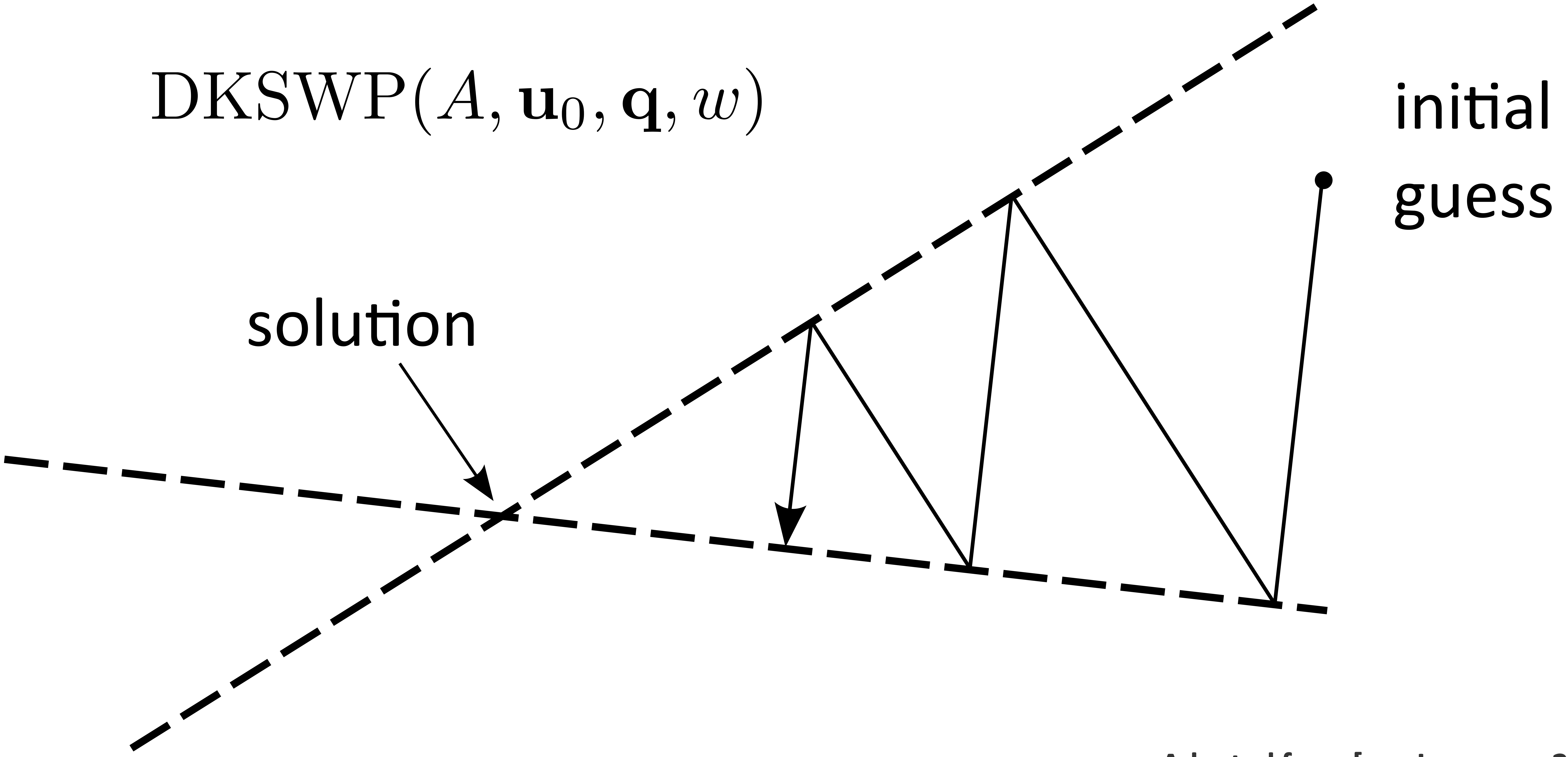
Solving the Helmholtz System

Problem: A is not symmetric positive semidefinite.

Solution: Precondition to obtain a different but equivalent system.

