Advancements in Eigenvalue Technology

Daniel Bielich[1], Roger Grimes[1], Francois-Henry Rouet[1]

[1]Ansys, Inc.

1 Introduction

The standard eigenvalue problem for Implicit Mechanics in LS-DYNA is $\mathrm{K}\Phi = \mathrm{M}\Phi\Lambda$, where K and M, the stiffness and mass matrices, are real and symmetric positive semi-definite in most applications. LS-DYNA offers three main algorithms for this problem, chosen with EIGMTH (field 7 of line 1 of **`*CONTROL_IMPLICIT_EIGENVALUE`**). The three algorithms currently available are

- Lanczos (EIGMTH=2);
- LOGPCG (EIGMTH=102);
- Fast Lanczos (EIGMTH=103).

This paper will give an overview of these methods, guidelines on when each should be used, performance comparisons, and recent enhancements.

For non-symmetric problems, LS-DYNA also has an SMP solver, which relies on ARPACK [1,2]. An MPP solver relying on a different algorithm is being actively developed and is briefly mentioned in the last section.

LS-DYNA also includes solvers for non-standard eigenvalue problems:

- MCMS (EIGMTH=101), a substructuring solver like AMLS [3], used for Frequency Response Analysis calculations for NVH applications [4].
- Sectoral Symmetry (EIGMTH=111), a solver for models with high degrees of symmetry [5].

We do not discuss these solvers here and refer the reader to the references listed above. Our focus here is on the three solvers for the standard, symmetric eigenvalue problem.

1.1 Lanczos

Lanczos was the only eigensolver in LS-DYNA for many years. It is based on the Block Shift and Invert Lanczos algorithm [6]. It offers the most robust and accurate computation of eigenmodes for eigenvalue problems arising in Implicit Mechanics. Lanczos attempts to compute the eigenmodes from smallest to largest by selecting a shift in the spectrum and running the Block Lanczos recurrence for a number of iterations. At the end of the Lanczos run, eigenmodes that are accurate enough are accepted. Those modes and the unconverged modes from the run are used to select the next shift. During the Lanczos recurrence much care is taken to maintain the orthogonality of the computed basis from the recurrence. Sturm Sequence checks are used to assure that all eigenmodes have been computed. The cost of the direct factorization, the solves with that factorization at each iteration, and the cost of maintaining orthogonality make Lanczos expensive. But it is those operations that make it robust and accurate.

1.2 LOBPCG

Algorithms relying on the Lanczos or Arnoldi iteration require "exact" solves with $K - \sigma M$ (for a given shift σ in the region of interest) to maintain the recurrence relation at the heart of the algorithm. On the other hand, *preconditioned* eigensolvers are algorithms that do not require an exact solve; instead, they only rely on an approximate solution, or preconditioner. LOBPCG (Locally Optimal Block Preconditioned Conjugate Gradient) [7] is such a preconditioned eigensolver for Hermitian problems. It is based on a Rayleigh quotient minimization technique.

LOBPCG was released in LS-DYNA R11 for SMP and LS-DYNA R13 for MPP [8]. One of the most important implementation aspects is the choice of the preconditioner. Our choice is the Block Low-Rank (BLR) mode of MUMPS. MUMPS (MUltifrontal Massively Parallel Solver) [9] is a sparse direct solver with a preconditioning mode based on low-rank approximations [10]. Dense matrices that appear throughout the factorization process are compressed (i.e., represented as a product of smaller matrices) following a dropping criterion chosen by the user. When the dropping criterion is close to machine precision, the factorization is almost exact; when one increases the dropping criterion, one gets an approximate factorization that uses less memory and is faster to compute and use.

The choice of this preconditioner (an accelerated direct solver) is motivated by the fact that our mechanical models are particularly challenging for iterative solvers and preconditioners. Preconditioners like diagonal scaling or simple incomplete factorizations almost never lead to convergence for our models. Multigrid solvers can be more successful, but they are notoriously hard to parametrize and require injecting significant amounts of information about the problem. MUMPS-BLR is fully algebraic (requiring only the matrices and vectors) and is easy to use. The BLR mode comes with strong approximation guarantees.

