Scalable Large Scale Process Modeling and Simulations in Liquid Composite Molding

Ram Mohan*, Dale Shires, and Andrew Mark

U. S. Army Research Laboratory [†]
High Performance Computing Division
Aberdeen Proving Ground, MD 21005

Abstract. Various composite structural configurations are increasingly manufactured using liquid composite molding processes. These processes provide unitized structures with repeatability and excellent dimensional tolerances. The new generation of materials and processes, however, pose significant challenges in affordable development and process optimization. Process modeling and simulations, based on physical models, play a significant role in understanding the process behavior and allow for optimizing the process variants in a virtual environment. Large scale composite structures require scalable process modeling and simulation software. An overview of various key issues for such scalable software developments are presented. Several items, including our experiences with the numerical techniques and algorithms for the solution of the representative physical models, are discussed. We also investigate and briefly discuss the various parallel programming models and software development complexities to achieve good performance. In this context, iterative solution techniques for large scale systems, based on a domain decomposition of the problem domain and non-invasive techniques to boost performance of scalable software, are discussed. Preliminary results of the performance of simulations in liquid composite molding are presented. The discussions presented, though in direct reference to scalable process modeling and simulations in liquid composite molding, are directly applicable to any unstructured finite element computations.

1 Introduction

Polymer composite materials have become the material of choice in various new aerospace, marine, and dual-use military and commercial applications. This is due to the high strength-to-weight ratio of these materials. However, manufacture of structural composite materials poses significant problems. These composite materials are made of a fiber matrix and are consolidated together by a polymeric resin. The strength of these composite materials is due to the directional

[†] This research was made possible by a grant of computer time and resources by the Department of Defense High Performance Computing Modernization Program.

* Guest Researcher

orientation of the individual fabric layers. Manufacturing techniques should ideally preserve this fiber orientation during processing for each of the repeated composite structural parts. Processes like prepreg and hand layup maintain the high quality of the layer orientations and good bonding between the individual layers. However, these tend to be highly time consuming and labor intensive. New processing methodologies based on liquid molding of composites have evolved over the past decade to provide a production oriented methodology for structural composite configurations. The process allows for repeated production of net-shape composite structures that maintain and preserve the desired fiber orientation in a production environment. One particular type of liquid composite molding is resin transfer molding (RTM) and its related variants. The process involves only a few steps. The first step involves construction of a tool or mold cavity that could be either one- or two-sided. This is followed by setting up a dry fiber preform with multiple individual layers of woven fabrics (normally stitched together) and preforming the dry fabric into a net-shape of the composite structural part configuration. The net-shaped composite structural part configuration is placed inside the mold cavity conforming to the tool surface. The polymeric resin is then injected into the mold cavity where it infiltrates and impregnates the dry fiber preform, thus consolidating into the net-shaped structural composite part configuration after curing and consolidation.

The challenges facing the new generation of material processes for producing these composite structures are how these processes and materials can be cost effectively developed and optimized for various processing conditions. The mold tooling costs in liquid molding applications are high and it is essential to develop optimal processing conditions. Physical process modeling and simulations enable understanding the physical behavior and optimizing the process variables in a virtual environment. The strength of the liquid molding processes such as RTM and its variants is in their ability to manufacture unitized, net-shape, large scale composite configurations. The physical and geometric complexity necessitate a need for scalable simulation software for the process modeling and simulations.

Physical process modeling and simulations begin with the understanding and description of the underlying physical phenomena by means of mathematical model equations representing the various conservation laws, and, appropriate material constitutive relations. The model equations are then discretized by numerical methods such as the finite element method. Computational algorithms play a critical role in obtaining physically and numerically accurate solutions for large scale problems. In particular, in scalable computational analysis, it is critical and important to employ optimal computational methodologies. This is in addition to having optimal data structures, data and processor layout, and communication strategies for scalable high performance computing architectures. A purely finite element based methodology [1,2,3] for modeling and analyzing the pressure driven transient flow through porous fiber media is briefly described.

