7th Workshop on Numerical Methods in Applied
Science and Engineering (NMASE 08)
Vall de Núria, 9 a 11 de enero de 2008
©LaCàN, www.lacan-upc.es

SOLMEC: AN EFFICIENT C++ LIBRARY TO SOLVE LINEAR AND NONLINEAR ELASTICITY PROBLEMS

A.M. Rosolen, R.D. Millán and M. Arroyo*

LaCàN, Universitat Politècnica de Catalunya,
C/Jordi Girona 1-3, Barcelona 08934, Spain.
e-mail: adrian.rosolen,daniel.millan,marino.arroyo@upc.edu
web: http://www-lacan.upc.es

Key words: linear and nonlinear elasticity, meshfree schemes, direct and iterative solvers, quasi-Newton methods, constrained optimization, C++ library

Abstract. *A C++ library based on local maximum-entropy approximation schemes to solve linear and nonlinear elasticity problem is presented. The available tools are briefly described, and several implementation details are also mentioned. Selected numerical examples are shown in order to illustrate the capabilities of the library. The current and future developments are indicated.*

1 INTRODUCTION

The aim of this work is to present an efficient C++ library to solve linear and nonlinear elasticity problems. The main objective is to show the most relevant available tools and the current state of development of the library. The design and programming of the library was motivated by the convenience of having an efficient, robust, flexible and totally controllable tool to attack applications of mechanics as diverse as multiscale modeling, quasi-continuum approaches, large scale problems, model reduction, and variational adaptivity in meshfree methods.

The library is based on local maximum-entropy (LME) approximation schemes [1] for the numerical solution of the elasticity problems. These schemes belonging to the family of meshfree methods present smooth LME shape functions, which make them appropriate to deal with the applications above mentioned. It is worth noting that, although the library is based on LME schemes, its flexible structure allows an easy inclusion of other numerical approaches such as FEM [6, 21].

At the present time, the library may be used for the full analysis of two and three-dimensional linear elasticity problems. Several kinds of nonlinear elasticity problems can be also solved, and the work in progress is focus mainly on the creation and optimization of algorithms to solve nodal-based numerical integration and variational adaptivity for finite-deformation elasticity.

The structure of the paper is as follows. In Section 2, the principal tools of the library are described and the most significant implementation details are mentioned. In Section 3, selected numerical examples are shown both for linear and nonlinear problems. Some concluding remarks and future work are finally indicated in Section 4.

2 LIBRARY DESCRIPTION

The library is specifically designed to solve linear and non linear elasticity problems. The necessary algorithms to complete all the steps of computation are implemented in an efficient and robust way. The algorithms are also designed to be flexible and easily extended.

The library can be compiled by using *gcc*, *intel*, *intel-OMP* or *MPI* [11] compilers. Many of the implemented algorithms are parallelized by using *OMP* [12], and the conjugate gradient method is also parallelized to be used with MPI. The design of the library permits the parallelization of the algorithms, which is necessary to obtain a good performance in large scale problems. The library also permits to include METIS [7], which is useful for domain decomposition.

The documentation of the library and its structure can be automatically generated. This is possible because the Doxygen [5] tool is employed and specific comments are suitably introduced in the code.

The analysis of linear and nonlinear elasticity problems involves three major stages: preprocessing or model definition, processing or computation, and postprocessing. The tools of the library for each one of this stages are described in this section. The more significant details of implementation are also commented.

2.1 Preprocessing

In this stage the data related with geometry, materials, and boundary conditions of the problem are defined. At the moment, all the tools associated to this stage are included in the library, but they will be separated in the future in order to maintain the natural independence between the different stages.

The support of a LME scheme is not given by elements, but a a set of sample points associated to each node. By that reason, the main effort is then applied to the generation of nodes and sample (or integration) points for the domain of the problem. Qhull library [13] is employed to make the simplex regions (triangles in 2D and tetrahedral in 3D). The ways for generating sample points currently available in the library are based on Gauss-Legendre and Gauss-Hermite cubatures [18]. More details about the implementation and accuracy of such cubature rules can be consulted in reference [8].

The data structure to store the sample points and its associated nodes is inspired in sparse matrix format. In the Figure 1 it is shown an example of the nodes (red) and the sample points (green) generated for a cylindrical domain. The current tools only permit to define and impose boundary conditions on simple geometries as squares, rectangles, prisms, cubes and cylinders.

