# Large-Scale Algorithm Design for Parallel FFT-based Simulations on GPUs

Anuva Kulkarni, Franz Franchetti, Jelena Kovačević
Department of Electrical and Computer Engineering
Carnegie Mellon University
Email: {anuvak, franzf, jelenak}@cmu.edu

***Abstract*—We describe a co-design of algorithm and software for high-performance simulation of a partial differential equation (PDE) numerical solver for large-scale datasets. Large-scale scientific simulations involving parallel Fast Fourier Transforms (FFTs) have extreme memory requirements and high communication cost. This prevents scaling to higher grid sizes, which is necessary for high resolution analysis. Moreover, legacy scientific codes used for numerical simulations are not highly efficient or scalable, and are not adapted for modern hardware accelerators. With a view to overcoming these challenges, our proposed method uses signal processing techniques such as lossy compression and domain-local FFTs to lower iteration cost without adversely impacting accuracy of the result.**

***Index Terms*—Irregular domain decomposition, algorithm design, GPU, lossy compression**

## I. Introduction

Scientific simulations developed in Fortran are used to study and model physical phenomena in many fields including physics, biology and materials science. These Fortran legacy codes are handed down and often evolve over the course of decades because rewriting them from scratch is both risky and costly. Discrete models are simulated on refined grids for high resolution analysis of interesting phenomena. However, refining a grid increases the scale of the problem, and the scaled implementations of old codes face significant memory and communication bottlenecks on modern multi-core machines. For example, large-scale FFT-based PDE solvers face limitations in scaling because all-all communication in parallel FFTs dominates runtime [1]. Hardware platforms such as Graphical Processing Units (GPUs) provide a lot of compute power and are a relatively inexpensive alternative for large computing clusters. But high memory requirement of Fortran code prevents GPUs from being used since they have small on-chip memory (less than 16 GB). *How can legacy scientific simulations be scaled to larger grid sizes using modern hardware such as GPUs?*

First, it is necessary to recognize that a co-design of algorithm and software is required to achieve maximum performance on modern hardware platforms [2]. Algorithms which address the issue of high communication must be developed using knowledge of hardware, while software design must take into account tolerable approximation error and potential savings in resources used. To show our approach towards developing such a co-design solution, we consider a particular simulation of stress-strain in composites, the Moulinec Suquet Composite (MSC) Basic Scheme. In this scheme, a PDE is solved iteratively using FFT-based convolution with the Green's function corresponding to the PDE, which is rapidly decaying in space domain but dense in Fourier domain [3]. Table I shows the prohibitive memory requirement for various simulation sizes.

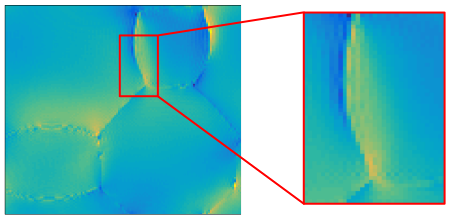
Fig. 1. Single component of stress field tensor in a sample composite microstructure simulated using MSC Basic Scheme. The image shows behavior of stress at the grain boundary between 3 grains, which is of interest.

TABLE I
Memory requirement of MSC Basic Scheme (serial version)

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

**Contribution.** In this paper, we present MSC Alternate Scheme, an algorithm design for heterogeneous platforms. Highlights of the method are:

1) Irregular domain decomposition
2) Lossy compression
3) Domain-local FFTs on different parts of the 3D volume.

(1) uses the observation that irregular grains are highly compressible domains because of the smooth-varying fields within them. (2) uses splines to compress data and hence reduce communication overhead. (3) ensures that convolutions can be performed on the limited memory of a GPU.

**Synopsis.** The next section describes MSC Basic Scheme, followed by sections describing MSC Alternate Scheme and early results.

## II. Background

**Tensor notation.** Throughout this paper, Einstein notation is used to represent tensor components and operations. Thus, $A_{mn}$ refers to component $(m, n)$ of the rank-2 tensor $A$.

Repetition of indices implies a summation over those particular indices. An important tensor operation is the contraction of indices (denoted by ':'). E.g., $C_{mnk\ell}{:}D_{mn} = \sum_m \sum_n C_{mnk\ell} D_{mn} = E_{k\ell}$ and yields a rank-2 tensor.

