Fast and Reliable Solution of GDoF-Problems on NAVO/BABBAGE and AFRL/HAWK Systems

Scott Fawaz
United States Air Force Academy
scott.fawaz@usafa.edu

Börje Andersson
Swedish Defense Research Agency
ba@foi.se

Abstract

Fast and reliable structures and materials analysis of full fuselage, wing, and/or empennage sections is now for first time possible with access to highly optimized software, called STRIPE, and the NAVO/BABBAGE and AFRL/HAWK systems. Analyses of this type have opened up the possibility for considering the statistical uncertainties in material data, geometry, crack locations, etc. Very detailed analysis can be performed on huge models which are geometrically exact down to rivet details where growth of numerous cracks at numerous locations is studied. Access to this type of analysis results can drastically reduce inspection requirements, prevent premature retirement of old aircraft and increase aircraft safety; all resulting in a potential to save billions of dollars if implemented across the USAF fleet.

By applying a novel mathematical multi-scale scheme, these problems can be split into solving many thousand smaller problems and one very large problem. The efficient implementation of these two different activities is crucial for computing in a shared environment. This technique results in solution of 10^9 sets of equations for the large problem with greater than 10^4 right hand sides and thousands of smaller problems having about 10^6 degrees of freedom (DoF) each. Iterative solvers are not competitive so direct solvers must be used. The major difficulty in achieving scalability is then related to the extensive I/O traffic characteristic of commercially available (MSC/NASTRAN and ABAQUS for example) software.

The paper describes various techniques adopted to achieve high system scalability when solving the world's largest strength of materials problem related to aircraft maintenance and design. Support from major software vendors (SGI, IBM), MSRC's support specialists as well as I/O specialists at the University of Tennessee (as a part of the PET-program) have strongly contributed to the successful results demonstrated. The technological capability developed is demonstrated by analyzing an idealized C-130 center wing box.

1. Introduction

The requirement to solve GDoF (10^9 DoF) finite element meshes is a result of trying to assess the residual strength of the C-130 center wing box (CWB). Residual strength is simply the maximum load carrying capability of a structure with a given damage condition. The structural integrity of the C-130 CWB has been under investigation for quite some time.[1] One damage condition of interest was determined during an extensive destructive teardown and failure analysis.[2] The computational methods developed control the error in the solution, are validated by strain data from test such as that shown in reference [3] and can be used to predict this specific damage condition as well as 1000's more.

In order to obtain reliable solutions, the contact problems riveted connections, multiple interacting fatigue cracks, corrosion damage, etc. must be considered in the large parts of the structural parts analyzed. Thus, statistical fatigue analysis is a necessity. We use a multi-scale scheme for solution of contact problems. In order to effectively use this scheme, the equation solver must efficiently handle many right hand sides. Although iterative solvers are superior if the ill-conditioning and load balancing problems can be solved, which is an open question for problems of this size, direct solvers are preferred when solving problems with hundreds of thousands of right hand sides. A direct equation solver designed for solution of problems with many right hand sides with emphasis on retaining good scalability on hardware systems having thousand of processors is described below. Very large meshes of a C-130 CWB resulting in problems of GDoF-size have been created at the USAF Academy's Center for Aircraft Structural Life Extension (CAStLE). Large-scale analyses on the AFRL/HAWK and NAVO/BABBAGE systems using thousands of processors have been performed.

2. A Direct Solver for Solution of GDoF Problems

Domain Decomposition

A domain decomposition approach is being developed where non-iterative techniques are used on each analysis level including the top level. The basic method used hence is the classical sub-structuring technique (i.e. computation of the Schur complement in mathematical notation). The system of algebraic equations to solve on each analysis level can then be written:

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{Bmatrix} x_i \\ x_b \end{Bmatrix} = \begin{Bmatrix} F_i \\ F_b \end{Bmatrix} \quad (1)$$

The internal degrees of freedom are denoted x_i and the boundary degrees of freedom x_b, respectively. The computational steps are,

- Assembly of domains from the analysis level below (or element matrices on lowest level) in order to form matrices *A, B* and *C*, respectively
- Factorize of the *A*-matrix
- Calculate $[A^{-1} \cdot B]$
- Create domain stiffness $[C - B^T \cdot A^{-1} \cdot B]$
- Create right hand side $\{F_b\}[B^T \cdot A^{-1}]\{F_i\}$

If the memory is large enough, the five steps are computed sequentially without writing any data to disk. If memory is insufficient, temporary data is written to disk creating an enormous I/O bottleneck which precludes scalability to large systems.

