# Algorithm Design at Scale: Porting Parallel FFT-based Fortran Simulations to GPUs

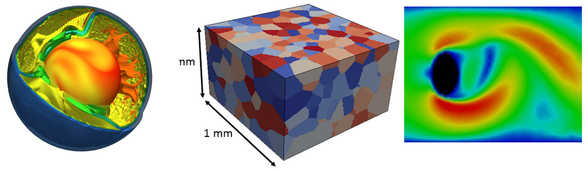
Anuva Kulkarni, Jelena Kovačević, Franz Franchetti

Carnegie Mellon University — Pittsburgh Pennsylvania 1900

Electrical & Computer ENGINEERING

## GPUs for Scientific Computing

General Purpose GPU for scientific computing:
- provide a lot of compute power
- are relatively cheap compared to large clusters.




However, accelerating legacy Fortran scientific codes with GPUs involves **rewriting and rethinking**.
For eg., consider the large memory requirements of the simulation (see table) and memory constraints of GPUs (few GB on-chip).

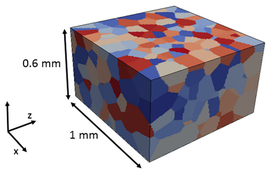
**Hooke's Law Simulation**

| Input data size | $32^3$ | $64^3$ | $128^3$ | $256^3$ | $512^3$ | $1024^3$ |
|---|---|---|---|---|---|---|
| Memory required (GB) | 0.07 | 0.55 | 4.44 | 35.5 | 284 | 2272 |

## Hooke's Law Simulation: Scaling Parallel FFTs

**3D Hooke's law simulates stress and strain (3x3 tensors) in crystals:**

$$\underset{\text{stress}}{\sigma_{ij}} = \underset{\text{3x3x3x3 stiffness tensor}}{C_{ijkl}} : \underset{\text{strain}}{\epsilon_{kl}}$$




**Partial Differential Equation:**

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

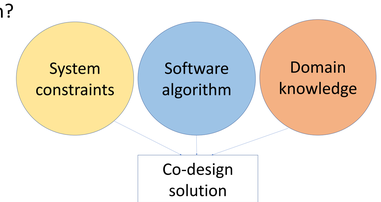
**Original method by Moulinec and Suquet**

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: $\hat{\sigma}^i_{mn}(\boldsymbol{\xi}) \leftarrow \text{FFT}(\sigma^i_{mn}(\mathbf{x}))$
4: Check convergence
5: $\Delta\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}) \leftarrow \hat{\Gamma}_{klmn}(\boldsymbol{\xi}) : \hat{\sigma}^i_{mn}(\boldsymbol{\xi})$
6: Update strain: $\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}) \leftarrow \hat{\epsilon}^i_{kl}(\boldsymbol{\xi}) - \Delta\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi})$
7: $\epsilon^{i+1}_{kl}(\mathbf{x}) \leftarrow \text{IFFT}(\hat{\epsilon}^{i+1}_{kl}(\boldsymbol{\xi}))$
8: Update stress: $\sigma^{i+1}_{mn}(\mathbf{x}) \leftarrow C_{mnkl}(\mathbf{x}) : \epsilon^{i+1}_{kl}(\mathbf{x})$

Increasing grid resolution leads to larger problem sizes.

Large parallel FFTs = **prohibitive memory requirement and all-all communication**.

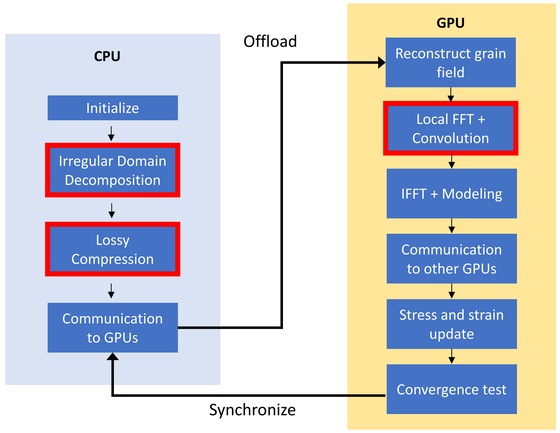
Current MPI simulation limit is $1024^3$ grid
Can using GPUs help us scale this further with a co-design solution?