**MSC Basic Scheme.** The MSC Basic Scheme is a fixed-point iterative numerical method that computes local stress and strain fields using Hooke's law. In this FFT-based PDE solver, the microstructure (arrangement of grains in the composite material) is discretized onto a regular grid and a PDE with periodic boundary conditions is formulated using the stress-strain constitutive relation and equilibrium conditions. The MSC Basic Scheme is enumerated as follows. $\epsilon(\mathbf{x})$ and $\sigma(\mathbf{x})$ are strain and stress tensor fields at 3-D grid point $\mathbf{x}$ respectively. $C_{mnkl}(\mathbf{x})$ is the rank-4 stiffness tensor. $E$ is initial average strain. $\hat{\Gamma}_{mnk\ell}(\boldsymbol{\xi})$ is the Green's operator in Fourier space at frequency point $\boldsymbol{\xi}$. The convergence error is $e_s$ and tolerance error is $e_{\text{tol}}$. $\Delta\epsilon_{k\ell}$ is the computed perturbation in component $(k,\ell)$ of the strain tensor. Superscripts indicate iteration number. The iterative scheme continues till convergence is reached. For more details, refer to [4] and [5]. This is a single time step simulation. An outer loop also simulates viscoplastic deformation over multiple time steps, however that is not discussed here.

**Algorithm.** Initialize strain $\epsilon^0$ and stress $\sigma^0$.

$$\epsilon^0 \leftarrow E, \quad \sigma^0_{mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^0_{k\ell}(\mathbf{x})$$

While $e_s > e_{\text{tol}}$, proceed with following steps.

1) Compute FFT of stress field.
$$\hat{\sigma}^i_{mn}(\boldsymbol{\xi}) \leftarrow \text{FFT}(\sigma^i_{mn}(\mathbf{x}))$$
2) Check convergence
3) Fourier domain convolution with Green's function.
$$\Delta\hat{\epsilon}^{i+1}_{k\ell}(\boldsymbol{\xi}) \leftarrow \hat{\Gamma}_{k\ell mn}(\boldsymbol{\xi}) : \hat{\sigma}^i_{mn}(\boldsymbol{\xi})$$
4) Update strain in Fourier domain.
$$\hat{\epsilon}^{i+1}_{k\ell}(\boldsymbol{\xi}) \leftarrow \hat{\epsilon}^i_{k\ell}(\boldsymbol{\xi}) - \Delta\hat{\epsilon}^{i+1}_{k\ell}(\boldsymbol{\xi})$$
5) Inverse transform updated strain field.
$$\epsilon^{i+1}_{k\ell}(\mathbf{x}) \leftarrow \text{iFFT}(\hat{\epsilon}^{i+1}_{k\ell}(\boldsymbol{\xi}))$$
6) Update stress field.
$$\sigma^{i+1}_{mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^{i+1}_{k\ell}(\mathbf{x})$$

The convolution with Green's function requires computation of 3D FFTs of tensor components of the stress field, hence the need for extensive resources for large grids. Other quantities such as crystallographic angles and stiffness tensors are stored and updated at each grid point. Hence, a simulation size of $1,024 \times 1,024 \times 1,024$ is also a large-scale dataset. The MSC Basic Scheme is implemented in Fortran serial and MPI parallel version with FFTW [6].

## III. MSC Alternate Scheme

This section briefly describes the proposed algorithm, designed to be implemented on a CPU-GPU hardware setup, and discusses its highlights, namely irregular domain decomposition, lossy compression and domain-local FFTs.

**Overview.** Initialization of the problem is performed on the CPU side. In the 3-D composite volume, we observe that grain interiors have smooth stress and strain fields that are compressible. Hence, an irregular domain decomposition method is used to decompose the volume into smaller domains (grains) with smooth fields. The smooth fields are compressed to data models that are communicated to $J$ GPUs, thus reducing data movement and ensuring that GPU memory is sufficient to process the grain volume. Fixed-point iterations are performed on the GPU side. In each GPU-based iteration, domain-local FFTs are used for convolution of the stress field of a single grain $\sigma_{\text{grain}}(\mathbf{x})$ with the Green's function in the Fourier domain. We use the term *domain-local* to refer to FFT of the signal on each grain. The local convolution result $(\Delta\epsilon_{\text{local}}(\mathbf{x}))_j$ (where $j$ refers to the GPU ID), is compressed using adaptive subsampling. Then, GPUs communicate between each other to transfer the compressed $(\Delta\epsilon_{\text{local}}(\mathbf{x}))_j$. The effect of convolution is summarized by the $\Delta\epsilon_{\text{total}}(\mathbf{x})$ field, which is the sum over all local convolutions $(\Delta\epsilon_{\text{local}}(\mathbf{x}))_j$. Once the effect of convolution from other grains has been taken into account, corresponding strain $\Delta\epsilon_{\text{grain}}(\mathbf{x})$ is extracted from $\Delta\epsilon_{\text{total}}(\mathbf{x})$ by each GPU using windows. Now stress and strain fields can be updated locally for the grain. This makes the GPU part of the code parallel for grain volumes and stress update for the grain is a self-contained problem. The algorithm flow is summarized in Figure 2.

