The Influence of Geometrical Aspects in the Condition Number of the Stiffness Matrix: An Empirical Approach.

Manuel J. Galán Moreno[1] Fidel García del Pino[1]

Abstract

This is a study about the influence of geometrical constraints and considerations in the condition number of the stiffness matrix that is obtained in a typical elasticity problem. Our approach is experimental as we will study the influence on one one-dimensional parameter ("thickness ratio") in the convergence speed of the method. For the resolution we will use an ANSI-C environment.

Introduction

It is a well known fact that in the resolution of P.D.E. problems by means of the Finite Element Method the geometrical considerations of the particular problem have a most important role in all the aspects of the resolution of the problem, most notably in the condition number of the stiffness matrix.

However, the inference of rules that could relate the geometrical aspects of the P.D.E. problem with particular aspects and parameters of the stiffness matrix regarding its condition number in a general way can be considered an extremely difficult functional analysis problem that can be only be solved under strong simplifications.

In this scenario we will take an experimental approach: we will study the convergence rate of several strongly convergent iterative methods in a particular problem in which we can provide a one-dimensional parameter that measures the "difficulty" of the problem.

Algorithmic Methodology

Structural analysis using FEM could be, generally speaking, divided into several phases, i.e.: physical analysis and restrictions, conditions, simplifications; mathematical development and election of a mathematical model for the physical problem which satisfies the required conditions, formulation of this mathematical model, election of the type of Finite Element formulation, restrictions due to boundary conditions; Linear Algebra problem, the assembly and resolution of stiffness matrices, eigenvectors and/or eigenvalues; and lastly the discussion and interpretation of results.

Clearly, the points that include mathematical and/or physical modelization are heavily dependent on the particular problem that is to be solved. In our study we will address the part of the Linear Algebra problem that deals with assembly and linear resolution of the stiffness matrix. These problems will be treated in a more independent and general way.

Resolution.

Assembly: First of all, we will address the problem of the assembly of the so called "stiffness" matrix. This matrix has a very important property: it is sparse. This property and its implications on the storage scheme will be a guideline to our approach, not only because of the memory requirements but also because the sparsity pattern of this matrix can be very complex (3D-case, complex/non-regular structures, etc.).

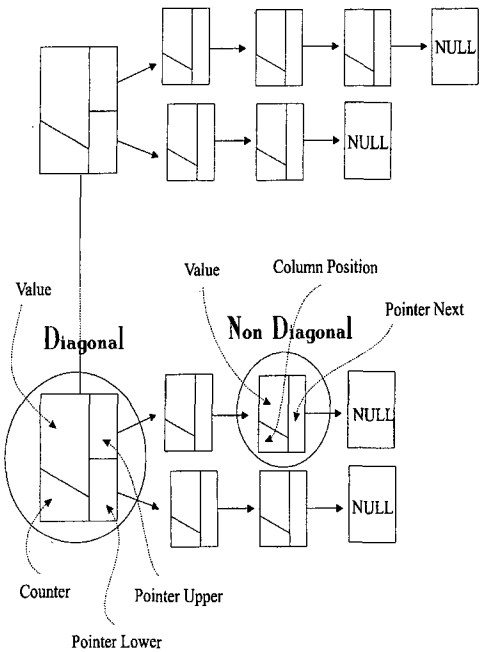

Fig. 1 Linked List storage scheme

Because of this (possible) complex sparsity pattern, we devise a new approach to the standard allocation schemes. Our aim will be to store only the relevant information keeping off any waste of memory and, at the same time, to ease the manipulation of the new entries of the matrix that are obtained from the contribution of the several elemental matrices.

We propose a robust, fast and simple C-scheme known as "simply linked lists" in this scheme we can view each row of the upper and lower triangles of the stiffness matrix as a (simply) linked list of structures each of them holding the column position, the value and a pointer to the next non-null element.

