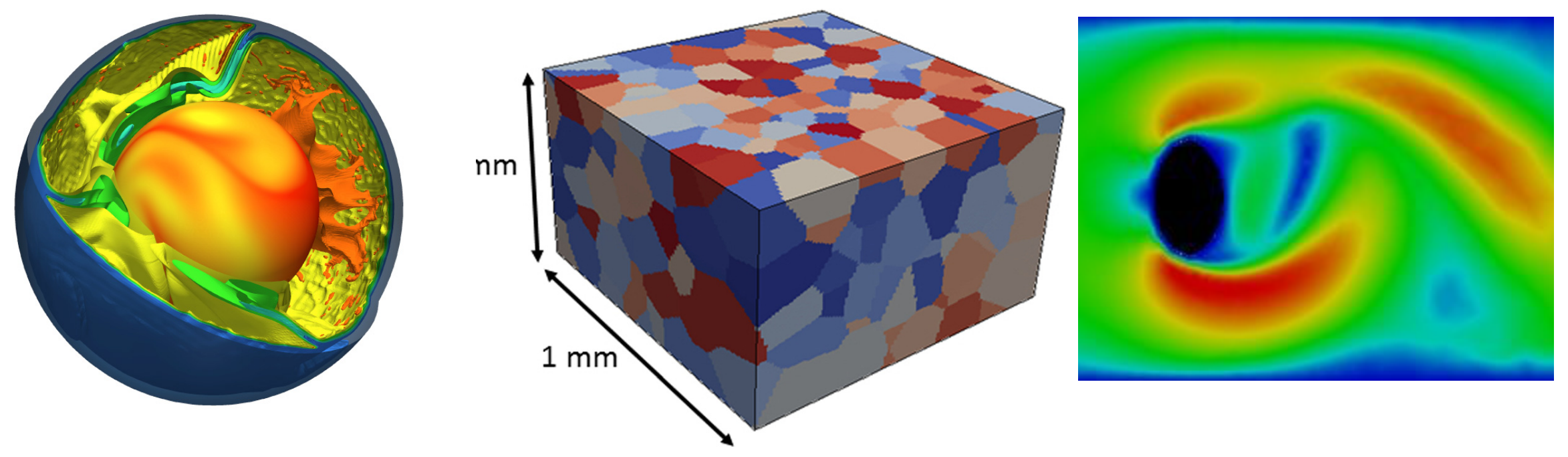
Large-scale Algorithm Design for Parallel FFT-based Simulations on GPUs

Anuva Kulkarni, Franz Franchetti, Jelena Kovačević

Carnegie Mellon University
Electrical & Computer ENGINEERING

FFT-based Simulations

Large-scale scientific simulations involving parallel Fast Fourier Transforms (FFTs) have extreme memory requirements and high communication overhead. It difficult to use GPUs to accelerate legacy Fortran codes because of memory constraints. But GPUs can provide a lot of inexpensive compute power. So how can we port memory-intensive simulations to GPUs?



A possible approach involves:

- **domain decomposition**
- **data compression**
- **pruned, domain-local FFTs**.

Background & Challenges

Consider Moulinec Suquet's Basic Scheme to compute local stress and strain fields in materials , a partial differential equation (PDE) simulation that uses FFTs

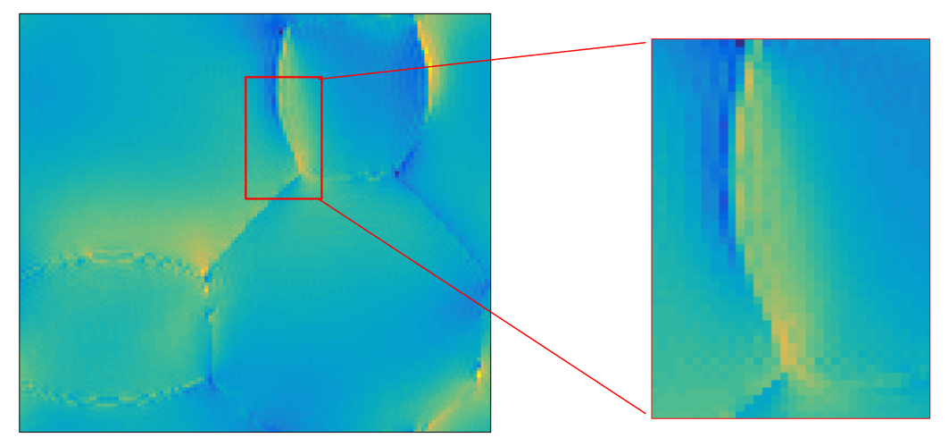
3D Hooke's law:

stress, strain

$$\sigma_{ij} = C_{ijkl} : \epsilon_{kl}$$

3x3x3x3 stiffness tensor

Stress in crystals



Elliptical PDE:

$$C^0_{ijkl} u_{k,lj}(\mathbf{x}) + \tau_{ij,j}(\mathbf{x}) = 0$$

Boundaries are regions of interest

MSC Basic Scheme is solved by convolution with Green's function using FFT.

