Efficient solution of mechanical and biomechanical problems by domain decomposition

Patrick Le Tallec, Marina Vidrascu

December 31, 1998

1 Introduction

The choice of an efficient and reliable method for the solution of large practical engineering problems is a difficult task. Direct solvers are robust and reliable but their storage requirements make them inappropriate for large three dimensional problems. On the other hand classical iterative solvers are more efficient but are sometimes unreliable in structural mechanics where the finite element matrices are very ill-conditioned.

Domain decomposition methods are a good alternative which combine advantages of both direct and iterative solvers. In these methods, the original domain is split into subdomains, local solutions are computed by direct solvers and an efficient iterative algebraic solver is used to properly interface these solutions. In addition these algorithms are appropriate for solving large scale problems on coarse grain parallel computers due to their built-in parallelism and good data localization. The key point to achieve robustness and performance is the choice of an efficient and parallel interface preconditioner.

For this purpose, we have developed and analyzed a generalized Neumann-Neumann preconditioner [9]. The objective of this paper is then to show the ability of this algorithm to handle real life problems as encountered in engineering and in biomechanical simulations.

Section 2 briefly reviews the main characteristics of the generalized Neumann-Neumann preconditioner. The convergence properties of the algorithm are then assessed in two different situations. The first one, described in Section 3 considers a challenging real life large scale engineering problem with large aspect ratios and very strong material heterogeneities. The second situation considers a biomechanical simulation, involving soft incompressible heterogeneous materials in very large deformations. Several classical algorithms of linear algebra will be used to solve this mid-size problem, the comparison of their performances being an alternative way to evaluate our algorithm.

2 The generalized Neumann-Neumann preconditioner

2.1 Iterative substructuring

A fundamental step in structural mechanics is to solve efficiently the algebraic system of equations obtained by finite element discretization of the equations of equilibrium. To reduce its size, iterative substructuring techniques split this original system into smaller coupled subsystems by partitioning the original domain of calculation Ω into the non-overlapping decomposition

$$\Omega = \bigcup_{i=1}^{N} \Omega_i, \quad \Omega_i \cap \Omega_j = \emptyset, \quad diam(\Omega_i) = O(H),$$

with interfaces

$$\Gamma_i = \partial\Omega_i \backslash \partial\Omega, \tag{1}$$

$$\Gamma = \cup_i \Gamma_i. \tag{2}$$

The resulting subsystems involve the stiffness matrix $\mathbf{K}_i$ of subdomain Ω_i defined by

$$(\mathbf{K}_i)_{lm} = \int_{\Omega_i} \Big(a(x) \cdot \nabla\phi_l(x) \Big) \cdot \nabla\phi_m(x) dx$$

and the corresponding right hand side

$$F_{il} = \int_{\Omega_i} f^{\Omega}(x) \cdot \phi_l(x) dx + \int_{\partial\Omega_N \cap \partial\Omega_i} f^{\Gamma}(x) \cdot \phi_l(x) da.$$

These matrices and right hand sides can obviously be computed independently on each subdomain. For three-dimensional elasticity problems involving complex heterogeneous anisotropic elasticity tensors $a(x)$ and second or higher order finite element shape functions $\phi_l(x)$, this local construction already involves a lot of computations, and is usefully done in parallel on different processors. For each subdomain the total set of degrees of freedom is then split into two subsets, the internal degrees of freedom $\mathring{X}_i$ associated to nodes which are strictly inside the subdomain Ω_i, and the interface degrees of freedom $\bar{X}_i$ associated to nodes lying on the boundary between two or more neighboring subdomains. With this partition, the subdomain stiffness matrix and right hand side take the form

$$\mathbf{K}_i = \begin{bmatrix} \mathring{\mathbf{K}}_i & \mathbf{B}_i \\ \mathbf{B}_i^t & \bar{\mathbf{K}}_i \end{bmatrix}, \qquad F_i = \begin{bmatrix} \mathring{F}_i \\ \bar{F}_i \end{bmatrix}. \tag{3}$$

Let us finally denote by $\bar{X} = \bigcup_i \bar{X}_i$ the entire set of degrees of freedom on the interfaces, and by $\bar{X}_i = \mathbf{R}_i \bar{X}$ the restriction of $\bar{X}$ on the boundary of Ω_i. In other words, $\mathbf{R}_i \bar{X}$ denotes the subset of interface degrees of freedom which lie on the boundary of Ω_i. Under this notation, after addition of the local contributions of all subdomains to the global stiffness matrix and right hand side, the linear system describing the global equilibrium of the domain Ω takes the standard block structured form

$$\begin{bmatrix} \mathring{\mathbf{K}}^1 & 0 & \cdot & 0 & \mathbf{B}^1\mathbf{R}^1 \\ 0 & \mathring{\mathbf{K}}^2 & \cdot & 0 & \mathbf{B}^2\mathbf{R}^2 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \mathring{\mathbf{K}}^N & \mathbf{B}^N\mathbf{R}^N \\ \mathbf{R}^{1t}\mathbf{B}^{1t} & \mathbf{R}^{2t}\mathbf{B}^{2t} & \cdot & \mathbf{R}^{Nt}\mathbf{B}^{Nt} & \sum_i \mathbf{R}_i{}^t\bar{\mathbf{K}}_i\mathbf{R}_i \end{bmatrix} \begin{pmatrix} \mathring{X}^1 \\ \mathring{X}^2 \\ \cdot \\ \mathring{X}^N \\ \bar{X} \end{pmatrix} = \begin{pmatrix} \mathring{F}^1 \\ \mathring{F}^2 \\ \cdot \\ \mathring{F}^N \\ \sum_i \mathbf{R}_i{}^t\bar{F}_i \end{pmatrix}.$$