In this scheme, the location process involves, as an average, n/2 comparisons + link displacements, being "n" the number of pre-existent terms in the linked list. To complete a row with "k" elements in the upper triangle (supposing the same number of terms in the lower one) we will have, approximately, $\frac{k(2k-1)}{2}$ comparisons for the full row, being comparisons in integer arithmetic, the overhead is minimal. The cost of the rest of operations can be considered negligible.

For the actual resolution of the linear system we have devised another storage scheme: "hypercompact matrices"; this is a scheme based on Radicatti's compact storage method [Radicatti,1986], somewhat improved to take full profit of the flexible memory allocation possibilities of ANSI-C.

In this scheme the sparse matrix is decomposed in two "ragged" (different row length) matrices: one of them holding the values and the other one holding the column positions, the first column in each one of these new "ragged" matrices is formed, respectively, with the diagonal entry of stiffness matrix and the counter for non-null terms in its corresponding row, the other columns hold the ordered off-diagonal entries and their column positions.

This storage scheme is preferred over the linked lists one, in the resolution phase, because ragged matrices are a metaphor for "vector of vectors" this way arithmetic

operations are done along "naturally" stored vectors which is much more compiler optimizable and with greater possibilities of vector/parallel implementation than any indirect addressing scheme like that of linked lists [Dongarra, 1991].

The linked list scheme is adapted to the assembly problem, being the "hypercompact" one well suited to matrix/vector multiplication (most time consuming operation in iterative solving). The conversion between these two schemes is done by a simple subroutine.

(F)GMRES Algorithm.

The resolution of the system is done by Generalized Minimum Residual (GMRES) proposed by Saad and Schultz [Saad, 1986] for the resolution of non symmetric and non singular systems of linear equations:

GMRES is able to solve that system even in a non restrictive environment in which A is a non positive $n \times n$ matrix, thus being one of the most serious alternatives to the general solvers for large sparse non-singular systems.

Recently, a flexible version of GMRES (FGMRES) has been proposed by Saad [Saad, 1993], this version of the algorithm allows several, possibly different, preconditioners at each step.

One of the keys in the success of an iterative method is the election of the stopping criterion, it seems that taking a "scaled" residual is an appropriate alternative. This way our election will be $\frac{\|\mathbf{A}\mathbf{x}_n - \mathbf{b}\|}{\|\mathbf{b}\|} < Tol$.

Another important question is to evaluate the "proximity" to the real solution, this is achieved through the implementation of "a posteriori" error estimator.

Being $\hat{\mathbf{x}}(x)$ the exact (approximate) solutions of Ax = b and being r = Ax-b the residual, then $\|\mathbf{x} - \hat{\mathbf{x}}\|_p = c \frac{\|\mathbf{r}\|_2^2}{\|\mathbf{A}^T\mathbf{r}\|_q}$ where $c \in [1, C_p(\mathbf{A})]$ [Auchmuty, 1992].

Preconditioners

(F)GMRES is a very robust iterative solver, however, being a long-term recurrence method, it is most important to decrease the dimension of the Krylov subspaces. A big dimension would involve the storage of big full matrices spoiling the savings due to sparsity. This can be achieved by the use of "strong" preconditioners.

The proposed (F)GMRES algorithm gives us, on the other side, ample possibilities to choose a preconditioner. Several tests have been done taking as preconditioner the coupling of several iterations of Van der Vorst's BiCGStab preconditioned itself with Incomplete LU Factorization (ILU) or iteratively refined ILU (ILU^r) [Golub, 1992]. We prefer this ILU preconditioner over Incomplete Cholesky Factorization not only because of its greater stability, but also by its greater generality, thus allowing the treatment of more general kind of problems (non-symmetric/non-singular matrices). On the other side the storage overhead involved with ILU only supposes a duplicate hypercompact value matrix as the position matrix is shared with the stiffness one.

BICGStab: BiCGStab algorithm is an iterative resolution scheme based in Conjugate Gradient. It can be considered a short-term recurrence iterative method, the memory overhead is small, on the other side its optimality properties make it a good candidate as preconditioner. The algorithm and its properties are described in [Van der Vorst, 1993].