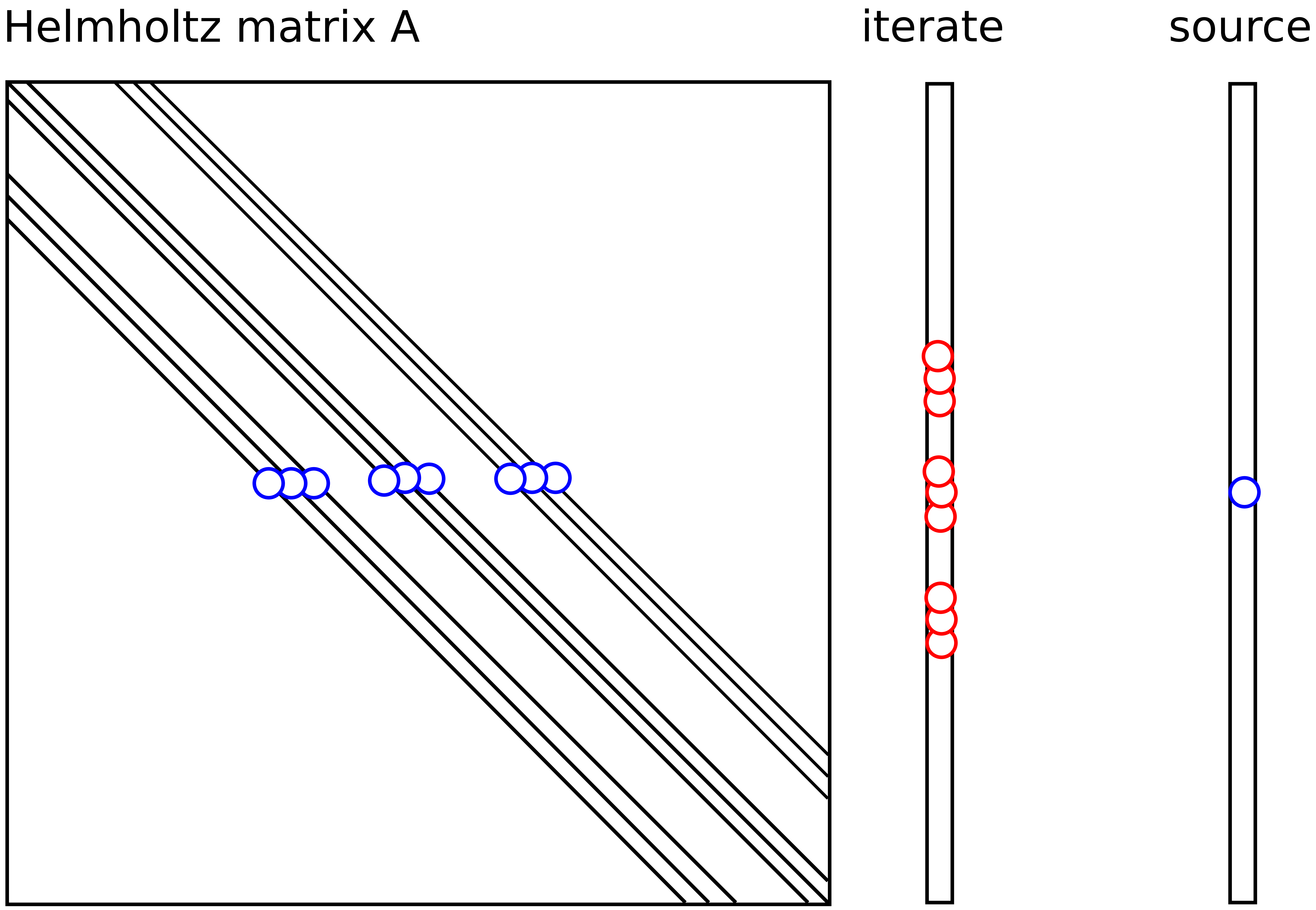
The Kaczmarz Algorithm
[Kaczmarz, 1937]

$$\mathrm{DKSWP}(A, \mathbf{u}_0, \mathbf{q}, w)$$



Adapted from [van Leeuwen, 2012]

The Kaczmarz Algorithm
one iteration



blue: values read
red: values read + modified

CG + Kaczmarz = CGMN
[Björck & Elfving 1979]

A different but equivalent system:

$$(I - \mathrm{DKSWP}(A, \cdot, 0, w))\mathbf{u} = \mathrm{DKSWP}(A, 0, \mathbf{q}, w)$$

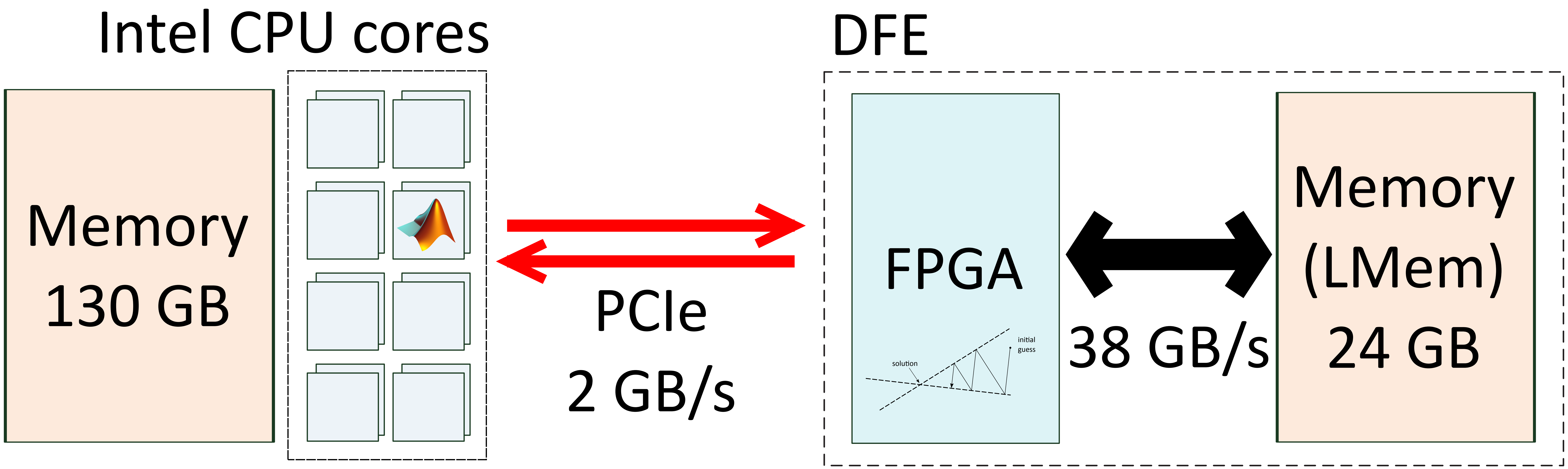
Double Kaczmarz sweep as preconditioner.