This system is ideally solved by block Gaussian elimination. More precisely, the block diagonal structure of $\mathring{\mathbf{K}}$ allows to eliminate the internal degrees of freedom $\mathring{X}_i$ by

$$\mathring{X}_i = (\mathring{\mathbf{K}}_i)^{-1}(\mathring{F}_i - \mathbf{B}_i\bar{X}_i). \tag{4}$$

In mathematical terms, this elimination amounts to the parallel solution of a local equilibrium problem on the subdomain Ω_i with fixed Dirichlet boundary conditions.

After elimination, we obtain the reduced interface system

$$\sum_i \mathbf{R}_i^t\left(\bar{\mathbf{K}}_i\mathbf{R}_i\bar{X} + \mathbf{B}_i^t(\mathring{\mathbf{K}}_i)^{-1}\left[\mathring{F}_i - \mathbf{B}_i\mathbf{R}_i\bar{X}\right]\right) = \sum_i \mathbf{R}_i^t\bar{F}_i.$$

Introducing the so-called local Schur complement matrix

$$\mathbf{S}_i := \bar{\mathbf{K}}_i - \mathbf{B}_i^t(\mathring{\mathbf{K}}_i)^{-1}\mathbf{B}_i, \tag{5}$$

this interface system writes :

$$(\sum_i \mathbf{R}_i^t\mathbf{S}_i\mathbf{R}_i)\bar{X} = \sum_i \mathbf{R}_i^t(\bar{F}_i - \mathbf{B}_i^t(\mathring{\mathbf{K}}_i)^{-1}\mathring{F}_i). \tag{6}$$

The dimension of $\mathbf{S} = \sum_i \mathbf{R}_i^t\mathbf{S}_i\mathbf{R}_i$ gets rapidly very large (the interface problem has often more than 10^4 unknowns). Therefore direct substructuring techniques [12], which solve the interface problem (6) by a direct solver cannot handle large three-dimensional structures. By contrast, iterative substructuring techniques solve the interface problem by a preconditioned gradient method. They do not require the computation of Schur complement matrix $\mathbf{S}$, since it needs to be available in terms of a matrix-vector product only, which can be calculated by solving in parallel a Dirichlet problem (4) on each subdomain. But they do require an efficient preconditioner $\mathbf{M}^{-1}$. Indeed, introducing the ratio

$$r(\bar{X}) = \frac{\sum_i\langle \mathbf{S}_i\mathbf{R}_i\bar{X}, \mathbf{R}_i\bar{X}\rangle}{\langle \bar{X}, \bar{X}\rangle}$$

the condition number of $\mathbf{S}$ can be characterized by

$$\text{cond } \mathbf{S} = \frac{\max_{\bar{X}\in\mathbf{V}} r(\bar{X})}{\min_{\bar{X}\in\mathbf{V}} r(\bar{X})}.$$

Since $r(\bar{X})$ reaches its minimal value $r(\bar{X}) \approx \dfrac{\text{diam } \Omega}{\text{area } \Gamma} \approx H$ for smooth functions and its maximal value $r(\bar{X}) \approx \dfrac{1}{h}$ for interface shape finite element functions, it is easily deduced that the condition number of $\mathbf{S}$ scales like the inverse of Hh and blows up either for small discretization steps h or for small subdomain diameters H (i.e. for partitions with many subdomains). A preconditioner is therefore necessary. It must have nice parallel properties, be able to handle arbitrary elliptic operators and discretization grids, and its performance must be insensitive to the discretization step h and to the number of subdomains. In particular, to be efficient, the preconditioner must couple all subdomains together; otherwise smooth errors with support included in a single subdomain will not be corrected.

2.2 Neumann-Neumann preconditioners

As observed in [5] or [4], the additive Schwarz decomposition provides such efficient parallel preconditioners for the interface problem (6). In fact, all recent efficient iterative substructuring techniques can be viewed as additive Schwarz algorithms partitioning the interface space $\mathbf{V}$ of interface unknowns $\bar{X}$ with scalar product $\mathbf{S}$ into a sum of a coarse interface subspace $\mathbf{V}_0$ and of local interface subspaces $\mathbf{V}_i$.

