Direct Or Iterative? A Dilemma For The User.

Petra Poschmann, Louis Komzsik, Stefan Mayer
The MacNeal-Schwendler Corporation
Pavel Babikov, Maria Babikova
Consultants

Abstract

The subject of this paper is the comparison of Direct and Iterative solvers in the Solution of large Finite Element Problems with MSC/NASTRAN. The results of such a comparison vary based on problem type and with new developments in both the direct and iterative solution techniques. This paper gives the current state of this comparison with MSC/NASTRAN V70.5.

Experiences gained in the past show that analysts are hesitant to use the iterative solver because the direct sparse solver has been in MSC/NASTRAN from the beginning, is very robust and has gone through several optimization phases. MSC's direct sparse solver has very low memory requirements, a very efficient out-of-core logic and very good re-ordering techniques.

The iterative solver, on the other hand, was first delivered only early in this decade. It has since been improved significantly by adding more and better preconditioning techniques, the latest one being BIC (Block Incomplete Cholesky) preconditioning which was introduced in V69 . Many clients started to use it for 3-D models and have reported positive results. More recently, in MSC/NASTRAN V70.5 the memory management and spill logic for the BIC preconditioner of the iterative solver have been improved significantly, particularly for jobs where the solution cannot be done completely in-core.

We will briefly discuss the methods used for the direct and iterative solvers in MSC/NASTRAN, the effects of new re-ordering schemes in the direct solver and the effects of the BIC preconditioning technique in the iterative solver. Moreover, we will present performance results for large models involving different types of elements to give guidelines on where the advantages of each solver lie and when is it more beneficial to use one solver over the other. For example, for models built from 2D elements it is always recommended to use the direct solver while for models of 3D elements the iterative solver should be preferred. Other issues like memory and disk space requirements and multiple load cases will also be discussed. The focus of the paper is on linear static analysis.

1. Introduction

The goal of this paper is to aid the user in a very frequent dilemma: to decide whether to use the direct or the iterative solver of MSC/NASTRAN.
The outcome of such decision has significant impact on the turn-around time of the analysis the user is executing. It also influences the amount of memory and space required for a particular problem. Finally, the appropriate selection may make it possible to run a job which may not run to completion with the given computer resources using a different selection.

Unfortunately, the decision is not easy. The decision depends heavily on analysis type and model characteristics. The analysis type, namely statics, or modes, linear vs. nonlinear is important because the iterative solution is not supported in all solutions yet. In fact, for some solutions the iterative solver is not appropriate. For normal modes analysis using the Lanczos method, for example, one decomposition is performed for every shift followed by many solutions with different right hand sides (RHSs). If the iterative solver was used for normal modes, we would need to solve the whole system not only once for each shift, but once for every RHS which is much less efficient than just doing a forward-backward substitution for each RHS. Moreover, the iterative solver would not give us the Sturm count, which is needed for the real Lanczos algorithm in MSC/NASTRAN.

The model characteristics, especially the type of the elements used to build the model as well as the modelled topology (2D vs. 3D) are also important decision making factors. This is due to the fact that the iterative solver may not converge in some cases, while the direct solver always provides a result.

2. Current status of direct solution technology in MSC/NASTRAN

The direct solution technique permeates all analysis types of MSC/NASTRAN.
Originally it has been the only solution technique for large systems of linear equations in MSC/NASTRAN and it has undergone significant changes during the years. The methodology was always based on the Gaussian elimination method, however, with significant implementation specialties. These specialties were aimed in part to take advantage of the structure of the matrices arising in finite element applications. Another reason to specialize our direct solver was to run efficiently on upcoming new computer architectures.

The current default method of direct solution is the well-known multi-frontal Gaussian elimination technique [Ref. 3]. For the considerable amount of numerical work in the decomposition process, MSC's proprietary implementation uses highly tuned so-called indexed kernels. In order to make the data access efficient, we are also using a sparse matrix storage technology, developed in-house to properly interact with our database management system.

In agreement with the current state of the art computers, the direct solver of MSC/NASTRAN may be executed in parallel. Both shared and distributed memory parallel implementations of the direct solver exist on certain computers, supporting one of the two paradigms. This enables the faster turn-around of analysis jobs, an eternal quest of the analyst.

Most of the recent development in the direct solver arena was focused on the reordering schemes. The performance of the direct solver very heavily depends on the order of the elimination. The newest reordering schemes introduced, the EXTREME and METIS methods, have significantly increased the direct solver performance. Section 4 will show results obtained from Version 70.5 using these new methods.

