Fourteenth International Conference on Domain Decomposition Methods
Editors: Ismael Herrera , David E. Keyes, Olof B. Widlund, Robert Yates

21. Experiences with FETI-DP in a Production Level Finite Element Application

K.H. Pierson[1], G.M. Reese[2], P. Raghavan[3]

Introduction The need for predictive, qualified models of very complex structures drives the requirements for large scale structural analysis. Reduced testing in the nuclear weapons program is a driving factor at the DOE labs. In addition, more detailed models reduce the need for engineering approximation, improve accuracy and often simplify model construction. Uncertainty in model parameters (for example, variations in joint preloads) can require multiple analyses for evaluation of a structure. Salinas was designed to meet the needs of a very large scale, general purpose, structural dynamics analysis ([1] and [14]).

Salinas was implemented with the goal of providing predictive modeling of complex structural systems. This necessitates a full suite of structural elements and solution methods which must perform reliably on serial and distributed memory machines. Robust solution methods and platform portability are critical. Sensitivity analysis and optimization capabilities are also required for application to the design and uncertainty quantification communities.

Salinas is implemented on a variety of Unix(tm) and Unix-like platforms. The core libraries are written in C++ using MPI communications. This facilitates extensibility to a full range of solvers, solution methods and element libraries. Scalability to thousands of processors is achieved through application of Finite Element Tearing and Interconnecting (FETI) methods ([2], [7], [6]). Recently, FETI-DP, the Dual-Primal Finite Element Tearing and Interconnecting method has been implemented as the replacement to the one-level FETI method previously used (see discussion below). High performance over a range of platforms is obtained through effective use of optimized BLAS routines. The BLAS routines are the building blocks for the sparse serial and parallel direct solvers used within Salinas/FETI-DP ([11], [13]).

Salinas has been used for production solutions of linear and nonlinear statics and implicit transient dynamics, and for eigen analysis and modal superposition solutions (such as frequency response, modal transient and random vibration). Extremely complex models have been analyzed utilizing combinations of beams, shells and solids. The models contain hundreds of different materials which may differ in modulus by ratios greater than 10^6. Models larger than 100M degrees of freedom (dof) have been solved with demonstrated scalability above 3000 processors. Salinas is limited to small deformation analysis, but some nonlinear elements have been added, and more are under development.

FETI-DP Overview We present an overview of the FETI-DP method to keep this paper self-contained. Let the global domain Ω be partitioned into a set of N_s, non-overlapping subdomains Ω^s. Select a set of corner points for each subdomain such that all zero energy modes are suppressed if Dirichlet boundary boundary conditions are applied to the set of corner points. The selected corner points remain primal unknowns which are used to define a sparse coarse grid matrix for FETI-DP. See [9] and more recent work by Lesoinne appearing in these proceedings about optimal corner point selection. Define u^s as the unknown solution vector associated with subdomain s. Split the global solution vector, u, into two sub-vectors

[1]Sandia National Laboratories, khpiers@sandia.gov
[2]Sandia National Laboratories, gmreese@sandia.gov
[3]The Pennsylvania State University, praghavan@psu.edu

such that:

$$u = \begin{bmatrix} u_r \\ u_c \end{bmatrix} = \begin{bmatrix} u_r^1 \\ \vdots \\ u_r^{N_s} \\ u_c \end{bmatrix} \tag{0.1}$$

where u_c is a primal unknown vector over all selected corner dof and u_r^s is the unknown vector for all remaining subdomain dof on subdomain s. The subdomain operator can be partitioned into the following 2x2 block matrix.

$$K^s = \begin{bmatrix} K_{rr}^s & K_{rc}^s \\ {K_{rc}^s}^T & K_{cc}^s \end{bmatrix} \tag{0.2}$$

Global equilibrium can be written by introducing unknown Lagrange multipliers exactly like the classical one-level FETI method.