Our Contribution

Implement Kaczmarz sweeps on **CPU + accelerator platform**.

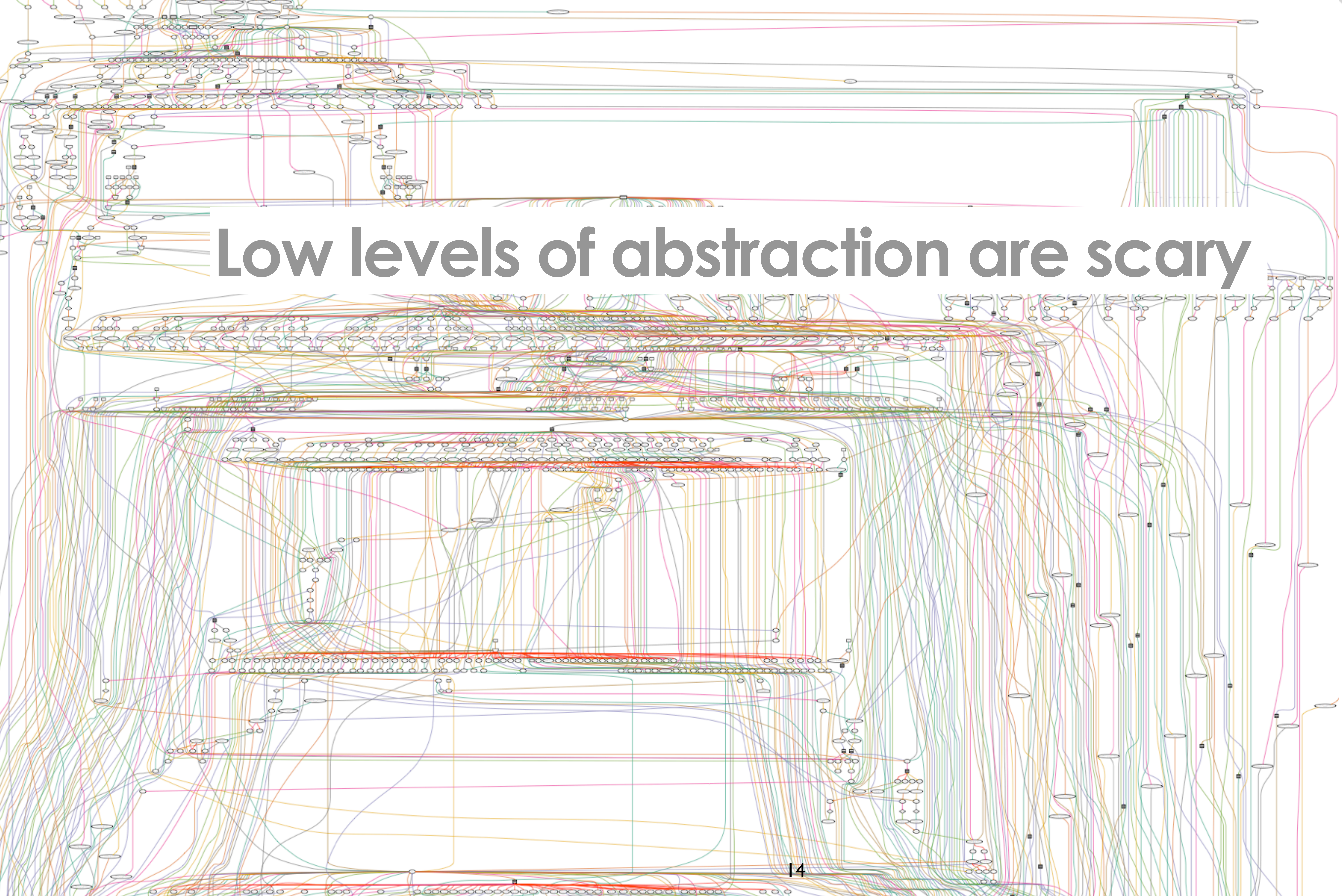
Integrate accelerated sweeps into SLIM's **MATLAB**, **object-oriented** workflow.