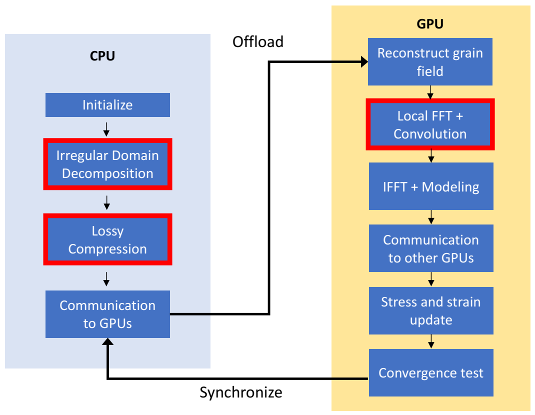

Fig. 2. CPU and GPU tasks for proposed MSC Alternate Scheme. Boxes highlighted in red are discussed in section III.

**CPU Algorithm.** Initialize strain $\epsilon^0$ and stress $\sigma^0$.

$$\epsilon^0 \leftarrow E, \quad \sigma^0_{mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^0_{k\ell}(\mathbf{x})$$

1) Decompose 3-D volume into grains (irregular domains) using windows.
2) Apply lossy compression to stress, strain fields in each grain.
3) Communicate data models to GPUs.

The GPU-side MSC Alternate Scheme for iteration $i$ is as follows.

**GPU Algorithm.** Reconstruct stress and strain fields for grain. While $e_s > e_{\text{tol}}$, proceed with following steps.

1) Compute domain-local FFT of stress field for grain.
$$\hat{\sigma}^i_{\text{grain},mn}(\boldsymbol{\xi}) \leftarrow \text{Domain-local FFT}(\sigma^i_{\text{grain},mn}(\mathbf{x}))$$
2) Compute Fourier domain convolution.

$$\Delta\hat{\epsilon}^{i+1}_{\text{local},k\ell}(\boldsymbol{\xi}) \leftarrow \hat{\Gamma}_{k\ell mn}(\boldsymbol{\xi}) : \hat{\sigma}^{i}_{\text{grain},mn}(\boldsymbol{\xi})$$

3) Inverse transform to get updated strain $\Delta\epsilon_{\text{local}}$.
$$\Delta\epsilon^{i+1}_{\text{local},k\ell}(\mathbf{x}) \leftarrow \text{iFFT}\big(\Delta\hat{\epsilon}^{i+1}_{\text{local},k\ell}(\boldsymbol{\xi})\big)$$
4) Compress sub-sampled field around grain. Communicate to other GPUs.
5) Obtain summarized convolution result $\Delta\epsilon_{\text{total}}$ by summing $\Delta\epsilon_{\text{local}}$ from other GPUs.
$$\Delta\epsilon^{i+1}_{\text{total}}(\mathbf{x}) = \sum_{j=1}^{J}\left(\Delta\epsilon^{i+1}_{\text{local}}(\mathbf{x})\right)_j$$
6) Obtain $\Delta\epsilon_{\text{grain}}$ in local grain volume from summarized convolution results using corresponding window $W(\mathbf{x})$.
$$\Delta\epsilon^{i+1}_{\text{grain}}(\mathbf{x}) = \Delta\epsilon^{i+1}_{\text{total}}(\mathbf{x}) \cdot W(\mathbf{x})$$
7) Update strain.
$$\epsilon^{i+1}_{\text{grain},k\ell}(\boldsymbol{\xi}) \leftarrow \epsilon^{i}_{\text{grain},k\ell}(\boldsymbol{\xi}) - \Delta\epsilon^{i+1}_{\text{grain},k\ell}(\boldsymbol{\xi})$$
8) Update stress.
$$\sigma^{i+1}_{\text{grain},mn}(\mathbf{x}) \leftarrow C_{mnk\ell}(\mathbf{x}) : \epsilon^{i+1}_{\text{grain},k\ell}(\mathbf{x})$$
9) Check convergence.

