Stress based finite element methods for solving contact problems: comparisons between various solution methods

François Kuss, Frédéric Lebon

Stress based finite element methods for solving contact problems: Comparisons between various solution methods

François Kuss*, Frédéric Lebon

Laboratoire de Mécanique et d'Acoustique, 31 Chemin Joseph Aiguier, F-13402 Marseille Cedex 20, France

This paper deals with numerical methods for solving unilateral contact problems with friction. Although these problems are usually defined in terms of the displacement, a stress based approach to the problem is developed here. The "equilibrium" finite elements method is therefore used. Using these elements make it possible to satisfy the local equilibrium condition *a priori*, but on the other hand, prescribed and contact forces have to be introduced using Lagrangian multipliers. The problem obtained is therefore a non-linear, constrained problem and the global system matrix is non-positive definite. Various solution algorithms are thus proposed and compared. Comparisons between the classical method and that developed here show that the stress formulation gives very satisfactory results in terms of the stresses.

Keywords: Contact, Friction, Equilibrium finite elements, Augmented Lagrangian, Condensation

1. Introduction

Here we present a non-classical approach for solving contact problems with friction. These problems are usually approached using the classical displacement based finite element method, with which many solution methods have been proposed and proved to be efficient. With approaches of this kind, the compatibility conditions are fulfilled, but not the local equilibrium conditions. A stress formulation for the problem, satisfying the local equilibrium conditions, is proposed, first because the stresses are usually the parameter of interest in engineering contexts, secondly with a view to using this approach for mesh refinement and limit analysis purposes. This paper is an extended and improved version of the paper [1] presented at the Fifth International Conference on Engineering Computational Technology. In the second section, the classical unilateral contact problem and its stress formulation are presented.

The method of discretizing the stress formulation presented in Section 3 involves equilibrium finite elements. The element used is described, it was found to be more suitable for contact and friction problems, since it makes it possible to impose the contact and friction conditions both strictly and simply. Some convergence results are presented in the appendix.

In Section 3, we show how equilibrium finite elements can be used to satisfy the local equilibrium conditions *a priori*, while imposing the use of degrees of freedom which are not directly stresses, prescribed forces and contact forces therefore have to be introduced using Lagrangian multipliers. The global system, which is presented in Section 4, is a non-linear problem with constraints due to the contact and friction conditions, and its matrix system is non-positive definite due to the Lagrangian multipliers involved. Various solvers are then presented: three based on condensation procedures and one on the Uzawa algorithm. In Section 5, the results obtained using the classical method and that presented here are compared. The validity of the present method is confirmed by these comparisons: it gives better results than the classical one in terms of the stresses. Lastly, the solvers presented here are compared. Various preconditioners are first compared by applying them to the stress discretization procedure. The computation times and memory requirements of each of the solvers applied to various problems are then compared.

2. The contact problem

2.1. Mechanical problem

In this section, the classical contact problem shown in Fig. 1 is presented. We consider a deformable body Ω in receding contact with a rigid foundation. The boundary Γ is split into three parts, Γ_D, Γ_F and Γ_C so that $\Gamma = \bar{\Gamma}_D \cup \bar{\Gamma}_F \cup \bar{\Gamma}_C$. We take $u = (u_i)$ to denote the displacement field, $\epsilon = \epsilon_{ij}(u) = \frac{1}{2}(u_{i,j} + u_{j,i})$ to denote the linearised strain tensor, $\sigma = \sigma_{ij}(u) = A_{ijkl}\epsilon_{kl}(u)$ to denote the stress tensor and $\alpha = A^{-1}$ to denote the flexibility tensor. The body is in contact with the foundation at Γ_C. The contact is governed by the Signorini unilateral conditions and the quasi Coulomb's friction law. At the contact boundary, the displacement and the stress vector are

* Corresponding author. Tel.: +33 491164438; fax: +33 491164481.
E-mail addresses: kuss@lma.cnrs-mrs.fr (F. Kuss), lebon@lma.cnrs-mrs.fr (F. Lebon).