Target Platform: Maxeler



Adapted from [Pell, 2013]

Low levels of abstraction are scary



Design at high level of abstraction

```
                             ((x[2]*R[24] + x[1]*R[25]) +
                               x[0]*R[26]));
DFEVar relaxationFactor = io.scalarInput("relaxationFactor", kaczmarzEngineCode.KaczmarzWriteLMemKernel.TruncatedFloatingPoint);
DFEComplex kaczmarz_numerator = computation_stage ? relaxationFactor*(b - dot_product) : 0;
DFEComplex[] R_conj = new DFEComplex[kaczmarzEngineCode.KaczmarzManager.array_size];
for(int j=0; j<kaczmarzEngineCode.KaczmarzManager.array_size; j++){
    R_conj[j] = kaczmarzEngineCode.KaczmarzWriteLMemKernel.ComplexTruncatedFloatingPoint.newInstance(this);
    R_conj[j].setReal(R[j].getReal());
    R_conj[j].setImaginary(-R[j].getImaginary());
}
DFEComplex[] R_scaled = new DFEComplex[kaczmarzEngineCode.KaczmarzManager.array_size];
for(int j=0; j<kaczmarzEngineCode.KaczmarzManager.array_size; j++){
    R_scaled[j] = kaczmarzEngineCode.KaczmarzWriteLMemKernel.ComplexTruncatedFloatingPoint.newInstance(this);
    R_scaled[j] = kaczmarz_numerator*R_conj[j];
}
//DFEComplex[] x_updated = new DFEComplex[kaczmarzEngineCode.KaczmarzManager.array_size];
for(int j=0; j<kaczmarzEngineCode.KaczmarzManager.array_size; j++){
    x_updated[j] <== x[j] + R_scaled[j];
}
```

Parallelism through Pipelining

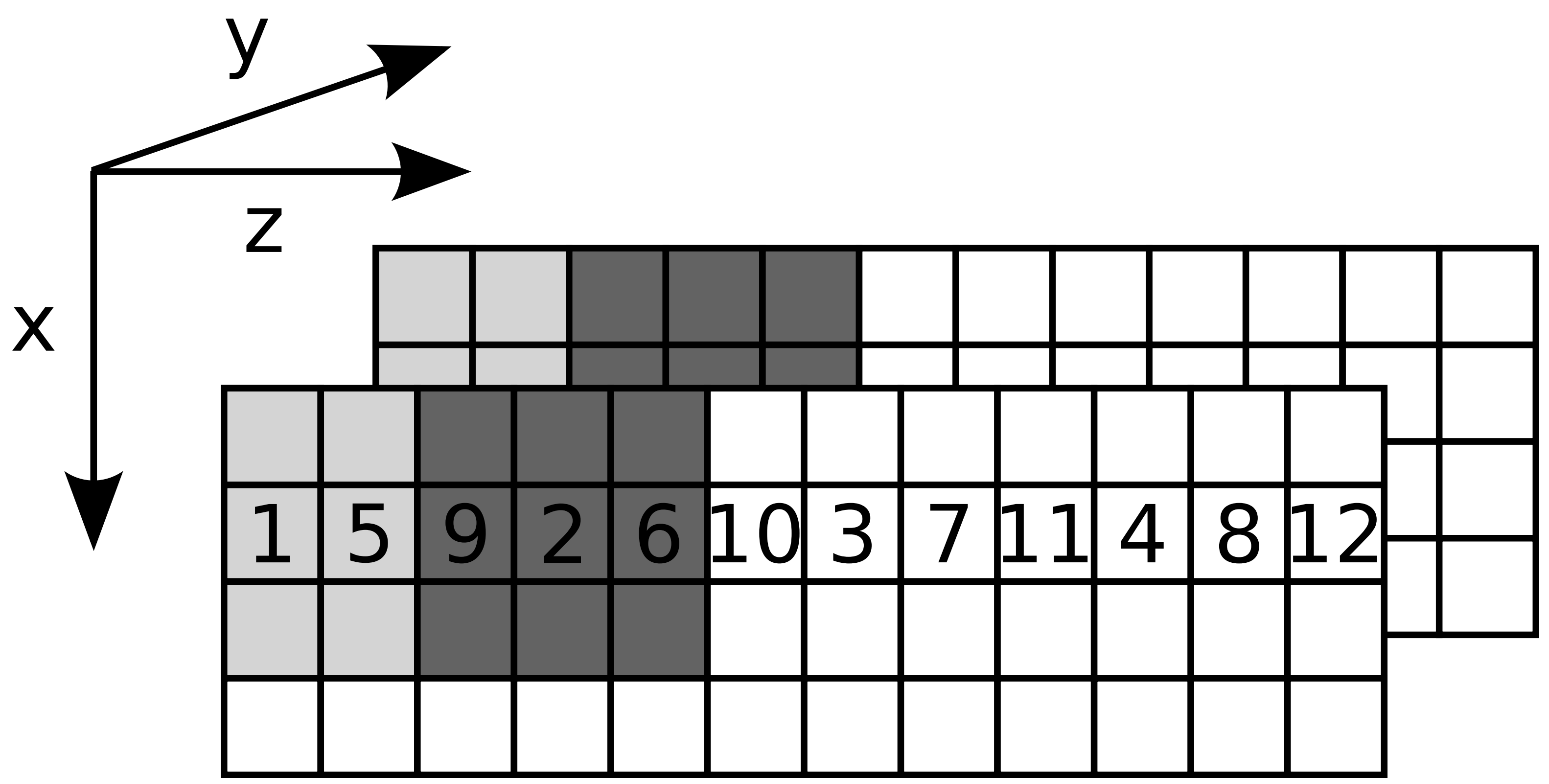
Challenge: Kaczmarz iterations are **sequential.** Each iteration takes many FPGA clock ticks.