$$\begin{bmatrix} K_{rr}^1 & \dots & 0 & K_{rc}^1 B_c^1 & {B_r^1}^T \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & K_{rr}^{N_s} & K_{rc}^{N_s} B_c^{N_s} & {B_r^{N_s}}^T \\ {B_c^1}^T {K_{rc}^1}^T & \dots & {B_c^{N_s}}^T {K_{rc}^{N_s}}^T & \sum_{s=1}^{N_s} {B_c^s}^T K_{cc}^s B_c^s & 0 \\ B_r^1 & \dots & B_r^{N_s} & 0 & 0 \end{bmatrix} \begin{bmatrix} u_r^1 \\ \vdots \\ u_r^{N_s} \\ u_c \\ \lambda \end{bmatrix} = \begin{bmatrix} f_r^1 \\ \vdots \\ f_r^{N_s} \\ \sum_{s=1}^{N_s} {B_c^s}^T f_c^s \\ 0 \end{bmatrix} \tag{0.3}$$

where B_c^s maps the local corner equation numbering to global corner equation numbering, f_c^s is the external force applied on the corner dof, f_r^s is the external force applied on the remaining dof, ${B_r^s}^T$ is a boolean matrix that extracts the interface of a subdomain, and λ are the Lagrange multipliers. Let K_{rr} denote the block diagonal matrix of subdomain operators restricted to the remaining, r, points, K_{rc} the block column vector of subdomain coupling operator matrices and f_r the block column vector of subdomain force vectors. Using the same corner/remaining degrees of freedom matrix partitioning, we can rewrite the equilibrium equations compactly.

$$\begin{bmatrix} K_{rr} & K_{rc} & B_r^T \\ K_{rc}^T & K_{cc} & 0 \\ B_r & 0 & 0 \end{bmatrix} \begin{bmatrix} u_r \\ u_c \\ \lambda \end{bmatrix} = \begin{bmatrix} f_r \\ f_c \\ 0 \end{bmatrix} \tag{0.4}$$

The first equation can be solved for u_r since K_{rr} is a symmetric positive definite matrix if the selected corner points remove all of the local singularities. Then substitute the result into the compatablity equation (last equation in 0.4). The FETI-DP interface problem can be derived with some algebraic manipulation where the unknowns are λ, the Lagrange multipliers and u_c, the global corner degrees of freedom.

$$\begin{bmatrix} F_{rr} & F_{rc} \\ F_{rc}^T & -K_{cc}^* \end{bmatrix} \begin{bmatrix} \lambda \\ u_c \end{bmatrix} = \begin{bmatrix} d_r \\ -f_c^* \end{bmatrix} \tag{0.5}$$

where $F_{rr} = \sum_{s=1}^{N_s} B_r^s {K_{rr}^s}^{-1} {B_r^s}^T$, $F_{rc} = \sum_{s=1}^{N_s} B_r^s {K_{rr}^s}^{-1} K_{rc}^s B_c^s$, $d_r = \sum_{s=1}^{N_s} B_r^s {K_{rr}^s}^{-1} f_r^s$, and $f_c^* = \sum_{s=1}^{N_s} [{B_c^s}^T (f_c^s - {K_{rc}^s}^T {K_{rr}^s}^{-1} f_r^s)]$. The corner degrees of freedom, u_c, are condensed out to form the following symmetric positive definite Dual-Primal FETI interface problem which we solve

using a preconditioned conjugate gradient method. For a detailed derivation of this equation, please see [3]. Because of the preconditioning, the number of cg iterations (or FETI iterations) required for the solution is independent of model size. This scaling is demonstrated in the following sections.

$$\left[F_{rr} + F_{rc}{K^*_{cc}}^{-1} F^T_{rc}\right] \lambda = d_r - F_{rc}{K^*_{cc}}^{-1} f^*_c \tag{0.6}$$

The FETI operator defined above has an embedded coarse grid problem which can be written in the following form.

$$K^*_{cc} = \sum_{s=1}^{N_s} \left[{B^s_c}^T (K^s_{cc} - K^s_{cr}{K^s_{rr}}^{-1} K^s_{rc}) B^s_c)\right] \tag{0.7}$$

This new coarse problem has some highly beneficial properties over the previously defined two-level FETI coarse problem ([5]). First, this new coarse problem is symmetric positive definite sparse matrix. Secondly, only one forward/backward substitution has to be performed per FETI iteration. The original FETI algorithms required two forward/backward substitution operations per iteration. For a detailed derivation of FETI-DP see [9], [3], [4] and [12]. For a detailed mathematical analysis of the dual-primal FETI method one can review [8] and [10].