In the generalized Neumann-Neumann algorithm which will be used in the present paper, this two scales interface partitioning uses [11],[10]

1. a partition of unity $\mathbf{D}^i : Tr\mathbf{V}_{|\Gamma_i} \to \mathbf{V}$ defined on the interface and satisfying

$$\sum_{i=1}^{N} \mathbf{D}^i \mathbf{R}^i \bar{X} = \bar{X}, \forall \bar{X}.$$

   For implementation reasons (flexibility and parallelism), the map $\mathbf{D}^i$ must be as local as possible. The generic choice consists in defining $\mathbf{D}^i$ on each interface point P_l of subdomain Ω_i as the relative value of the subdomain contribution to the stifness matrix at point P_l to the sum of these coefficients for all subdomains Ω_j containing the point P_l.

2. an approximate local operator $\tilde{\mathbf{S}}^i$ such that

$$\mathbf{S} \approx \tilde{\mathbf{S}} = \sum \mathbf{R}^{i^t} \tilde{\mathbf{S}}^i \mathbf{R}^i.$$

   In nonlinear elasticity, one may use the local Schur complement of the linearized elasticity operator computed once for all in the undeformed configuration. Computing its inverse will then amount to solve an approximate Neumann problem on each subdomain with fixed stiffness matrix $\mathbf{K}_i$

3. a $\tilde{\mathbf{S}}^i$ orthogonal decomposition of each local space $Tr\mathbf{V}_{|\Gamma_i}, i = 1, N$, into

$$Tr\mathbf{V}_{|\Gamma_i} = \mathbf{V}_i \oplus \mathbf{Z}_i.$$

Above, the local coarse space $\mathbf{Z}_i$ contains all potential local singularities, that is functions whose extensions are of very large (usually H dependent) energy. In particular, it must be such that

$$Ker\tilde{\mathbf{S}}^i \subset \mathbf{Z}_i \subset Tr\mathbf{V}_{|\Gamma_i}.$$

For elasticity problems, $\mathbf{Z}_i$ is usually taken as the space of local rigid body motions.

We then define the generalized Neumann-Neumann domain decomposition technique as the additive Schwarz algorithm solving $\mathbf{S}$ on the space $\mathbf{V}$ of restrictions with coarse space $\mathbf{V}_0 = \sum_{i=1}^{N} \mathbf{D}^i \mathbf{Z}_i \subset \mathbf{V}$, endowed with the scalar product $\tilde{\mathbf{S}}$, with local spaces $\mathbf{V}_i, i = 1, N$ endowed with the scalar product $\mathbf{B}_i = \tilde{\mathbf{S}}^i$, and with extensions $\mathbf{I}_i = (I - \mathbf{P})\mathbf{D}_i$, where $\mathbf{P}$ denotes the $\tilde{\mathbf{S}}$ orthogonal projection of $\mathbf{V}$ onto $\mathbf{V}_0$. The above choice of extension map is in fact the key characteristic point of the Neumann-Neumann algorithm.

By construction, this Neumann-Neumann algorithm corresponds then to the classical additive preconditioning operator

$$\mathbf{M}^{-1} = \tilde{\mathbf{S}}_0^{-1} + \sum_i (I - \mathbf{P})\mathbf{D}^i \tilde{\mathbf{S}}_i^{-1} \mathbf{D}^{i^t} (I - \mathbf{P})^t.$$

Its theoretical efficiency has been analyzed in different papers. Using the above Neumann-Neumann preconditioner in the framework of three-dimensional linear elasticity problems or of shell problems, the condition number of the operator $\mathbf{M}^{-1}\mathbf{S}$ has been proved to be bounded by

$$\text{Cond}(\mathbf{M}^{-1}\mathbf{S}) \leq \frac{C}{\alpha_i^2}[1 + ln \max_i \frac{H_i}{h_i}]^2, \tag{7}$$

the constant C being independent of the subdomains diameters H_i, discretization steps h_i, aspect ratios α_i and averaged stiffness coefficients ρ_i.

The above result guarantees the scalability of the proposed algorithm with respect to the number of subdomains (H_i independence), and its robustness with respect to strongly heterogeneous elasticity coefficients (ρ_i independence). This independence with respect to coefficient jumps is due to our specific choice of weighting factors

$$\mathbf{D}_i v(P_l) = \frac{\rho_i}{\rho} v(P_l).$$

2.3 Implementation

The above algorithm has been implemented on different parallel computers in the framework of the general purpose finite element library Modulef. The PVM library is used for message passing. Thus the software is portable and can run on different platforms including standard parallel computers and clusters of workstations.

The first step common to all domain decomposition methods is to divide the original domain into small non overlapping subdomains. The quality of this partition is essential : on one hand, the convergence properties of most iterative substructuring algorithms depend on the subdomains' aspect ratio. On the other hand, the parallel performance depends on the load balancing induced by this partitioning strategy. How to measure such loads as a function of the number of nodes, elements or sizes of the local matrices is not an obvious problem and is beyond the aim of this paper. In our practical applications we will use the automatic mesh partitioner **decomp** based on K means techniques, available in the Modulef library. It gives reasonable partitions both with respect to aspect ratio and load balancing.