For more details on the usage and mathematical aspects of the direct method please see Ref.1.

3. Current status of iterative solution technology in MSC/NASTRAN

Iterative solvers use a completely different approach to solving large systems of linear equations than direct solvers. The iterative solver used inside MSC/NASTRAN is based on the preconditioned conjugate gradient method [Ref. 4]. It starts out with an initial guess which may be, for example, the zero vector and goes through an iterative process to update the solution vector in every iteration using the system matrix and a preconditioner matrix to converge to the solution. A convergence criterion is used to determine whether the accuracy of the solution is acceptable or more iteration is needed to improve the accuracy. If a pre-determined maximum number of iterations is reached without obtaining the desired accuracy of the solution, the solver exits with the message that it could not converge to the correct solution.

Convergence of the solver and the convergence rate are highly dependent on the preconditioner used. For a simple preconditioner (for example, Jacobi), less work is done in every iteration and less memory is required, however, the solver may take many iterations to convergence or will not converge at all. More sophisticated preconditioners require more memory and more work in every iterations, but they may converge quickly to the correct solution.

The iterative solver was first introduced in MSC/NASTRAN V67.5 with very limited preconditioning capabilities. Since then it has been improved and tuned continuously with regard to applicability, convergence rate, performance, memory usage and spill logic. The most significant improvement was introduced in MSC/NASTRAN V69 with the BIC preconditioner (Block Incomplete Cholesky). In our latest release, i.e. V70.5, we delivered BIC preconditioning with a very efficient spill logic and improved memory management which allows the execution of even very large jobs in a reasonable amount of time. For more details please see Ref. 2.

Currently the iterative solver is available for linear and non-linear static analysis (SOL 101, 106) and for direct and modal frequency response (108, 111). For linear statics, the iterative solver with Jacobi preconditioning has been parallelized for distributed memory machines such as the IBM SP2. For well-conditioned models where the solver converges with the simple Jacobi preconditioner, the SOLVIT module scales quite well. Parallel distributed SOLVIT as well as parallel direct frequency response for the IBM SP2 system was delivered in MSC/NASTRAN V69.2.

4. Comparison of performance of the direct and iterative solution techniques

The choice of the direct versus the iterative solution method depends on the type of elements in the model, the shape of the model, the computer resources and the type of analysis. The user should ask whether the main goal is to run a job as fast as possible or to be able to run a job at all given particular computer resources. The following paragraphs compare the 2 methods regarding computer resources, performance and applicability.

It is very important to realize that the actual performance of the iterative solution procedure as well as the required computer resources strongly depend on predefined parameters of the preconditioner. The quality of the BIC preconditioner for a particular model is determined by a single integer parameter called the generation number or padding level; in the MSC/NASTRAN user interface, this parameter is referred to by IPAD on the ITER bulk data entry. The accuracy of the incomplete decomposition increases with higher values of IPAD resulting in fewer iterations required for convergence at the expense of CPU time increase per iteration. The optimal IPAD value is problem dependent. Well-conditioned linear systems may run well with IPAD = 2 or 3 while ill-conditioned problems may require an IPAD value of 3, 4 or higher in order to converge in a reasonable time. The defaults in MSC/NASTRAN have been chosen according to the type of elements present in the model since they influence the conditioning of the linear system. Table 4.6 illustrates the dependencies of CPU time and memory on IPAD for a very well conditioned benchmark, a well-conditioned small crankshaft model and an ill-conditioned carbody model.

The disk space requirements of the iterative solver are much less than the disk space requirements of the direct solver due to the fact that there is no need to create and store big factor matrices. For complete linear statics runs we have seen disk space savings of up to 80 %. This fact may enable the user to run the job with the iterative solver when the direct solver's factor matrix could exceed the disk capacity of the machine.

With regard to memory the requirements of the iterative solver are usually higher than those of the direct solver because it needs to fit both, the matrix and the preconditioner, in memory for optimal performance. This OPTIMUM MEMORY is listed under the iterative solver diagnostics in the .f04 output file of an MSC/NASTRAN run. The sparse decomposition only needs to fit a portion of the system matrix in memory for good performance, which is reflected by the SUGGESTED MEMORY output in the .f04 file. If the iterative solver runs completely incore, its performance is bound by the CPU speed and sometimes by the memory access speed (for vector machines). If parts of the matrix and/or preconditioner need to be stored out-of-core, the solver needs to read those parts from disk in every iteration so that the performance is bound by both, the CPU speed and the I/O performance of the computer. The performance of the direct solver is bound by the I/O performance of the computer because the decomposition writes the big factor to disk which then has to be read twice by the FBS module once in a forward and once in a backward pass. Usually, the number of terms in the factor is greater than the number of terms in the matrix and the preconditioner by one order of magnitude.