Scaled Problem Size Scalability We generate a series of model cube problems to assess scalability of Salinas and the underlying FETI-DP linear solver. The target platforms for assessing the scaled problem scalability of Salinas and FETI-DP are ASCI-Red, ASCI-Cplant and ASCI-White. The model cube problem is 13x13x13 hex elements per subdomain on ASCI-Red and ASCI-Cplant. On ASCI-White, we increased the model cube problem to 18x18x18 hex elements per subdomain to utilize the additional memory available. We scale the model cube problem with the number of processors keeping the size of the subdomains fixed. The number of subdomains is equal to the number of processors for all of our scalability experiments. The eight processor model cube problem is shown in figure 0.4.

For each of the platforms we evaluate the number of FETI iterations, the solver time, and the total time. The solver time (or FETI-DP time) represents the total time spent in the solver. This includes setup, factorization and solve time. The total time represents the time it takes to read the input geometry files, generate the subdomain matrices, solve a single $Ax = b$ problem and output the solution. The right hand side vector in all cases was a pressure load applied to the face opposite of the face where the Dirichlet boundary conditions were applied. The convergence tolerance was 0.001 for all platforms.

ASCI-Red The ASCI Option Red supercomputer, also known as the Intel Teraflops machine, is the first large-scale supercomputer built mostly of commodity, commercial, off-the-shelf (COTS) components. It has 4,536 compute and 72 service nodes each with 2 Pentium Pro processors. The system was delivered with 128 Mbytes of memory per node, but has been upgraded to 256 Mbytes of memory per node. The Pentium Pro processor runs at 333 MHz and has a peak floating-point rate of 333 Mflops. The system has over 1 Terabyte of real memory, and two independent 1-Terabyte disk systems. The system's 9216 Pentium Pro processors are connected by a 38x32x2 custom interconnect mesh with a bandwidth of 800 MB/s.

We show scalability results for up to 1000 nodes on ASCI-Red. Scaling the problem from sixty-four processors to one-thousand processors saw the number of iterations increase from 45 to 53. In figure 0.1 the total execution time is plotted for Salinas and FETI-DP running on ASCI-Red.

ASCI-Cplant CPlant is a large-scale massively parallel computer built from commodity computing and network computing components with a theoretical peak performance of

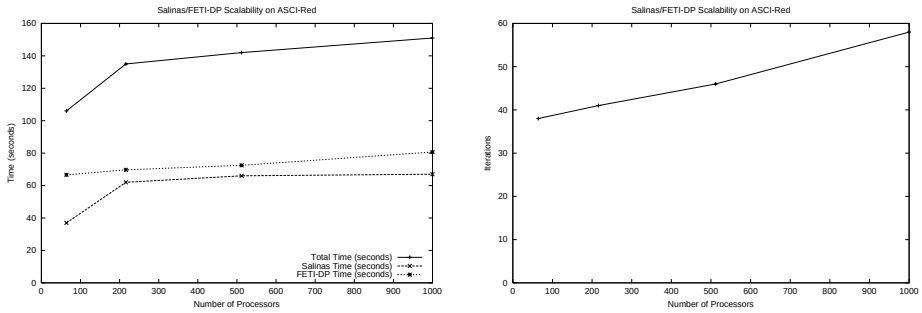

Figure 0.1: Performance on ASCI-Red for the following linear system sizes (processors): 446631 (64), 1479117 (216), 3472875 (512) and 6744273 (1000)

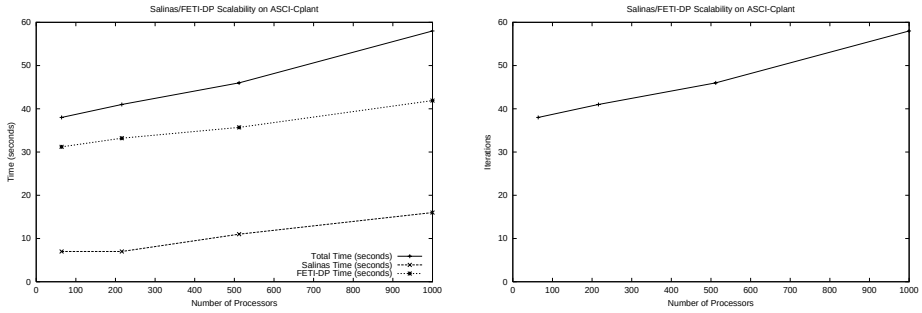

Figure 0.2: Performance on ASCI-Cplant for the following linear system sizes (processors): 446631 (64), 1479117 (216), 3472875 (512) and 6744273 (1000)