1.3 Fast Lanczos

Fast Lanczos is an adaptation of the Lanczos algorithm to meet the requirements of computing thousands of eigenmodes for the Noise-Vibration-Harshness application at a lower cost. To reduce cost, the eigenmodes are computed to a lower accuracy, 2 or 3 digits instead of 8 to 10 for the Lanczos solver. Fast Lanczos became available in LS-DYNA R15. The Fast Lanczos implementation runs totally in memory and is designed from the ground up for MPP performance. It utilizes a block size much larger than that of (standard) Lanczos, and it usually computes hundreds of eigenmodes per shift. Fast Lanczos is a competitor to the AMLS algorithm in applications where large numbers of modes are needed. More implementation details can be found in previous proceedings [8].

1.4 Which method to use

A general rule of thumb is that LOBPCG is often the fastest method when the number of modes to compute (NEIG) is very small. This can be used as debugging tool, for example to look for rigid body modes (by setting NEIG to 6 or a little over 6). Lanczos is more efficient when computing more than a few modes; it is also the most robust of the three solvers. and it usually converges even for the most numerically difficult problems. Lanczos will probably stay the main workhorse for most analysts. Fast Lanczos is designed for computing thousands or tens of thousands of eigenmodes. The modes will not be as accurate as what Lanczos would compute, but they are usually accurate enough for the type of workflows that require that many modes, and usually more accurate than the approximate modes calculated by AMLS. Section 3 offers a comparison study of the 3 solvers.

2 Enhancements to existing solvers

As problem size grows and hardware architectures become more complex, Ansys is continually improving its software, including eigensolvers. Both Lanczos and Fast Lanczos utilize the sparse matrix factorization and solution package MF2 used in both LS-DYNA and MAPDL. Improvements to MF2 will automatically improve Lanczos and Fast Lanczos, especially as we march towards implementations on GPUs. Here we describe a few improvements that are specific to the eigensolvers themselves.

2.1 Lanczos

The computation of thousands of eigenmodes for automotive models made us aware that the eigenvalues tend to become denser along the real line. Starting in R16 the shifting strategy was improved to have better performance when computing thousands of modes by avoiding shifts that are too aggressive.

We illustrate this using a model of a Honda Accord car with 35 million degrees of freedom. The model originally came from NHTSA and was refined and set up for modal analysis by Arup. We compute 3000 modes; this is not necessarily a typical use case for Lanczos (to compute this many modes, we recommend using Fast Lanczos, unless high accuracy is needed), but it illustrates the improvements to the shifting strategy. Total time using R15 is 10,332 seconds, while total time using R16 is 6,842 seconds, a 34% improvement; R15 used 38 shifts (38 values of σ, and therefore 38 factorizations), while R16 used 26 shifts. Fig. 1 shows the (cumulated) number of modes captured by the algorithm throughout the shifting sequence. It shows that the shifts that R16 used were more effective at finding eigenvalues, while R15 had to backtrack (decrease the shift) because the shifts were too aggressive to capture any mode; this is shown by the plateaus in the convergence curve for R15.