The development of large scale, parallel scalable software involves a clear trade-off between the amount of work and the information the developer has to perform and provide, and the amount of effort the compiler has to expend to generate the optimal scalable parallel software code. This depends mainly on parallel programming paradigms. Programming paradigms based on high level parallel languages (High Performance Fortran (HPF), CM-Fortran[4] in CM-5 systems) and parallelizing compilers permit a data parallel mode of computing with no user defined explicit communication across multiple processors. Another parallel programming paradigm involves explicit message passing across the multiple processors. With the development and maturation of the message passing standards from the early 1990s, this approach has gained wider usage for scalable parallel software development. Various issues of the message passing interface (MPI) parallel software developments are discussed and presented. Illustrative examples demonstrating the computational time speed up for multiprocessor runs are presented. Preliminary results comparing the execution times between the HPF and MPI parallel paradigms are provided in the context of these simulations.

2 Process Modeling of Resin Impregnation

The resin impregnation process in liquid molding processes such as RTM involves the flow of a polymeric resin through a fiber network until the network is completely filled. The macroscopic flow of the resin through the complex fiber medium is usually treated as a pressure driven flow through a porous media and is characterized by Darcy's law. Darcy's law relates the macroscopic flow velocity field to the pressure gradient through fluid viscosity and fiber preform permeability as

$$\boldsymbol{u} = \frac{K}{\mu} \bigtriangledown P, \tag{1}$$

where $\boldsymbol{u}$ is the velocity field, μ is the fluid viscosity, and P is the pressure. Permeability K is a measure of the resistance a fluid experiences when flowing through a porous medium and is an important material characteristic in the process flow impregnation simulations. Physically, resin impregnation and flow permeating through a porous media is a free surface moving boundary value problem whereby the field equations and the free surface have to be solved and tracked. In the liquid composite molding processes such as RTM, the primary interest is in the temporal progression of the resin inside a complex mold cavity representing the net-shape composite structural part. Eulerian fixed-mesh approaches are employed for numerical finite element computations of the complex diverging/merging flow front progressions.

2.1 Numerical Methodologies

Computational methodologies employed in process flow modeling simulations for solving the pressure field and the free surface include (1) an explicit finite element-control volume method and (2) the pure finite element method originally developed by Mohan et al. [1,2,3].

In the finite element-control volume technique [5,6,7], the transient resin impregnation problem is treated as a quasi-steady-state problem solving for the quasi-steady incompressible continuity equation. This leads to stringent numerical restrictions based upon the Courant stability conditions. The time step increments are highly restricted across each of the quasi-steady steps ensuring the numerical stability of the quasi-steady approximations based on the Courant stability conditions. Such restrictions increase the number of quasi-steady state steps needed for the analysis. The power of high performance computing lies in its ability to enable practical large scale simulations in a reasonable time. For large scale problem sizes and computational domains, the quasi-steady time increment sizes are extremely small compared to the resolution needed in the application. This makes it impossible to complete large scale simulations, even on scalable high performance computing platforms in any reasonable time. These difficulties are overcome by the pure finite element methodology [1,2,3] briefly described next.

The pure finite element methodology is based on the transient mass conservation law modeling the resin mass balance inside a mold cavity involving the state variable $\Psi(0 \leq \Psi \leq 1)$, where $\Psi = 0$ represents the unimpregnated regions of the dry fiber preform, and $\Psi = 1$ represents the completely impregnated regions of the dry fiber preform. The velocity field is modeled with Darcy's law. The governing transient equation is represented by

$$\frac{\partial \Psi}{\partial t} = \bigtriangledown \cdot \left(\frac{K}{\mu} \bigtriangledown P \right), \tag{2}$$

where μ is the resin viscosity and K is the permeability tensor of the fiber preform. For thin preforms (2.5 D physics) with variations in the in-plane velocity fields, the permeability tensor K is a matrix of order two; for thick preforms, the permeability tensor is a matrix of order three. Based on a Galerkin-weighted residual formulation, and, introducing finite element approximations for both the state variable $\Psi(N_i \Psi_i)$ and the pressure field $P(N_i P_i)$, where N_i is the finite element shape function that depends on the element type employed; P_i, Ψ_i are the nodal values, leads to a discretized system of equations given by

$$C \left[\Psi^{n+1} - \Psi^{n} \right] + \Delta t \left[K \right] P = \Delta t q. \tag{3}$$

In equation 3, C is the mass matrix representing the pore volume, $[K]$ is the stiffness matrix associated with the pressure field, Δt is the time step size for the transient problem, and q is the force vector representing the injection conditions. The pure finite element methodology solves for the fill factor and the pressure field associated with the finite element nodes in an iterative manner until complete mass conservation is achieved at each time step.