1.5 Tflops. The project goal of CPlant is to develop an architecture similar to the ASCI Red machine with entirely off-the-shelf commodity parts. Cplant uses the same partition model as the ASCI Red machine where pools of nodes can be divided into different categories, such as service, compute, and IO nodes.The compute processors are Compaq XP1000 workstations each containing a 500 MHz EV6 21264 microprocessor with 256 MB ECC SDRAM. The memory subsystem includes a 64KB instruction L1 cache and 4 MB L2 cache. The EV6 can issue four instructions per clock cycle and has two floating point units which amounts to a theoretical peak performance of 1 Gigaflops. The compute nodes are connected using Myrinet gigabit network.

We show scalability results for up to 1000 nodes on ASCI-Cplant. Scaling the problem from sixty-four processors to one-thousand processors saw the number of iterations increase from 45 to 53, identical to the ASCI-Red results. In figure 0.2 the total execution time is plotted for Salinas and FETI-DP running on ASCI-Cplant. The same model cube problem was tested on ASCI-Red and ASCI-Cplant. Therefore, one can directly compare the scalablity results for ASCI-Red in figure 0.1 versus the scalability results for ASCI-Cplant shown in figure 0.2. Based on these results, one can conclude that the total analysis time is approximately three times faster on ASCI-Cplant compared to ASCI-Red. This can be readily explained by the faster processors available on ASCI-Cplant. As expected, communication affects the overall scalability for large number of processors.

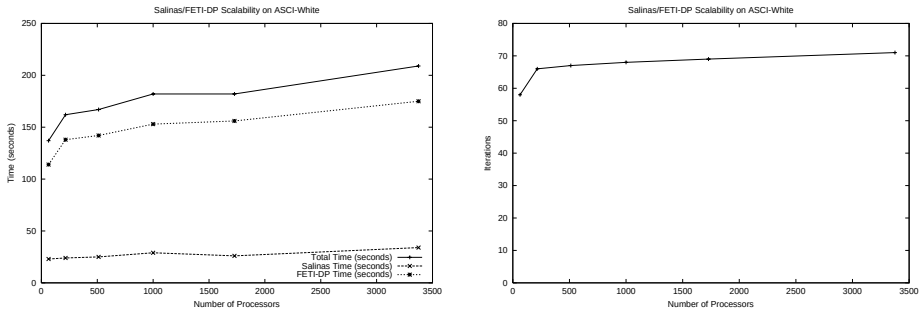

Figure 0.3: Performance on ASCI-White for the following linear system sizes (processors): 1167051 (64), 3885087 (216), 9145875 (512), 17789223 (1000), 30654939 (1728), 59707533 (3375)

ASCI-White ASCI White, the third step in a five step computational platform ladder, is currently the world's fastest computer with a peak speed slightly greater than 12 Tflops. The final ASCI platform has a goal to reach 100 Tflops peak performance by 2004. ASCI White is based upon IBM's latest SP technology. It comprises 512 symmetric multi-processor (SMP) machines, each possessing 16 processors, for a total of 8192 processing units. Each node consists of IBM's RS/6000 POWER3 symmetric multiprocessor (64 bit) architecture and this Nighthawk-2 SMP node is a stand-alone system with its own memory, operating system, local disk and 16 CPUs. POWER3-II processors are super-scalar pipelined 64 bit RISC chips with two floating point units and three integer units, capable of executing up to eight instructions per clock cycle and up to two floating point operations per cycle. At 375 Mhertz this processor is capable of producing a peak performance of 750 Mflops peak. The one cycle latency L1 cache is 128-way set associative and consists of 64KB data cache and a 32 KB instructions cache. The 4 MB L2 cache runs typically at half the processor speed and uses a direct mapped approach. Each processor has 1 GigaByte of available memory. All nodes are interconnected by the internal SP switch network, which has a bidirectional bandwidth of 300MB/second.

We show scalability results for up to 3375 nodes on ASCI-White. Scaling the problem from sixty-four processors to 3375 processors saw the number of iterations increase from 58 to 71. In figure 0.3 the total execution time is plotted for Salinas and FETI-DP running on ASCI-White.

