Sparse solver – 64 bit and out-of-core addition

Prepared By:
Richard Link
Brian Yuen

Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8

PWGSC Contract Number: W7707-145679

Contract Scientific Authority:
Malcolm Smith, Defence Scientist, 902-426-3100 x383

Contract Report
DRDC-RDDC-2014-C301
May 2014

Lloyd's Register Marine

Working together for a safer world

Sparse Solver – 64 Bit And Out-Of-Core Addition

Martec Technical Report # TR-14-55
Martec Reference Number: 14.28008.1104

May 2014

Prepared for:

Malcolm Smith
DRDC Atlantic
P.O. Box 1012
Dartmouth, Nova Scotia
Canada
B2Y-3Z7

Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8 Canada

tel. 902.425.5101
fax. 902.421.1923
email. martec-info@lr.org
www.martec.com

REVISION CONTROL

REVISION	REVISION DATE
1.0	April 30, 2014
2.0	May 6, 2014
3.0	May 12, 2014

PROPRIETARY NOTICE

This report was prepared under Contract **W7707-145679 DRDC Atlantic** and contains information proprietary to Martec Limited.

The information contained herein may be used and/or further developed by **DRDC Atlantic** for their purposes only.

Complete use and disclosure limitations are contained in Contract **W7707-145679 DRDC Atlantic.**

SIGNATURE PAGE

MODELLING THE FRACTURE BEHAVIOR OF A 350WT STEEL

Technical Report # TR-14-56 Rev 02
Martec Reference Number: 13.40156
12 May 2014

Prepared by: _____ **Date:** May 12/14
Rick Link
Senior Research Engineer

Prepared by: _____ **Date:** May 12/14
Brian Yuen
Research Engineer

Reviewed by: _____ **Date:** May 12, 2014
Lei Jiang
Senior Research Engineer

Approved by: _____ **Date:** May 12, 2014
David Whitehouse
Manager, Research and Product Development

EXECUTIVE SUMMARY

The ability to analyze large FE problems is crucial in investigations involving large ship models. This report describes the modifications made to the VAST sparse solver for the analysis of large problems exceeding the 32-bit limit (2 GBytes) of RAM storage. This includes the extension of the sparse solvers to 64-bit technology, and the addition of an out-of-core (OOC) capability to the new supernodal sparse solver. In addition, the VAST future development code, VAST API, is upgraded to 64-bit.

The 64-bit and OOC solvers were tested on large problems in both the linear and nonlinear range. Results using the current version of VAST indicate a dramatic increase in speed over the existing sparse solver. The VAST API code provided even further speed increase due to the minimization of disk I/O in the element stiffness and stress calculation routines.

From these results, it is clear that large strides have been taken to make the VAST FE solver competitive with other commercial products such as NASTRAN and ANSYS. However, other areas of speed optimization were identified, such as parallelism of the element stiffness matrix and stress calculation routines, and the use of quasi-Newton methods for nonlinear problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The ability to analyze large FE problems is crucial in investigations involving large ship models. This report describes the modifications made to the VAST sparse solver for the analysis of large problems exceeding the 32-bit limit (2 GBytes) of RAM storage. In previous work [1], a supernodal solver was implemented in the existing VAST sparse solver C++ code. The new solver utilizes supernodes in conjunction with parallel optimized Basic Linear Algebra Subroutines (BLAS) libraries in order to significantly reduce matrix factorization times. The new solver was compared with the existing sparse solver by running a static analysis of a medium-sized ship model, which had approximately 71,000 nodes (426,000 degrees of freedom resulting in a 1.3 GByte stiffness matrix). The factorization times were 226.40 sec and 10.87 sec for the current and new solvers respectively. This represents a large speedup of 20.83X, a successful proof-of-concept test. This new sparse solver was incorporated into an improved version of VAST, known as VAST API, in which an in-core database was implemented to eliminate all the intermediate I/O operations in finite element calculations. Benchmark results for nonlinear analyses of practical engineering problems have demonstrated that by combining the savings gained by using improved sparse solver and the database, the computational efficiency of VAST was increased by a factor of five. Following the developments described above, additional funding was obtained from Lloyd's Register Marine Product Development (MPD) for adding new solver functionality for transient, eigenvalue, and restarts; this work has been completed. It remained to extend the sparse solver capability to problems greater than 2 GBytes by using both 64-bit technology, and out-of-core solution techniques.