ILU and iteratively Refined ILU: Incomplete LU Factorization is based on the direct LU solver. However, when used as preconditioner, the factorization is done ONLY for the non-null entries of the matrix. It preserves the sparsity of the original matrix so, as said, the memory overhead is only a copy of the "values matrix".

We can improve the "strength" of ILU preconditioner by iterative improvement. This is another technique borrowed from the direct solvers' field [Golub, 1992].

If we have a matrix $\tilde{\mathbf{A}}$ that is an approximate inverse of A, defining the residual matrix as $\mathbf{R} = \mathbf{1} - \tilde{\mathbf{A}} \cdot \mathbf{A}$, we consider the following formalism: take $\mathbf{A}^{-1} = \left(\mathbf{A}^{-1} \cdot \tilde{\mathbf{A}}^{-1}\right) \cdot \tilde{\mathbf{A}}$, then $\mathbf{A}^{-1} = \left(\tilde{\mathbf{A}} \cdot \mathbf{A}\right)^{-1} \cdot \tilde{\mathbf{A}}$ and substituting, we have that $\mathbf{A}^{-1} = (\mathbf{1} - \mathbf{R})^{-1} \cdot \tilde{\mathbf{A}}$ $\mathbf{A}^{-1} = \left(\mathbf{1} + \mathbf{R} + \mathbf{R}^2 + \ldots\right) \cdot \tilde{\mathbf{A}}$. If $\|\mathbf{R}\| < 1$ we have that, being $\tilde{\mathbf{A}}_0 = \tilde{\mathbf{A}}$, and $\tilde{\mathbf{A}}_n = \left(\sum_{0 \le i \le n} \mathbf{R}^i\right) \cdot \tilde{\mathbf{A}}$, then $\lim_{n \to \infty} \tilde{\mathbf{A}}_n = \mathbf{A}^{-1}$, so follows the recursive relation: denoting by $\mathbf{x}_n = \tilde{\mathbf{A}}_n \mathbf{v}$, then $\mathbf{x}_{n+1} = \mathbf{x}_n + \tilde{\mathbf{A}}(\mathbf{v} - \mathbf{A}\mathbf{x}_n)$.

In our case A will be the ILU preconditioner. We will denote by ILU^r the application of "r" stages of iterative improvement to standard ILU preconditioner.

Validation of our Methodology.

General considerations.

We will, now, put together all these features to build a program for the resolution of a 3D truss structure. Our aim will be to check the possibilities of our environment when applied to really large linear systems.

We use the standard classical formulation for the formation of elemental matrices that are, eventually, assembled into the stiffness one.

In this particular simple case, further memory savings can be done as for the symmetry of the matrices involved (roughly speaking, we can reduce storage requirements to the half).

Comparison of results.

To validate the results of our program, we have compared them with FElt-3.02 [Atkinson, 1996]. We have solved a truss tower made up of 52 nodes and 160 elements, being 144 the order of stiffness matrix. The solutions given in this case using both methods are, essentially, the same (up to the 7^{th} decimal digit).

Case Study.

Now we are going to solve several large ill-conditioned linear systems. The test case structure is a dome composed of two connected layers with a total of 6,002 nodes and 29,581 elements, in this case the order of the stiffness matrix was 17,646.

The special characteristics of this structure, when the distance between layers (dome thickness) is a small fraction of the radius of the circumference give rise to a quite ill-conditioned stiffness matrix.

In our case we will start with a thickness of 1% and will increase this in 0.5 increments until we reach a 5% of thickness.

Our hypercompact and iterative scheme is able to solve the problem with our strong tolerance requirements of 10^{-9}.

We implement several methods, using several combinations of preconditioners. Main preconditioner is ILU(0) which, in turn, is reinforced with several loops of BiCGStab and Iterative Refinement.

The lines in the following graphics refer to the "thickness" of the structure $100\,\frac{R-r}{r}$, where *"R"* and *"r"* refer to the outer, inner radius of the dome, respectively.

As we can see there is an initial "divergence" of the method, (due to eigenvalue dispersion) which, eventually, is defeated as the dimension of Krylov subspaces increases.