To obtain good scalability, the subdivision of the domain into sub-domains must

- Minimise the total number of floating point operations required
- Equally distribute the computational load over the hardware system used
- Consider memory limitations on distributed memory systems like NAVO/BABBAGE
- Consider number of partitions used on shared memory systems like AFRL/HAWK

Presently, an automated bottom-up approach is used to define the domains on various levels. In the first step the finite element mesh is divided into a number of element groups containing typically 30-40 finite elements in case of stiffened shell structures. The process is repeated on level after level until only one global element remains (top level). Typically 3-12 levels are used for problems ranging from MDoFs to GDoFs. The load balancing algorithm is iterative and is more difficult on the higher domain levels. The measure of the computational work in the load balancing algorithm is the number of floating point operations which can be calculated exactly since a direct solution technique is used. Ideally the amount of I/O needed on the different levels should be considered too when optimizing the load balancing.

Table 1 exemplifies the approximate number of domains created at different analysis levels during the decomposition of a generic finite element mesh having 13 million finite elements, Figure 7.

Table 1 Example of the number of domains used per level when analyzing a mesh with 13 million finite elements

Level	1	2	3	4	5	6	7	8	9	10
Number of Domains	360,000	90,000	16,000	4,000	1,000	260	80	25	7	1

The solver uses mixed OpenMP and MPI software where OpenMP is used in each MPI-process. On distributed memory machines, one MPI process is typically allocated to one computational node. The main reason for using mixed MPI/OpenMP is that each MPI process must have access to sufficient memory. As mentioned, the load balancing algorithm, which governs the subdivision into domains, is based on the criterion that each MPI process shall perform roughly the same number of floating point operations. Initially each MPI process, computational node, works completely independently and analyzes a number of levels, starting from the lowest level, completely independent of the other MPI processes. The number of levels to be analyzed by one MPI process is selected from the criterion that the computational work in the different MPI-processes should be equal. This is often achieved within a few percent variation.

Practical experience shows that for large wing type structures like the C-130 CWB it is optimal if the different MPI processes compute completely independent of one another during the 5-8 lowest levels.

When close to perfect load balancing can no longer be achieved by using completely isolated branches, one branch per MPI process, the solver shifts mode and assigns several MPI processes to calculate the domain stiffness and right hand sides. Each analysis level has the optimization task to find to the optimum distribution of parallel jobs, i.e. the selection of domain sizes, their distribution and the number of MPI-processes to assign to each parallel job. Since the individual MPI-processes can not share a common memory this part of the solution scheme becomes more I/O-intensive something that prevents good scaling properties on hardware systems like the AFRL/HAWK.

We first discuss performance on the AFRL/HAWK system. The learning curve was quite steep on the SGI Altix shared memory system HAWK at AFRL. Several months were spent simply porting the source code from AFRL/EAGLE (SGI Altix as well) to HAWK. The source code and makefiles did not change between the two platforms. EAGLE was using SLES 9 and Intel FORTRAN and C/C++ compiler versions 8.1; whereas, HAWK was using SLES 10 and compiler versions 9.1 initially. STRIPE could not be recompiled and produce a properly executing code under version 9.1 on HAWK. The same problem would have existed on EAGLE using compiler versions 9.1; thus, the problem appears to be with the compiler and not with the hardware. Not until version 10.0.023 was available did we have a working executable on HAWK.

By default, MPI processes are always placed by (SGI) mpirun and (LSF) pam commands in a round-robin fashion. The omplace command was introduced in MPT 1.15, and was new to HAWK. Simply a dplace wrapper script, it is not "required", but is much easier to use than dplace. Using the wrong dplace options can result in all processes being placed on just one CPU.[4] Many unsuccessful attempts were made using dplace before exclusive use of omplace began in Sep 2007.

As our scaling attempts increased, we did so by increasing the size of the job (computational effort), number of CPUs and number of MPI processes. Regardless of number of CPUs and MPI processes, as the computational effort increased, the analysis would abort with no information regarding the cause. After close observation of executing jobs by the SGI experts, the cause was determined to be that the MPI processes were exceeding their available stacksize. Increasing the value of the stacksize available to each MPI thread via the KMP_STACKSIZE environment variable from the default (1MB) to 1GB in addition to reducing the number of OpenMP threads resulted in successful job execution.

For meshes above 742 MDoF, the analysis requires more than 12 MPI processes to finish within the maximum two week queue. However, increasing the number of MPI process requires more memory than is available on the HAWK fat partitions which have 4 GB memory per CPU. Two possible solutions exist; one, execute the 12 MPI process jobs for more than two weeks; two, execute analysis across partitions to access more available memory. The two fat partitions on HAWK have 500 CPUs and 2 TB memory; whereas requesting the maximum CPUs per job allowed, 2000, makes 4 TB of memory available per job but across multiple partitions.