The practical implementation of all these substructuring algorithms requires then two successive steps. First, the action of the operator $\mathbf{S}$ on the direction d is computed by the sum

$$\bar{r} = \sum_i \mathbf{R}^{i^t} \mathbf{K}^i \begin{bmatrix} \mathring{u}_i \\ \mathbf{R}^i d \end{bmatrix},$$

of the solutions $\mathring{u}_i$ of the local Dirichlet problems (4) (to be computed in parallel)

$$\mathring{\mathbf{K}}^i \mathring{u}_i = -\mathbf{B}^i(\mathbf{R}^i d).$$

The preconditioning step then amounts to add the solutions of the local Schur complement problems (once projected onto the orthogonal to the coarse space)

$$\mathbf{M}^{-1}\bar{r} = (I - \mathbf{P}) \sum_i \mathbf{D}^i \bar{u}_i.$$

Here, the solutions $\bar{u}_i$ are obtained by solving in parallel the local approximate Neumann problems

$$\begin{aligned} \bar{u}_i \in \mathbf{V}_i \quad : \quad & \langle \mathbf{K}_i^{ap} \bar{u}_i, \bar{v}_i \rangle = \langle \bar{r}, I_i \bar{v}_i \rangle, \\ & \forall \bar{v}_i \in \mathbf{V}_i. \end{aligned}$$

All local problems are solved by direct solvers.

3 Application to a large scale engineering problem

We consider herein the numerical static simulation of a model three-dimensional slender structure described in Figure 1. This structure is reinforced by very stiff steel rods and thin layers of anisotropic materials and is supported on small soft contact areas. Additional periodic boundary conditions are imposed on both ends.

Such a structure is very typical of what can be expected in many engineering applications. All components are considered here to be linearly elastic. The steel

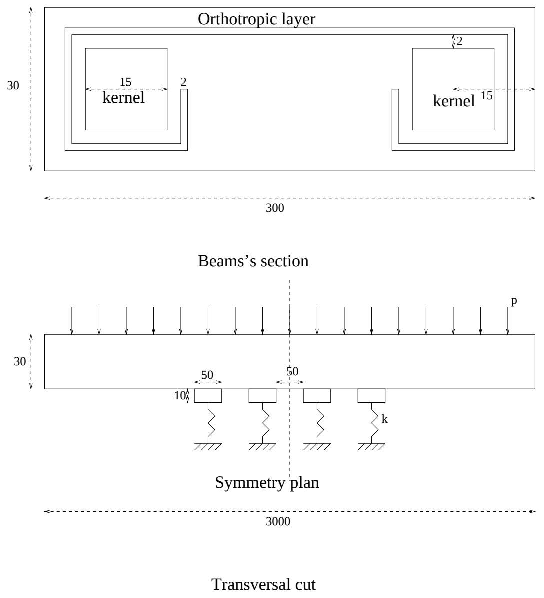

Figure 1: *Example of a linear elasticity problem.*

in the rods has a Young modulus of $E = 15000 daN/mm^2$ and a Poisson ratio of $\nu = 0.3$. The rubber is supposed to have a Young modulus of $E = 5 daN/mm^2$ and a Poisson ratio of $\nu = 0.45$. In the same system of units, the elasticity tensor in the orthotropic material of the thin layer is given by

$$EL = \begin{bmatrix} 2000 & & & & & \\ 50 & 50 & & & & \\ 52 & 50 & 50 & & & \\ & & & 2 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix}$$

For solving this problem, we have used a global finite element mesh made of 14 688 second order hexaedra, that is 67 633 nodes and 202 899 degrees of freedom. This is still a very coarse mesh for this problem : the thickness of a single finite element in the z direction is about half the total thickness of the structure along the y direction.

Despite the rather coarse character of the finite element discretization, the resulting linear system is quite difficult to solve. The matrix is large and the addition of periodic boundary conditions increases the size required for its solution by a direct method : the space needed to store the matrix in a skyline form which is $0.785\,10^9$ words without boundary conditions reaches $1.3\ 10^9$ words when one takes the periodic boundary conditions into account. Moreover, the matrix has a very bad condition number, because of the structure's slenderness, of the material heterogeneities, and of the second order elements used in the simulation.

In such cases, the domain decomposition algorithm is a good alternative. The periodic boundary conditions can be imposed in a natural easy way by treating them as a fictitious interface boundary separating two neighboring subdomains. During the computations, this new boundary will therefore play exactly the same role as the other ones. In this framework, the domain is decomposed manually into 24 subdomains. The load balancing is not optimal (each domain contains between 520 and 720 elements, that is between 2760 and 3733 nodes) but the inter-domain boundaries are very regular (Figure 2) which makes the substructuring algorithm converge faster.