**Irregular Domain Decompositions.** The large 3-D volume is decomposed into smaller volumes (grain interiors) using windows. Number of voxels to be excluded at the grain boundary must be specified depending on the requirements of the simulation. For the test case discussed in this paper, we use a dataset with cubic grains. The cubical shape makes it easier to test the prototype with simple tapering windowing techniques such as trapezium or Tukey windows.

Datasets with irregular grains present a challenge in description of smooth regions because they do not have a compact representation. More complicated pre-processing is used in such cases. For e.g., a heuristic packing algorithm is used to pack the irregular shape with compressible regular shapes (such as rectangles).

**Lossy Compression.** Lossy compression methods can lead to higher compression as compared to lossless methods [7], [8], [9]. A number of lossy compression techniques are being increasingly used to deal with the problem of storing exponentially growing scientific data [10], [7]. One such scheme is ISABELA [11], which uses sorting and B-spline fitting [12] to compress random, noisy data. However, there is a tradeoff between compression ratio and error because indices of the sorted data have a storage requirement. In our case, we observe that grain interiors have smooth stress and strain fields. Hence, there is no need for sorting data. The stress field in each XY-plane of the windowed grain volume is modeled using cubic B-splines, which gives high compression ratios.

**Domain-local FFTs.** For large grids, GPU memory is a limiting factor in computing Fourier domain convolution of the $N_1 \times N_2 \times N_3$ stress signal with Green's function. However, by processing each grain separately on the GPU and using adaptive downsampling, we can compute the full convolution without storing entire large 3-D signals.

FFT of the sub-volume is computed pencil-wise, one dimension at a time. Pencils which are all zeros can be ignored. Transformation in the X-dimension gives a beam of $N_1 \times k_2 \times k_3$ non-zeros. By transforming in Y dimension, we get a slab of $N_1 \times N_2 \times k_3$ non-zeros. This is illustrated in Figure 4. In the Z-dimension, we transform one pencil at a time. Then, element-wise convolution is performed with Green's function, which is computed at required frequency points using the closed form in [4]. The pencil is immediately inverse transformed so that $N_1 \times N_2 \times N_3$ grid does not need to be stored. Since the Green's function is rapidly decaying in space domain, convolution of the two signals results in a low-magnitude and low-varying field in the volume surrounding the grain. This important observation allows us to develop an adaptive downsampling scheme to compress the convolution result and further reduce memory required. Low-resolution sampling is used in the low-varying part and sampling rate is higher around grain boundary. This is illustrated for a 1-D example in Figure 3. Similarly in 3-D, downsampling in the areas around the $k_1 \times k_2 \times k_3$ sub-volume yields a compressed model of the convolution result in that region.

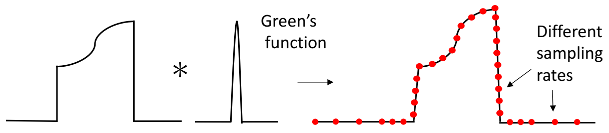

Fig. 3. Adaptive downsampling scheme to reduce memory for storage of convolution result.

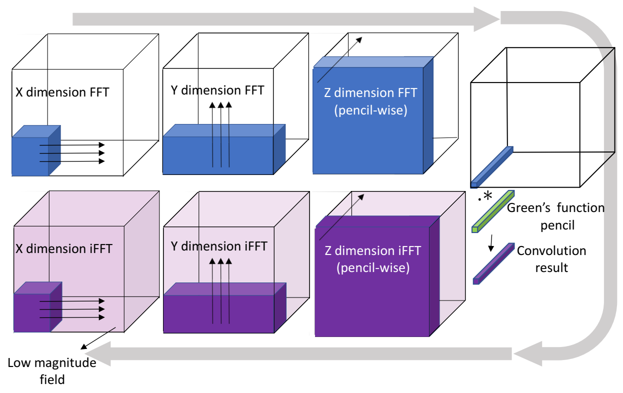

Fig. 4. Illustration of convolution of isolated grain using domain-local FFT. The $.*$ indicates element-wise multiplication.