Improvements were made to the domain decomposition phase of the code in an attempt to solve the larger jobs within the queue time limit. A 30% increase in domain decomposition was achieved; however, meshes larger than 742 MDoF still could not be solved.

Our success in executing cross partition jobs has been limited. The largest job executed to date is 418 MDoF which is the idealized C-130 CWB with a *p-level* of 3. This size job is easily 10 times smaller than what is required to analyze the full C-130 CWB. The cause of the job crashes above $p=3$ is due to exceeding the requested memory due to the unpredictable memory required for the system buffer cache. This problem exists on the single partitions jobs as well, but is somehow exacerbated in the cross-partition jobs. Currently, there is no solution to this problem.

Another unresolved issue with cross-partition jobs is long wait times on messages sent from the metadata server (MDS). The SGI experts noticed a correlation between long wait times on messages sent from the MDS and hard crashes of the cross-partitions jobs. The cause of the long wait times has yet to be determined.

Lastly, excessive ethernet errors were occurring on three of the batch nodes which caused cross-partition MPI jobs to crash.[4] We learned of this issue on 3 June 08 and have not determined how many jobs in the past year have crashed for this reason.

Many right hand sides

For problems of GDoF size, the number of right hand sides can exceed 50,000. The strategy used for obtaining good computational efficiency in such a case is

- Calculate all solution vectors on the top level and write solution data only for boundary nodes for those domains forming the top level domain
- For each level, and each domain on that level, calculate the solution data for the boundary nodes for domains on the level directly below
- On the lowest domain level, assemble all local solution vectors for each load case to a global vector with degrees of freedom

As mentioned previously, on each domain level the different MPI processes work independently. Even for a very large number of right hand sides, the stiffness matrix needs to be read from disk only once which keeps the I/O operations at a minimum.

The wall time, T_{LC}, needed to decompose a problem with LC right hand sides has, for a 50 MDoF problem with 50,000 right hand sides been measured on NAVO/BABBAGE to

$$T_{LC} = T_{up} \cdot (1 + LC/22000) \qquad (2)$$

Where T_{up} is the wall time needed to decompose a problem with just one right hand side. The wall time needed to calculate the internal degrees of freedom once the top level problem has been solved, T_{LC2}, was observed to be:

$$T_{LC2} = T_{down} \cdot (1 + LC/4500) \qquad (3)$$

Where T_{down} is the wall time needed with just one right hand side.

Accurate and Fast Solution of 3D Contact Problems

In order to accurately analyze the C-130 CWB mesh, the contact problem, which generates 10^4 to 10^5 right hand sides, between the rivets/bolts and aircraft structure must be solved. Due to the nonlinear character of the contact problem an iterative scheme has been designed which is both robust, virtually exact, converges fast and simultaneously takes the mathematical character of the problems appearing during the iteration steps into account. The contact problem, Figure 2, of interest is split into three problems. The global problems a) and c) in Figure 2 are analyzed without considering either the contact surfaces or the crack surfaces, i.e., all surfaces are in perfect sliding-free contact. The splitting scheme used then has the advantage that the global analysis a) and c) which is very costly for problems of GDoF-size do not have to be repeated for varying crack sizes and contact surfaces. The local problems b), see also Figure 3, are analyzed for fixed (but *á priori* unknown) locations of the contact surfaces for a set of loading functions defined by carefully selected traction basis functions $Q_1 - Q_4$ and scaling factors $\beta_{(m,1)}$, the primary unknowns.

A necessary condition to be satisfied at the two points separating contact and no-contact is:

$$K_I = 0 \qquad (4)$$

and that normal stresses are compressive at the part of the circular boundary being in contact. Equation (4) can be used as a very precise criterion to find the two points separating contact and no-contact. A simple Newton-type algorithm can then be used to find the contact points satisfying Eq. (4) for each bolt/rivet. The splitting scheme is simply used as a very fast solver for determination of stress intensity factors at the crack-tips and at potentially correct contact positions during iterations. Note that during iterations and before convergence is achieved $K_I \neq 0$ implying that there is a strong stress singularity at the assumed contact positions. Robustness of the approach is obtained by using a *hp*-type of mesh moving with the location of the assumed contact points. Experience shows that the algorithm converges to about 3 digits accuracy in less than 5 iterations. Figure 4 gives an overview of the contact solution scheme in a 3D setting.