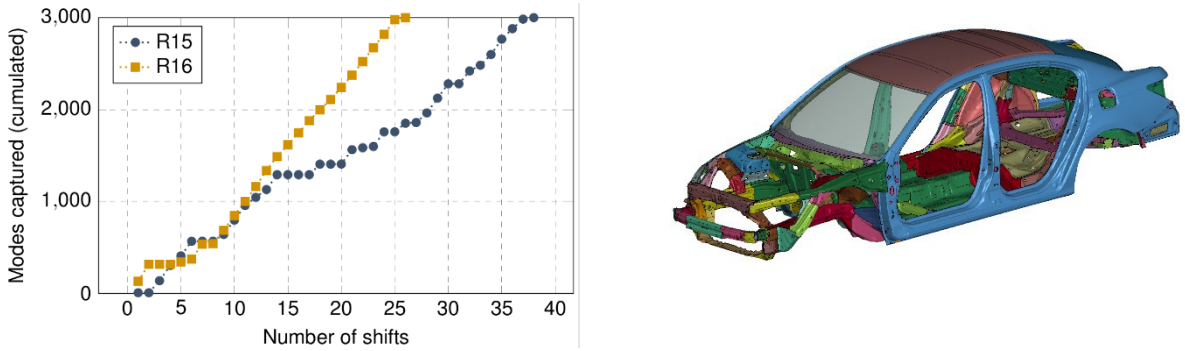

Fig.1: Convergence curve for Lanczos, Honda Accord model.

2.2 Fast Lanczos

We have also improved the shift logic and memory management for Fast Lanczos. For small problems or poorly chosen initial shifts there may be no eigenvalues to one side or the other of the first shift. This issue was corrected in R16. Improvements were made to enable Fast Lanczos to stop earlier when all requested modes have been computed. For example, for the same configuration as above (Honda Accord model, 3000 modes), LS-DYNA R15 with Fast Lanczos takes 4,812 seconds; R16 takes 4,218 seconds, a 12% improvement. As usual, the impact of these modifications is problem dependent, and it also depends on the number of modes requested by the user.

2.3 LOBPCG

LOBPCG is faster thanks to improvements in the orthogonalization process, data communication, and an updated version of the MUMPS solver (used as the preconditioner, as explained above). Table 1. Illustrates this for the Honda Accord model mentioned above. We compute 200 modes with LOBPCG, using different number of MPI ranks. For this configuration, LS-DYNA R16 is about 25% faster than R15.

LS-DYNA	#MPI ranks		
	32	64	128
R15	5500.2	2134.4	1275.1
R16	3656.9	1782.2	1018.5

Table 1: LOBPCG performance improvements

Another improvement worth mentioning is that, as of R16, the MPP version of LOBPCG can now perform interval computations (LFTEND, RHTEND).

3 Comparison study

3.1 LOBPCG vs Lanczos

A key factor in our implementation of LOBPCG is the Block Low-Rank preconditioner. Such preconditioners tend to be more effective when the geometry is strongly 3D, bulky. However, they are not as effective for “2.5D” geometries, i.e., hollow objects like car bodies. For example, for the Honda Accord model mentioned above, Lanczos is always faster than LOBPCG, no matter the number of modes to be computed. Fig. 2. shows results for a bulky geometry, a model of mining equipment with 10 million degrees of freedom that came from Predictive Engineering. Here LOBPCG has an advantage for low numbers of modes, until a crossover point is reached and Lanczos is faster. This is because Lanczos can usually capture more eigenmodes than LOBPCG with the same shift; LOBPCG convergence becomes too slow when the number of modes is large, and run time is spent either waiting for convergence, or shifting and factoring a new shifted matrix. However, where the crossover point is, if it exists, is very application dependent, and we encourage users to try LOBPCG for themselves.

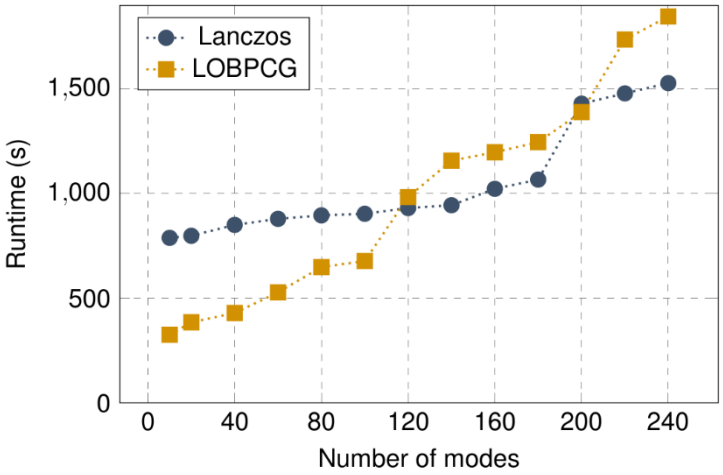

Fig.2: Performance of LOBPCG and Lanczos for a bulky 3D model.