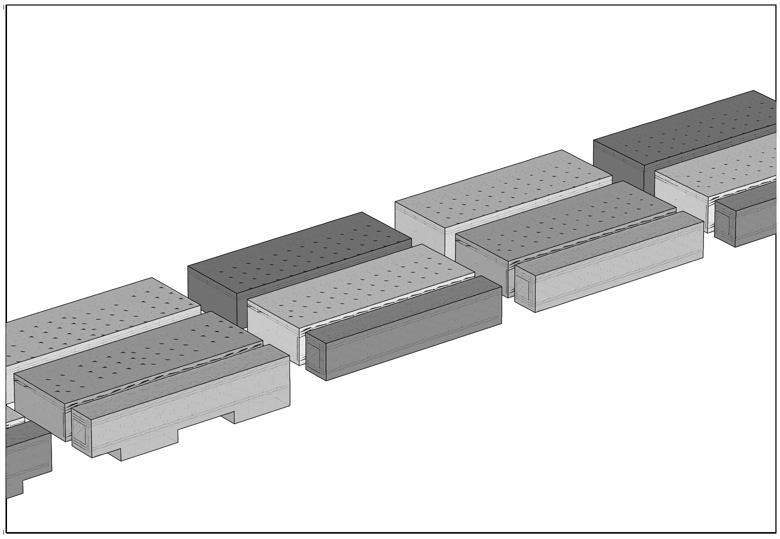

Figure 2: *Partial vue of the decomposition used for the model problem.*

Our algorithm has in fact successfully converged to a relative residual of $0.87\ 10^{-6}$ after 73 iterations. The condition number for the preconditioned Schur interface problem, as obtained by monitoring the iteration coefficients of the conjugate gradient algorithm, is 713. Our computation was performed on a cluster of HP workstations, with full parallelization of the matrix construction, each subdomain being processed on a different processor. The reduction in size induced by subdomain partitioning, combined with the distribution of these stiffness matrices was the only way for us to fit into the available computer memory.

4 A biomechanical simulation

4.1 Surgery simulation

Laparoscopic techniques in liver surgery reduce operating time and morbidity. As explained in [2], this technique uses one small incision through which a video camera and a surgical tool can be inserted. This novel technique has several advantages for the patients, but it is more complex and in particular it requires a perfect hand eye coordination. Surgeons need therefore to practice, on dead humans, on live animals or on simulators. As the mechanical properties of the tissues drastically change after death and, since for obvious ethical reasons, the use of living animals is limited, surgery simulation remains as the only appropriate training tool.

A Surgery Simulator was developed at INRIA in collaboration with the IR-CAD institute. In order to allow for real time simulations, the liver is modeled by a simplified linear model [3], [1]. The liver is treated as a linearly elastic incompressible isotropic body. The actions of the bistoury or of the pliers on the liver are modeled by imposing the displacement of the nodes which are in contact with the tools. The problem to solve is then a standard linear elasticity problem. Real time simulation can then be achieved by precomputing the response of the structure to a set of imposed displacements and constructing the corresponding Schur complement (transfert) matrix.

Our aim here is to develop a more realistic biomechanical model and compare its numerical predictions to those obtained by the real-time simulator.

4.2 Mechanical model

4.2.1 Geometrical model

The proposed model must first respect the real geometry of the liver. For this purpose, the contour of the liver under study is first reconstructed automatically from a three dimensional medical image, using image reconstruction techniques as described in [1]. This step produces a smooth definition of the external surface endowed with a triangular surface grid. This first mesh of the external surface can be regularised or refined if needed [6]. Then, an unstructured tetrahedal mesh of the interior is obtained by using GHS3D, an automatic three dimensional mesh generator [7]. Because of the geometric complexity of the data and to be able to obtain a reliable finite element mesh for each individual patient, it is mandatory to use tetrahedral finite elements. This is not standard in nonlinear structural mechanics where hexahedral finite elements are usually prefered because of their better accuracy.

Because of this geometric requirement, and in order to properly handle the incompressibility of the liver, we have used second order $P2$ Lagrange finite elements for approximating the displacement fields, and piecewise constant $P0$ elements for imposing the incompressibility constraint and approximating the hydrostatic pressure field. This is not an accurate choice, but it is stable (in the sense of the Ladyzenskaya-Babuska-Brezzi compatibility condition) and simple. Moreover, the liver is not too much confined, which means that the pressure field should be rather smooth and of small intensity, and that therefore a piecewise constant approximation should be sufficient. Finally, the anisotropic membrane which envelops the liver has been treated as a hyperelastic thin shell in large displacements, and its displacements are approximated by nonlinear DKT shell triangular finite elements. For kinematic compatibility reasons, we have identified the shell nodes with the three dimensional internal nodes. This means that each boundary face of a volumic second order finite element corresponds to four DKT triangles. In this framework, we have used in our first simulations a midsize three-dimensional mesh of 6342 elements. The corresponding volumic model has therefore 9741 nodes and 29 223 degrees of freedom.