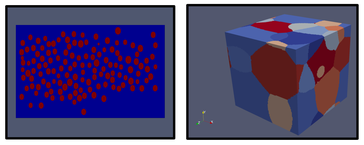

Images:
http://www.ethlife.ethz.ch/archive_articles/120223_Simulation_subduktion_su/1202XX_einseitigeSubduktion_L1-hires.jpeg
http://public.lanl.gov/lebenso/2012-IJP-RLetal-EVPFFT.pdf
https://www.researchgate.net/profile/Tamas_Csanadi/publication/311421026/figure/fig2/AS:489562741972997@1493732261970/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

## Co-design Solution

**This work presents algorithm development and analysis of the proposed solution in MATLAB.**




**Irregular domains:** Microstructure is decomposed into grains, which are the irregular domains. We are working with 2D and 3D spherical domains, elliptical domains, triangular/polygonal domains and cubic domains.

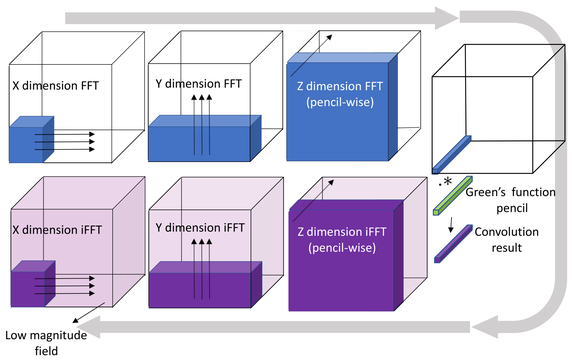


**Lossy compression:** Adaptive downsampling used to reduce storage of convolution result. Sampling strategy is chosen based on desired reconstruction accuracy and a given prior on the field being sampled. For example, from the Eshelby solution, we could have:

**Eshelby tensors**

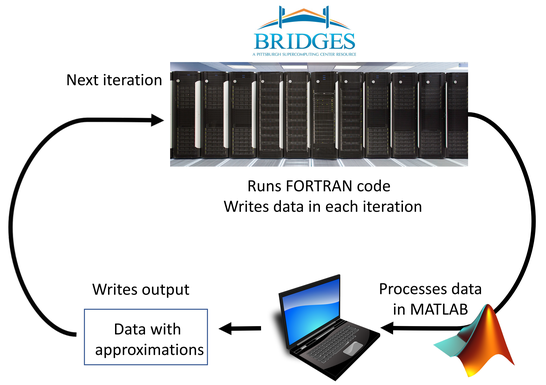
$$\begin{aligned}8\pi(1-\nu)D_{ijkl}(x) =& 8\pi(1-\nu)S_{ijkl}(\lambda) + 2\nu\delta_{kl}x_i I_{I,j}(\lambda) \\ &+(1-\nu)\left(\delta_{il}x_k I_{K,j}(\lambda) + \delta_{jl}x_k I_{K,i}(\lambda)\right. \\ &\left.+\ \delta_{ik}x_l I_{L,j}(\lambda) + \delta_{jk}x_l I_{L,i}(\lambda)\right) \\ &-\delta_{ij}x_k\left(I_K(\lambda) - a_I^2 I_{KI}(\lambda)\right)_{,l} \\ &-(\delta_{ik}x_j + \delta_{jk}x_i)\left(I_J(\lambda) - a_I^2 I_{IJ}(\lambda)\right)_{,l} \\ &-(\delta_{il}x_j + \delta_{jl}x_i)\left(I_J(\lambda) - a_I^2 I_{IJ}(\lambda)\right)_{,k} \\ &-x_i x_j\left(I_J(\lambda) - a_I^2 I_{IJ}(\lambda)\right)_{,lk}.\end{aligned}$$