Coarse Grid Solution Options We describe two FETI coarse grid solver technologies which are implemented in Salinas. The FETI-DP coarse grid as described above is a sparse matrix that couples all of the subdomains. This coarse sparse matrix has to be factored during the FETI-DP initialization step. During the FETI-DP solve step, one coarse matrix forward/backward solve is performed per iteration. The two coarse grid solver technologies are listed below.

- Redundant storage, factorization and forward/backward solves on each Salinas processor. In this option, a distributed inverse is computed and the relevant columns are stored on each processor. Solves are accomplished with local matrix-vector products.

- Distributed storage, parallel factorization and parallel forward/backward solves on a subset of the Salinas processors.

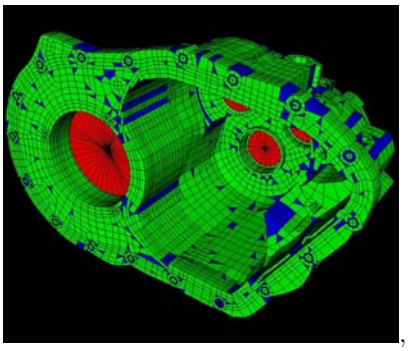
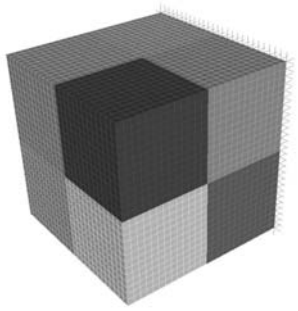

Figure 0.4: Engine block finite element model and Model cube

$N_p = N_s$	N_{eqn}^{coarse}	Serial Sparse	Parallel Sparse	Memory
25	831	8.3 sec.	7.9 sec. (8 processors)	1.7 MB
115	3999	9.7 sec.	5.9 sec. (16 processors)	14.6 MB
137	4815	14.0 sec.	8.4 sec. (32 processors)	19.9 MB
222	7425	25.8 sec.	11.4 sec. (32 processors)	34.0 MB
276	9339	36.4 sec.	12.8 sec. (32 processors)	45.6 MB

Table 0.1: Coarse grid setup and factorization time for engine block on ASCI-Red

For the redundant storage case, we choose a sparse matrix solver based on a multiple minimum degree ordering ([11]). The parallel factorization case is accomplished by the Domain Separator Cholesky package (DSCpack) ([13]).

Parallel Sparse Solver Experiments We experiment with solving large scale problems using large numbers of processors which results in increasingly larger FETI-DP coarse grid problems. A comparision is done between the redundant factorization and subsequent coarse grid matrix inverse technique versus using a parallel distributed memory sparse solver. At each iteration of FETI-DP, the parallel sparse solver does forward/backward solves in parallel. In the future, further studies will be conducted to determine if the parallel sparse solver in conjunction with the coarse grid matrix inverse technique will result in optimal CPU time.

Coarse Grid Scalability A finite element model of an engine block was chosen for coarse grid scalability studies. A picture of the engine block finite element model is shown in figure 0.4. This model is available in three increasing larger sizes. We choose the smallest model to illustrate the affect of increasing the number of subdomains for a fixed size problem on the FETI-DP coarse grid matrix. The small engine block model has 28498 nodes, 24363 elements and approximately 75000 degrees of freedom. This problem contains hex, wedge and triangular shell elements. We partition this model into 25, 115, 137, 222 and 276 subdomains respectively. We then solve an eigenvalue using Salinas and FETI-DP. In this Salinas solution method, FETI-DP is employed as the linear solver inside of a Lanzcos based parallel eigensolver. Table 0.1 shows the factorization times of the two coarse grid options on an increasing number of processors. Table 0.1 also shows the memory requirements of the serial sparse solver. The parallel sparse solver distributes the coarse grid LDL^T factor over the number of coarse solver processors. This effectively reduces the per processor memory requirements. For small coarse grid sizes, the redundant sparse direct method