Algorithm 1 MSC Basic Scheme

```
1: Initialize:
       $\epsilon^0 \leftarrow E$,
       $\sigma^0_{mn}(\mathbf{x}) \leftarrow C_{mnkl}(\mathbf{x}) : \epsilon^0_{kl}(\mathbf{x})$
2: while $e_s > e_{tol}$ do
3:     $\hat{\sigma}^i_{mn}(\xi) \leftarrow \mathrm{FFT}(\sigma^i_{mn}(\mathbf{x}))$
4:     Check convergence
5:     $\Delta\hat{\epsilon}^{i+1}_{kl}(\xi) \leftarrow \hat{\Gamma}_{klmn}(\xi) : \hat{\sigma}^i_{mn}(\xi)$
6:     Update strain: $\hat{\epsilon}^{i+1}_{kl}(\xi) \leftarrow \hat{\epsilon}^i_{kl}(\xi) - \Delta\hat{\epsilon}^{i+1}_{kl}(\xi)$
7:     $\epsilon^{i+1}_{kl}(\mathbf{x}) \leftarrow \mathrm{IFFT}(\hat{\epsilon}^{i+1}_{kl}(\xi))$
8:     Update stress: $\sigma^{i+1}_{mn}(\mathbf{x}) \leftarrow C_{mnkl}(\mathbf{x}) : \epsilon^{i+1}_{kl}(\mathbf{x})$
```

Original method by Moulinec and Suquet

Increasing grid resolution leads to larger problem sizes, which must be run with parallelized code. Large parallel FFT computations on stress tensors means **high memory usage and all-all communication**.

How can we go bigger?

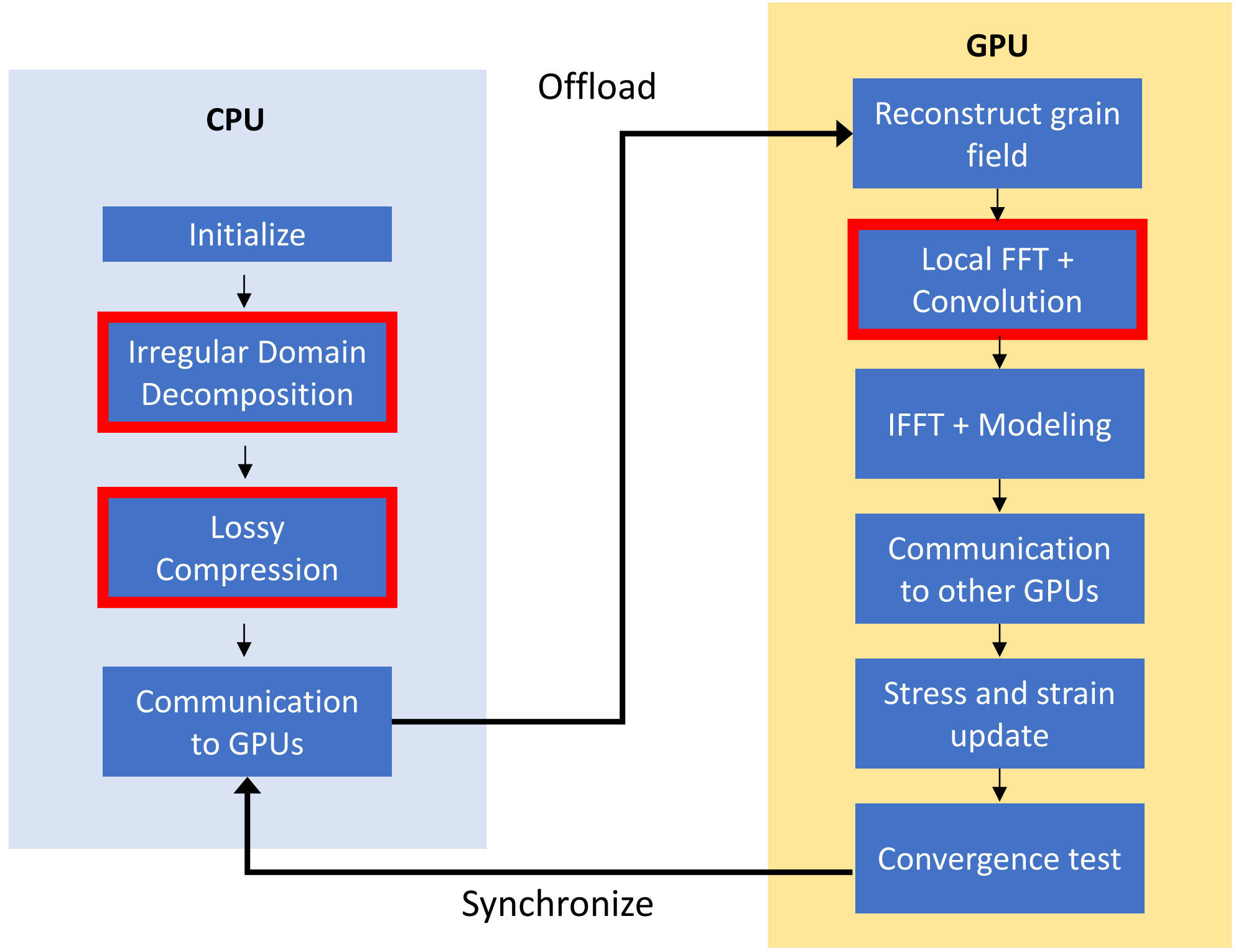
Our solution: An algorithm and software co-design for heterogeneous platforms using irregular domain decomposition and domain local FFTs.

This work presents algorithm development and analysis in MATLAB with GPUs for the proposed solution.

Images:
http://www.ethlife.ethz.ch/archive_articles/120223_Simulation_subduktion_su/1202XX_einseitigeSubduktion_L1-hires.jpeg
http://public.lanl.gov/lebenso/2012-IJP-RLetal-EVPFFT.pdf
https://www.researchgate.net/profile/Tamas_Csanadi/publication/311421026/figure/fig2/AS:489562741972997@1493732261970/ Effect-of-load-and-crystal-orientation-on-crack-formation-during-scratch-test-b.ppm

Proposed Method

The proposed MSC Alternate Scheme is a co-design of algorithm and software for heterogeneous platforms. It enables scaling of stress-strain simulations to large grids by overcoming high memory requirements and communication bottlenecks.