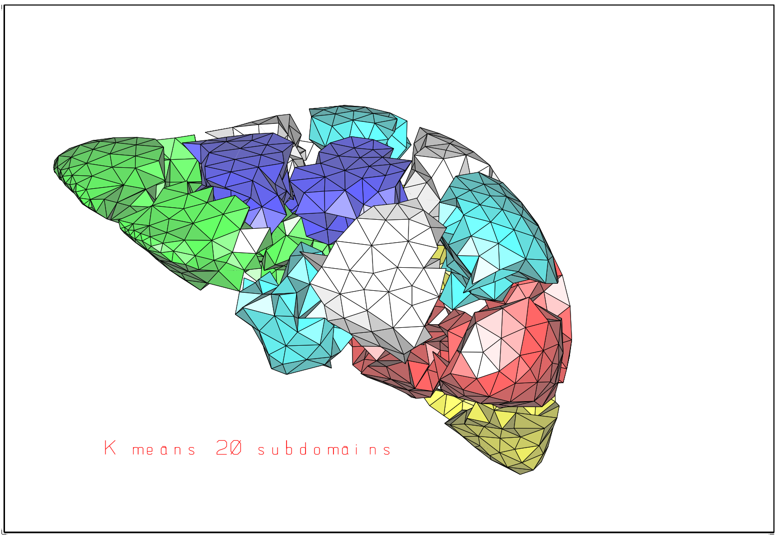

Figure 3: *Mesh of the liver.*

On this real geometry described in Figure 3, different types of boundary conditions were imposed. The first type of boundary conditions describes the part of the boundary which is in direct contact with stiff neighboring organs, typically the suspensory ligament and the vena cava. We impose there zero displacements. Non zero imposed displacements were applied to the nodes in direct contact with the tools, and model the motion imposed by the surgeon. Finally, the rest of the boundary is supposed to be free of any external loads or imposed displacements. More realistic boundary conditions will be investigated later.

4.2.2 Material characteristics

Modeling the deformation of human organs for surgery simulations is a challenge. Very little is known about the physical properties of general human tissues. In addition, these tissues are usually incompressible and undergo large displacements or deformations. Therefore one must use three-dimensional nonlinear incompressible hyperelastic models [8].

In this framework, we have treated the liver as a hyperelastic material with constitutive energy

$$\mathcal{W}(F) = C_1(I_1 - 3) + C_2(I_2 - 3) + a(J^2 - 1) - (2C_1 + 4C_2 + 2a) log J, \qquad (8)$$

where a, C_1, C_2 are material dependent constants, and $I_1, I_2, J = \det F$ are the invariants of the right Cauchy Green tensor $F^tF = (Id+\nabla u)^t(Id+\nabla u)$. In small deformations, this model reduces to a standard isotropic elastic constitutive law with Lamé coefficients

$$\begin{aligned} 2C_1 + 2C_2 &= \mu, \\ C_2 + a &= \frac{\lambda}{4.} \end{aligned} \qquad (9)$$

The first computations will use a value of Young modulus $E = 10000$. The corresponding values of the constants C_1, C_2, a in (8) will be $C_1 = 1515$, $C_2 = 151.5$ and $a = C_1 * 10^4$.

4.3 Solution of the linear problem

A linear problem is first considered for comparison purposes. As described in 4.1 the starting point for the real time simulator is the solution of a linear problem for an isotropic material ($E = 10000$ and $\nu = .4999$). We use here full second order tetraedric finite elements $P2$ on the finite element mesh described in 4.2.1. The liver is subjected to an imposed displacement of a few particular nodes.

The matrix has 1172865 non zero entries. Several classical algorithms were used to solve the problem :

- Direct method using Cholesky factorization. After renumbering, by Gibbs algorithm, 92 736 689 words are needed for the skyline storage of the matrix, and the solution time is approximatively of 10 hours on a HP900 workstation ;

- Preconditioned Conjugate Gradient (PGC) with different preconditionners :
  - incomplete Cholesky factorization : this requires 2345730 words for the storage of the matrix and of the preconditioner; droping the fill-in terms make it impossible to compute the Cholesky factor because of lack of positivity.
  - incomplete Crout factorization : the algorithm does not converge ;
  - incomplete Crout factorization with fill-in: the algorithm does not converge for level 1 and 2. In this last case 23 239 968 words are needed to store the preconditionner ;
  - the cost of a fill-in level 3 is similar to the cost of the full computation.
- domain decomposition algorithm : using the partition described in the next section, the algorithm converges in 74 iterations, using a computing time of approximatively 10 mn.

We can notice that the choice of Crout incomplete factorization ($A = LDL^t$) is more robust that Cholesky ($A = LL^t$). The previous results show once again the difficulty to solve very ill conditioned problems by PGC methods. In this particular case, as we do not have an M matrix, the robustness of the above preconditioners is not guaranteed. On the other hand, the direct method is indeed robust but it is rather time consuming. In any case, the domain decomposition algorithm is by far the most efficient. Using a non dedicated cluster of HP worktations, standard Ethernet with only a few workstations having a 100-Mbit connection, and being in competition with large number of other applications running at the same time (including applications as netscape or sendmail which are demanding in communications), the CPU time drops from ten hours to about ten to fifteen minutes. These computational times are not very accurate because of the heterogeneous environment, but give a general idea of the potential of the method.