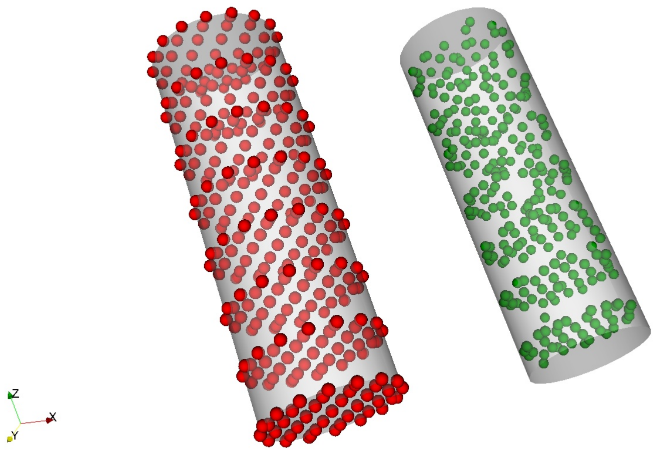

Figure 1: Nodes (red) and sample points (green) for a cylindrical domain. The sample points correspond to Gauss-Legendre points generated by applying QHULL.

2.2 Processing

This stage is related with the assembly of the stiffness and tangent matrix, and with the computation of the solutions. As the tools employed during the calculation process change if the problem is linear or nonlinear, the logical order of computation is not followed below to explain the developed algorithms, but they are only enumerated and described.

2.2.1 Shape Functions

A considerable difference between LME and FEM is given by the computation of the shape functions. While the shape functions for FEM depend only on the type of element, a local minimization problem must be solved in each node for LME in order to determine the value of the shape functions. It means that the calculation of the shape functions is computationally more expensive for LME schemes.

Even though the data structures to store the shape functions and its gradients are not detailed here, it can be asserted that they are optimal and very efficient because they are inspired in a sparse matrix format [14].

2.2.2 Matrix Assembly

All the algorithms of the library related with matrices were designed and optimized to work with sparse matrices in CSR or CSC format [14, 16, 17]. However, the extension of the algorithms to work with full matrices can be easily done if it is necessary for some specific application.

The assembly of the stiffness matrix [6, 21] for linear elasticity problems and the tangent matrix [2, 3] for nonlinear elasticity applications follow a similar process to that of FEM. The design of the data structure of the nodes and sample points facilitate and optimize the assembly process.

2.2.3 Direct Solvers

Efficient algorithms for LU and Cholesky decomposition [14] are currently implemented both for sparse and full matrices. The structure of the library allows to include easily other matrix decomposition algorithms.

The solvers associated to each matrix decomposition are also implemented for the data structure used in the library. They can be applied to solve a complete matrix decomposition when a direct method is used, or an incomplete matrix decomposition when a preconditioner is used within an iterative method.

2.2.4 Preconditioners

The ilut and the incomplete Cholesky preconditioners [14] are implemented. The algorithms are very efficient and optimize considerably the computational time when an iterative solver is used. They are implemented both for sparse and full matrices.

2.2.5 Iterative Solvers

The gradient conjugate algorithm [15] is only implemented to solve systems of linear equations. Different preconditioners can be chosen in order to reduce the computational time, and different criteria can be used to finalize the computation. Although the current conjugate gradient algorithm does not permit to solve systems of nonlinear functions, an extension of the implemented algorithm can be done with a little effort.

The Newton-Raphson algorithm [9] is only implemented for specific problems, and the design of a general structure for the algorithm is a work in progress.

The L-BFGS Quasi-Newton method is also available. This was not implemented, but it was taken from the work of Nocedal and others [4] and adapted to the structure of the library.

2.2.6 Nonlinear Elasticity

Algorithms to compute energy, forces and stiffness (tangent matrix) for materials defined by a strain energy function [2, 3] are implemented. The algorithms are efficient, and its flexible structure allows an easy inclusion of new materials.

2.2.7 Optimization

The optimization problems without constraints can be solved by applying one of the iterative solvers described above.

The L-BFGSB method (developed by Nocedal and co-workers [10]) is available to solve constrained optimization problems. This kind of problems can be also solved by using a combination of iterative solvers and an special dealing of the constraints.