The pure finite element methodology briefly described here does not involve the restrictions of the time step increments seen in the explicit finite element-control volume method for this free surface flow problem and is second-order accurate in time. Rather, it computes the position of the flow front at each of the discrete time steps that are selected by the analyst. This leads to significant reductions in the computing time required to complete the execution of process modeling and simulations. These reductions are quite dramatic in the case of large-scale computational models for composite structural configurations. This methodology is proven to provide a more physically accurate, computationally faster, and algorithmically better solution strategies for the finite element modeling of the resin impregnation in liquid composite molding processes [1,2,3]. The computational advantage demonstrated by the method in comparison to the finite element-control volume technique for liquid resin impregnation composite molding simulations is solely due to the computational methodology and the algorithmic solution strategy. The computational effectiveness of the strategy is independent of the computational platform. Large-scale process modeling and simulations that were impossible earlier are now possible.

When the thermal and curing effects are considered during the resin impregnation process, the governing model equations based on a volume-averaged energy balance is given by

$$\rho C_p \frac{\partial \hat{T}}{\partial t} + \rho_f C_{pf} \hat{\boldsymbol{u}} \cdot \bigtriangledown \hat{T} = \bigtriangledown \cdot K_T \bigtriangledown \hat{T} + \phi \hat{\dot{G}}. \tag{4}$$

The subscript f denotes the liquid phase, ϕ is the porosity of the fiber medium, and ρ and c_p are the average density and specific heat, respectively. The curing reaction based on species balance can be written as

$$\phi \frac{\partial \hat{\alpha}}{\partial t} + \hat{u} \cdot \bigtriangledown \hat{\alpha} = \phi \hat{R_\alpha}. \tag{5}$$

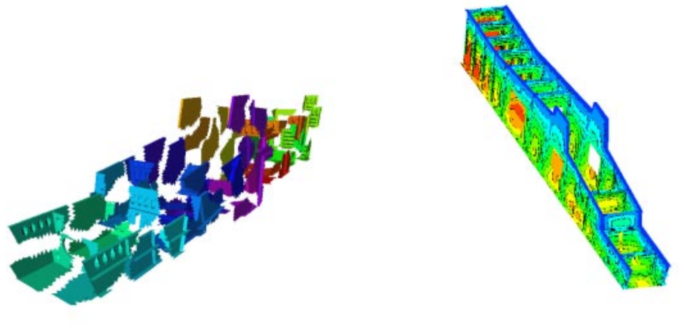
In most cases, the impregnation phase is completed before the initiation of thermal and cure reactions. The equations are presented here for illustration and the present scalable developments focus only on the resin impregnation phase.

3 Scalable Parallel Software Paradigms

The development of parallel software involves various levels of effort based on the programming paradigms employed. There is a clear correlation between the work that the parallel software developer has to perform and the performance of a parallel code. The paradigms based on parallelizing compilers leave the full responsibility of extracting the parallelism to the compilers and tend to perform poorly, though a minimal level of effort is needed from the developer. At the intermediate level are the parallel languages in the form of HPF, where the developer and the compiler/runtime system share the responsibility of extracting the parallelism. This is based on Fortran directives that allow the developer to express the parallelism (normally data parallel mode in finite element computations) and control the data locality at a very high level. The developer-defined data layout is then utilized in a compiler which generates the low-level details, including the required communications. At the highest level of developer responsibility are explicitly-parallel formulations, such as MPI. Here, the developer has to explicitly code all the parallelism, synchronization, and masking based on the analysis schemes and interprocedural data requirements. In the MPI based paradigms, parallelism is obtained through a single-program, multiple-data (SPMD) programming model. This is achieved using appropriate decomposition of the computational domain into multiple domains assigned to various processors. Specific interprocessor and required processor masking is specified by the developer with explicit MPI communication and synchronization calls. Both HPF- and MPI- based scalable parallel software have been developed. An illustrative preliminary comparison of total execution times, excluding input and output operations, is presented next.