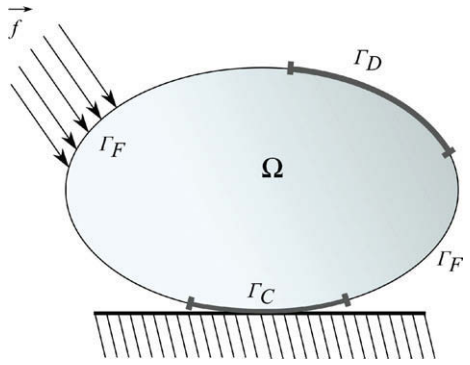

Fig. 1. The classical contact problem.

decomposed by introducing n, the outward normal unit vector to Γ:

$$\begin{aligned} &u_n = u_i n_i, \quad u_t = u - u_n \cdot n, \\ &\sigma_n = \sigma_{ij} n_i n_j, \quad (\sigma_t)_i = \sigma_{ij} n_j - \sigma_n n_i. \end{aligned} \tag{1}$$

The body is subjected to volume forces $F = (F_i)$ in Ω and to surface forces $f = (f_i)$ on Γ_F (f could possibly be zero at a part of Γ_F). It is assumed that $F \in [L^2(\Omega)]^d$ and $f \in [L^2(\Gamma_F)]^d$. The displacement is prescribed on Γ_D. The mechanical problem involving an elastic body in contact with a rigid obstacle is classically written as follows:

Problem (**P**)**.** Find u and σ such that

$$\begin{cases} \sigma = A\epsilon(u) \quad \text{in } \Omega, \\ div\,\sigma = -F \quad \text{in } \Omega, \\ u = u_0 \quad \text{on } \Gamma_D, \\ \sigma \cdot n = f \quad \text{on } \Gamma_F, \\ u_n \leqslant 0, \quad \sigma_n \leqslant 0, \quad u_n \sigma_n = 0 \ \text{ on } \Gamma_C, \\ |\sigma_t| \leqslant -\mu\sigma_n \quad \text{on } \Gamma_C \text{ and } \begin{cases} |\sigma_t| < -\mu\sigma_n \Rightarrow u_t = 0, \\ |\sigma_t| = -\mu\sigma_n \Rightarrow \exists \lambda \geqslant 0, u_t = -\lambda\sigma_t, \end{cases} \end{cases} \tag{2}$$

where μ is the friction coefficient. Classically, this problem is treated using a kinematically admissible formulation, written in terms of the displacements. Many studies have been carried out on these lines (for further details, see [2]). We adopt a statically admissible formulation of the problem presented above.

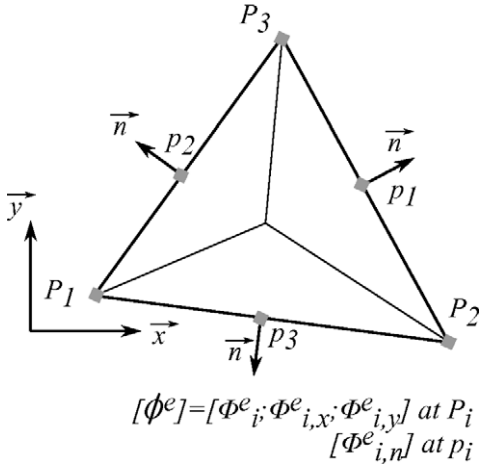

Fig. 2. The Hsieh–Clough–Tocher (HCT) element.