## IV. Results and Discussion

We developed a MATLAB-Fortran prototype workflow of the proposed method prior to GPU implementation. This section describes results and analysis for the prototype. For proof-of-concept results, a simple microstructure test dataset with two types of grain orientations was created using MATLAB for various grid sizes. Grains in each grid of size $N^3$ are 8 cubes of $\left(\frac{N}{2}\right)^3$ points arranged in a periodic lattice. We also demonstrate results on grids of size $N^2 \times 8$ with 4 grains of size $\left(\frac{N}{2}\right)^2 \times 8$ points arranged in a periodic lattice. These microstructures simulate thin composite sheets.

**Lossy Compression.** Table IV shows results for two different values of grain boundary width (GBW), or the number of voxels excluded from lossy compression at the boundary. Reconstruction error (RE) is computed using Frobenius norm and compression ratio (CR) with respect to original memory requirement is as shown. For lower GBW (thinner boundaries), CR is higher. Reconstruction accuracy for grain interior is not affected adversely although low GBW might introduce more error at grain boundaries. Domain scientists can select appropriate boundary width required to study various physical phenomena while keeping this trade-off in mind. High compression and low error for very large grain sizes justifies using lossy compression since we have higher savings in memory and communication cost than smaller grains.

TABLE II
RECONSTRUCTION ERROR AND COMPRESSION RATIO

| **Grain Size** | $\mathbf{64^3}$ | $\mathbf{128^3}$ | $\mathbf{128^2 \times 8}$ | $\mathbf{256^2 \times 8}$ | $\mathbf{512^2 \times 8}$ |
|---|---|---|---|---|---|
| | Grain Boundary Width = 4 | | | | |
| **RE** | 1.49% | 1.11% | 0.52 % | 0.44% | 0.41% |
| **CR** | 2.48 | 8.01 | 5.47 | 13 | 28.69 |
| | Grain Boundary Width = 8 | | | | |
| **RE** | 1.99% | 1.25% | 0.51 % | 0.43% | 0.38% |
| **CR** | 1.46 | 4.19 | 3.37 | 7.32 | 15.29 |

TABLE III
ERROR IN CONVOLUTION BY LOCAL FFT METHOD

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

**Domain-local FFTs.** Using domain-local FFTs introduces some approximations due to the post-convolution downsampling of the smooth field around the grain. Table III shows error in convolution result for various grain sizes when domain-local FFT is used to compute convolution grain-by-grain. When field around the grain is downsampled by a factor of 8, memory required for grains of size $64^3$ and $128^3$ is reduced to only 12.7% of original value. This enables computation of the convolution on the limited resources of a GPU but grain edges show larger error. An adaptive method can be used to reduce this error by sampling more finely at the edges, with tradeoffs in compression. For example, downsampling by a factor of 2 reduces convolution error for these grains to 0.52% and 0.92% respectively and increases memory requirement by up to 1%.

**Sample convergence results.** Next, we confirm that our method converges in a comparable number of iterations from the original method. In the MSC Alternate Scheme, $n$ lower cost iterations of the fixed-point method are to be performed on GPUs. A few (3 to 4) high cost iterations are performed using the MSC Basic Scheme on the CPU side as a part of initialization to reduce approximation errors in the final answer. The plots in Figures 5 and 6 show the convergence of stress and strain fields in MSC Basic Scheme and MSC Alternate Scheme for different values of $n$ in a simulation of size $128 \times 128 \times 128$. We observe that for equal error thresholds for both methods, number of iterations for convergence changes depending on $n$, but not drastically. For this test example with $n = 7, 10, 15$, the simulations converge to a final answer with a deviation of about $2-4\%$ from the original ($n = 0$). This deviation can be due to convergence to a different local minimum. Additionally,

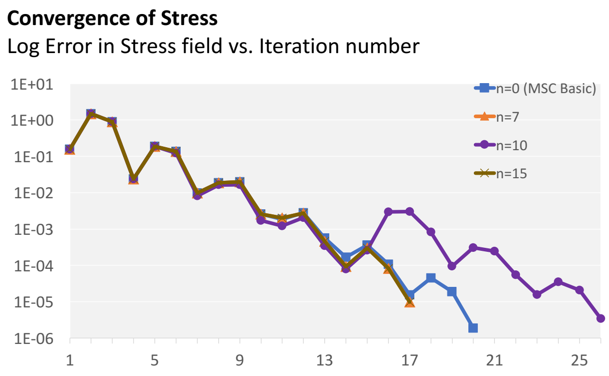

Fig. 5. Convergence in stress for problem size $128^3$. Number of iterations for convergence changes based on $n$, the number of iterations performed on GPUs using approximations.