3.1 HPF and MPI Comparisons

Illustrative analysis resulting from the scalable software developments is shown in figure 1. A domain decomposed partition for MPI-based scalable simulations



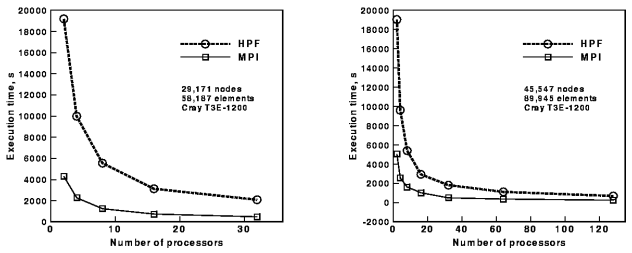
(a) Partitioned Domain for MPI

(b) Temporal resin progression contours

Fig. 1. Scalable process modeling and simulations of a complex structure

is shown in figure 1(a). The nonoverlapping finite element mesh decompositions have been obtained using the graph partitioning software Metis and/or ParMetis [8]. The temporal resin progression contours, based on representative injection conditions for a large, complex aerospace composite structural configuration, are shown in Figure 1(b).

The total execution times for complete analysis based on two finite element mesh configurations are shown in figure 2. The timing comparisons are based on a Cray T3E-1200 system. We chose this system since the HPF compiler employed (Portland Group PGHPF 3.0) was more heavily optimized for this architecture [9]. A preconditioned conjugate gradient iterative solver is employed to solve the linear system of equations. The HPF implementation of the iterative solver is based on a element-by-element formulation where the residuals are determined at the element level before they are assembled to the global level. The MPI implementation is based on a sub-domain assembled (no communication involved within a processor) sparse implementation and is discussed further in the next section. It is clearly seen from figure 2, that the execution time for the HPF



(a) Configuration 1 (b) Configuration 2

Fig. 2. Total execution time comparison of HPF and MPI scalable software

version of scalable software is higher compared to the MPI solver. This illustrates that significantly better performance can be obtained with MPI-based paradigms that require a full developer involvement. However, new approaches to boost performance of data parallel approaches for unstructured mesh problems are coming to fruition. For example, the new PGHPF 3.2 compiler supports asymmetric block data distribution, and the Japan Association for HPF has proposed techniques to reuse communication schedules inside of gather-type operations. Further studies on the computation, communication costs, memory usage, and linear system solver performance are currently in progress.

4 Linear System Solver with MPI

The process modeling and simulations in liquid composite molding involve the solution of a linear system of equations of the form $Ax = b$, based on a sparse, symmetric positive definite matrix. An iterative conjugate gradient algorithm, based on the developments [11] for multiple instruction, multiple data (MIMD) computers, is employed. A similar procedure has been followed and extended to include diagonal preconditioners [10]. The assembled and distributed vector forms are employed. Assembled forms are denoted by $(\hat{\cdot})$ [12]. The iterative procedure is thus

```
– $\{x_0^{SD}\} = 0$, $\{r_0^{SD}\} = \{b^{SD}\}$
– Do $i = 0, 1, \ldots\ldots$ until convergence

1. If $i = 0$ Go To Step 5
2. $\{u^{SD}\} = \left[A^{SD}\right]\{p^{SD}\}$
   $\beta^{SD} = \{p^{SD}\}^T\{u^{SD}\}$
   $\sigma_{SD} = \beta^{SD}/\gamma$
3. MPI_ALLREDUCE $\frac{1}{\alpha} = \Sigma\sigma^{SD}$
4. $\left\{x^{SD}\right\} = \left\{x^{SD}\right\} + \alpha\left\{p^{SD}\right\}$
   $\left\{r^{SD}\right\} = \left\{r^{SD}\right\} - \alpha\left\{u^{SD}\right\}$
5. SEND $\{r^{SB}\}$; RECEIVE $\{r^{NB}\}$
   $\left\{\hat{r}^{SD}\right\} = \left\{r^{SD}\right\} + \Sigma\{r^{NB}\}$
   $\rho^{SD} = \left\{\hat{r}^{SD}\right\}^T\left\{r^{SD}\right\}$
6. MPI_ALLREDUCE $\gamma_{new} = \Sigma\rho^{SD}$
7. If $(i = 0)$ Go to Step 9
8. If $\left(\frac{\gamma_{new}}{\gamma} < \epsilon\right)$ STOP
9. $\left\{p^{SD}\right\} = \left\{\hat{r}^{SD}\right\} + \frac{\gamma_{new}}{\gamma}\left\{p^{SD}\right\}$
10. $\gamma = \gamma_{new}$
```

The method is demanding in terms of both communication and computations and is called repeatedly during the analysis runs.