2.2. Dual (stress) formulation

First, we define the statically admissible sets:

$$\begin{aligned} &H = \{\sigma = (\sigma_{ij}), \sigma_{ij} = \sigma_{ji} \in L^2(\Omega)\}, \\ &H_{F,f} = \{\sigma \in H; \sigma_{ij,j} + F_i = 0 \text{ in } \Omega, \sigma_{ij} n_j = f_i \text{ on } \Gamma_F\}, \\ &\Sigma(\tau) = \{\sigma \in H_{F,f}; \sigma_n \leqslant 0, |\sigma_t| \leqslant -\mu\tau_n \text{ on } \Gamma_C\} \end{aligned} \tag{3}$$

to ensure the symmetry of the stress tensor, the local equilibrium conditions and the contact and friction conditions.

The variational formulation presented in [3] which is written in terms of the stresses and is known as the dual formulation, is used here:

Problem ($\mathbf{P_s}$). Find a stress field $\sigma : \Omega \to H$ such that

$$\begin{cases} \sigma \in \Sigma(\sigma), \\ b(\sigma, \tau - \sigma) \geqslant l(\tau - \sigma) \quad \forall \tau \in \Sigma(\sigma) \end{cases} \tag{4}$$

with $b(\sigma,\tau) = \int_\Omega A^{-1}\sigma \cdot \tau\, dx$ and $l(\tau) = \int_{\Gamma_D} u_0 \tau \cdot n\, ds$.

3. Discretization of stress formulations

3.1. Discrete formulation

In this part, which deals with the discretization of the dual problem, the problem is reduced to that of the complementary energy for the sake of simplification. The problem is then expressed without any contact or friction; it is discretized using the equilibrium finite elements method introduced by Fraeijs de Veubeke in [4] and recently used by Kempeneers et al. in [5]. The elementary stress field has to satisfy local equilibrium conditions; for this purpose, the stress field is obtained from an Airy stress function. This type of element has been used in some numerical cases: by Zavelani-Rossi [6] in a study on plane structures in the case of plasticity and cracks, by Wieckowski and et al. [7] in the elastoplastic analysis of plane structures, and by Bisegna and et al. [8,9] to solve the Signorini and Coulomb problem in the case of a plane elastic structure. In the last part, we describe how the prescribed and contact forces are accounted in this system.

The problem presented above can be written starting with the complementary energy and solved using a relaxation algorithm. We then write the complementary energy functional π as follows:

$$\pi = \frac{1}{2} b(\tau,\tau) - l(\tau). \tag{5}$$

The discretization of the problem leads to transforming the functional π into a sum of elementary functionals:

$$\begin{aligned} \pi = \sum_{e=1}^{N} \pi_e \quad \text{with}: \ \pi_e &= \frac{1}{2} b(\tau^e, \tau^e) - l(\tau^e) \quad \text{and} \\ b(\sigma^e, \tau^e) &= \int_{\Omega_e} \alpha^e_{ijkl} \sigma^e_{ij} \tau^e_{kl}\, dx_e \\ l(\tau^e) &= \int_{\Gamma_{D_e}} \tau^e_{ij} n_j u_{0i}\, ds_e \text{ if the element edge} \in \Gamma_D \end{aligned} \tag{6}$$

In each element, one has to satisfy the local equilibrium condition, to satisfy the continuity of the stress vector between two neighbouring elements K and K' in order to satisfy the local equilibrium condition on the whole structure, and to introduce the external (prescribed or contact) forces if the edge of the element is in Γ_F or Γ_C. These conditions can be written respectively as follows:

$$\vec{div}\,\overline{\overline{\tau^e}} + \overline{F} = \vec{0} \quad \text{in } K, \tag{7}$$

$$\overline{\overline{\tau^e}}\vec{n} = \overline{\overline{\tau^{e'}}}\vec{n} \quad \text{if } \Gamma_K = \Gamma_{K'}, \tag{8}$$

$$\overline{\overline{\tau^e}}\vec{n} = \vec{f} \quad \text{if } \Gamma_K \in \Gamma_F, \tag{9}$$

$$\overline{\overline{\tau^e}}\vec{n} = \vec{f_C} \quad \text{if } \Gamma_K \in \Gamma_C. \tag{10}$$