CPU
Initialize
Irregular Domain Decomposition
Lossy Compression
Communication to GPUs

Offload

GPU
Reconstruct grain field
Local FFT + Convolution
IFFT + Modeling
Communication to other GPUs
Stress and strain update
Convergence test

Synchronize

Irregular domains: Stress/strain in grains is smooth, hence grains are domains assigned to each GPU. Grains of size N x N x M used.

Lossy Compression with B-splines:
B-splines are composed of polynomial pieces and are generalizations of Bezier curves with breakpoints called 'knots'. For a knot sequence,

$$t_0 \leq t_1 \leq \cdots \leq t_{N+1}$$

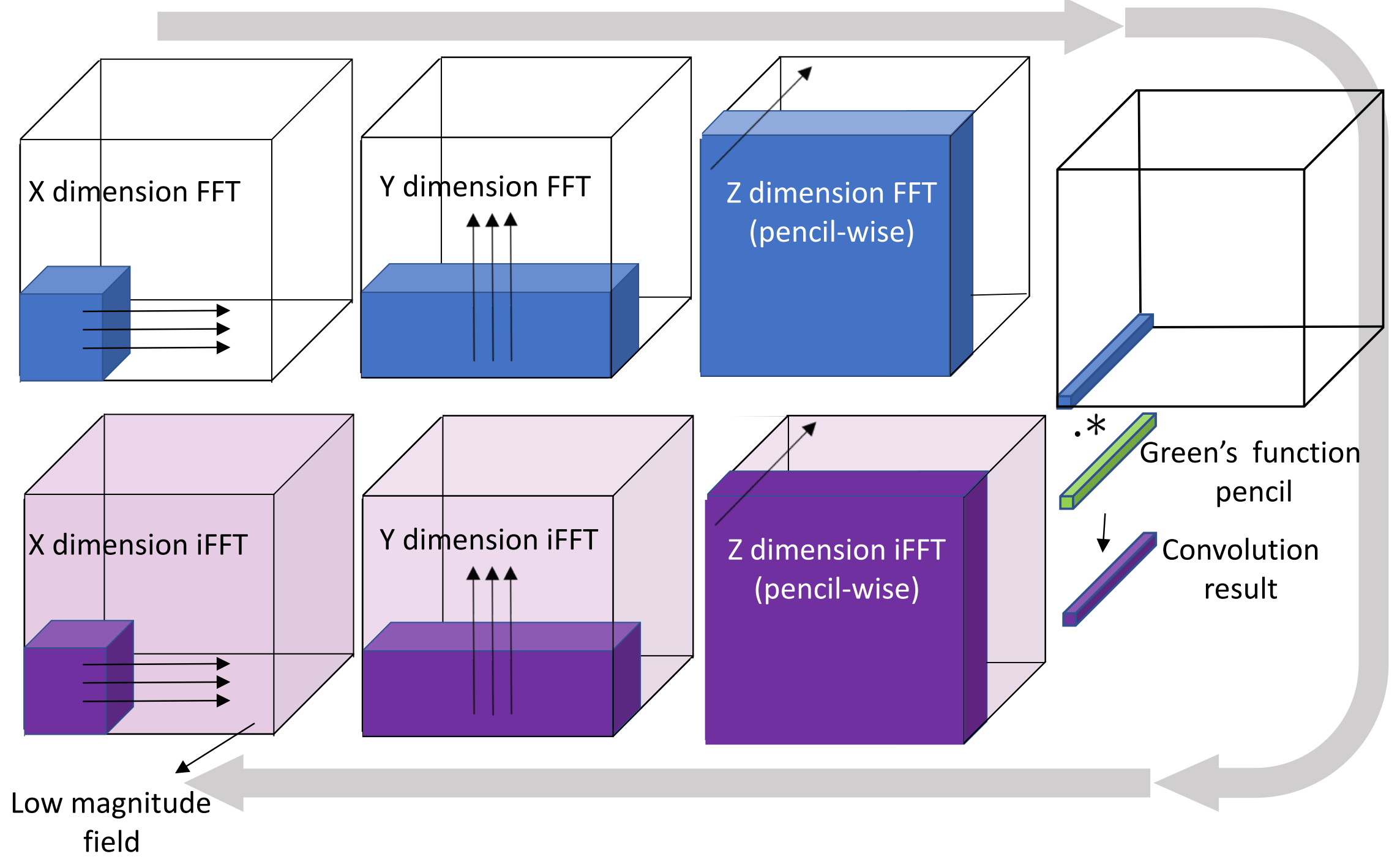
The i^{th} B-spline basis function of order j is

$$B_{i,j+1}(x) = \alpha_{i,j+1}(x) B_{i,j}(x) + [1 - \alpha_{i+1,j+1}(x)] B_{i+1,j}(x)$$

where

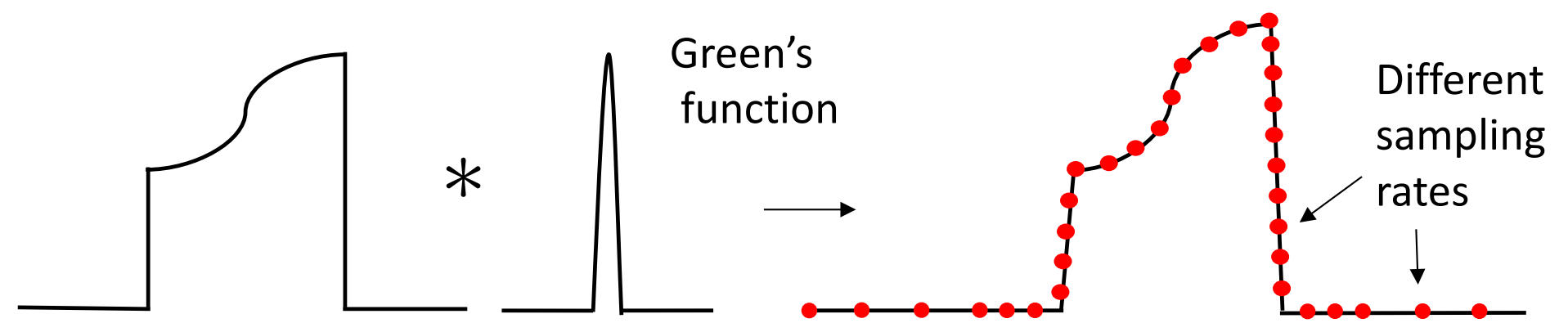
$$\alpha_{i,j}(x) = \begin{cases} \dfrac{x - t_i}{t_{i+j} - t_i} & \text{if} \quad t_{i+j} \neq t_i \\ 0 & \text{otherwise} \end{cases}$$