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

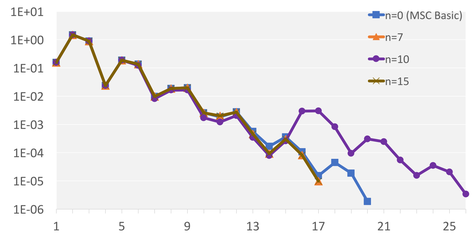



## Phase I: Successes

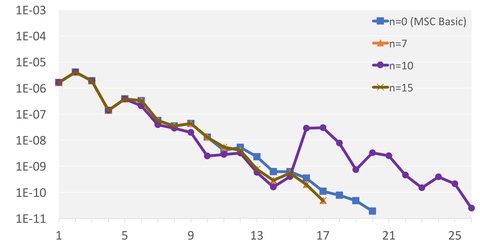
**Computational aspects for prototype development**: **MATLAB-FORTRAN workflow** for convergence analysis is as follows.




**Convergence of Stress**
Log Error in Stress field vs. Iteration number



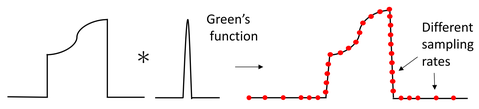

**Convergence of Strain**
Log Error in Strain field vs. Iteration number



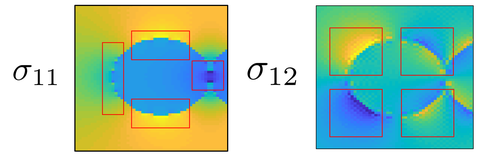

**MATLAB-GPU** Interface to obtain proof-of-concept results for domain-local FFTs. 700 x 700 x 700 size convolution possible grain-by-grain with irregular domain decomposition on NVIDIA Quadro K2200 with 640 CUDA cores and 4 GB GPU memory.

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

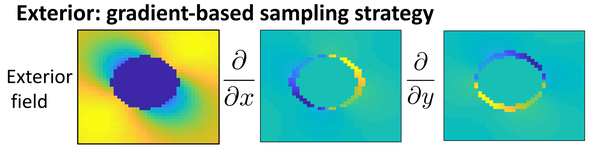
**Lossy compression:** Convolution with impulse-like Greens' function results in a solution which can be sampled at lower rates as distance from the grain increases.
Identifying the regions of interest correctly is important.
Region of interest is sampled at high rates.




**Interior: Identifying contributions from surface stresses**

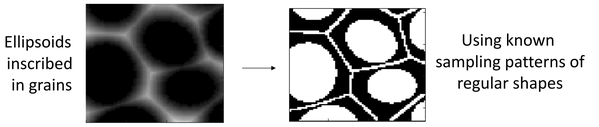



**Exterior: gradient-based sampling strategy**




## Phase II: Plans

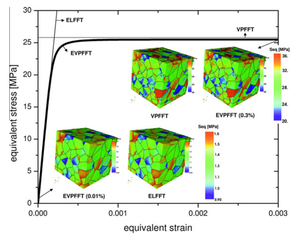
**Thrust 1: Domain decomposition framework for irregular grains**
Irregular grain as a 'packing' of regular shapes (Eg., ellipsoids), for which sampling patterns have been derived

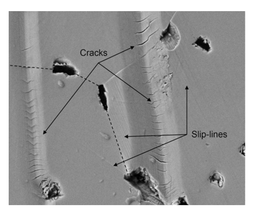



**Thrust 2: Deployment on heterogeneous system with Tesla K80 and Fortran-GPU testing**

Tesla K80
- 24 GB of GDDR5 memory
- 480 GB/s aggregate memory bandwidth

**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. This work used XSEDE, supported by NSF grant ACI-1548562.

### 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.