3.2. Elementary stress field

In order to reduce the computational cost, the elementary stress field interpolation is constructed in such a way that it fulfills the condition (7) *a priori*. The stress field in an element K is defined by the curl of an Airy stress function ($\Phi^e = \Phi^e(x,y)$), such that:

$$\tau^e_{\alpha\beta} = e_{\alpha\gamma}e_{\beta\delta}\Phi^e(x,y)_{,\gamma\delta} + \delta_{\alpha\beta}\psi \Rightarrow \begin{cases} \tau^e_{xx} = \Phi^e(x,y)_{,yy} + \psi, \\ \tau^e_{yy} = \Phi^e(x,y)_{,xx} + \psi, \\ \tau^e_{xy} = -\Phi^e(x,y)_{,xy}. \end{cases} \tag{11}$$

With ψ such that $Fi = -\psi_{,i}$, where Fi denotes the volume forces in the direction i. For the sake of simplicity, the problem will be written with no volume forces in the following part of the paper.

The Airy stress function is interpolated on an element by:

$$\Phi^e = [Y^e(x,y)][\phi^e], \tag{12}$$

where ϕ^e are the degrees of freedom. The stress field is then linked to the degrees of freedom by differentiating the Airy stress function:

$$[\tau^e] = \begin{pmatrix} \tau^e_{xx} \\ \tau^e_{yy} \\ \tau^e_{xy} \end{pmatrix} = [B^e][\phi^e]. \tag{13}$$

Using Airy stress functions also makes it possible to meet condition (8) very simply. This condition requires the stress vector to be continuous across elements. Since the stress vector is completely defined on one edge by the value of the Airy stress function and that of its first derivatives, C^1 regular finite elements were used to ensure the continuity of the stress vector, by performing nodal degrees of freedom assembly.

Here, the triangular composite Hsieh–Clough–Tocher (HCT) (Fig. 2) element is used. This element is built using three Airy subfunctions which allow to define three linear stress fields on each sub-element. The elementary stress field is therefore linear in parts and defined by 12 degrees of freedom. it can be shown that using this element also simplifies the procedure used to impose the contact and friction conditions, since the stress vector is linear on the edges of the element.

3.3. Contact and prescribed forces

At this point, conditions (9) and (10) have not yet been imposed. Since the degrees of freedom are not forces, this cannot be done directly; the stress vector is obtained by projecting the stress field onto an element edge:

$$T(\vec{M(s)}, n) = [n][B^e(s)][\phi^e], \tag{14}$$

where s is the coordinate of a point M on the edge and $[n]$ is the projection matrix associated with the outward normal vector $\vec{n}$ of the edge:

$$[n] = \begin{bmatrix} l & 0 & m \\ 0 & m & l \end{bmatrix} \quad \text{with } \vec{n} = \begin{vmatrix} l \\ m \end{vmatrix} \text{ in (x, y)}. \tag{15}$$

We then introduce the force vector $[T^e]$. To completely define the stress vector on one edge, one has to apply its value at different points on an element edge, the number of which depends on the degree of the stress interpolation function. In the case of HCT, since both components are linear, four values are required.

These values are then placed on the force vector $[T^e]$, which can be directly related to the degrees of freedom by the force interpolation matrix $[C^e]$:

$$[T^e] = [C^e][\phi^e]. \tag{16}$$

Conditions (9) and (10) are then directly imposed by introducing the vector $[F^e]$ (respectively $[F^e_C]$), which gives the discrete values of the external forces f prescribed on the element edges (contact and friction forces f_C, respectively):

$$[F^e] = [C^e_F][\phi^e], \tag{17}$$

$$[F^e_C] = [C^e_C][\phi^e]. \tag{18}$$

The external forces are then prescribed by fixing the values of $[F^e]$.