Domain local FFT: Performed on GPU for each domain. Platform used is MATLAB-GPU interface, using **NVIDIA Quadro K2200.**



X dimension FFT
Y dimension FFT
Z dimension FFT (pencil-wise)
.* Green's function pencil
Convolution result
X dimension iFFT
Y dimension iFFT
Z dimension iFFT (pencil-wise)
Low magnitude field

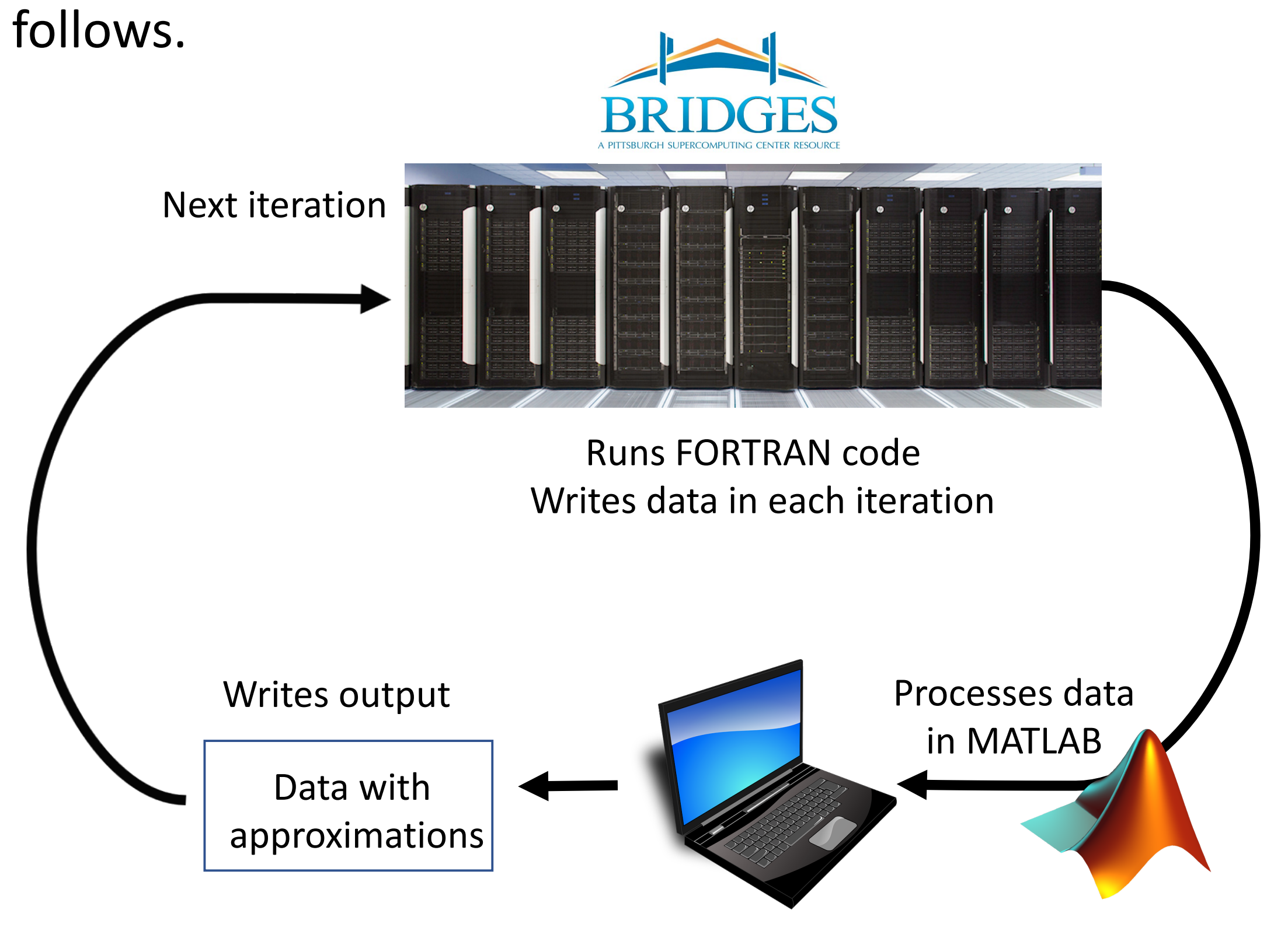
Adaptive downsampling used to reduce storage of convolution result



*
Green's function
Different sampling rates

Phase I: Successes

Computational aspects for prototype development:
MATLAB-FORTRAN workflow for convergence analysis is as follows.