Solution: Independent Kaczmarz iterations are used for pipelining.

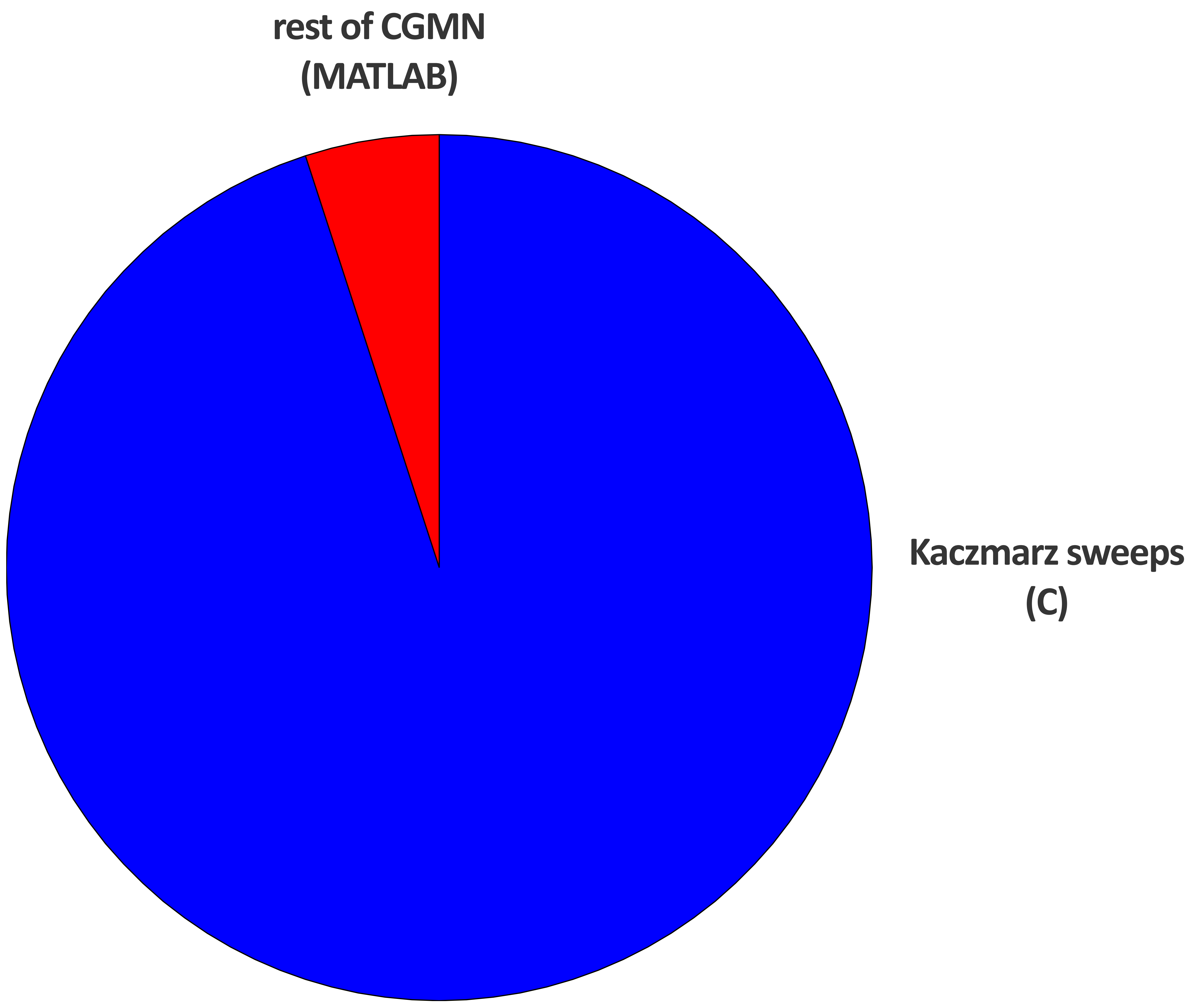
Parallelism through Pipelining

Solve several forward problems.