3.2 Lanczos vs Fast Lanczos

In this section we compare Lanczos and Fast Lanczos for a model of an electric sedan car with 21.9 million degrees of freedom. Fig 3. (a) shows run time as a function of the number of modes, for a fixed number of MPI ranks (36). Fig. 3 (b) on the other hand shows run time as a function of the number of MPI ranks, for a fixed number of modes (500).

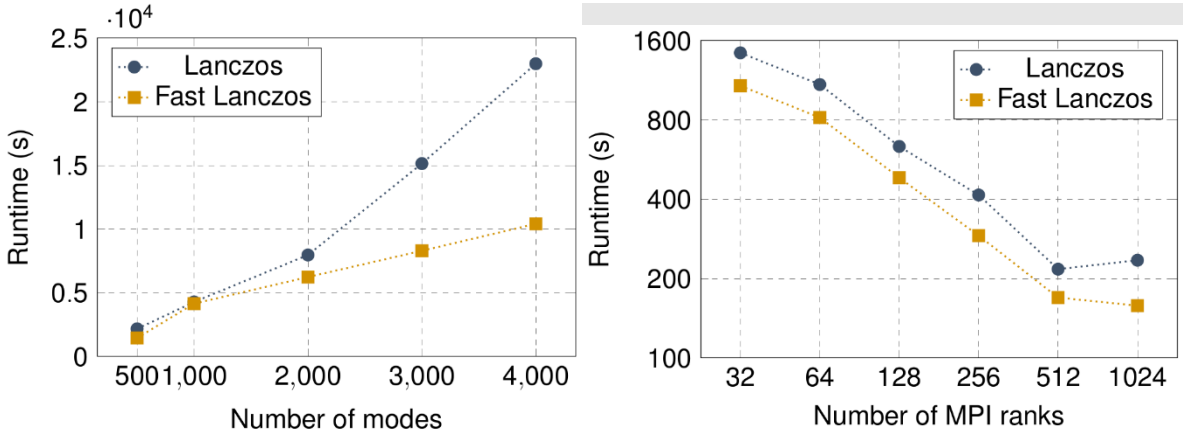

Fig.3: Performance of Lanczos and Fast Lanczos. (a) varying number of modes; (b): strong scaling.

The figure shows that Fast Lanczos is very scalable and outperforms the standard Lanczos algorithm when the number of modes becomes large. Note that this depends very much on the structure of the spectrum; not all problems might benefit from Fast Lanczos. In any case, we encourage users to try Fast Lanczos when the number of modes that they need gets into the thousands.

4 Output controls

One of the significant costs of computing eigenvectors in a distributed memory environment, especially when computing thousands of modes for problems of order tens of millions of rows, is the cost of outputting those eigenvectors. In this section we introduce two new options which give the user more control over the output of eigenvectors. We also introduce a new database which uses the ADAMS MNF format. On the third line of **`*CONTROL_IMPLICIT_EIGENVALUE`** are two new parameters, IPARM3 and RPARM4, which allow the user to control the type and volume of the output for the eigenvectors.

4.1 Controlling the d3eigv database

Parameter RPARM4 (field 8, line 3 of **`*CONTROL_IMPLICIT_EIGENVALUE`**) allows the user to specify how many modes (columns of the eigenvectors) are written to the d3eigv database. RPARM4 can be used to control the number of eigenvectors written to the d3eigv database. RPARM4 > 0 means that only the first RPARM4 eigenvectors (corresponding to the RPARM4 lowest eigenvalues) are written.

RPARM4 = 0, the default, means all vectors are written. RPARM = -1 means that nothing is written in the d3eigv database

4.2 Controlling the LSDA database

Parameter IPARM3 (field 8, line 3 of ***CONTROL_IMPLICIT_EIGENVALUE**) defines a node set. When a node set is specified, only a subset of each eigenvector is output for the degrees of freedoms on those nodes. This is especially useful for Frequency Range Computations. The output based on a node set is written to an LSDA database.

