EVPFFTX: A First Look at FFTX Applications in Material Science

Het Mankad*, Andrea Rovinelli†, Miroslav Zecevic†, Peter McCorquodale‡, Franz Franchetti*, Naifeng Zhang*, Sanil Rao*, R. A. Lebensohn†, Laurent Capolungo†

*Electrical and Computer Engineering Department Carnegie Mellon University, Pittsburgh, PA, USA

†Los Alamos National Laboratory, Los Alamos, NM, USA

‡Lawrence Berekely National Laboratory, Berkeley, CA, USA

*{hmankad, franzf, naifengz, sanilr}@andrew.cmu.edu, †{arovinelli, miroslav, lebenso,laurent}@lanl.gov,

‡{PWMcCorquodale}@lbl.gov

***Abstract*—This work presents a first look of EVPFFTX. It is an FFTX based library for the elasto-viscoplastic FFT (EVPFFT) algorithm used in material science. FFTX is a high performance library with a SPIRAL backend, build for running the fast Fourier transform (FFT) based applications on the latest exascale machines. SPIRAL is a code generation system that was developed for generating highly optimized code for different types of linear transforms.The use of EVPFFT is to study polycrystalline material which forms an integral part of the molten salt reactors. The aim of this work is to provide a brief overview of the procedure required to translate the EVPFFT algorithm into the SPIRAL framework.**

***Index Terms*—FFTX, molten salt reactor, code generation, viscoplastic polycrystals, SPIRAL**

I. Introduction

Molten salt reactors are a type of nuclear reactors that use molten fluoride salts as the fuel and/or primary coolant. The main advantage is that they are cheaper, safer and can generate a huge amount of energy while producing less waste. However, one of the challenges that is faced is that of the corrosivity of the molten salt. This occurs due to the long time effect of salt on the metals used to build the reactor. Hence, understanding the properties of different materials at is an important factor in producing safe energy but it is very difficult to do so in a laboratory environment. This provides the opportunity to use advanced scientific algorithms to perform simulations that help in the understanding of these microstructures. One of the algorithms used for this purpose is the elasto-viscoplastic FFT (EVPFFT) algorithm [1]. It is FORTRAN based code that studies polycrystals using FFTs. Some past work in this area can be seen in [2]. In order to boost the performance of this code on various computing architectures, we propose EVPFFTX, a FFTX based [3] EVPFFT algorithm which also has optimized code generated using SPIRAL [4]. [5] is another example where efforts were made to improve the performance of the existing stress-strain code on GPUs.

II. Background

SPIRAL and FFTX. SPIRAL is a code generation system that was initially developed to produce high performing optimized code for various linear transforms [4], [6]. It initially used the Signal Processing Language (SPL) to express the computation described in the given problem specification. It is now generalized into the Operator Language (OL). It is a mathematical declarative language which helps translates the given semantics of the problem specification into something that helps SPIRAL generate the optimized code for. This is then broken down into $\Sigma-$OL which translates this into a loop based rule system that final provides the user with the optimized code. FFTX [3] is a high performance library for FFTs which has SPIRAL and Just -In Time compilation as its backend and was mainly developed to run FFT based applications on exascale machines.

EVPFFT. Elasto-viscoplatic FFT algorithm [1] was developed to predict the micromechanical fields in materials that contain polycrystals. It is a FORTRAN based code which consists of many complex computations including FFTs. A code snippet of the EVPFFT FORTRAN code is shown in Fig. 1. We can observe that there are multiple calls to perform either the DFT or inverse DFT of the given quantities inside the `OuterLoopStep` subroutine.

III. EVPFFTX

In this section we will introduce EVPFFTX which is the SPIRAL generated and FFTX based EVPFFT code. The idea is to recognize the current EVPFFT algorithm as a problem specification in SPIRAL and then understand the data flow and use OL to translate this data flow into a language that SPIRAL understands.