2.2.8 Error Norms

Several algorithms are implemented in order to compute different error norms, both for scalar and vectorial magnitudes.

2.3 Postprocessing

It has not been implemented any tool related with the postprocess of the results obtained from the numerical simulations. However, VTK [20] can be included as tool in the structure of the library, by which the visualization of the domain, nodes, sample points, and numerical results is possible during execution time.

3 SELECTED NUMERICAL EXAMPLES

The four numerical examples presented in this section illustrate some possible applications of the library tools. The statement of the problems and the numerical approximation schemes are not shown, but they can be consulted in the documentation of the library or in the indicated references.

3.1 Poisson's PDE

The problem corresponds to a distribution of temperature for a square plate without any heat source but with a parabolic prescribed temperature on one of the edges. The analytical solution and the numerical results obtained by using LME are shown in the Figure 2. It can be noticed that analytical (green) and numerical (red) solutions can be exactly superposed due to the accuracy of the numerical approximation scheme.

3.2 Cantilever beam

This is a standard benchmark problem of linear elasticity. A cantilever beam is subjected to a parabolic distribution of tractions at one end and built-in boundary conditions

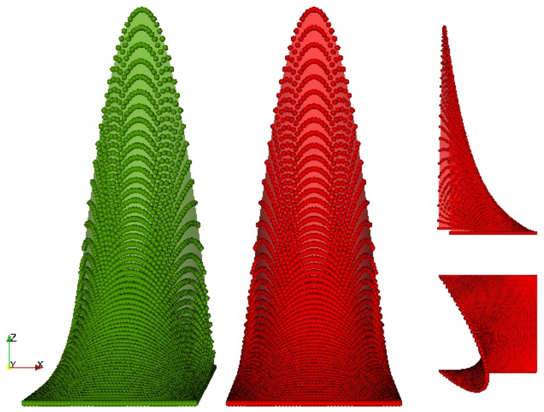

Figure 2: Temperature distribution for a square plate.

at the other end. The analytical solution of this problem is known [19]. The numerical results (red) obtained by using LME approximants and the initial (green) configuration of the cantilever beam are shown in the Figure 3.

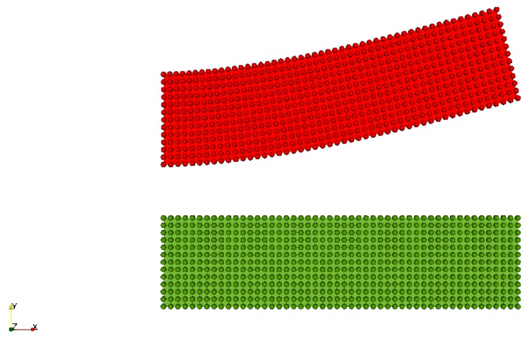

Figure 3: Linear elastic built-in cantilever beam loaded at the tip. Initial (green) and deformed (red) configurations.

3.3 Finite-Deformation compression

This is a strongly nonlinear elasticity problem. A hyper-elastic sheet of compressible neo-Hookean material is subjected to compression. The load is only applied to two edges of the solid sheet. The numerical results (red) obtained by using LME approximants and the initial (green) configuration of the sheet are shown in the Figure 4.

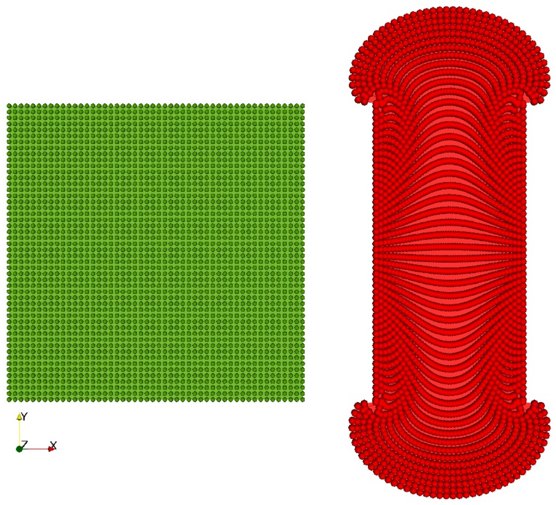

Figure 4: Hyper-elastic sheet of compressible neo-Hookean material subjected to compression. Initial (green) and deformed (red) configurations.