IPARM3 and RPARM4 can be used in combination with each other. See the Keyword Manual for full details on these two controls.

4.3 ADAMS MNF

LS-DYNA can also generate an Adams Modal Neutral File (MNF), which contains geometry information (nodal coordinates, mesh connectivity, nodal masses and inertias) and modal information (eigenvalues and mode shapes, generalized stiffness and mass) [11]. The MNF database can be used with other solvers, such as Hexagon's Adams Flex.

IPARM3 (mentioned above) can be used to restrict the MNF database to a nodal set. Only the corresponding node coordinates and masses, and the corresponding rows of the eigenvectors (modal shapes) are written in the MNF database. Connectivity information is not written since some faces (elements) can involve a mixture of nodes that belong or do not belong to the nodal set.

5 Summary and future work

We presented recent work on LS-DYNA's solvers for the standard, symmetric eigenvalue problem. All three solvers (Lanczos, LOBPCG, Fast Lanczos) have received significant improvements in the upcoming R16 release, both in terms of performance and robustness. We also added options that give the user more control over the output of eigenvectors.

Our solvers are continuously improved, and we are actively working on solvers for other eigenvalue problems. For example, for nonsymmetric problems (arising, e.g., in Rotational Dynamics), we are working on an implementation of the GPLHR algorithm (Generalized Preconditioned Locally Harmonic Residual) [12]; GPLHR can be seen an extension of LOBPCG to non-Hermitian problems; just like LOBPCG, it relies on a preconditioner instead of exact solves with the shifted stiffness matrix.

6 Acknowledgements

We thank Richard Sturt (Arup) and George Laird (Predictive Engineering) for providing us with test problems.

7 Literature

[1] Sorensen, D. C., "Implicit application of polynomial filters in a k-step Arnoldi method", SIAM Journal on Matrix Analysis and Applications, 13, 1992, 357–385.
[2] Lehoucq, R. B., Sorensen, D. C., and Yang, C., "ARPACK users' guide", Society for Industrial and Applied Mathematics, 1998.
[3] Bennighof, J., Kim, C., "An adaptive multi-level substructuring method for efficient modeling of complex structures", 33rd Structures, Structural Dynamics and Materials Conference, 1992, 23-27.
[4] Kim, C.-W. and Grimes, R. "A new eigensolver for high performance NVH analysis: MCMS (Multi-level Component Mode Synthesis)", 11th European LS-DYNA Conference, 2017.
[5] Grimes, R., Weisbecker, C., Ashcraft, C., Vecharynski, E., Rouet, F.-H., Anton, J., Li, L., and Lucas, R., "Enhancements to Implicit Mechanics", 11th European LS-DYNA Conference, 2017.
[6] Grimes, R., Lewis J. G., and Simon, H. D., "A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems", SIAM Journal on Matrix Analysis and Applications, 15(1), 1994, 228-272.
[7] Knyazev, A. V., "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method", SIAM Journal on Scientific Computing, 23, 2001, 517–541.

[8] Grimes, R. and Rouet, F.-H., "New eigen solver technology in LS-DYNA", 14th European LS-DYNA Conference, 2023.
[9] Amestoy, P. R., Duff, I. S., L'Excellent, J.-Y., and Koster, J., "A fully asynchronous multifrontal solver using distributed dynamic scheduling", SIAM Journal on Matrix Analysis and Applications (SIMAX), 23, 2001, 15–41.
[10] Amestoy, P. R., Buttari, A., L'Excellent, J.-Y., and Mary, T., "Performance and scalability of the block low-rank multifrontal factorization on multicore architectures", ACM Transactions on Mathematical Software (TOMS), 45, 2019. 1–26.
[11] Hexagon AB, Adams Flex User's Guide, 2024.
[12] Vecharynski, E., Yang, C., and Xue, F., "Generalized preconditioned locally harmonic residual method for non-Hermitian eigenproblems", SIAM Journal on Scientific Computing, 2016, 38, A500-A527.