The callup task [2] is divided into several sub-tasks:

Sub-task 1: Conversion of the prototype sparse solver to 64-bit technology.
Sub-task 2: Conversion of the VAST API to 64-bit.
Sub-task 3: Addition of out-of-core (OOC) capability to the prototype sparse solver.

Completion of each task is marked with successful testing using large problems representative of large ship analysis problems.

Section 2 describes the modifications made to the sparse solver to accommodate both 64-bit and OOC technology, along with prototype solver tests. Section 3 describes the VAST API changes, as well as solver tests for nonlinear problems. Finally, Section 4 summarizes results, and makes recommendations for future work.

2.0 SPARSE SOLVER MODIFICATIONS

This section describes the basic modifications made to the current sparse solver code (both column and supernodal versions) in order to accommodate 64-bit addressing and out-of-core (hence denoted as OOC) technology.

2.1 64-BIT MODIFICATIONS

The previous version of the new sparse solver supported 32 bit addressing, which has serious limitations for large problem sizes. For problems exceeding 2 GBytes of RAM, the stiffness matrix must be factorized using OOC techniques, since the largest addressable data space is 2 GBytes. For problems with relatively few forward and back solve evaluations, such as static analysis of a few load cases, this does not result in significant slowdown, because the OOC factorization process does not add significant solver time. However, for cases such as dynamic and eigenvalue analysis, multiple forward/back solve evaluations are required, and solver time increases dramatically.

Two basic changes were made to the 32-bit code to make it 64-bit compatible. First, all 4 byte integers (data type int) associated with the stiffness matrix data sizes were changed to 8 bytes (data type __int64). This allowed memory allocation and management of arrays larger than 2 GBytes. The second modification made was to the data pointers to the storage locations of the supernodes from int to __int64. This allowed the management of the stiffness matrix array for problem sizes exceeding $2^{31} - 1$ (approximately 2.1 billion) 8 byte words (16 GBytes). These two modifications should be sufficient to run extremely large problems, although there may be other bottlenecks as problem sizes increase. Examples include the number of nodes and elements, node renumbering software (currently 32-bit), auxiliary arrays, etc. exceeding $2^{31} - 1$ in either memory size or addressing.

It is noted here that these modifications were made to both the current and new sparse solvers, and these were fully tested for large problem sizes, as described subsequently.

2.2 OUT-OF-CORE MODIFICATIONS

2.2.1 Factorization

The factorization procedure for the new solver is similar to that of the current one, except that operations are performed at the supernode level. Figure 2.1 shows the factorization process. Initially, the stiffness matrix is assembled and stored on disk in nearly equally sized partitions. The partitions are started on distinct supernodal boundaries. The factorization proceeds in three distinct steps; the OOC update, the in-core update, and the supernodal factorization. The OOC update consists of reading the supernodes in partition j (OOC) one at a time, and forward updating the connected supernodes in partition i (in-core). After that, the factorization proceeds for partition i using the existing in-core techniques. Backward updates are performed

in-core in partition i at the supernode level, and finally the supernode is factorized. Once all supernodes are factorized, partition i is written to disk.