3.4. Elementary problem

The integral $l(\tau^e)$ is discretized as follows:

$$l(\tau^e) = \int_{\Gamma_{D_e}} ([\tau^e][n])^T[u]ds_e = [\phi^e]^T \int_{\Gamma_{D_e}} [C^e]^T[N]^T[u]ds_e, \tag{19}$$

where $[N]$ is the displacement interpolation matrix and $[u]$ the nodal displacements. Using Gauss integration formulas, we obtain:

$$l(\tau^e) = [\phi^e]^T[q^e]. \tag{20}$$

The flexibility matrix $[S^e]$ is obtained by Gauss integration and the elementary functional becomes:

$$\pi_e = \frac{1}{2}[\phi^e]^T[S^e][\phi^e] - [\phi^e]^T[q^e], \tag{21}$$

$$[F^e_F] = [C^e_F][\phi^e] \quad \text{if } \Gamma_K \in \Gamma_F, \tag{22}$$

$$[F^e_C] = [C^e_C][\phi^e] \quad \text{if } \Gamma_K \in \Gamma_C. \tag{23}$$

4. Global system and solution

Having defined the elementary system, the global system is now obtained directly by assembly and written:

$$\pi = \frac{1}{2}[\phi]^T[S][\phi] - [\phi]^T[q], \tag{24}$$

$$[C_F][\phi] = [F] \text{on} \Gamma_F, \tag{25}$$

$$[C_C][\phi] = [F_C] \text{on} \Gamma_C. \tag{26}$$

The last two conditions are included using Lagrangian multipliers. We obtain the following augmented complementary energy functional:

$$\pi^* = \frac{1}{2}[\phi]^T[S][\phi] - [\phi]^T[q] + [\lambda]([C_F][\phi] - [F]) + [\lambda']([C_C][\phi] - [F_C]). \tag{27}$$

Cancelling the first variation of this functional with respect to the variables ϕ, λ, λ', and F_C gives the global matrix system:

$$\begin{bmatrix} S & C_F^T & C_C^T & 0 \\ C_F & 0 & 0 & 0 \\ C_C & 0 & 0 & -I \\ 0 & 0 & -I & 0 \end{bmatrix} \cdot \begin{bmatrix} \phi \\ \lambda \\ \lambda' \\ F_C \end{bmatrix} = \begin{bmatrix} q \\ F \\ 0 \\ 0 \end{bmatrix}. \tag{28}$$

At this point, the contact and friction conditions have to be applied. In the primal case, when dealing with displacements, the matrix is positive definite and the system can be solved using relaxation algorithms. Here the matrix is not positive definite. We therefore compared two ways of solving this system: by using a Gauss–Seidel algorithm to solve a smaller condensed system, which can be obtained using various condensation strategies, and by solving the problem, given by the augmented Lagrangian formulation, using the Uzawa algorithm.

The size and structure of each matrix contribute importantly to the efficiency and the memory requirements of the method. Let n_ϕ be the number of degrees of freedom used for the discretization

procedure, and n_f be the number of prescribed forces (which usually amount to less than 20% of n_ϕ in many contact problems) and n_C be the number of contact forces (which usually amount to less than 10% of n_ϕ in many contact problems). Table 1 summarises the characteristics of each matrix and gives storage schemes that can possibly be used.

4.1. Condensation on the contact boundary

The goal of the condensation procedure is to obtain the reduced (condensed) system:

$$[D][F_C] = [G] \tag{29}$$

to be solved with a relaxation algorithm. This condensed system can be obtained using various methods: the "blind condensation" method, or condensation using either the SYMMLQ solver or a three-step solver.