CPU efficiency: The code is optimized for efficient memory and cache utilization and executes typically with 70-80% of the theoretical peek processor speeds (typically 3-8 GFLOPs/CPu on state of the art hardware 2007).

I/O efficiency: Scalability to thousands of processors on the GDoF problems of interest is prevented by the significant I/O activities needed in a direct solver. Different techniques are used in the solver to reduce the time needed for I/O. The techniques used are use of large memory buffers, recalculation of data, reuse of data in memory buffer, building local data structures optimized for each MPI process etc.

Table 2 exemplifies the scaling performance for a small shell problem having about 20 MDOFs. Speed up is measured in wall time, hence data shown includes all I/O-activities. The table shows that by increasing the number of processors by a factor five, from 2 nodes to 10 nodes on NAVO/BABBAGE, the time needed to solve the problem decreases by a factor four. The high theoretical maximum processor speed, i.e., 7.6 GFLOPS and the very efficient utilization of processors in the code, 70-80% of theoretical speed, makes I/O effects more critical when it comes to scalability.

Table 2 Scaling properties measured in wall time during solution of a shell problem with 20 MDoF

Relative Performance	CPU	8	16	32	64	96	128	160	192
babbage		-	-	1.00	1.88	2.68	3.36	3.97	4.63
hawk		1.00	0.59	7.00	7.00	-	4.04	4.29	3.75

On the AFRL/HAWK system with theoretical maximum processor speed of 6.4 GFLOPS; time to solution is extremely sensitive to I/O performance just as with NAVO/BABBAGE. On AFRL/HAWK, workspace for executing jobs reside on two large file systems (/workspace and /large). If all of the I/O is with files residing in the same directory, the files will all reside on a small subset of disks which severely limits the available I/O bandwidth. If each file is placed in a different directory, they would be spread across many disks and provide much greater I/O performance.[4] This concept was investigated by executing a 133 MDoF analysis using 256 CPU, 512 GB memory, and 8 MPI processes. This analysis was run ten times with all file I/O on a single directory and ten times with all file I/O on multiple (roughly 20) directories. The mean and standard deviation wall time to solution, which includes all I/O activity, for the single directory execution was 72.3 hrs and 5.6 hrs and mulitiple directory runs was 56.6 hrs and 14.7 hrs; respectively. Thus, using multiple directories increases the time to solution by approximately 22%.

The I/O performance is also sensitive to overal I/O load on the entire system. During the runs mentioned above, multiple jobs (over 2.4 GDoF) were executing concurrently; therefore creating competition in available bandwidth amongst the jobs. The fastest times to solution are obtained when only one or two jobs are running simultaneously. Specifically, when these jobs are run by themselves, the time to solution is about 2.6 times faster.

Large-Scale analysis of a generic wing on NAVO/BABBAGE and AFRL/HAWK

A little more than 50% of the large C-130 CWB mesh has been developed. Figure 1 shows the structural domain currently being meshed at CAStLE. A stringer detail is shown in order to exemplify the mesh density used. The mesh generator TrueGrid™, developed by XYZ Scientific, is used to create the large finite element mesh which is estimated to consist of about 25 million finite elements. A representative mesh of the C-130 CWB is shown in Figure 5 with a close-up view of the skin-stringer interface shown in Figure 6.

In order to fine tune software optimisation, a very fine generic mesh of a wing was generated. Figure 7 shows details from this mesh. Skin, rivets, frames and stiffeners were all modelled in great detail as 3-dimensional objects. The full mesh has 13.3 million hexahedral elements.

The problem was solved using the *hp*-version of the finite element method. Table 3 shows the number of degrees of freedom, the number of processors used and the wall time needed. The 256 processor job was analysed on a very large SGI-system which makes the execution times vary a lot due to entire I/O load on the full system.

A closer look on solution characteristics from the three executions shows that the load balancing over the execution period was excellent.

Table 3. Analysis of the generic wing on SGI/Altix 4700 and NAVO/BABBAGE systems

p-level	Host	MDoF	Processors	Wall time (*h*)
2	puma	243	48	43
	eagle		64	30
	hawk		256	28
3	eagle	418	64	55
	hawk		256	60
4	puma	742	64	288
	hawk		192	252
5	babbage	1242	1728	15

puma → SGI Altix 4700 BX2, 78 CPU, 624 GB memory, 100 TB disk at USAFA
eagle → SGI Altix 3700, 2048 CPU, 2 TB memory, 100 TB disk at AFRL
hawk → SGI Altix 4700, 9000 CPU, 40 TB memory, 300 TB disk at AFRL
babbage →IBM P5+, 3072 CPU, 6.4 TB memory, 139 TB disk at NAVO

Fast solution of GDOF problems.