Another criterion in choosing between direct and iterative methods is the number of load vectors. If multiple load vectors are present, the direct methods needs to do one decomposition followed by multiple (block) FBSs. Using the iterative method, the iteration loop has to be repeated for each right hand side unless a special algorithm is implemented to exploit multiple RHSs for accelerated convergence using Krylov subspace extension (Ref. 2). This was done for BIC preconditioning in MSC/NASTRAN which makes the iterative solver attractive for up to some limiting number of load vectors (see example 2, table 4.2). However, for bigger numbers of right hand sides, two opposite effects start fighting each other: the reduction of the number of iterations due to Krylov subspace extension and the increase of the number of iterations due to loss of stability introduced by round-off errors. The actual limiting number of load vectors is typically about 5, but for some specific cases the iteration stability can be lost already for 3 right hand sides, or vice versa it can be kept up to 10 or even more right hand sides. It is recommended to use the direct solver for a higher number of load cases, in addition to the considerations explained earlier.

The major disadvantage of iterative methods is that they may not converge for shell models. This fact makes the direct solution method indispensable for finite element analysis. In almost all cases the sparse direct solution is faster on shell models which result in very sparse and more ill-conditioned matrices.

The tables below demonstrate the considerations above for several different examples from industry. Unless noted otherwise, all jobs were run with MSC/NASTRAN V70.5.

Example 1: from automotive industry (courtesy of Audi, Germany)

Table 4.1:performance studies, run on SGI Challenge, MSC/NASTRAN V70

	Crankshaft		**Transmission Housing**	
	Iter	Direct	Iter	Direct
TET10 Elements:	~510000		~440000	
DOF:	~2.1 mio		2.2 mio	
CPU Time:	13:27 h	10:23h	20:19 h	8:10 h
Time in SEQP:	7:05 h		2:26 h	
SOLVIT / DCMP	5:18 h	17:59 h	16:48 h	6:47 h
# of Iterations	354	N/A	1380	N/A
Disk:	12.3 GB	31.6 GB	11.1 GB	20.0 GB
Mem. Given	2.8 GB	2.5 GB	2.3 GB	2.5 GB

Example 2: Shocktower (3,775 QUAD4, 126 TRIA3, 23,055 DOFs), run on IBM RS/6000 (390)

Table 4.2: performance studies, multiple loads, iterative solver

	CPU in SOLVIT	# Iterations
Run with Load 1	84.64 s	411
Run with Load 2	86.66 s	427
Run with Load 3	87.98 s	433
Run with all 3 Loads	98.10 s	208

Example 3: From automotive industry (86,500 HEXA, 23,350 PENTA, 850 TETRA, 388,488 DOFs), run on Cray C90. In the following table MMD and Extreme are two different reordering methods in the sparse direct solver.

Table 4.3: Performance studies, disk, memory, re-ordering

Method	Disk Space	Memory	Total CPU
Iterative (In-Core)	1.7 GB	780 MB	3,725 s
Direct (MMD)	12.7 GB	197 MB	6,456 s
Direct (Extreme)	5.4 GB	197 MB	3,126 s

Example 4: One half of a pressure vessel cap modeled for topology optimization (MSC/CONSTRUCT) (18,821 HEXA8, 277 PENTA6, 64 TETRA4, 69,051 DOFs), run on IBM RS/6000 (390)

Table 4.4: Performance studies, disk, CPU time

Method	Disk Space	Memory	Total CPU
Iterative	184 GB	78 MB	261 s
Direct	725 GB	78 MB	1,237 s

Example 5: Half tube model (27,081 TETRA, 134,333 DOFs), run on IBM RS/6000 (390)

Table 4.5: performance improvements - MSC/NASTRAN V70 / V70.5

Method	Disk Space (MB)		Memory	CPU (s)	
	V70	V70.5		V70	V70.5
Iter	585.6	585.6	144 MB	865.6	775.1
Iter	645.8	585.8	96 MB	2.939.5	777.9
Iter	645.8	585.8	60 MB	2.035.6	917.1
Direct	1.066.2	902.8	140 MB	1.219.3	892.3
Direct	1.067.5	904.0	60 MB	1.291.2	912.4