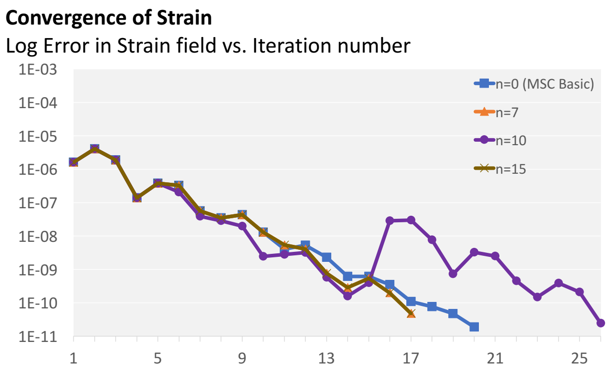

Fig. 6. Convergence in strain for problem size $128^3$. Strain converges more rapidly than stress in both MSC Basic and MSC Alternate Schemes.

B-spline compression on grain interiors smooths out artifacts which are otherwise carried forward in MSC Basic Scheme iterations.

## V. CONCLUSIONS

The proposed MSC Alternate Scheme is a co-design of algorithm and software for heterogenous platforms that enables scaling of stress-strain simulations to large grids by overcoming high memory requirements and communication bottlenecks. The algorithm uses domain-local FFTs and data modeling to perform iterations with a lowered cost, which converge to the same solution as the MSC-Basic Scheme with a small accuracy tradeoff, as is seen in proof-of-concept results presented here. The analysis presented here is an important step in the process of algorithm development at large scales. Ongoing work includes using a setup of CPUs and GPUs to implement the MSC-Alternate Scheme to observe performance metrics on varied datasets.

## VI. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Anthony Rollett, Dr. Vahid Tari, Dr. Anirban Jana and Dr. Roberto Gomez for their assistance and collaboration. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562.

## REFERENCES

[1] Y. Sabharwal, S. Garg, R. Garg, J. Gunnels, and R. K. Sahoo, "Optimization of fast fourier transforms on the blue gene/l supercomputer," *High Performance Computing - HiPC 2008: 15th International Conference, Bangalore, India, December 17-20, 2008. Proceedings*, pp. 309–322, 2008.

[2] J. Sun, C. Yang, and X.-C. Cai, "Algorithm development for extreme-scale computing," *National Science Review*, vol. 3, no. 1, pp. 26–27, 2016. [Online]. Available: http://dx.doi.org/10.1093/nsr/nwv088

[3] T. Mura, "Micromechanics of defects in solids," *The Journal of the Acoustical Society of America*, vol. 73, no. 6, pp. 2237–2237, 1983. [Online]. Available: http://dx.doi.org/10.1121/1.389536

[4] H. Moulinec and P. Suquet, "A numerical method for computing the overall response of nonlinear composites with complex microstructure," *Computer methods in applied mechanics and engineering*, vol. 157, no. 1-2, pp. 69–94, 1998.

[5] R. A. Lebensohn, "N-site modeling of a 3d viscoplastic polycrystal using fast fourier transform," *Acta Materialia*, vol. 49, no. 14, pp. 2723–2737, 2001.

[6] M. Frigo and S. G. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, Feb 2005.

[7] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec 2014.

[8] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 18–31, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1109/TC.2008.131

[9] J.-l. Gailly, "gzip: The data compression program," 2016. [Online]. Available: https://www.gnu.org/software/gzip/manual/gzip.pdf

[10] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *IPDPS 2016*, IEEE. Chicago, IL: IEEE, 06/2016 2015.

[11] L. Sriram, S. Neil, E. Stephane, K. S. Hoe, C. C. S., K. Scott, L. Rob, R. Rob, and S. N. F., "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 524–540. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.2887

[12] M. Unser, "Splines: a perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, Nov 1999.