The long execution times exemplified in Table 3 are due to the huge computational work needed for these large problems. In order to get reasonable execution times, a hardware system with many more processors and, ideally, higher I/O-capacity has to be used. The IBM POWER5/6 series at NAVO are hardware with excellent properties in this respect. Since these machines are distributed memory machines, the limited memory issue is to be considered carefully when designing the load balancing algorithm.

The domain subdivision described above is based on a bottom-up approach and results in excellent load balancing. The memory requirement is determined, if extensive I/O is to be avoided, by the size of the largest domain on the top level. This approach does not presently consider the memory requirement on the top level. Thus, the memory needed might be rather non-optimal (unnecessarily large) which might prevent solution of very large problems on distributed memory machines with limited memory per computational node. The present approach, adapted to the development resource available, is to use a combined top-down and bottom-up approach for the domain division, i.e. the global mesh is in a first step divided into a small number (4-64, say) of large domains where the division is made with respect to available memory per node on the hardware platform and load balancing on the level just below the top level. The bottom-up domain division technique is then used for subsequent mesh subdivision of the resulting global domains.

The usefulness of such a technique for solution of GDoF size problems on distributed memory machines was first tested semi-manually. The global domain shown in Figure 7 was used in a series of tests. Polynomial order p=5 resulting in a problem with 1.24 GDOF's was used for testing. Problems of this size are needed to achieve the objectives of the present Challenge project. The authors are unaware of any finite element analysis using more DoF. Figure 8 summarize results from the most recent test. The 13.3M element mesh was first manually split into 8 sub-domains, using a top-down approach, of rather different computational sizes in order to the fit into the memory available per computational node on NAVO/BABBAGE. Load balancing was obtained by assigning 8-20 MPI-processes to the different domains. The top level analysis and restarts were done manually. In all, 1728 processors were used. The figure shows that the manual domain subdivision was not perfect since execution times for domain creation, 10 levels, varied between 11.1 and 12.8 hours. The total execution time was around 15 hours for this large 1.214 GDOF problem on NAVO/BABBAGE. A similar analysis was conducted on AFRL/HAWK using 256 processors, 732 GB memory, and 20 mpi. A comparison is made in Table 4. Columns

Table 4 Comparison of Domain Decomposition for 1.2 GDoF Mesh

	p-level = 5		
MDoF	Hawk Relative Performance	Babbage Relative Performance	Babbage/Hawk Relative Performance
152.3	1.00	1.01	1.45
152.3	6.30	1.82	2.60
152.5	6.22	1.90	2.72
149.8	1.71	1.68	2.40
152.3	1.70	1.30	1.87
152.3	1.71	1.28	1.83
152.5	1.35	1.00	1.43
149.8	1.79	1.65	2.36

two and three show the relative performance on the given system. The last column shows a comparison of the fastest job, which was on AFRL/HAWK, to jobs on both AFRL/HAWK and NAVO/BABBAGE. In other words, the NAVO/BABBAGE jobs are 1.43 to 2.72 times slower than the fastest job on AFRL/HAWK in the domain decomposition phase of the solution.

The top-down, bottom-up approach is presently being implemented together with MPI-2 functionalities for improved I/O performance. With these software improvements and using faster hardware, IBM/POWER6, the turnaround time will decrease significantly.

The I/O performance on NAVO/BABBAGE was measured by Sameer Shende and David Cronk using the TAU Performance System.[5] An aircraft stiffened shell mesh with 133 MDoF was analyzed using 1,024 cores and 16 MPI ranks in 9 hours wall time. They observed that I/O operations on rank 0 of each node took over two hours longer than the I/O operations on the other ranks.[5] The two hours on rank 0 was spent in an assembler subroutine. Since the time of the measurement, the assembler has been rewritten and assembly time will be reduced by at least a factor of 10. Thus, a time to solution savings of 22% (2 hrs / 9 hrs) can be realized. In addition, the mean peak write and read bandwith was 4.7 and 4.2 GB/s; respectively.[5] The peak write bandwidth achieved was 5.62 GB/s.[5] Not until now has the I/O performance of STRIPE been observed to such detail.