4.1.1. Blind condensation

The first method of obtaining this system consists in applying the basic idea of condensation, starting with system (28) and taking the first three groups of unknowns as the internal unknowns ϕ_i. The symbolic system to be reduced is therefore:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & 0 \end{bmatrix} \cdot \begin{bmatrix} \phi_i \\ F_C \end{bmatrix} = \begin{bmatrix} U_i \\ 0 \end{bmatrix}. \tag{30}$$

By solving:

$$\phi_i = M_{11}^{-1} U_i - M_{11}^{-1} M_{12} F_C \tag{31}$$

the following condensed system is obtained:

$$M_{21} M_{11}^{-1} M_{12} F_C = M_{21} M_{11}^{-1} U_i \quad \text{denoted } [D][F_C] = [G], \tag{32}$$

where

$$M_{11} = \begin{bmatrix} S & C_F^T & C_C^T \\ C_F & 0 & 0 \\ C_C & 0 & 0 \end{bmatrix}. \tag{33}$$

Since the computational cost of inversing M_{11} could be very high, (31) can be solved using a linear solver: M_{11} is factorised once and ϕ_i is obtained by $(1 + n_C)$ triangular matrix multiplications. Since the matrix M_{11} is not positive definite and cannot be stored using an efficient storage scheme, the first method is very costly and it has not been used here.

4.1.2. SYMMLQ condensation

Since the slowness of the above condensation strategy is due to the inefficiency of the direct solvers applied to non-positive definite matrices, the SYMMLQ iterative method was used to solve the previous $(1 + n_C)$ systems. The SYMMLQ method has been presented in [12]: it is a conjugate gradient like method for solving symmetric indefinite linear systems. It solves the projected system and keeps the residual vectors orthogonal to all the previous ones. The efficiency of this algorithm depends heavily on the choice of the preconditioner. An efficient diagonal block preconditioner is used in [13]. Let us simplify the notation of (33) as follows:

$$M_{11} = \begin{bmatrix} S & C^T \\ C & 0 \end{bmatrix}. \tag{34}$$

The preconditioner matrix P is the block diagonal matrix:

$$P = \begin{bmatrix} S^* & 0 \\ 0 & -T \end{bmatrix}, \tag{35}$$

where S^* is an approximation for S. Various preconditioners were tested in this study: incomplete Cholesky factorizations of S, the SSOR preconditioner and, since the SYMMLQ solver is used $(1 + n_C)$ times during the condensation process, complete Cholesky factorization of S was also tested. The second block preconditioner used here is such that $T = CS^{*-1}C^T$.

4.1.3. Three-step condensation

An efficient condensation method was developed to be able to take the structure of each matrix into account. Note that in what follows, the brackets will be omitted from the matrices to simplify the notations. The matricial system (28) can be written as a system of four equations as follows:

$$\begin{cases} S\phi + C_F^T\lambda + C_C^T\lambda' = q, \\ C_F\phi = F, \\ C_C\phi = F_C, \\ I\lambda' = 0. \end{cases} \tag{36}$$

The condensation procedure involves three main steps:

Step 1: Evaluating ϕ so that:

$$\phi = S^{-1}q - S^{-1}C_F^T\lambda - S^{-1}C_C^T\lambda' \tag{37}$$

can be achieved in two ways:

- Using a direct solver: S is factorized once by performing LDL^T factorization with skyline storage and the products $S^{-1}q$, $S^{-1}C_F^T$ and $S^{-1}C_C^T$ are obtained by $(1 + n_f + n_C)$ system resolutions. The costliest part of this step is therefore carried out only once.
- Using a conjugate gradient (CG) algorithm: an efficient preconditioner is computed once and used to quickly obtain the products given above as the result of the $(1 + n_f + n_C)$ system resolutions.

ϕ is then introduced into lines 2 and 3 of (36).

Step 2: The system then becomes:

$$\begin{cases} C_FS^{-1}q - C_FS^{-1}C_F^T\lambda - C_FS^{-1}C_C^T\lambda' = F, \\ C_CS^{-1}q - C_CS^{-1}C_F^T\lambda - C_CS^{-1}C_C^T\lambda' = F_C, \\ I\lambda' = 0. \end{cases} \tag{38}$$

λ is obtained by performing $(1 + n_C)$ resolutions of a full and symmetric n_f by n_f matrix. Here again, the matrix is factorised only once.