One of the major steps in the previous iterative procedure is matrix-vector multiplication. The sparse matrix is repeatedly multiplied by a vector. The cache performance of SAXPY with the matrix A in sparse format depends on the finite element connectivity. Performance gains are possible by noninvasive techniques that actually modify the data sets without affecting the physics behind the simulation. The techniques presented next are apropos for reduced instruction set (RISC) based computers currently in use.

Most RISC-based processors utilize various levels (at least two) of cache to overcome the CPU/memory system performance gap [13]. A poorly structured connectivity of an unstructured finite element mesh can lead to poor cache affinity. The sparse matrix A is in a compressed row format, and the node-based computations are based on element connectivity. For instance, matrix-vector multiplication with an element composed of nodes numbered 3, 100, and 200 will require a load of data from the node-based vector *vec* at addresses &*vec*[3], &*vec*[100], and &*vec*[200]. A load of data for *vec*[3] will likely load data into cache for entries *vec*[4] and *vec*[5], but hardly data for nodes 100 and 200. A node renumbering that will improve the proximity of node numbers will also improve the cache efficiency. Various techniques are available to renumber the meshes to promote cache efficiency [14]. Reordering finite element nodes based on the Reverse Cuthill McKee (RCM) scheme [15] significantly improved the closeness of the data for the matrix-vector multiplications. Figure 3(a) shows the distribution of non-zero entries of a matrix A before reordering, and figure 3(b) shows the distribution of entries after reordering. The closeness of the non-

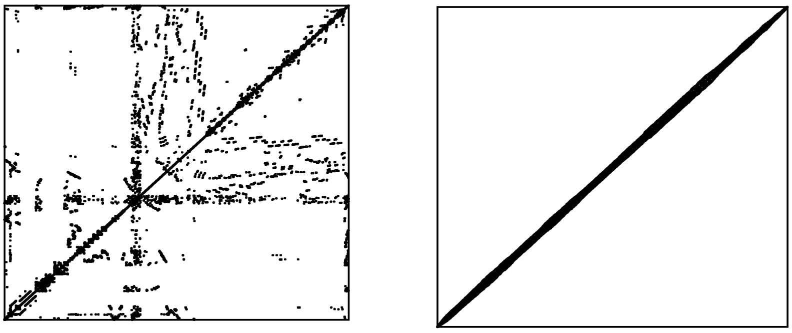
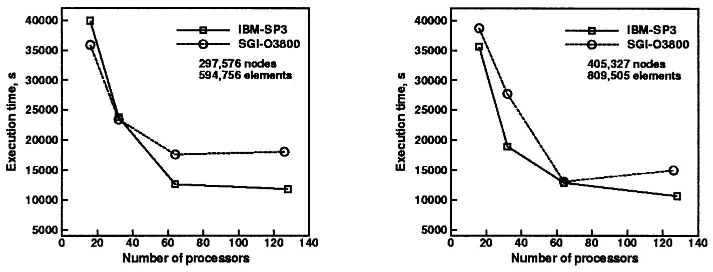

(a) Before reordering (b) After reordering

Fig. 3. Distribution of non-zero entries in a finite element sparse matrix A

zero entries after reordering permits loading of all required nodal vectors into the cache. This is important in the case of high performance computing systems with small cache sizes for combined data and instruction. For example, the Cray T3E processors only contain 8KB of primary cache and 96KB of secondary cache. An RCM processed finite element mesh executed roughly 10% faster on the Cray T3E than an unprocessed mesh. However, on machines with a large cache, such as the SGI Origin 3800 with an 8MB secondary cache, the wall clock time is unaffected. A detailed analysis of the underlying memory system, however, did show dramatic improvements in cache performance with reduced cache misses and improved memory performance. It is surmised that the large cache, coupled with optimizations such as software and hardware prefetching, mitigated wall clock improvements. Further detailed analysis is currently being conducted. Similar approaches improve the data locality in HPF implementations [16].

5 Scalable Simulations

The total execution times based on scalable multiprocessor MPI simulation runs for two different mesh configurations are shown in figure 4. These runs represent different processing conditions for the composite structural configuration shown in figure 1(a). The results imply a good scalability and portability of the scalable



(a) Configuration 1 (b) Configuration 2

Fig. 4. Execution time and speedup from scalable MPI simulations

software, and demonstrate the scalable process modeling and simulation capability to perform realistic simulations in a reasonable time. Large-scale process modeling simulations are now routinely possible for complex composite structural configurations manufactured by liquid molding processes such as RTM and its variants. Further performance studies are currently in progress.