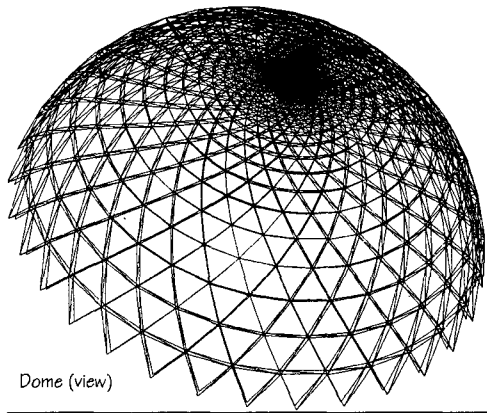

Fig. 2: Dome view

Clearly the better convergence properties appear when using ILU/BiCGStab(5)-RefIter(3), however, considering the computational costs of these different preconditioners, it seems that using ILU/BiCGStab(3)-RefIter(2) could be sensible (except when the memory constraints -Krylov dimension- are determinant).

We should mention that for the 1 % thickness case, convergence was NOT achieved using the algorithm ILU(0)/BiCGStab(1)-RefIter(1) for a dimension of Krylov Subspaces of 350.

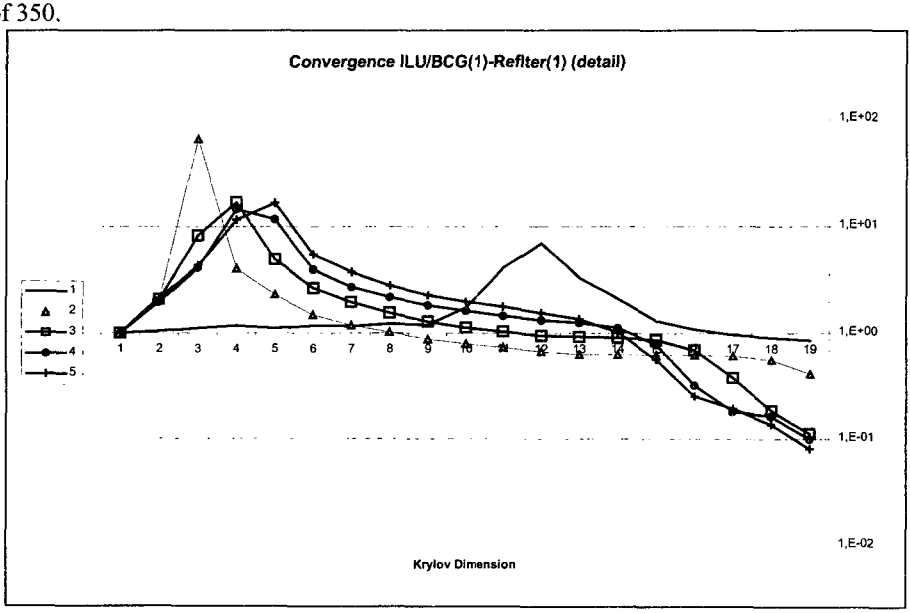

Fig. 3 Convergence Graph ILU(0)/BiCGStab(1)-RefIter(1) (detail)