Step 3: λ is then introduced into line 2 of (38) and λ' is obtained in the same way (by performing $(1 + n_C)$ resolutions of a full and symmetric n_C by n_C matrix). Lastly, λ' is introduced into the last line of (38) and the condensed system is obtained. Note that the last two steps involve full and symmetric matrices, but since n_f and n_C are small in comparison with n_ϕ, their factorization is not too costly.

4.1.4. Preconditioners

Since the efficiency of iterative methods depends strongly on the preconditioning procedure, the following preconditioners have been used to accelerate both the SYMMLQ and CG methods.

Table 1
Main characteristics of matrices.

Matrix	Size	Storage
S	n_ϕ by n_ϕ	Symmetric (skyline or sparse)
C_F	n_f by n_ϕ	Sparse
C_C	n_C by n_ϕ	Sparse

4.1.5. Incomplete Cholesky factorization

When direct solvers are used, the symmetric positive definite matrix S is decomposed into a lower and an upper matrix so that:

$$S = LL^T. \tag{39}$$

The aim of incomplete Cholesky factorization is to find an approximation for L named L^*. The classical incomplete Cholesky factorisation strategies are reviewed in [14]. We keep only the most important terms of L in L^*. For this purpose, two methods are used:

- The drop tolerance factorization strategy:The term $L^*(i,j)$ is kept only if: $L^*(i,j) \geqslant 10^{-c} * \sqrt{L^*(i,i) * L^*(j,j)}$.
- The filling level factorization strategy:The largest $n_{fl} * n_i$ terms in line i of L^* are kept, where n_i is the size of line i of L and n_{fl} is a parameter on which the filling level of L^* depends.

With both methods, two strategies can be used, and the least important terms of L^* can be removed during the factorization procedure or removed after performing the full Cholesky factorization procedure. The former strategy is usually used in order to minimise the computational cost of L^* and the memory requirements, but this sometimes makes it necessary to add small diagonal terms δ to S. Since the preconditioner is used many times in the condensation process, the second strategy can be a useful means of maximizing the efficiency of the preconditioner, although the computational cost of the preconditioner will be greater in this case.

In order to compare those two strategies, the first one was applied to the filling level factorization strategy, and the second one to the drop tolerance factorization strategy.

4.1.6. SSOR preconditioner

The matrix S is decomposed so that:

$$S = D - E - E^T, \tag{40}$$

Until convergence:

For $i = 1$ to n_f:

$$F_C^* = \frac{1}{D(i,i)}[G(i) - \sum_{j=1}^{i-1} D(i,j)F_C(j) - \sum_{j=i+1}^{n_f} D(i,j)F_C(j)]$$

- **If** $F_C(i)$ is normal: **If** $F_C^* \geq 0$

  - **Then**: $F_C(i) = 0$
  - **Else**: $F_C(i) = F_C^*$

- **If** $F_C(i)$ is tangent: **If** $|F_C^*| > \mu |F_{C_N}|$

  (Where F_{C_N} is the normal force corresponding to $F_C(i)$)

  - **Then**: $F_C(i) = \mu \ sign(F_C^*)|F_{C_N}|$
  - **Else**: $F_C(i) = F_C^*$

End of loop

Convergence if i_C^k small

End of loop

Fig. 3. The Gauss–Seidel algorithm.

where D is the diagonal of S and $-E$ is its lower triangular part. The preconditioner S^* is defined by:

$$S^* = \frac{1}{\omega(2-\omega)}(D - \omega E)D^{-1}(D - \omega E)^T, \tag{41}$$

where ω is the relaxation parameter, which is taken to range between 0 and 2.