Table 4.6: performance studies, memory and CPU time vs IPAD

IPAD	Bcell 18 (3-D)			Small Crankshaft (mixed)			Carbody (2-D)		
	#iter	Opt. Mem	CPU's	#iter	Opt. Mem	CPU's	#iter	Opt. Mem	CPU's
1	206	9.4 MW	87.6	815	4.9 MW	230	4455	5 MW	1,225
2	(def) 79	13.6 MW	66.4	317	6.7 MW	102	3406	5.8 MW	1,095
3	52	19.8 MW	94.8	(def) 209	8.8 MW	94	(def) 2267	6.7 MW	842
4	*41	27.4 MW	170.2	160	11.2 MW	101	1708	7.8 MW	747
5	*34	36.1 MW	276.1	137	13.6 MW	117	1516	9.0 MW	795
6	*30	45.6 MW	446.7	113	16.0 MW	180	460	10.3 MW	269

(def): MSC/NASTRAN default

* : spill logic was activated

5. Recommendations on direct vs. iterative solution selection.

This section summarizes the pros and cons of the iterative and direct methods respectively, giving recommendations to the user of which solver to use for what kind of models and/or resources.

The optimal situation for iterative solutions is a linear statics analysis with quadratic solid elements (HEXA20). Please note that quadratic solid elements create denser matrices than linear solid elements. Moreover, we noticed that topologicaly cube like geometries favor the iterative solution most since the resulting matrices have a higher connectivity and are thus denser than for topologically long structures. A good example for this behavior is shown above in table 4.1, which compares the performance of the iterative and direct methods for a crankshaft (compact model) and transmission housing (highly irregular). For the crankshaft, the iterative solver is much faster than the direct solver; however, for the transmission housing, the direct solver is superior. This demonstrates quite well how the topology of the model influences the performance of the two solvers since in this case, both models consist of 3-D TET10 elements and have about the same number of degrees of freedom.

Table 4.6 illustrates a strong sensitivity of the iterative solver with respect to the preconditioner parameter IPAD. It can be seen that an optimal IPAD value depends on the particular problem's features. Typically, if the amount of memory is large enough to store both the matrix and preconditioner completely incore, then for well-conditioned problems small IPAD values (2 to 4) appear to be optimal and for ill-conditioned problems, large IPAD values (4 to 6 and more) are preferable. Thus, any apriori knowledge about the problem conditioning would help to tune the performance of the iterative solver.

Another criterion discussed is the number of load vectors. Table 4.2 shows the performance of the iterative solver. If the model consists of 3-D elements and has 1-5 load vectors, it is recommended to use the iterative solver. For more than 5 load vectors, the performance of the direct solver will be superior even for 3-D models. The only reason to use the iterative solver for more than 5 load vectors is to save disk space.

An interesting option is available for direct frequency response analyses: the user can specify an option (PRECOND=USER) to do a direct solution for the first frequency and then use the factor matrix as the preconditioner in the iterative solution for all subsequent frequencies. However, one has to consider that the quality of the factor matrix as a preconditioner decreases the further the frequencies are from the first frequency and seriously diminishes when stepping beyond a resonance frequency.

From table 4.5 it becomes clear that the iterative solver has a very competitive out-of-core logic starting with MSC/NASTRAN V70.5, which prevented some users to use the iterative method in versions prior to MSC/NASTRAN V70.5 because the out-of-core performance was not acceptable. Fortunately, this restriction has been removed.

If the iterative solver is going to be used then a preliminary investigation of a small version of the model becomes particularly important as it allows to choose an optimal preconditioning set-up without significant computational expenses.

6. Conclusions

The final recommendation for choosing the best solver for the user's model is to do some studies when starting to work with a new model type. Usually the user does not run an analysis for a particular model only once, but runs the analysis many times possibly making some adjustments to the model or giving it different boundary conditions etc. It would be useful to run the direct and the iterative solver on that model (or possibly on a small version of the model since the characteristics usually do not change with increasing the size of the model), study the performance, memory and disk space requirements and decide which solver is best for that type of model for future analysis.

References

1. Komzsik, L., MSC ‘s Numerical User's guide V70.5, The MacNeal-Schwendler Corporation, 1998

2. Babikov, P., Babikova, M., “An Improved Version of Iterative Solvers for Positive-Definite Symmetric Real and Non-Hermitian Symmetric Complex Problems”, INTECO srl, Report JA-A801, 1995.

3. Duff, I., Reid, J.K., The Mult-frontal Solution of Indefinite Sparse Symmetric Linear Equations, ACM Transactions on Mathematical Software, Vol. 9, No. 3, September 1983, pp. 302-325.

4. Golub, G., van Loan, C.F., Matrix Computations, 2nd Edition, The Johns Hopkins University Press, Baltimore and London, 1989