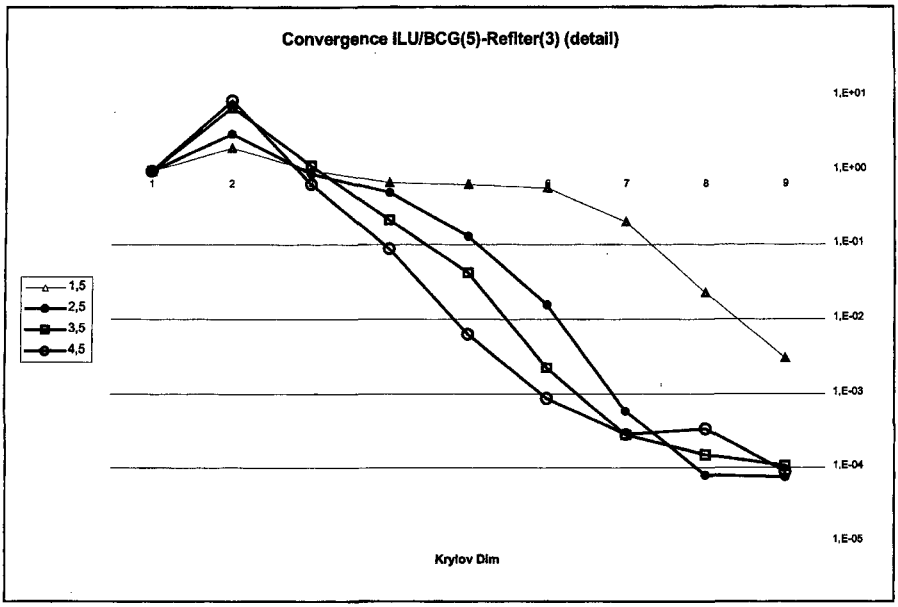

Fig. 4: Convergence Graph ILU(0)/BiCGStab(1)-RefIter(1) (detail)

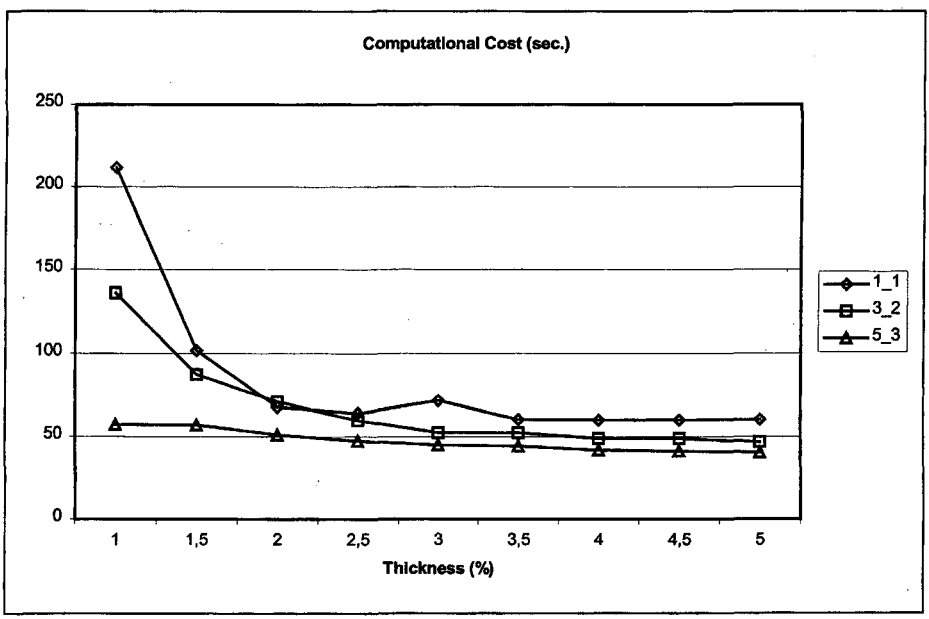

Fig. 5: Computational. Cost ILU(0)/BiCGStab(X)-RefIter(Y)