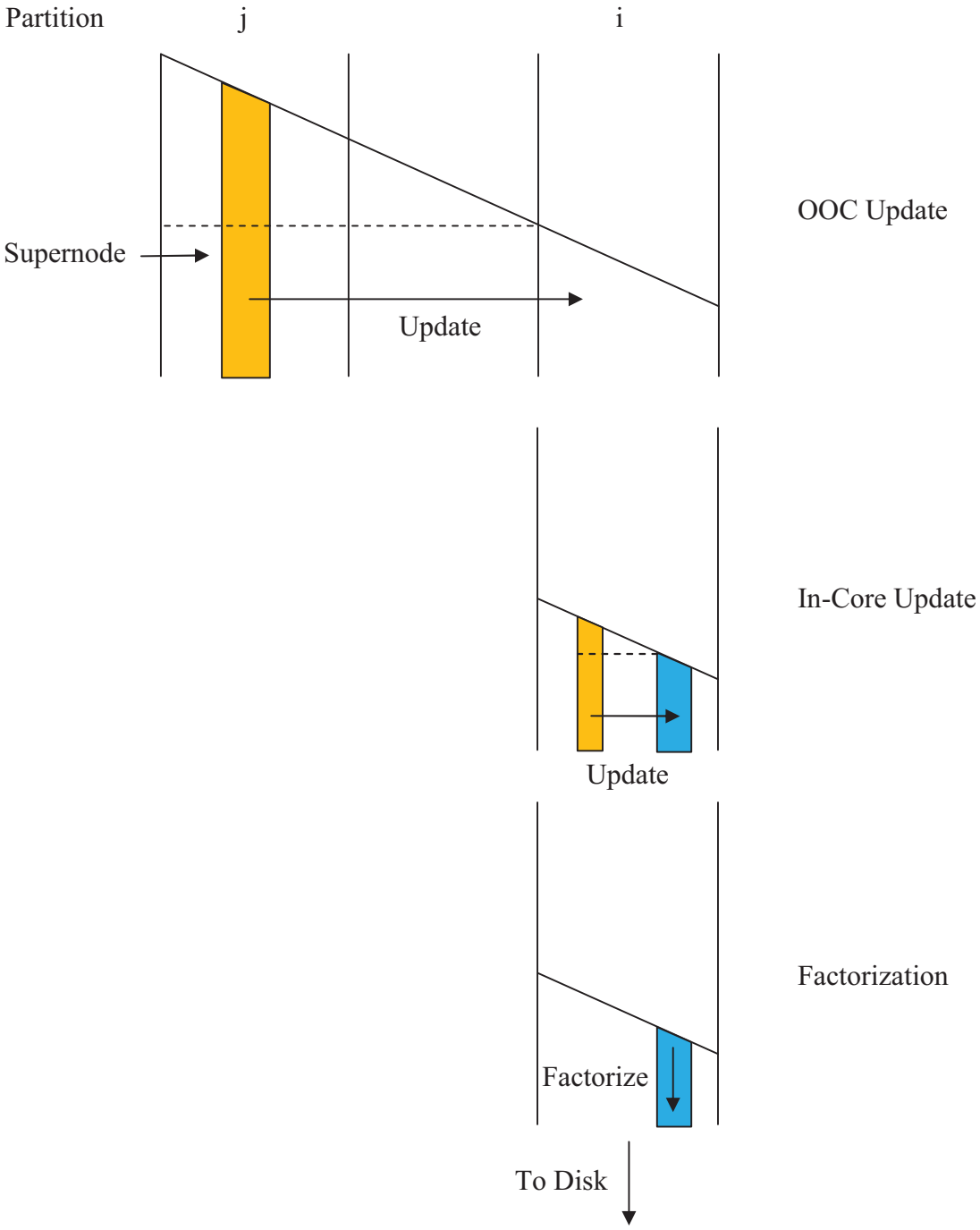

Figure 2.1: OOC Factorization Process

Pseudocode for the OOC update is given below in Figure 2.2.

```
For i = 1, # of partitions
{
        For j = 1, i – 1, j not equal to i
        {
                For k = 1, # of supernodes in j
                {
                        Read k'th factorized supernode
                        Forward update i'th partition
                }
        }
}
```

Figure 2.2: OOC Update Pseudocode

2.2.2 Forward and Back Substitution

For both forward and back substitution, the displacement vector is updated one partition at a time. The partition is read, then the entire displacement vector is updated. Initially, the back substitution routine could not be performed partition-wise, and required information from other partitions. This was changed, with the additional benefit of a substantial speedup of that module.

2.3 SOLVER TESTS

A series of larger analysis cases has been run on three computers with differing CPU's and disk configurations. Table 2.1 describes the problems that were analysed, along with problem size descriptors. The matrix size refers to the memory occupied by the number of matrix fill-ins. The first three problems were chosen to significantly exceed the 32-bit barrier, with the third problem exceeding 2 billion matrix elements. The container ship was included to test the out-of-core eigensolver, and the final 2 problems were to test the nonlinear analysis capabilities.

Table 2.1: Solver Tests

Problem	Description	# Of DOF's ($x10^6$)	Matrix Size (GB)
LM400K	Linear static analysis of plate with transverse loading	2.41	5.79
LM800K	Linear static analysis of plate with transverse loading	4.81	12.73
LM1200K	Linear static analysis of plate with transverse loading	7.21	20.00
Container Ship	Natural frequency analysis of a container ship	0.13	0.39
Cylinder	Nonlinear buckling analysis of a	0.13	0.32

	stiffened cylinder		
Box Girder	Nonlinear buckling analysis of a stiffened box girder. Obtained from the SA.	0.26	0.50

Figures 2.3 and 2.4 show the container ship and box girder problems, respectively. The plate problems are not shown because their geometry is simple; a highly refined simple plate is used.