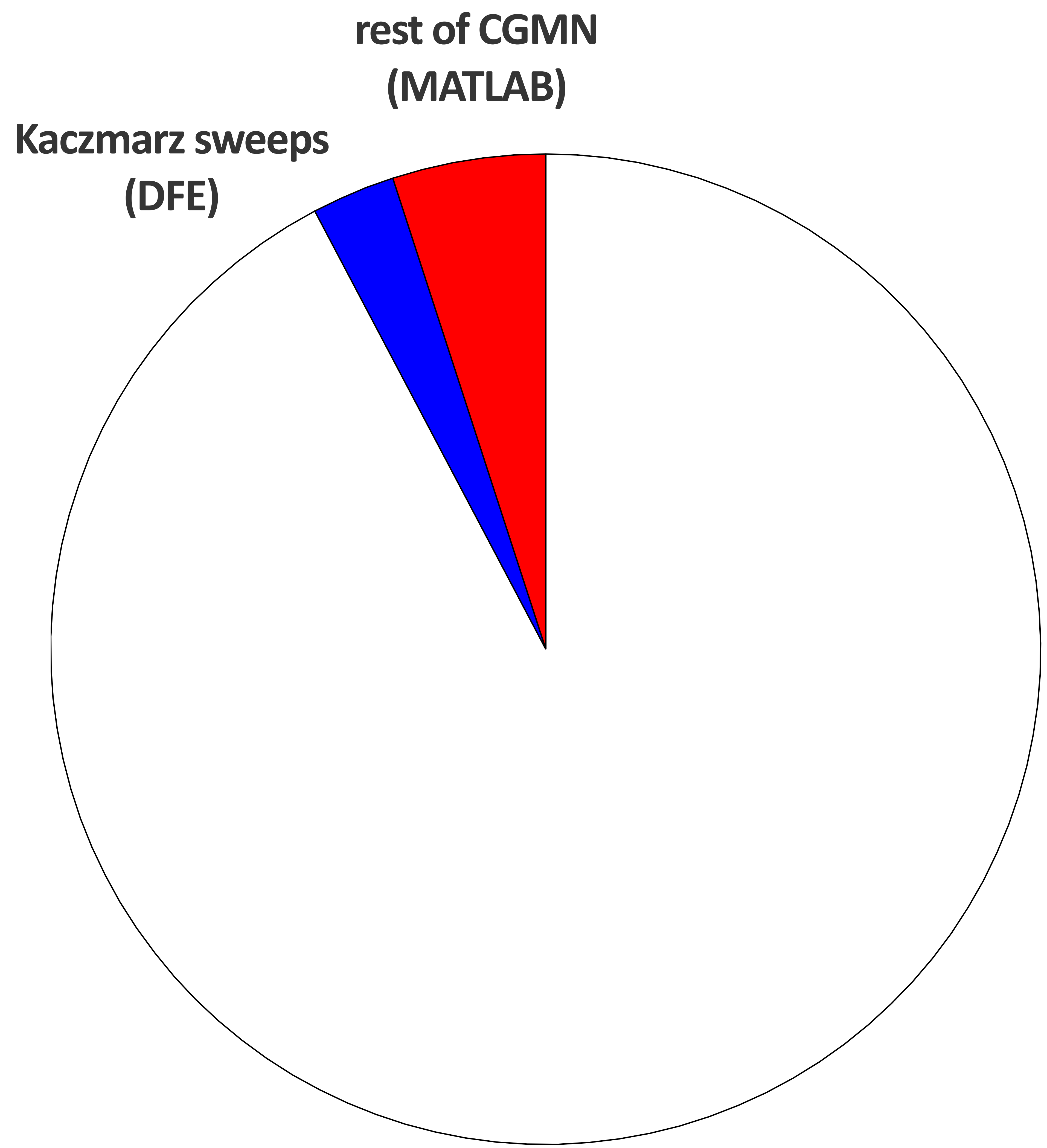
Re-order elements of the Kaczmarz iterate.