outperforms the parallel sparse solver. Please note that this is mainly due to the calculation of the coarse grid inverse (a detailed description can be found in [6]) and subsequent replacement of sparse forward/backward substitution by matrix-vector multiplication during the FETI-DP iterations. For a sufficiently large problem, the factorization of the coarse grid becomes the dominant time and the parallel sparse solver begins to out-perform the serial sparse direct method. More importantly, the coarse grid eventually becomes too large to store on every processor and the only option is to use the parallel sparse distributed solver. Future studies will investigate tiling the parallel sparse solver on $N_p/N_{cs} = N_{tiles}$ while N_{cs} equals the number of processors solving the coarse grid matrix. This approach leaves an integer number of processors, $N_{rem} = mod(N_p/N_{cs})$ idle while the current parallel sparse solver implementation leaves $N_p - N_{cs}$ processors idle during coarse grid factorization and subsequent forward/backward solves required during FETI-DP iterations.

Conclusion We have shown that FETI-DP performs well in a production finite element application on a variety of massively parallel platforms. Scalablity was demonstrated on ASCI-Red, ASCI-Cplant and ASCI-White. We are actively pursuing parallel factorization of the FETI-DP coarse grid to enable further improvements in scalability.

Acknowledgements The authors would like to thank Charbel Farhat and Michel Lesoinne, from the University of Colorado at Boulder for all of their time, energy, and dedication in researching and implementing FETI methods into Salinas. The authors would also like to thank David Day, Manoj Bhardwaj, Tim Walsh, Ken Alvin, David Martinez, James Peery, Dan Segalman, and Sam Key for their support, contributions and insightful discussions. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

REFERENCES

[1] M. Bhardwaj, D. Day, C. Farhat, M. Lesoinne, K. Pierson, and D. Rixen. Application of the FETI method to ASCI problems - scalability results on one thousand processors and discussion of highly heterogeneous problems. *Int. J. Numer. Meth. Engrg.*, 47:513–535, 2000.

[2] C. Farhat. A Lagrange multiplier based on divide and conquer finite element algorithm. *J. Comput. System Engrg*, 2:149–156, 1991.

[3] C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D. Rixen. FETI-DP: A dual-primal unified FETI method – part I: A faster alternative to the two-level FETI method. *Int. J. Numer. Meth. Engrg.*, 50:1523–1544, 2001.

[4] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numer. Lin. Alg. Appl.*, 7:687–714, 2000.

[5] C. Farhat and J. Mandel. The two-level feti method for static and dynamic plate problems - part i: an optimal iterative solver for biharmonic systems. *Computer Methods in Applied Mechanics and Engineering*, 155:129–152, 1998.

[6] C. Farhat, K. H. Pierson, and M. Lesoinne. The second generation of feti methods and their application to the parallel solution of large-scale linear and geometrically nonlinear structural analysis problems. *Computer Methods in Applied Mechanics and Engineering*, 184:333–374, 2000.

[7] C. Farhat and F.-X. Roux. A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm. *Int. J. Numer. Meth. Engrg.*, 32:1205–1227, 1991.

[8] A. Klawonn and O. B. Widlund. FETI-DP Methods for Three-Dimensional Elliptic Problems with Heterogeneous Coefficients. Technical report, Courant Institute of Mathematical Sciences, 2000. In Preparation.

[9] M. Lesoinne and K. Pierson. Feti-dp: An efficient, scalable, and unified dual-primal feti method. In T. Chan, T. Kako, H. Kawarada, and O. Pironneau, editors, *Domain Decomposition Methods in Sciences and Engineering*, pages 421–428. DDM.org, 1999.

[10] J. Mandel and R. Tezaur. On the convergence of a dual-primal substructuring method. Technical report, University of Colorado at Denver, Department of Mathematics, January 2000. To appear in Numer. Math. `URL: http://www-math.cudenver.edu/ jmandel/papers/dp.ps.gz`.

[11] E. G. Ng and B. W. Peyton. Block sparse Cholesky algorithms on. *SIAM J. Sci. Stat. Comput.*, 14:1034–1056, 1993.

[12] K. H. Pierson. *A family of domain decomposition methods for the massively parallel solution of computational mechanics problems*. PhD thesis, University of Colorado at Boulder, Aerospace Engineering, 2000.

[13] P. Raghavan. Efficient parallel triangular solution with selective inversion. *Parallel Processing Letters*, 8:29–40, 1998.

[14] G. Reese, M. Bhardwaj, D. Segalman, K. Alvin, and B. Driessen. *Salinas User's Manual*. Sandia National Laboratories.