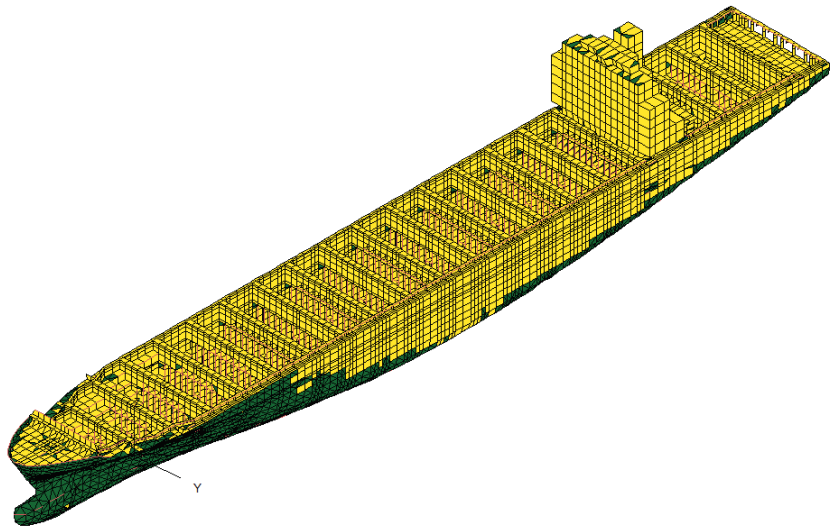

Figure 2.3: Container Ship Problem

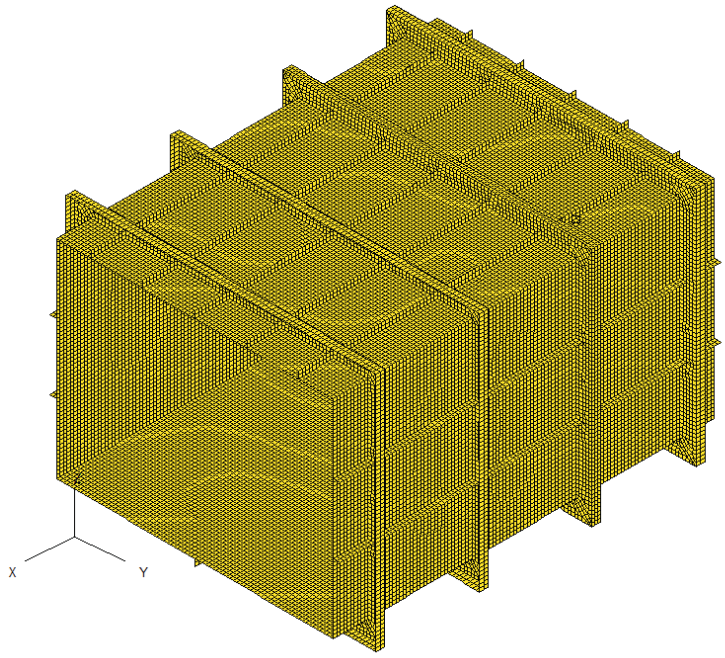

Figure 2.4: Box Girder Problem

Table 2.2 shows the computers used for the tests, including Passmark rating, RAM, and disk configuration. One machine has a solid-state drive (SSD) in order to determine the effect of disk operations on solution time. The Passmark rating is a common CPU performance rating used to compare overall CPU performance; the higher the number, the better the performace. All processors are Intel quad cores.

Table 2.2: Test Machines

Processor	Machine Type	RAM (GB)	Disk Controller	Passmark
Core I7-3720QM@2.60 GHz	Laptop	16.0	AHCI	8322
Core I7-2600K@3.40 GHz	Desktop	16.0	RAID	8604
Core I5-2500K@3.30 GHz	Desktop	32.0	AHCI (SSD)	6507

In the following sections, the current solver is non-supernodal modified to 64-bit. The new solver is supernodal modified for 64-bit and OOC. For both solver types, if the matrix can be accommodated with one partition, the stiffness matrix is not written to disk.