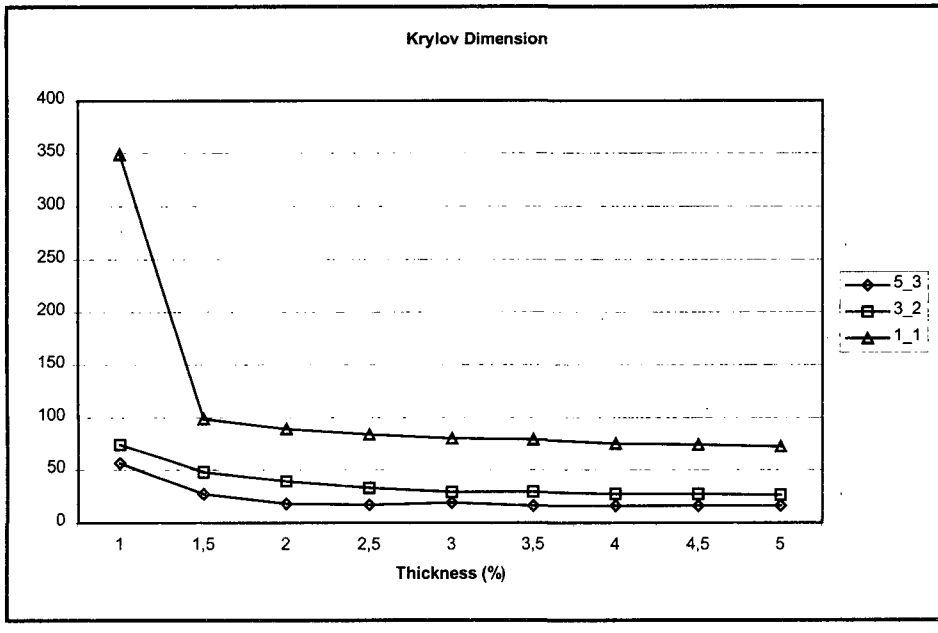

Fig. 6: Krylov Subspaces Dimension on reaching tolerance.

Conclusions

In this work, we have shown the strong influence of the geometry of the P.D.E. problem in the condition number of its stiffness matrix from an empiric/experimental point of view.

We have been able to correlate a one-dimensional parameter (thickness) with an estimation of the condition number of the matrix measured through the convergence speed of a strongly convergent iterative resolution scheme.

All through the algorithmic resolution we have used an alternative implementation of the structure analysis using ANSI-C, storage saving allocation schemes and strong iterative solvers with several advantages as:

- High portability due to the ANSI-C strong emphasis in the possibility of porting source code with small or no effort.
- Flexible and memory saving storage schemes (linked lists for assembly and hypercompact matrices for computational operations), which allow us to deal with very large sparse linear systems.
- Good convergence properties of "robust" iterative solvers coupled with strong preconditioners, which can help us to tame down ill-conditioned systems, otherwise unsolvable (direct methods).

Clearly, there are, some drawbacks in the iterative implementation, among them we could mention that in the case of small linear systems, there seems to be no gain in using iterative solvers. However in the case that we have to deal with big and very ill-conditioned systems, the iterative approach seems to be the only sensible one.

References

[1] Atkinson, D. and Gobat, J., ftp://ftp.cs.ucsd.edu/pub/atkinson/felt/FElt-3.0.2.tar.gz, (1996)

[2] Auchmuty, G., "A posteriori error estimates for linear equations", *Numer. Math.*, 61, (1992).

[3] Dongarra, J. J., Duff, I. S., Sorensen, D. C. and Van der Vorst, H. A., "Solving Linear Systems on Vector and Shared Memory Computers", *SIAM, Philadelphia*, (1991).

[4] Galán, M., Montero, G. and Winter, G., "A direct solver for the Least Square Problem arising from GMRES(K)", *Com. in Num. Meth. in Eng.*, 12, (1994a).

[5] Golub, G. and Van Loan, C. F., "MATRIX COMPUTATIONS" 2nd. Ed., The John Hopkins University Press, Baltimore, (1992).

[6] Radicati, G. and Vitaletti, M., "Sparse matrix-vector product and storage representations on the IBM 3090 with Vector Facility", *Report G513-4098, IBM-ECSEC, Rome*, (1986).

[7] Saad, Y., "A flexible inner-outer preconditioned GMRES algorithm", *SIAM J. Sci. Statist. Comput.*, 14-2, (1993).

[8] Saad, Y. and M. H. Schultz, M. H., "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.*, 7, (1986).

[9] Van der Vorst, H. A., "Bi-CGStab: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Statis. Comput*, 7, (1993).

1. Department of Mathematics. University of Las Palmas de Gran Canaria. Edificio de Arquitectura, Campus Universitario de Tafira Baja. 35017 Las Palmas de Gran Canaria. SPAIN (manolo@polaris.ulpgc.es)