3.4 Finite-Deformation buckling

This is other strongly nonlinear elasticity problem. A long slender column is subjected to compression. A buckling effect appears as results of that load. The numerical results (red) obtained by using LME approximants and the initial (green) configuration of the column are shown in the Figure 5.

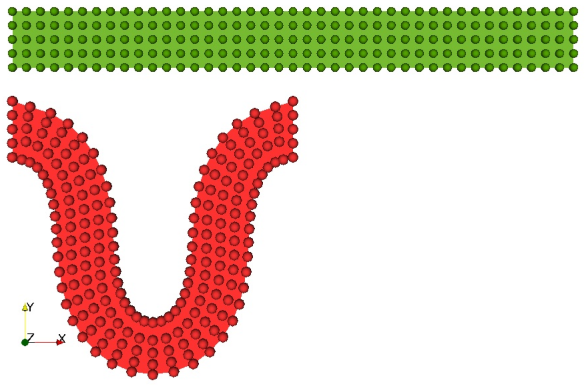

Figure 5: Long slender column subjected to compression. Initial (green) and deformed (red) configurations. A buckling effect appears.

4 CONCLUSIONS

It was presented a robust and efficient library to solve linear and nonlinear elasticity problems. The more relevant implemented tools were also briefly described. Several selected numerical examples were shown in order to illustrate the capabilities of the library.

It is important to remark the facilities that the structure of the library presents to include external libraries (as METIS or VTK) and to compile with *gcc*, *intel*, *intel-OMP* or *MPI*.

The work in progress is focus mainly on the creation and optimization of algorithms to solve nodal-based numerical integration and variational adaptivity for finite-deformation elasticity. The future algorithms of the library will be related with meshless applications.

REFERENCES

[1] M. Arroyo and M. Ortiz. "Local *maximum-entropy* approximation schemes: a seamless bridge between finite elements and meshfree methods". *Int. J. Numer. Meth. Engng*, Vol. **65**, 2167–2202, 2006.

[2] T. Belytschko, W.K. Liu and B. Moran, *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons, 2001.

[3] J. Bonet and R.D. Wood, *Nonlinear continuum mechanics for finite element analysis*, Cambridge University Press, 1997.

[4] R.H. Byrd, P. Lu and J. Nocedal. "A Limited Memory Algorithm for Bound Constrained Optimization", *SIAM Journal on Scientific and Statistical Computing*, **16:5**, 1190–1208, 1995.

[5] Doxygen: `www.stack.nl/~dimitri/doxygen`

[6] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, 2000.

[7] G. Karypis and V. Kumar. METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Department of Computer Science, University of Minnesota, 1998.

[8] R.D. Millán, A.M. Rosolen and M. Arroyo, "Local–node Gauss–Hermite cubature for numerical integration", NMASE 08, Vall de Núria, Spain, 2008.

[9] Newton-Raphson's Method: `en.wikipedia.org/wiki/Newton's_method`

[10] J. Nocedal. "Updating Quasi-Newton Matrices with Limited Storage", *Mathematics of Computation*, **35**, 773–782, 1980.

[11] MPI: `www-unix.mcs.anl.gov/mpi`

[12] OMP: `developers.sun.com/solaris/articles/omp-intro.html`

[13] QHULL: `www.qhull.org`

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Second Edition, SIAM, 2003. An Introduction to the

[15] J. R. Shewchuk, Conjugate Gradient Method Without the Agonizing Pain, School of Computer Graphics, Carnegie Mellon University, 1994.

[16] F. Smailbegovic, G.N. Gaydadjiev and S. Vassiliadis, "Sparse Matrix Storage Format", Computer Engineering Laboratory, TU Delft University.

[17] Sparse Matrix Storage Formats:
`www.intel.com/software/products/mkl/docs/webhelp/appendices/mkl_appA_SMSF.html`

[18] A.H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[19] Timoshenko and Goodier, *Theory of Elasticity*, Second Edition, McGraw-Hill, 1951.

[20] `VTK: www.vtk.org`

[21] O.C. Zienkiewicz and R.L. Taylor, *The finite element method*, McGraw Hill, Vol. I., (1989), Vol. II., (1991).

Con el apoyo de *Universitat Politècnica de Catalunya* y *E.T.S. d'Enginyers de Camins, Canals i Ports de Barcelona*