3. Error Estimation

We employ an *hp*-version of the finite element method in order to estimate the errors in any quantity of interest. Hence, very accurate solutions are derived by using properly designed meshes and increasing the number of degrees of freedom per element (i.e. the order p of the basis functions in the finite element approximation). The number of degrees of freedom needed in order to obtain a solution of high accuracy depends on the mathematical properties of the exact solution, u. The solutions sought do in all practical cases exhibit singularities at geometrical edges, vertices and at faces where material data changes abruptly. The solution can in the neighborhood of such a singularity be expressed as $u_{local} = K \cdot r^{\lambda} \cdot f(\theta)$ where r is the distance to the edge/vertex/material interface and θ a polar angle. K is a complex valued stress intensity function which varies along edges.The complex scalar, or function in case of edges and for non uniform material data, λ and the function f depend on local data at the edge/vertex, that is local material properties, boundary conditions and geometry. The intensity function K depends on global data of the problem.

For reliable solution of the equations approximate knowledge of (f, λ) is of primary importance since,

- the regularity of the exact solution u is determined by (f, λ) and hence the approximation properties of the numerical solution scheme used to derive approximate solutions
- for calculation of stress functions which are important engineering parameters in fatigue and damage tolerant design

We are interested in displacement approximations $\bar{u}$, stress approximations $\bar{\sigma}$, and stress intensity function approximations $\bar{K}$, at points (x,y,z) to the exact mathematical values u, σ, K such that

$$|(u(x) - \bar{u}(x))/u(x)| \leq \varepsilon_1 \quad (5)$$

$$|(\sigma(x) - \bar{\sigma}(x))/\sigma(x)| \leq \varepsilon_2 \quad (6)$$

$$|(K(x) - \bar{K}(x))/K(x)| \leq \varepsilon_3 \quad (7)$$

$|\blacksquare|$ denote the infinity norm, that is point wise values at (x,y,z) and ε_i denote tolerances.

Relative errors of the order 0.01 to 0.001 are rather easy to obtain at any point (x) when the *hp*-version of the finite element method (FEM) is employed. The approach used to derive accurate solutions with control of the error in the solution is the following

- Use mesh generation principles which are based on *á priori* known properties of the exact mathematical solution near geometrical edges/vertices/material interfaces which, in linear cases, leads to an exponentially decreasing error with increasing number of degrees of freedom, *N*
- Use hierarchical properties of the basis functions for fast calculation of element stiffness matrices and residuals to be used large in scale
- Use advanced post-treatment for accurate extraction of solution data like stress intensity factors, *K* etc.

In Figure 9, the relative percent error in the Von Mises stresses in the vicinity of the crack tip using $p = 2$ and $p = 4$ are shown. The error is quite large, ±20%, which has safety implications since the $p = 4$ solution gives higher stresses indicating a lower residual strength. Error estimation with this level of fidelity is not usually obtained using traditional *h*-version finite element analysis.

The software used for large scale analysis can only consider finite elements of solid type, brick and wedge type of elements.

For each of the three element types considered, a local coordinate system (ξ, η, ζ) may be assigned and topological entities as corners, edges and faces defined. The corresponding element basis functions are denoted corner, edge, face and internal basis functions, respectively. Table 5 shows the number of basis functions as function of the polynomial order *p* used.

Table 5 Number of vertex, edge, face, and internal basis function for *p*-type finite elements available

Element	η_v	η_e	η_f	η_i
Brick Q	8	$12(p-1), p \geq 2$	$6(p-1)^2, p \geq 2$	$(p-1)^3, p \geq 2$
Brick Q'	8	$12(p-1), p \geq 2$	$3(p-2)(p-3), p \geq 2$	$(p-3)(p-4)(p-5)/6, p \geq 6$
Wedge Q	6	$9(p-1), p \geq 2$	$3(p-1)^2 \mid p \geq 2 + (p-1)(p-2) \mid p \geq 3$	$(p-1)^2(p-2)/2, p \geq 3$
Wedge Q'	6	$9(p-1), p \geq 2$	$3(p-2)(p-3)/2 \mid p \geq 4 + (p-1)(p-2) \mid p \geq 3$	$(p-2)(p-3)(p-4)/6, p \geq 5$
Simplex	4	$6(p-1), p \geq 2$	$2(p-1)(p-2), p \geq 3$	$(p-1)(p-2)(p-3)/6, p \geq 4$

The number of nodes in a Q'-type brick element is for example 20, 50, 105, and 192 for polynomial orders *p*= 2, 4, 6 and 8, respectively. The solutions described are all obtained with Q' type of finite elements. Q-type elements often show better approximation properties and scalability than Q'-type elements. The draw back with Q-elements is that the number of degrees of freedom increases very fast with *p* so for big meshes only a few solutions are affordable, hence making it difficult to estimate the solution accuracy. Non-homogenous *p*-distributions can be assigned automatically via a self-adaptive scheme, or manually.