6 Concluding Remarks

Modeling and simulation, coupled with scalable high performance computing, can play a significant role in various new composite manufacturing applications. These techniques, available because of new numerical approaches such as the pure finite element method and large scale parallel computing assets, reduce the risks associated with manufacturing large, geometrically complex composite structural components. The impacts are wide ranging, and applicable to numerous military and commercial uses.

This paper has demonstrated how various parallel methodologies can be applied to complex engineering problems. High level approaches to parallelism, such as data parallelism through HPF, are applicable to this class of problem. These approaches continue to mature through more robust compilers and an enhanced standard. Large scale industrial applications may benefit more from an explicitly parallel approach found in message passing, where the MPI libraries are providing fast solutions to grand-scale problems on numerous parallel architectures.

References

[1] R. V. Mohan, N. D. Ngo, and K. K. Tamma. On a Pure Finite-Element-Based Methodology for Resin Transfer Mold Filling Simulations. *Polymer Engineering and Science*, Vol. 39, no.1, January 1999.

[2] R. V. Mohan, N. D. Ngo, K. K. Tamma, and K. D. Fickie. A Finite Element Based Methodology for Resin Transfer Mold Filling Simulations. In R. W. Lewis and P. Durbetaki, editors, *Numerical Methods for Thermal Problems*, Vol. IX, pp. 1287–1310, Atlanta, GA, July 1995. Pineridge Press.

[3] R. V. Mohan, N. D. Ngo, K. K. Tamma, and D. R. Shires. Three-Dimensional Resin Transfer Molding: Isothermal Process Modeling and Implicit Tracking of Moving Fronts for Thick, Geometrically Complex Composite Manufacturing Applications - Part 2. *Numerical Heat Transfer-Part A, Applications*, Vol. 35, no. 8, 1999.

[4] R. V. Mohan, D. R. Shires, A. Mark, and K. K. Tamma. Advanced Manufacturing of Large Scale Composite Structures : Process Modeling, Manufacturing Simulations and Massively Parallel Computing Platforms. *Journal of Advances in Engineering Software*, Vol. 29, no. (3-6), pp. 249–264, 1998.

[5] C. A. Fracchia, J. Castro, and C. L. Tucker. A Finite Element/Control Volume simulation of Resin Transfer Mold Filling. In *Proc. of the American Society For Composites, 4th technical conference*, pages 157–166, Lancaster, PA, 1989.

[6] M. V. Bruschke and S. G. Advani. A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media. *Polymer Composites*, Vol. 11, no. 6, pp. 398–405, 1990.

[7] F. Trouchu, R. Gauvin, and D. M. Gao. Numerical Analysis of the Resin Transfer Molding Process by the Finite Element Method. *Advances in Polymer Technology*, 12(4):329–342, 1993.

[8] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices.* University of Minnesota and the Army HPC Research Center, 1997.

[9] Portland Group, Major Shared Resource Center Symposium Series, U. S. Army Research Laboratory, 2000.

[10] R. Kanapady. Parallel Implementation of Large Scale Finite Element Computations on a Multiprocessor Machine: Applications to Process Modeling and Manufacturing of Composites. Masters Thesis, 1998. University of Minnesota.

[11] K. H. Law. A Parallel Finite Element Solution Method. *Computers & Structures*, Vol. 23, no. 6, pp. 845–858, 1985.

[12] D. R. Shires, R. V. Mohan, and A. Mark A Study of Parallel Software Development with HPF and MPI for Composite Process Modeling Simulations DOD Users Group Conference, Albuquerque, NM, 2000.

[13] T. Chilimbi, M. Hill, and J. Larus. Making Pointer-Based Data Structures Cache Conscious. *Computer*, vol. 33, no. 12, pp. 67–74, 2000.

[14] G. Kumfert and A. Pothen. Two Improved Algorithms for Envelope and Wavefront Reduction. *BIT*, Vol. 37, no. 3, pp. 559–590, 1997.

[15] E. Cuthill and J. McKee. Reducing the Bandwidth of Sparse Symmetric Matrices. *24th National Conference*. Association for Computing Machinery, pp. 157–172, 1969.

[16] D. R. Shires, R. V. Mohan, and A. Mark. Improving Data Locality and Expanding the Use of HPF in Parallel FEM Implementations. International Conference on Parallel and Distributed Processing Techniques and Applications, LasVegas, NV, 2000.