In this work we will mainly focus on the OL representation of the code snipped shown in Fig. 1. OL representation is based on concepts from linear algebra and some other basic mathematical concepts. Some of the notation used to formulate an OL expression are shown in table I. For example using the notations shown in the table I, the multidimensional DFT and inverse DFT are defined in SPIRAL with the help of Kronecker product and 1D DFTs.

$$\mathrm{DFT}_{k\times m\times n} = \mathrm{DFT}_k \otimes \mathrm{DFT}_m \otimes \mathrm{DFT}_n, \quad (1)$$

$$\mathrm{iDFT}_{k\times m\times n} = \mathrm{iDFT}_k \otimes \mathrm{iDFT}_m \otimes \mathrm{iDFT}_n\,. \quad (2)$$

Now using a combination of these notations helps us to formulate an OL expression for the EVPFFT algorithm shown in Fig. 1. Here we focus on the `OuterLoopStep` subroutine

only. The OL formulation for that is provided in (3) - (7). PRDFT and iPRDFT are part of the FFTX function calls to compute the forward and the inverse DFT of the given input data.

Operation	matrix
$A \circ B : \mathbb{R}^q \to \mathbb{R}^m$	matrix product, AB
$A \oplus B : \mathbb{R}^{n+q} \to \mathbb{R}^{m+p}$	$\begin{bmatrix} A & \\ & B \end{bmatrix}$
$A \otimes B : \mathbb{R}^{nq} \to \mathbb{R}^{mp}$	$\begin{bmatrix} A_{00}B & \cdots & A_{0,n-1}B \\ \vdots & \ddots & \vdots \\ A_{m-1,0}B & \cdots & A_{m-1,n-1}B \end{bmatrix}$
$A + B : \mathbb{R}^n \to \mathbb{R}^m$	matrix sum, $A + B$
$A \cdot B : \mathbb{R}^n \to \mathbb{R}^m$	pointwise matrix product, $(A \cdot B)_{i,j} = A_{i,j}B_{i,j}$

TABLE I: If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, then A and B can be interpreted as operators, respectively $A : \mathbb{R}^n \to \mathbb{R}^m$ and $B : \mathbb{R}^q \to \mathbb{R}^p$.

```
module evpfft_algorithm_mod
    ...
  subroutine outerLoopStep(dt, L0, L0_T4,L0_NEW,
             vel_grad_correction)
    ...
    ! the fft related Variables
    real(r64) :: K33(3,3), & ! frequency tensor
                 G33(3,3), G33_inv(3,3), &
                 Gamma3333(3,3,3,3), &
                 M0(6,6), M0_T4(3,3,3,3)
    ...
    complex(C_DOUBLE_COMPLEX) ::
        lambda_fourier_space(3,3), &
        vel_grad_fourier_space(3,3)
    ...
    ! perform forward DFT fo the stress field
    call my_fft_data_container%executeForwardDFT()
    ...
             ! and finally the velocity gradient
            ...
          ! copy the velcoity gradient fluctuation
          ! tensor into the FFT container
        call my_fft_data_container%setPointFromComplexTensor
           (ix,iy,iz, vel_grad_fourier_space)
     ! perform backward DFT
     call my_fft_data_container%executeBackwardDFT()
    ! comnpute the updated velocity gradient
    ...
        vel_grad(:,:,ix,iy,iz) =
        vel_grad(:,:,ix,iy,iz) + vel_grad_correction + &
        my_fft_data_container%getPointDataReal(ix,iy,iz)
        point_weight
    ...
    ! compute material constituive response
    call elastic_response_augmented_lagrangian
    (dims_rank, dt, L0, L0_NEW)
   end subroutine
end module evpfft_algorithm_mod
```

Fig. 1: Sample code from EVPFFT. One can observe that at various stages FFT or inverse FFT is being computed.