Where CGMN used to spend its time



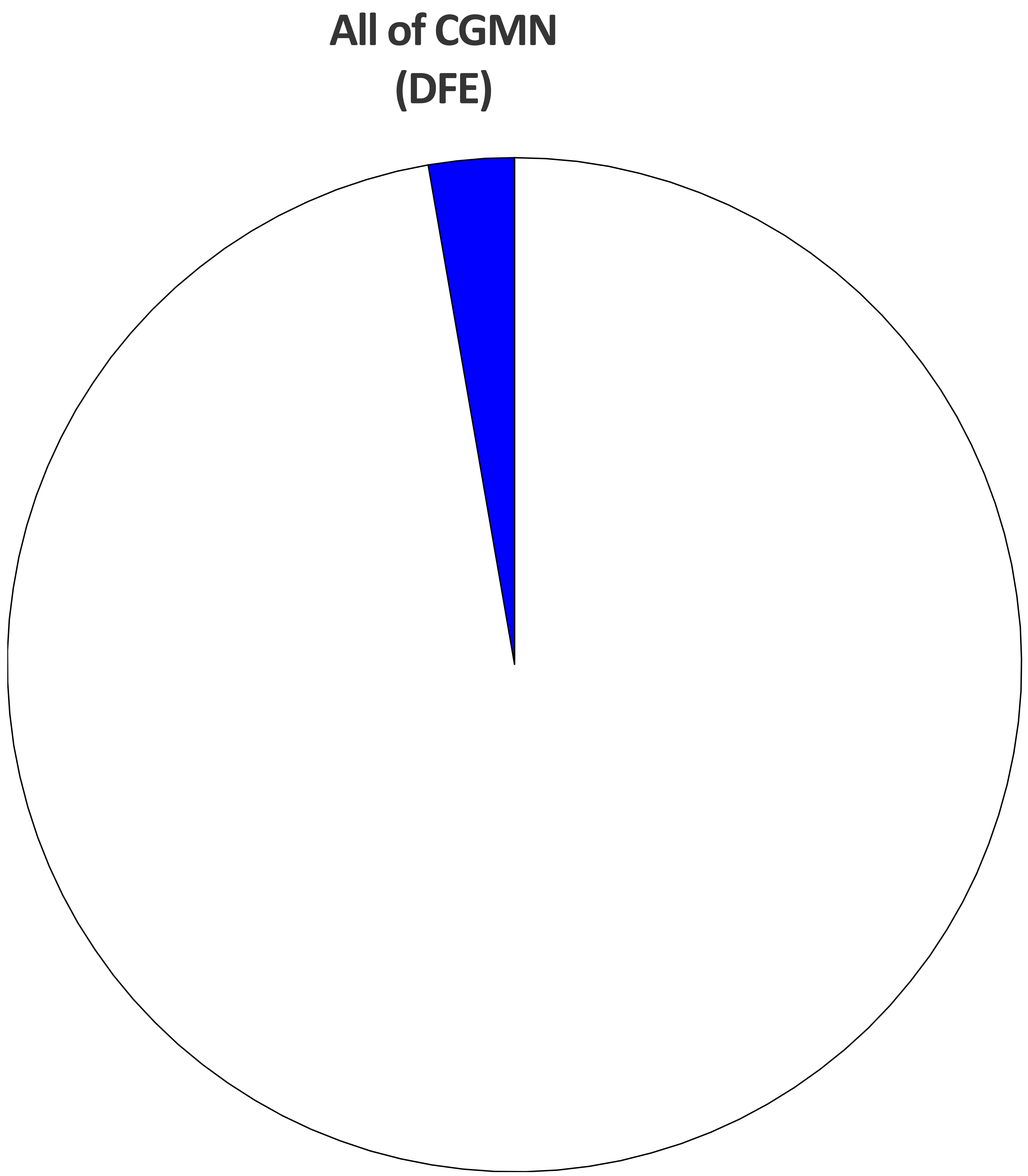
Where CGMN spends its time now*



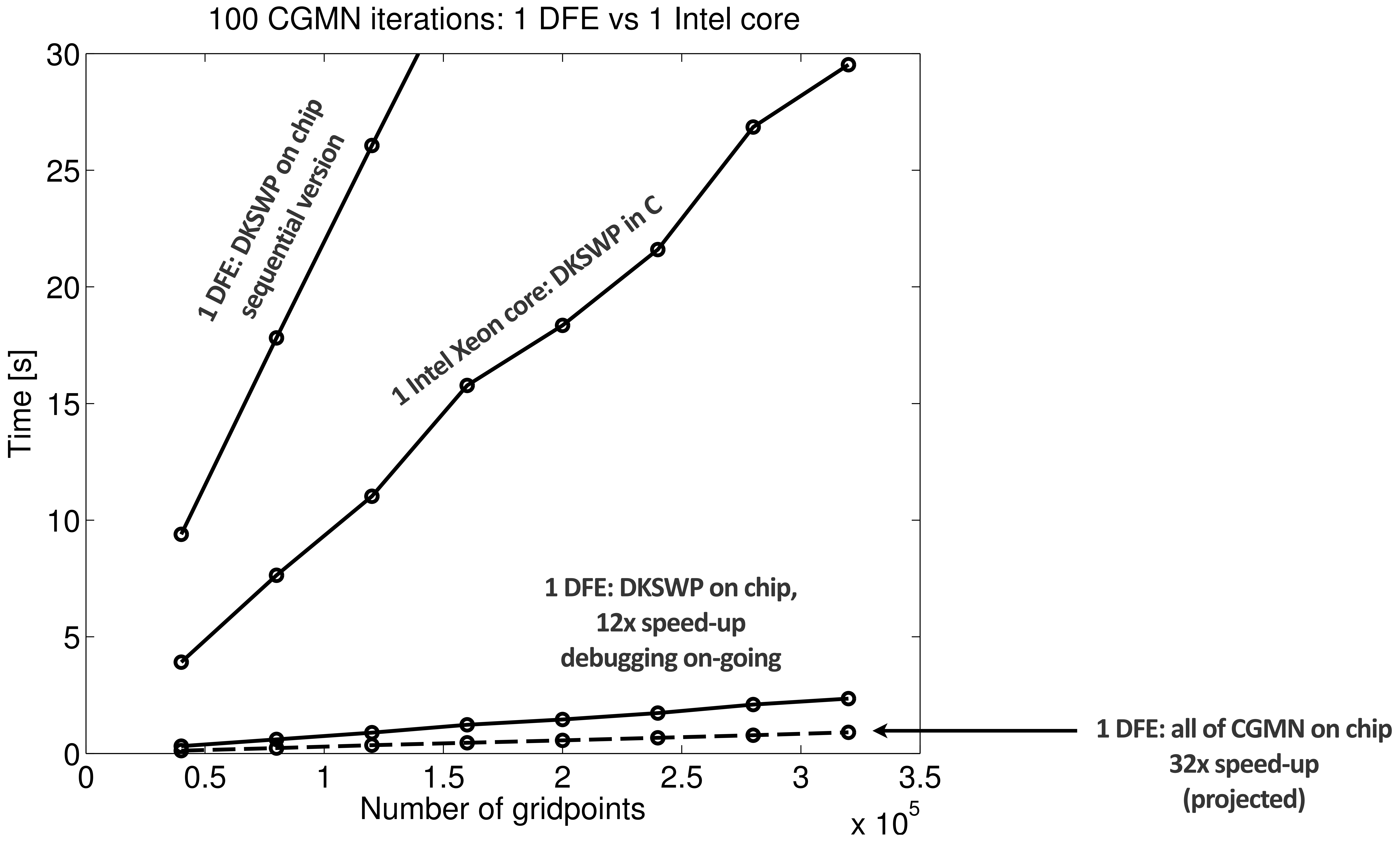
Total speed up: 12 x Intel Xeon core

*Debugging ongoing

Where CGMN will spend its time: projection



Total speed up: 32 x Intel Xeon core (projected)



100 CGMN iterations: 1 DFE vs 1 Intel core
Time [s]
Number of gridpoints
x 10^5
0
5
10
15
20
25
30
0
0.5
1
1.5
2
2.5
3
3.5
1 DFE: DKSWP on chip
sequential version
1 Intel Xeon core: DKSWP in C
1 DFE: DKSWP on chip,
12x speed-up
debugging on-going
1 DFE: all of CGMN on chip
32x speed-up
(projected)

What remains to be done

Fact: Kaczmarz sweeps account for 95% of CGMN time.

Result: Port **all of CGMN** to the DFE.

What remains to be done

Fact: On-chip memory (4 MB) limits block size to **300 x 300** in the two faster dimensions.

Result: Implement **domain decomposition** for larger systems.

What remains to be done

Fact: Currently only 1 (of 4) chips is used.

Result: Parallelize CGMN to CARP-CG [Gordon & Gordon, 2010] *or* solve **several forward problems** at once.

What remains to be done

Estimate: Reading A from memory limits optimizations like increasing FPGA frequency.

Result: Read **only earth model** $\mathbf{m}$ and generate A on the DFE.

Conclusion

Have **implemented** frequency-domain wave simulation using reconfigurable hardware.

A projected acceleration of 32 x 1 Intel Xeon core results from a **dataflow computing** paradigm.

Acknowledgements

Thank you to:

- Henryk Modzelewski for support of the hardware and software used at SLIM.
- Eddie Hung for insight during development and feedback on a draft of this talk.
- Rafael Lago for guidance about Krylov solvers.
- Maxeler Technologies for allowing Art Petrenko to visit for a two week period in November.