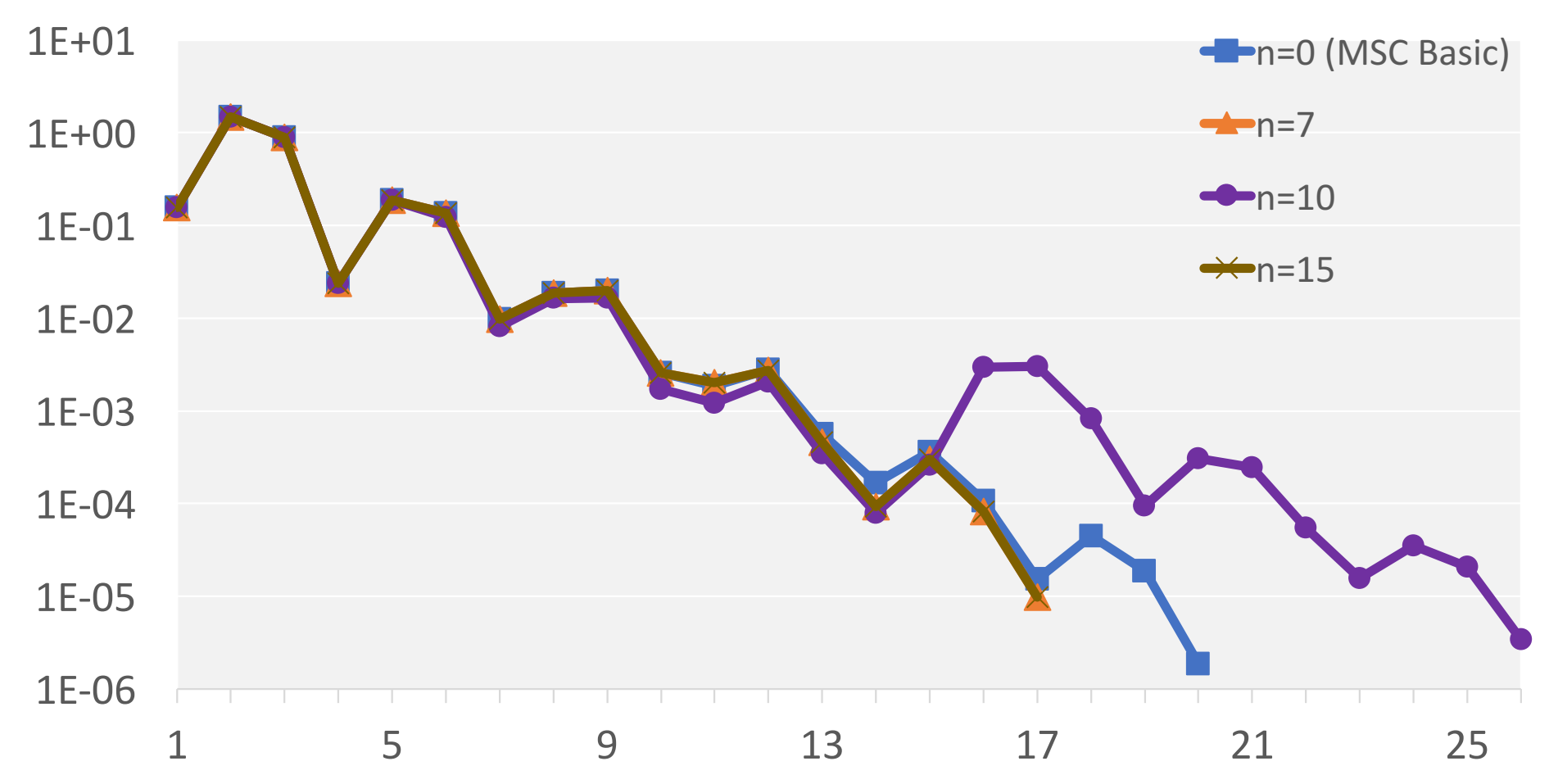


BRIDGES
Next iteration
Runs FORTRAN code
Writes data in each iteration
Writes output
Data with approximations
Processes data in MATLAB

Analysis of results:

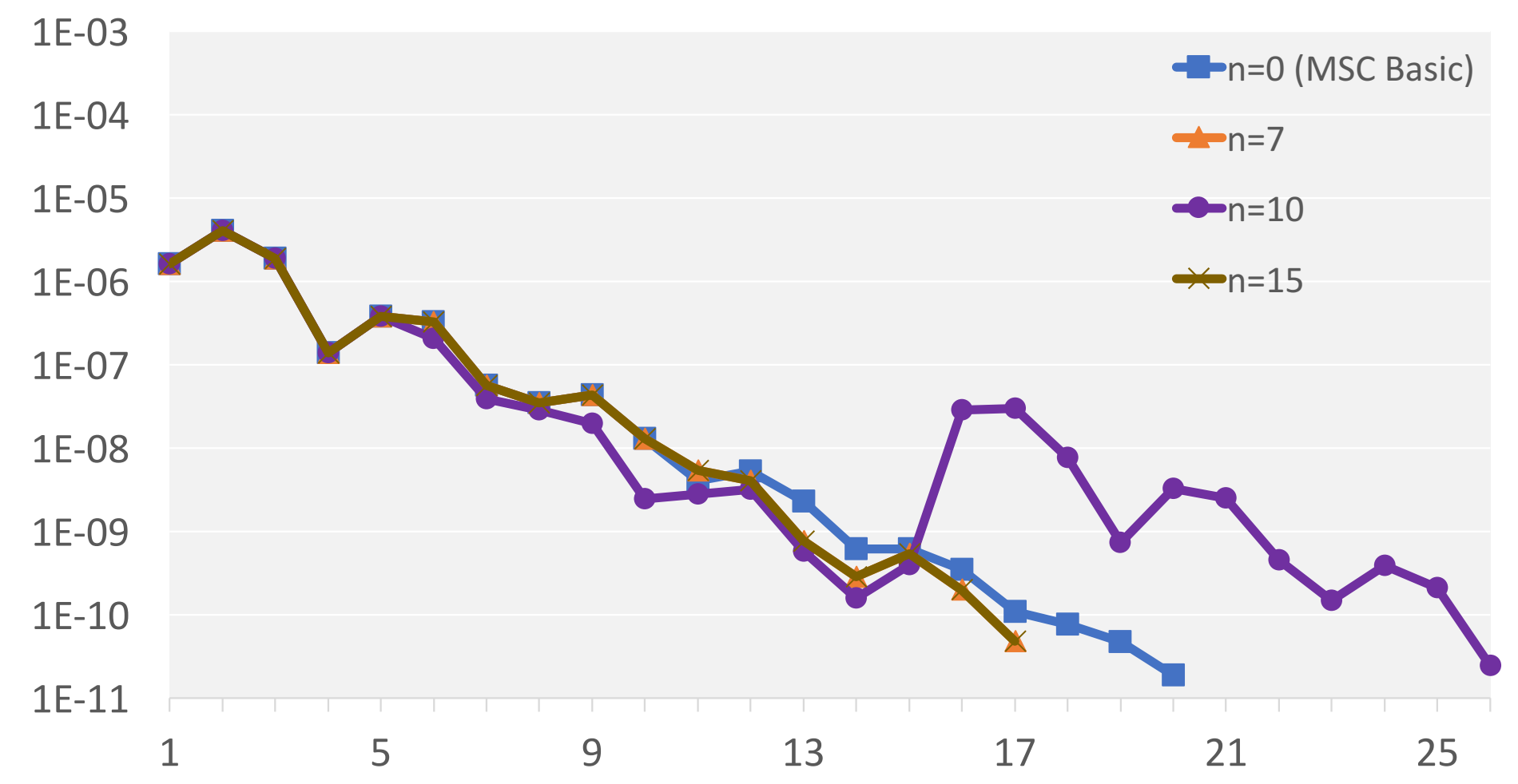
Convergence of Stress

Log Error in Stress field vs. Iteration number



Convergence of Strain

Log Error in Strain field vs. Iteration number



Lossy Compression Reconstruction Error (RE) and Compression Ratio (CR) for various grain sizes:

Grain Size	64^3	128^3	$128^2 \times 8$	$256^2 \times 8$	$512^2 \times 8$
RE	1.49%	1.11%	0.52 %	0.44%	0.41%
CR	2.48	8.01	5.47	13	28.69

Table 1: Reconstruction error and compression ratio for lossy compression in various grain sizes.

MATLAB-GPU Interface to obtain proof-of-concept results for domain-local FFTs. 700 x 700 x 700 size convolution possible grain-by-grain with irregular domain decomposition on NVIDIA Quadro K2200 with 640 CUDA cores and 4 GB GPU memory.

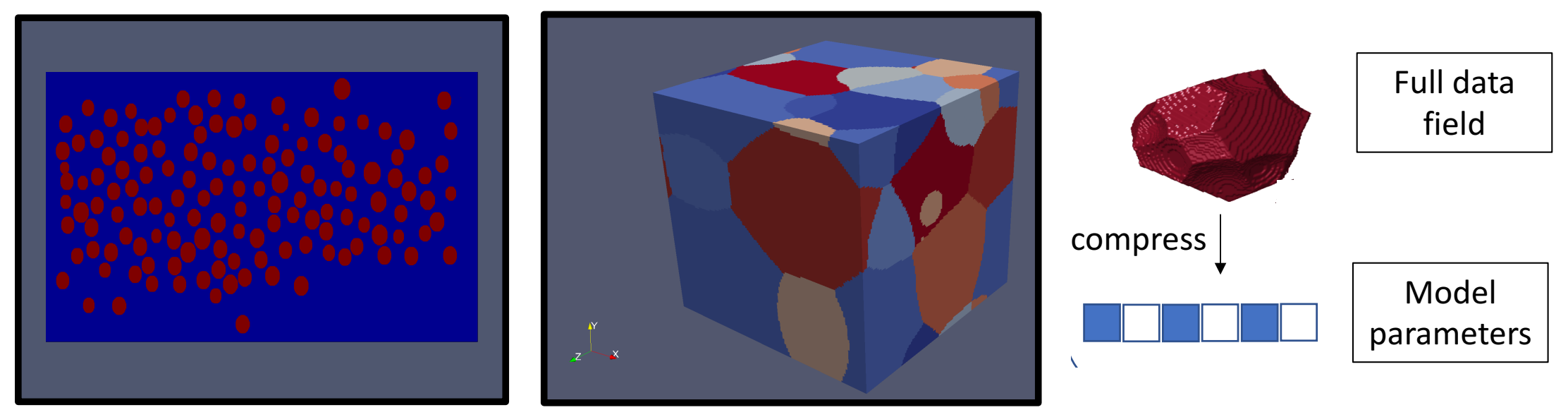
Grain size	64^3	128^3	$128^2 \times 8$	$256^2 \times 8$	$512^2 \times 8$
Error(%)	1.79	3.03	$3.74 \cdot 10^{-14}$	$4.11 \cdot 10^{-14}$	$4.32 \cdot 10^{-14}$