$$\begin{aligned}\hat{\Gamma}^{i,j,k,\ell}_{p,q,r} \circledast \hat{[.]}^{k,\ell}_{p,q,r} \to &\big(\,\mathrm{iPRDFT}_{N_z\times N_y\times N_x} \otimes \mathrm{I}_{3\times 3}\,\big) \\ &\circ \big(\underline{\Gamma^{i,j,k,\ell}_{p,q,r} : [.]^{k,\ell}_{p,q,r}}\big) \\ &\circ \big(\,\mathrm{PRDFT}_{N_z\times N_y\times N_x} \otimes \mathrm{I}_{3\times 3}\,\big)\end{aligned} \tag{3}$$

$$\begin{aligned}\underline{\Gamma^{i,j,k,\ell}_{p,q,r} : [.]^{k,\ell}_{p,q,r}} \to &\,\mathrm{I}_{N_z\times N_y\times N_x/2+1} \\ &\otimes_{r,q,p} \big(\overline{\Gamma^{i,j,k,\ell}_{p,q,r}|_{p,q,r} : [.]^{k,\ell}_{p,q,r}|_{p,q,r}}\big)\end{aligned} \tag{4}$$

$$\Gamma^{i,j,k,\ell}_{p,q,r}|_{p,q,r} \to \begin{cases} 0 \in \mathbb{R}^{3\times3\times3\times3}, & \text{if } (p,q,r) = 0; \\ -M_0^{i,j,k,\ell}, & \text{if } p = \frac{N_x}{2} \text{ or } q = \frac{N_y}{2} \\ & \text{or } r = \frac{N_z}{2}; \\ \Gamma^{i,j,k,\ell}(\nu^u_{p,q,r}), & \text{else} \end{cases} \tag{5}$$

$$\begin{aligned}\Gamma^{i,j,k,\ell}(.) \to &\big(([.]^\top[.]) \otimes \mathrm{I}_{3\times3}\big) \\ &\circ \big([.]^{-1} \circ (-L_0^{i,j,k,\ell} : [.])[-](\mathrm{I}_{3\times3})\big) \circ \big([.]^\top[.]\big)\end{aligned} \tag{6}$$

$$\nu^u_{p,q,r} = 2\pi \begin{bmatrix} \frac{p - N_x\chi_{p\geq N_x/2}}{\Delta_x N_x} \\ \frac{q - N_y\chi_{q\geq N_y/2}}{\Delta_y N_y} \\ \frac{r - N_z\chi_{r\geq N_z/2}}{\Delta_z N_z} \end{bmatrix} \tag{7}$$

IV. Conclusion

In this work we present a glimpse of our initial efforts to generate an optimized code for the existing EVPFFT algorithm. This is done by using the FFTX library and by translating the existing code into an OL representation that is used by SPIRAL to produce a high performing optimized code for the study of polycrystals. It is an ongoing work and the future work is to make an entire end-to-end representation in SPIRAL for this problem along with testing on different target platforms.

V. Acknowledgment

This project is supported by DOE ECP project 2.2.6.04 and DOE SciDAC project DE-SC0023523.

References

[1] R. A. Lebensohn, A. K. Kanjarla, and P. Eisenlohr, "An elasto-viscoplastic formulation based on fast fourier transforms for the prediction of micromechanical fields in polycrystalline materials," *International Journal of Plasticity*, vol. 32-33, pp. 59–69, 2012.

[2] 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.

[3] F. Franchetti, D. G. Spampinato, A. Kulkarni, D. Thom Popovici, T. M. Low, M. Franusich, A. Canning, P. McCorquodale, B. V. Straalen, and P. Colella, "Fftx and spectralpack: A first look," in *2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)*, 2018, pp. 18–27.

[4] M. Puschel, J. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. Johnson, and N. Rizzolo, "Spiral: Code generation for dsp transforms," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 232–275, 2005.

[5] A. Kulkarni, J. Kovačević, and F. Franchetti, "Massive scaling of massif: Algorithm development and analysis for simulation on gpus," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, ser. PASC '20. New York, NY, USA: Association for Computing Machinery, 2020.

[6] F. Franchetti, Y. Voronenko, and M. Püschel, "Formal loop merging for signal transforms," *SIGPLAN Not.*, vol. 40, no. 6, p. 315–326, jun 2005.