4. Summary

The focus of the present paper is the development of advanced computational methods for reliable fatigue and residual strength analysis of bolted-riveted-bonded metallic structures. The main accomplishments were

- Created FE-meshes for majority of C-130 CWB
- Fast solver for solution of GDOF-problems with 10^4 to 10^5 right hand sides
- Benchmarking on generic C-130 models with up to 1.2 GDoFs
    - GDoF problems require top-down, bottom-up domain decomposition for solution

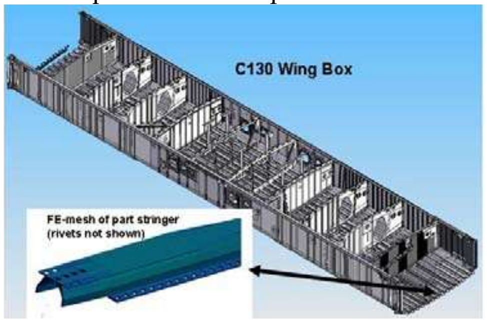

Figure 1. C-130 CWB. Stringer shown exemplifies a characteristic mesh density

- Factors affecting time to solution
    - Memory per CPU (AFRL/HAWK), Memory per node (NAVO/BABBAGE)
    - I/O performance
    - Constraints imposed by shared environment

Acknowledgements

The authors wish to express their gratitude to the Materials Directorate, Air Force Research Laboratory, Wright Patterson AFB for sponsoring this work. We also gratefully acknowledge the support of Brent Anderson, AFRL/MSRC, John Skinner, NAVO/MSRC, and Steve Senator, USAFA/M&SRC in development of the computational platform. This work could not have been accomplished without the support of the DoD High Performance Computing Modernization Program.

Reference

[1] Bateman, Geoffrey and Peter Christiansen, "C-130 Center Wing Fatigue Cracking -- A Risk Management Approach," Proc. of the 2005 USAF Aircraft Structural Integrity Program Conference, 29 Nov – 1 Dec 2005, Memphis, TN.
[2] Shoales, Gregory, Sandeep R. Shah, Justin W. Rausch, Molly R. Walters, Saravanan R. Arunachalam, and Matthew J. Hammond, "C-130 Center Wing Box Structural Teardown Analysis Final Report," USAFA-TR-2006-11.
[3] Snider, Lawrence H., Franklin L. Reeder, and William Dirkin, "Residual Strength and Crack Propagation Tests on C-130 Airplane Center Wings With Service-Imposed Fatigue Damage," NASA_CR-2075, 1972.
[4] Andersen, Brent, Personal Communication, 13 Mar 08.
[5] Shende, Sameer, Allen Malony, Alan Morris and David Cronk, "Observing Parallel Phase and I/O Performance Using TAU," 2008 High Performance Computing Modernization Office User Group Conference, Seattle, WA, 13-18 July 2008.

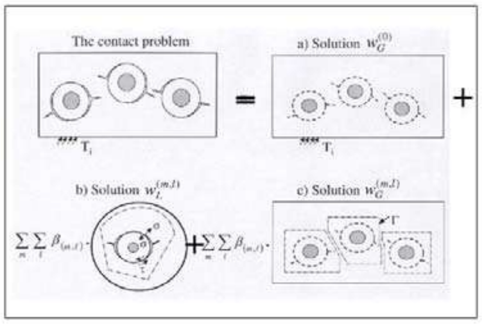
The contact problem
a) Solution $w_G^{(0)}$
b) Solution $w_L^{(m,l)}$
c) Solution $w_G^{(m,l)}$
$\sum_m \sum_l \beta_{(m,l)}$
T_i
Γ

Figure 2. Schematic of splitting scheme in a 2D setting and in case stress intensity factor calculation for the case with unknown contact surfaces

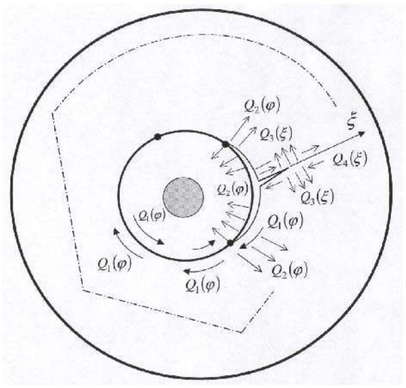

Figure 3. Loads on local problems are shear tractions *$Q_1(\varphi)$* on the entire circular boundary, normal tractions *$Q_2(\varphi)$* on the part of the circular boundary not in contact, shear tractions *$Q_3(\varphi)$* and normal tractions *$Q_4(\varphi)$* on the crack face, respectively. All loading cases are treated separately.