2.3.1 In-Core Tests

Tables 2.3-2.5 show the in-core solution times for the various problems. For the laptop and 2500K SSD, the LM1200K problem was not run, because the matrix size is substantially larger than 16 GBytes, and would cause the computer to run in virtual memory, slowing it down considerably. It is seen that the solution times decrease considerably when the new solver is used. For the static problems, the decomposition (including matrix assembly) time exhibits a speedup of 4.6-10.8X, while the overall solution time is decreased 3.7-8.0X. The container ship overall time speedup is approximately 2X, and the nonlinear problems from about 2.2-2.9X.

It can be seen that VAST performance increases substantially with increasing disk speed. The laptop has a substantially slower disk drive than the desktops, and this is reflected in the solution times. The 2500K SSD results show significantly faster times than those from the 2600K desktop and the laptop, even though the processor is somewhat slower.

Table 2.3: Laptop In-Core Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	663.44	69.34	9.57	750.38	126.59	5.93
LM800K	2536.89	549.79	4.61	2700.95	730.54	3.70
Container Ship	29.44	4.66	6.32	211.29	110.94	1.90
Cylinder	8859.12	2064.77	4.29	13392.48	6056.45	2.21
Box Girder	15936.91	3713.09	4.29	21345.29	9882.35	2.16

Table 2.4: 2600K In-Core Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	681.52	64.62	10.55	798.20	145.97	5.47
LM800K	2432.34	327.66	7.42	2672.36	575.91	4.64
Container Ship	28.28	5.01	5.64	213.56	106.51	2.01
Cylinder	7866.58	1362.29	5.77	11066.57	5046.02	2.19
Box Girder	14310.10	2538.76	5.64	18487.76	7489.51	2.47

Table 2.5: 2500K SSD In-Core Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	687.31	78.23	8.79	739.47	129.56	5.71
LM800K	2113.45	216.06	9.78	2215.66	318.27	6.96
LM1200K	4161.85	384.43	10.83	4316.37	538.20	8.02
Container Ship	29.41	3.95	7.45	224.22	120.85	1.86
Cylinder	8083.01	1257.44	6.43	11422.30	4692.69	2.44
Box Girder	15143.68	2196.90	6.89	19636.32	6729.21	2.92

The container ship, cylinder, and box girder problems are run significantly faster using the new sparse solver, but not proportionate to the speedup seen in the large static problems. For the container ship, a relatively short amount of time is spent in the factorization routine; a large portion of time is spent in inverse vector iteration, which utilizes forward and back substitution modules, and are not sped up to the extent of the factorization module. The cylinder and box girder problems are nonlinear, which means that for each solver, a substantial amount of time is spent in disk I/O-intensive (slow) element stiffness matrix and stress calculations, thus reducing the relative speedup. The VAST API minimizes disk operations, and speedups are reported in Section 3.0.

2.3.2 Out-Of-Core Tests

Tables 2.6-2.8 show the timings for the out-of-core solver. Only the larger problems were run to compare timing performances with the in-core solver. The container ship was also run to demonstrate that the OOC eigensolver is functional. The new OOC solver exhibits similar speedups for the static problems as the in-core solver, with the decomposition (including matrix assembly) time exhibiting a speedup of 4.1-12.4X, and an overall solution time decrease of 3.1-8.4X. However, the container ship overall solution time is slower than the current solver by approximately 10%. While undoubtedly linked to I/O disk operations due to inverse iteration procedures, it is unknown as to the exact reason of the OOC slowdown. However, it is noted that the OOC eigensolver is substantially slower (in some cases, an order

of magnitude) than its in-core counterpart, and it is not recommended for use in eigenproblems.

Table 2.6: Laptop OOC Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	657.70	144.38	4.56	756.37	229.98	3.29
LM800K	2214.38	532.22	4.16	2600.40	837.02	3.11
LM1200K	4499.84	967.50	4.65	5154.62	1575.27	3.27
Container Ship	31.09	6.30	4.93	1333.35	1508.01	0.88

Table 2.7: 2600K OOC Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	634.34	80.01	7.93	764.85	162.38	4.71
LM800K	2136.95	311.72	6.86	2513.24	633.02	3.97
LM1200K	4406.05	672.99	6.55	5102.26	1256.82	4.06
Container Ship	28.69	4.73	6.07	744.82	778.55	0.96

Table 2.8: 2500K SSD OOC Times

Problem	Matrix Decomposition Time			Total Time		
	Current	New	Factor	Current	New	Factor
LM400K	679.30	64.51	10.53	736.58	122.24	6.03
LM800K	2130.26	174.24	12.23	2247.82	294.16	7.64
LM1200K	4167.59	337.59	12.35	4344.45	518.30	8.38
Container Ship	30.53	4.43	6.89	776.29	840.22	0.92

It is also observed here that in some cases, the new OOC solver performs matrix decompositions faster than the in-core solver. The reasons for this are unknown, but the results show that the OOC factorization process incurs little or no CPU time penalty for use.