Table 2: Error in convolution by domain-local FFT method

Phase II: Plans

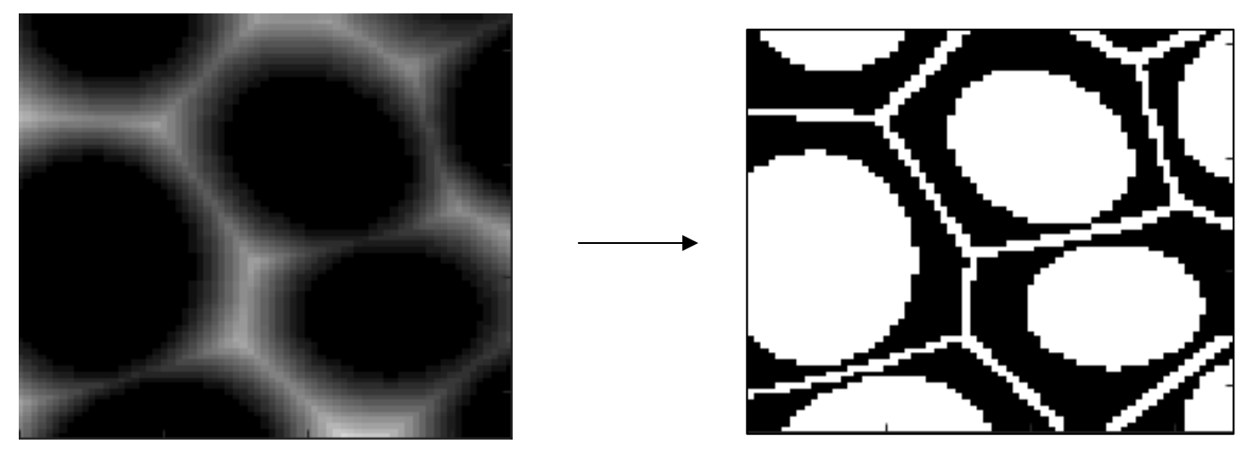
Thrust 1: Domain decomposition framework for various datasets

- Extend to different datasets with irregular grains



Full data field
compress
Model parameters

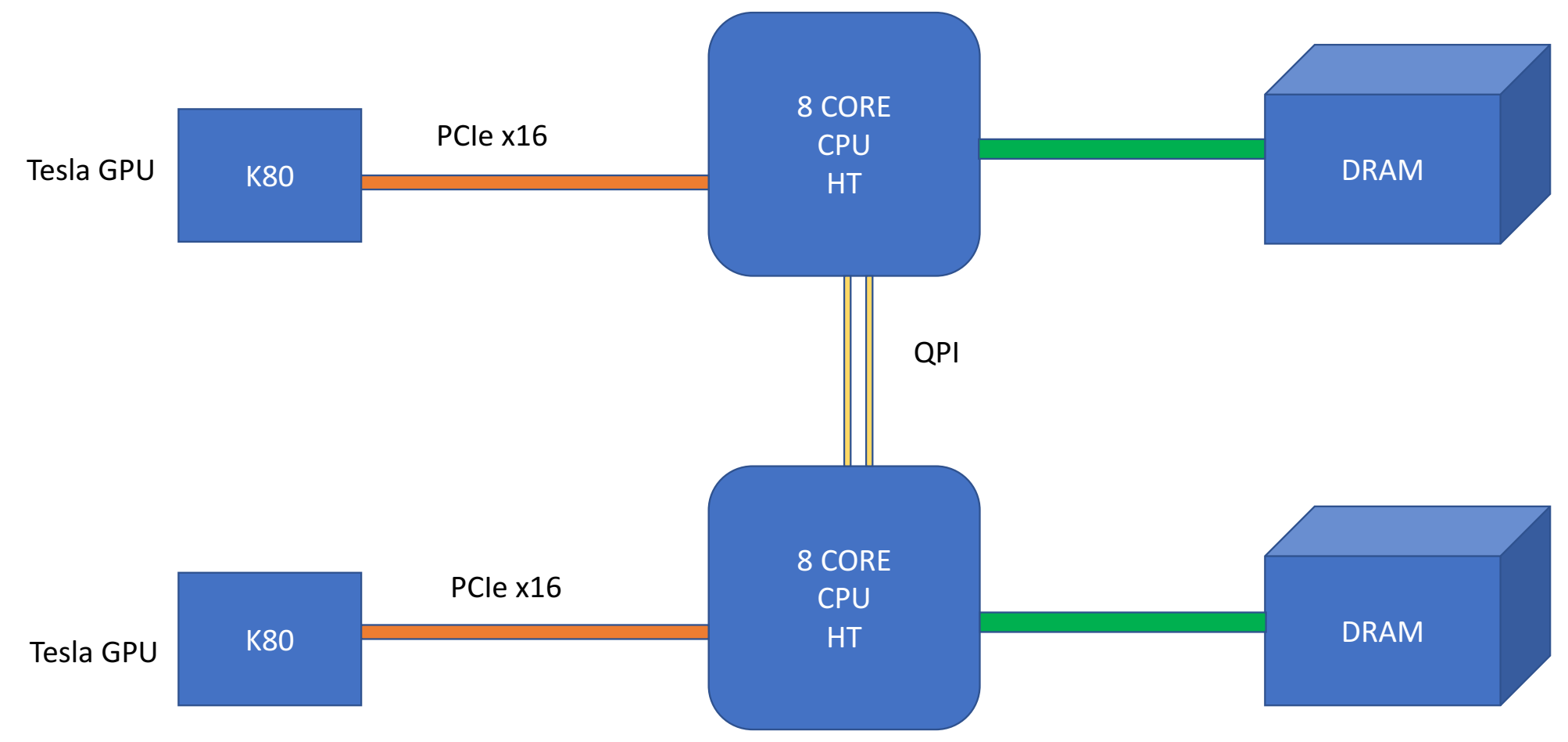
Irregular grain as a 'packing' of regular shapes (Eg., ellipsoids). Synthetic dataset generated using watershed algorithm, ellipsoid packing and Voronoi maps.