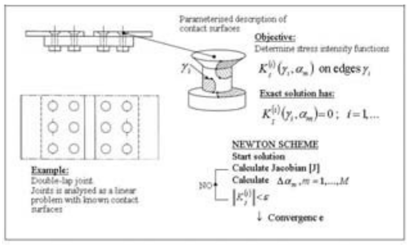
Parameterised description of contact surfaces
Objective:
Determine stress intensity functions
$K_I^{(i)}(\gamma_i, \alpha_m)$ on edges γ_i
γ_i
Exact solution has:
$K_I^{(i)}(\gamma_i, \alpha_m) = 0$; $i = 1, \ldots$
Example:
Double-lap joint.
Joints is analysed as a linear problem with known contact surfaces
NEWTON SCHEME
Start solution
Calculate Jacobian [J]
Calculate $\Delta\alpha_m, m = 1, \ldots, M$
$|K_I^{(i)}| < \varepsilon$
NO
Convergenc e

Figure 4. Steps in solution of 3D contact problems using a fracture mechanics approach to find contact areas and crack data with high accuracy in a few iterations

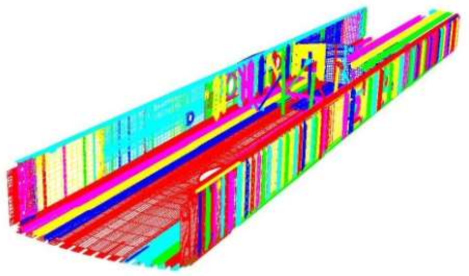

Figure 5. Cutaway view of C-130 CWB Finite Element Mesh

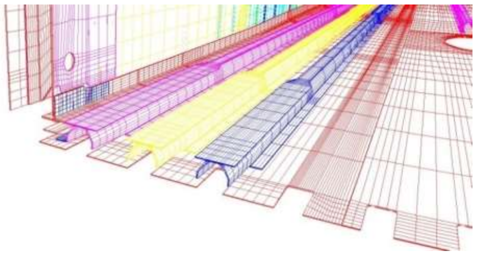

Figure 6. Close-up view of stringer-skin interface

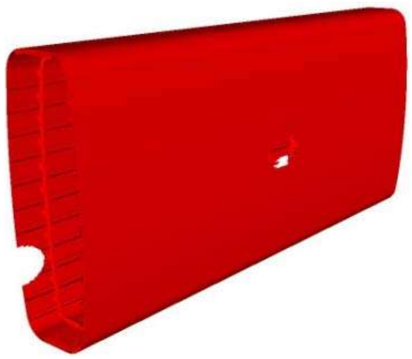

Figure 7. Generic 1/2 mesh of C-130 CWB consisting of 7M hexahedral elements

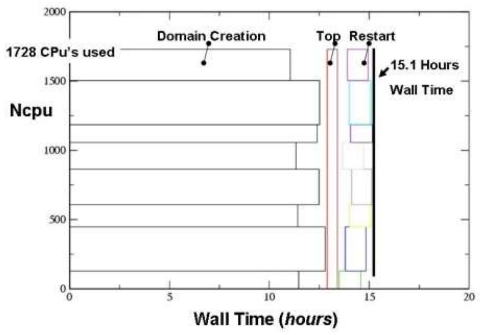

Figure 8. Benchmark results from analysis of generic wing consisting of 13.3M finite elements. A semi-automatic approach was used to solve the problem.

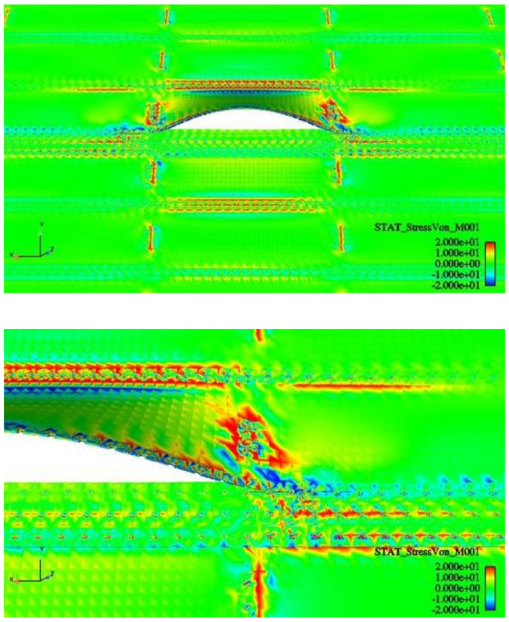

Figure9. Relative error in Von Mises stress solutions for a curved, stiffened panel using *p* = 2 and *p* = 4