3.0 VAST API

The VAST API represents the future direction of the VAST FE code, with new developments being implemented and tested on that platform. In previous work [3], the element stiffness matrix and stress calculations were moved in-core through a series of interface routines that wrote data to RAM rather than disk, resulting in significantly faster solution times. A 64-bit version was built with the new 64-bit sparse solver and tested on the two nonlinear problems for the purposes of speed comparison. Nonlinear problems were selected as test cases because they are disk I/O intensive, and the effects of minimizing disk interaction can be more readily seen.

3.1 IN-CORE TESTS

The cylinder and box girder tests from Section 2 are used for comparison purposes. The test cases were run on the 2500K SSD machine. Table 3.1 shows the results comparison; the speedup factor refers to the ratio of the current solver solution time to that of the VAST API.

Table 3.1: VAST API Times – 2500K SSD

Problem	Matrix Decomposition Time				Total Time			
	Current	New	VAST API	Factor	Current	New	VAST API	Factor
Cylinder	8083.01	1257.44	1153.45	7.01	11422.30	4692.69	2735.43	4.18
Box Girder	15143.68	2196.90	1361.08	11.13	19636.32	6729.21	3214.37	6.11

It can be seen that the minimization of disk I/O has a significant effect on reducing the total solution time. The element stiffness matrix and stress calculations are all performed in-core, and result in approximately doubling the speedup of the total solution time for the nonlinear problems considered here.

4.0 CONCLUSIONS AND RECOMMENDATIONS

In this work, the 32-bit VAST prototype sparse solver was extended to 64-bit in order to access machine RAM exceeding the 32-bit barrier ($2^{31} - 1$ bytes), and to solve problems involving stiffness matrices with greater than $2^{31} - 1$ entries. This was achieved by changing all pertinent memory pointers to 64-bit integers.

For extremely large problems, or for machines with limited RAM size, the prototype solver was extended to out-of-core using techniques similar to those adopted by the current solver. Out-of-core partition updates are performed by using a supernode forward updating scheme, followed by a regular in-partition update and factorization. The forward and back-substitution modules were performed at a partition level; the existing back-substitution technique had to be modified in order to accommodate this.

The prototype solver was tested using both in-core and OOC modes. Large static, eigenvalue, and nonlinear problems were used to ensure functionality over a large problem range. The solver performed very well, resulting in large speedups by up to an order of magnitude. In some cases, the OOC solver resulted in faster solution times than the in-core solver.

The VAST API development platform was also extended to 64-bit in order to add the prototype sparse solver capability for further FE solver development. It was expected that minimization of disk I/O operations would result in further solution time reductions; tests using large nonlinear problems bore this out, even reducing the prototype solver solution times by a factor of two.

It is clear that large strides have been taken to make the VAST FE solver competitive with other commercial products such as NASTRAN and ANSYS. However, there are more areas that can be optimized for speed including:

1) Parallel element matrix, stress, and stiffness matrix assembly – using OpenMP, these embarrassingly parallelizable modules can fully utilize machine CPU throughput.
2) Quasi-Newton methods – these nonlinear solution methods use approximations of the stiffness matrix inverse between matrix updates. This means that the stiffness matrix does not have to be recalculated and inverted for every equilibrium iteration. Preliminary investigations using LS-DYNA indicate a potential speedup of 2-4X.
3) Reduced integration elements – one-point shell elements are commonly used in commercial FE programs. Zero-energy deformation modes are treated using stiffness formulations.

5.0 REFERENCES

(1) Link, R.A. (2013). VAST Parallel Sparse Solver. Martec Technical Report TR-13-39.

(2) Link, R.A. (2014). Sparse Solver – 64-Bit and OOC Addition. Martec Call-Up CU-14-39.

(3) Jiang, L., and Macadam, T. (2013). Improving Computational Efficiency of VAST. Martec Technical Report TR-13-42.