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, Petrobras, PGS, Total SA, WesternGeco, and Woodside.

References

Å. Björck and T. Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT Numerical Mathematics, 19(2):145–163, 1979. ISSN 0006-3835. doi: 10.1007/BF01930845. URL http://dx.doi.org/10.1007/BF01930845.

D. Gordon and R. Gordon. Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems. SIAM Journal on Scientific Computing, 27(3):1092–1117, 2005. doi: 10.1137/040609458. URL http://epubs.siam.org/doi/abs/10.1137/040609458.

D. Gordon and R. Gordon. CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs. Parallel Computing, 36(9): 495–515, 2010. ISSN 0167-8191. doi: 10.1016/j.parco.2010.05.004. URL http://www.sciencedirect.com/science/article/pii/S0167819110000827.

S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. Bulletin International de l'Academie Polonaise des Sciences et des Lettres, 35:355–357, 1937.

S. Kaczmarz. Approximate solution of systems of linear equations. International Journal of Control, 57(6):1269–1271, 1993. doi: 10.1080/00207179308934446. (translation)

T. van Leeuwen, D. Gordon, R. Gordon, and F. J. Herrmann. Preconditioning the Helmholtz equation via row-projections. In EAGE technical program. EAGE, 2012. URL https://www.slim.eos.ubc.ca/Publications/Public/Conferences/EAGE/2012/vanleeuwen2012EAGEcarpcg/vanleeuwen2012EAGEcarpcg.pdf.

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, 72(5):SM195–SM211, 2007. doi: 10.1190/1.2759835. URL http://geophysics.geoscienceworld.org/content/72/5/SM195.abstract.

O. Pell, J. Bower, R. Dimond, O. Mencer, and M. J. Flynn. Finite-difference wave propagation modeling on special-purpose dataflow machines. Parallel and Distributed Systems, IEEE Transactions on, 24(5):906–915, 2013. ISSN 1045-9219. doi: 10.1109/TPDS.2012.198.