4.4 Solution of the non linear problem

A robust algorithm to solve the full nonlinear problem involving soft hyperelastic materials is the Newton's method with arc length continuation [8]. This continuation strategy is applied here to control the magnitude of the imposed displacement. Each step of this Newton's algorithm with arclength continuation is then a linear elasticity problem, on which we apply the standard Neumann-Neumann domain decomposition algorithm. For robustness, since the tangent stifness matrix can loose its positivity, we have replaced the conjugate gradient algorithm by a GMRES technique using the same preconditioner to solve the interface problem. A further simplification can be added in this nonlinear case by keeping the same interface preconditioner during several consecutive Newton iterations.

The initial domain is decomposed into 20 subdomains as described in Table 1.

elements	nodes	Matrix size	elements	nodes	Matrix size
318	628	**1185150**	321	622	1102836
316	627	924372	315	611	1068594
316	**592**	**699096**	317	600	977670
323	616	1123482	321	626	966984
316	604	817230	317	628	858468
319	621	792900	316	608	1027212
321	610	1155180	316	605	1048236
313	634	754482	314	619	1065414
314	628	916716	**310**	**663**	857556
321	612	1014282	318	602	1082076

Table 1: *Description of the decomposition : number of elements, nodes and matrix size per subdomain*

This decomposition is reasonable and gives nice results. It nevertheless illustrates the difficulty to define load balancing. Indeed, the relevant parameter for the matrix computation is the number of elements in each subdomain , and for the solution step the relevant parameter is the matrix size. In table 1 the minimum and maximum values are in bold characters. Observe that the same subdomain has the smallest number of elements and larger number of nodes and that there is no connection between number of nodes and matrix size.

The corresponding software is parallel and uses the message passing paradigm and a **master-slave** approach. Here **the master** pilots the entire Newton algorithm for the interface problem and each **slave** is in charge of the local computations by subdomain. In practice, the user starts the master process. The number of subdomains $NDSD$ is a data for the computation, the master process starts NDSD slave processes.

The same loads as in the linear case are considered : the liver is clamped on a part of the boundary and a normal displacement of 0.6 cm is imposed on one side of a tetrahedra (compared to a global size of 20 cm for the full liver). The algorithm converged in 15 continuation iterations, for each of them about 6 Newton steps are used. The convergence of the domain decomposition algorithm at one Newton iteration needs between 140 and 400 GMRES iterations.

4.5 Comparison between linear and nonlinear

In the computations described the only load imposed is a prescribed displacement of one side. Far from this particular side the displacement of the structure are not very important. In order to compare the linear and nonlinear model we selected a particular node in the part of the structure where the displacements are significative. For the differents loads used in the continaution process we computed the displacement of this selected node.

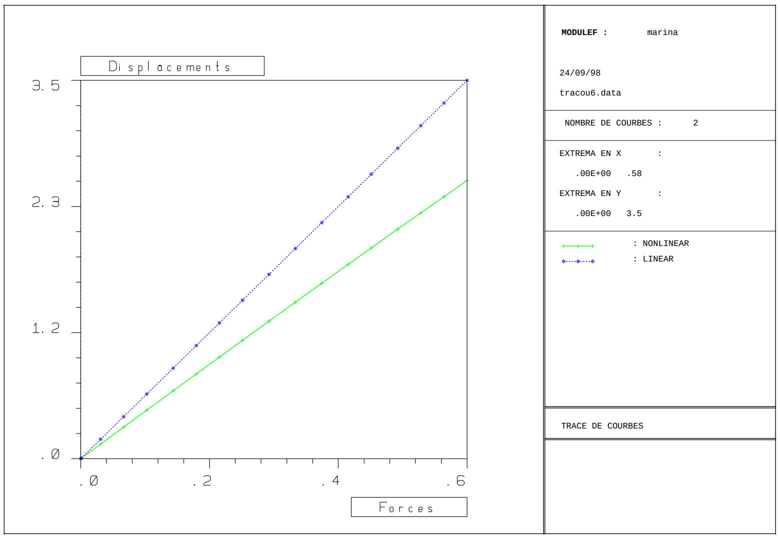

Figure 4: *Comparison between the linear and nonlinear model.*

Figure 4 presents the displacement of the selected node as a function of the load for the linear and the nonlinear case. It is interesting to notice that the problem is highly non linear, even for very small displacements : an imposed displacement of 0.001 cm requires 4 Newton iterations.

4.6 The full test with internal body and external shell

We present bellow a first test of the coupled model with 3d and shell elements. At this point the material characteristics of the shell are not known. In the present computation the shell Young modulus was taken 100 times larger than the 3d modulus and a thickness of 0.02 cm.