Distance map after random ellipsoid packing

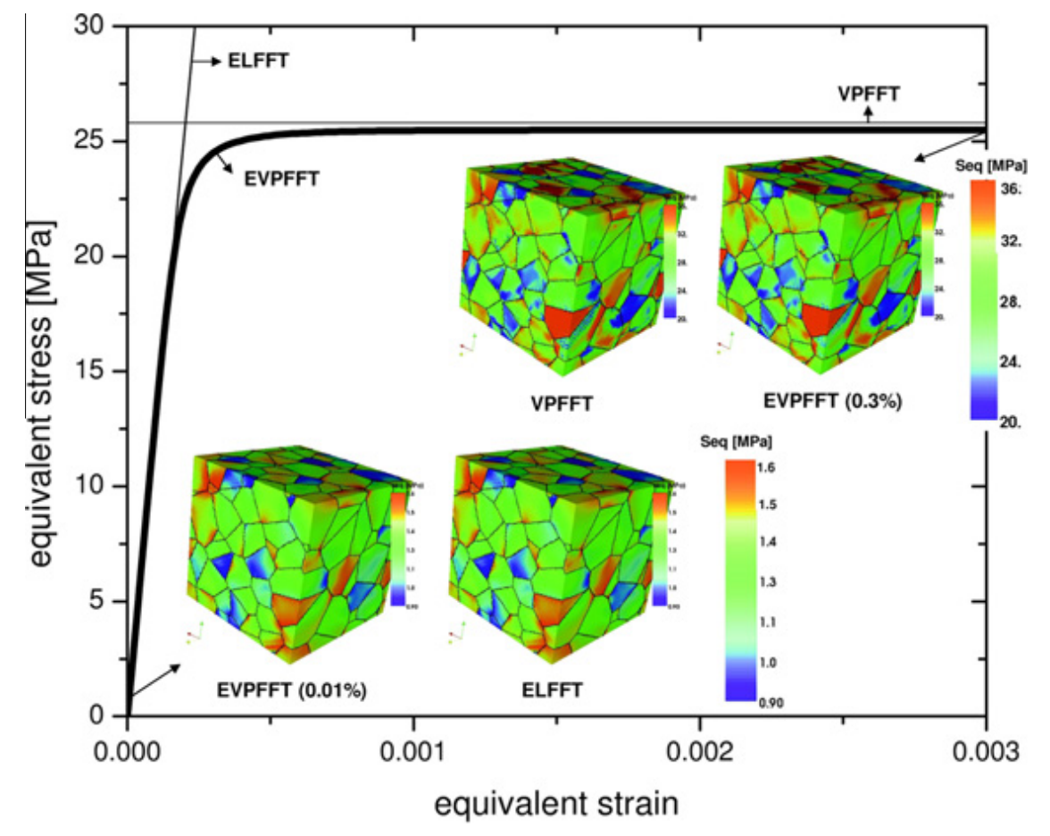
Watershed algorithm gives lines for Voronoi diagram

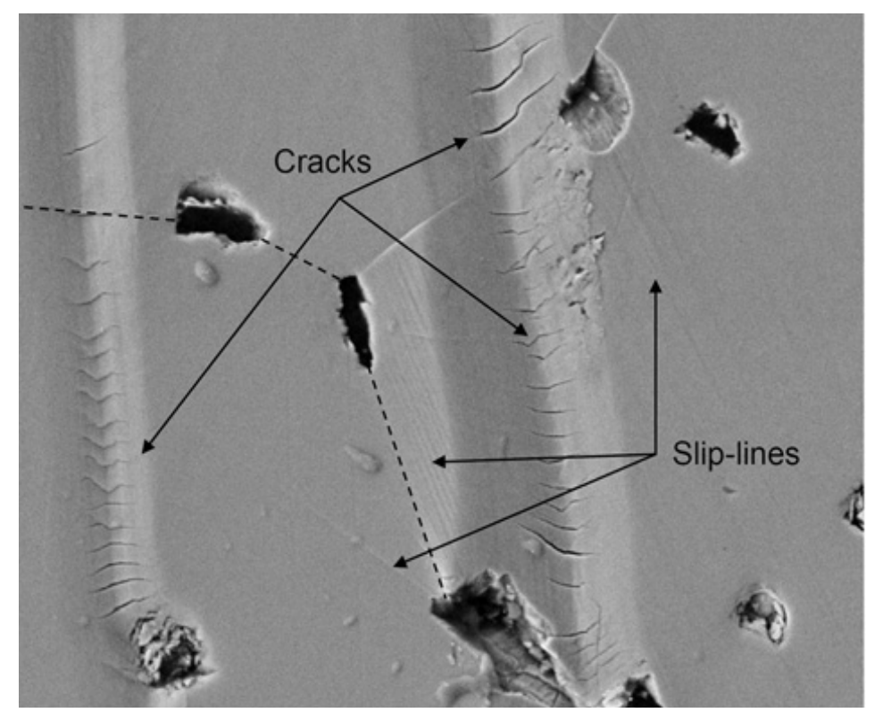
Thrust 2: Deployment on heterogeneous system with Tesla K80 and Fortran-GPU testing



Tesla GPU
K80
PCIe x16
8 CORE CPU HT
DRAM
QPI
Tesla GPU
K80
PCIe x16
8 CORE CPU HT
DRAM

Thrust 3: Extend to visco-plastic code [2] which includes deformation of crystals and studies cracking and fracture formation





Thrust 4: Use FFTX [3], a new framework for building high performance FFT-based applications on exascale machines, for domain-local FFTs.
FFTX is backwards compatible to FFTW and has a SPIRAL-based back end for advanced performance optimizations.

Acknowledgements

The authors would like to thank Dr. Anthony Rollett, Dr. Vahid Tari at CMU and Dr. Anirban Jana and Dr. Roberto Gomez at Pittsburgh Supercomputing Center for all their assistance and collaboration with this project.

References

[1] H. Moulinec and P. Suquet. 1998. A numerical method for computing the over-all response of nonlinear composites with complex microstructure. Computer methods in applied mechanics and engineering 157, 1-2 (1998), 69–94.
[2] R. A. Lebensohn. 2001. N-site modeling of a 3D viscoplastic polycrystal using fast Fourier transform. Acta Materialia 49, 14 (2001), 2723–2737.
[3] F. Franchetti et al. 2018. FFTX and SpectralPack: A First Look. PFFT Workshop, 2018.