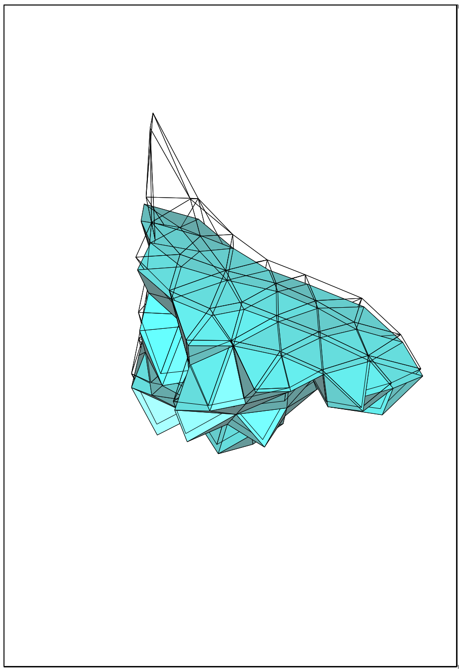

Figure 5: *Deformations for the 3d model*

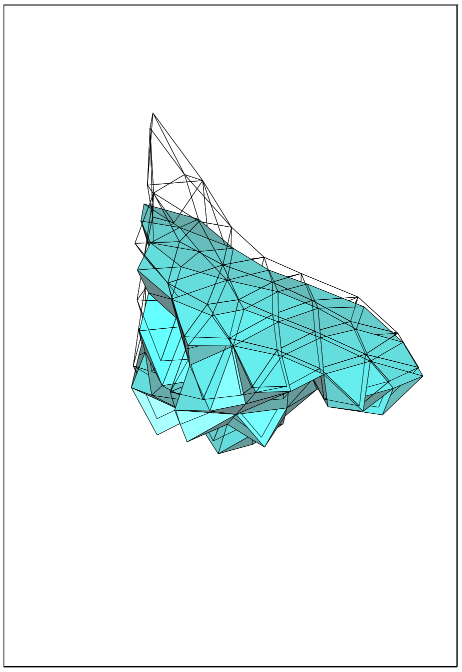

Figure 6: *Deformation for the 3d model with external shells*

For a small load coresponding to an imposed displacement of 0.001 cm convergencewas reached in 2 Newton iterations with 48 (respectively 44) GMRES iterations for the domain decomposition. Figures 5 and 6 display the initial and deformed configurations for both models.

We observe that, as expected, the model with external shells is smoother.

5 Conclusion

The results presented show that the generalized Neumann - Neumann algorithm is affordable and efficient for real life problems arising from linear and nonlinear elasticity. This algorithm is easy to use and implement and can handle problems with highly non homogeneous materials. In addition the software is appropriate for clusters of workstations which make the computation cheap.

On the other hand, solving nonlinear problems involving soft tissues remains expensive and questionable due to their severe nonlinearities and to a lack of mechanical data. Additional work is therefore needed to improve the mechanical relevance of the model and to better control its nonlinear behavior. In such a framework, the tools described in the paper, such as iterative substructuring and models coupling 3D volumic and nonlinear shell elements will play a key role.

References

[1] S. Cotin. *Modèles anatomiques déformables en temps-réel: Application à la simulation de chirurgie avec retour d'effort.* Docteur en informatique, Université de Nice Sophia Antipolis, INRIA Sophia Antipolis, 19 novembre 1997.

[2] S. Cotin, H. Delingette, and N. Ayache. Real time volumetric deformable models for surgery simulation. *Lecture Notes in Computer Science*, 1131:535–540, 1996. Visualization in Biomedical Computing.

[3] H. Delingette. Toward realistic soft-tissue modeling in medical simulation. *Proceedings of the IEEE*, 86(3):512–523, march 1998. Special issue on virtual and augmented reality in medicine.

[4] M. Dryja and O. Widlund. Towards an unified theory of domain decomposition algorithm for elliptic problems. In R. Glowinski T. Chan, editor, *Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Houston, Texas, March 20-22, 1989*, Philadelphia, 1990. SIAM.

[5] M. Dryja and O. Widlund. Additive Schwarz Methods for Elliptic Finite Element Problems in Three-Dimensions. In T. Chan, D. Keyes, G. Meurantand S. Scroggs, and R. Voigt, editors, *Proceedings of the fifth international symposium on Domain Decomposition Methods for Partial Differential Equations, Norfolk, May 1991*, Philadelphia, 1992. SIAM.

[6] P.J. Frey and H. Borouchaki. Geometric surface mesh optimization. *Comput. Visual Sci.*, 1:113–121, 1998.

[7] P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Comput. Methods Appl. Mech. Engrg*, 92:269–288, 1991.

[8] P. Le Tallec. *Numerical Methods for Nonlinear Three-Dimensional Elasticity*, volume 3, pages 465–622. P.G. Ciarlet and J.L. Lions, 1994.

[9] P. Le Tallec and M. Vidrascu. Solving large scale structural problems on parallel computers using domain decomposition. In *Solving Large Scale Problems in Mechanics*, chapter 2, pages 49–82. John Wiley & Sons, 1997.

[10] P. Le Tallec and M. Vidrascu. Generalized neumann-neumann preconditioners for iterative substructuring. In Magne Espedal Petter Bjørstad and David Keyes, editors, *Proceedings of the ninth international symposium on Domain Decomposition Methods for Partial Differential Equations, Bergen, June 96*, 1998.

[11] J. Mandel. Balancing domain decomposition. *Communications in numerical methods in engineering*, 9:233–241, 1993.

[12] W. Przemieniecki. Matrix structural analysis of substructures. *Am. Inst. Aero. Astro. J*, 1:138–147, 1963.