4.2. Condensed system solver

The condensed system obtained using one of the above condensation methods is solved using a Gauss–Seidel algorithm, and it makes it possible to apply contact and friction conditions throughout the iterations. This algorithm is presented in Fig. 3. The global resolution time used by this algorithm usually amount to less than 1%. The effects of ordering F_C and that of the value of the friction coefficient on the convergence of the algorithm are not very large and are not presented here, since they do not significantly affect the overall computation time. It is assumed that the convergence has been obtained by defining the indicator of convergence i_C^k at step k:

$$i_C^k = \sup_i \left\{ 0; \frac{\|F_C^k(i) - F_C^{k-1}(i)\|}{\|F_C^k(i)\|} \right\} \quad \text{with } F_C^k(i) \neq 0 \tag{42}$$

to determine at what point no further changes in the contact forces vector occur. The limit value of i_C^k has been taken to be 10^{-5}, which is the usual choice with algorithms of this kind [15].

4.3. Augmented Lagrangian formulation

Another method of solving (27) consists in transforming this equation to obtain an augmented Lagrangian formulation. The functional then becomes:

$$\pi_a* = \frac{1}{2}\phi^T S\phi - \phi^T q + \lambda(C_F\phi - F) + \lambda'(C_C\phi - F_C) + \frac{r}{2}(C_F\phi - F)^2 + \frac{r}{2}(C_C\phi - F_C)^2 \quad \text{with } F_C \in \Sigma(F_C), \tag{43}$$

where r is the penalty parameter. Cancelling the first variation of this functional in comparison with the variables ϕ, λ, λ' and F_C gives the system:

$$\begin{cases} \tilde{S}\phi = \tilde{U} \\ \quad \text{with} : \tilde{S} = S + rC_F^TC_F + rC_C^TC_C \\ \quad \text{and} : \tilde{U} = q - \lambda C_F - \lambda' C_C + rC_F^TF + rC_C^TF_C \\ C_F\phi - F = 0 \\ C_C\phi - F_C = 0 \\ \lambda' - r(C_C\phi - F_C) = 0 \text{with} F_C \in \Sigma(F_C) \end{cases} \tag{44}$$

In order to apply the contact and friction conditions, the Uzawa algorithm presented in Fig. 4 was used, where ρ is the Uzawa's step. Note that the augmented Lagrangian formulation was used to improve the convergence rate of the Uzawa algorithm. Since many iterations are usually required to obtain convergence, the first step in this algorithm is performed using a direct skyline solver. The factorization of $\tilde{S}$, which is the costliest operation, is thus performed only once and the factorised matrix is used at each step in the algorithm. The convergence indicator was defined such that:

$$i_C^k = sup_i \|r^k(i)\| \tag{45}$$

with

$$r^k = C_C\phi^k - F_C^{k+1} + C_F\phi^k - F \tag{46}$$

the limit value of i_C was taken to be 10^{-3} MPa, which is an appropriate value with the examples treated here.

Initial guess of ϕ, λ, λ', and F_C

Until convergence, iteration k:

1- Determination of $\tilde{U}^k(\lambda^k, \lambda'^k, F_C^k)$

2- Determination of ϕ^k by solving $\tilde{S}\phi^k = \tilde{U}^k$

3- Update of F_C:

$$F_C^{k+1} = P_{\Sigma(F_C)}(C_C\phi^k - \tfrac{\lambda' k}{2})$$

4- Update of λ and λ':

$$\lambda^{k+1} = \lambda^k + \rho(C_F\phi^k - F)$$

$$\lambda'^{k+1} = \lambda'^k + \rho(C_C\phi^k - F_C)$$

Convergence if i_C^k small

End of loop

Where $P_{\Sigma(F_C)}$ is the projection operator in the Signorini Coulomb friction cone.

Fig. 4. The Uzawa algorithm.

5. Numerical results

The algorithms presented here have been implemented in the computer code LMGC90 (http://www.lmgc.univ-montp2.fr/~dubois/LMGC90/) and tested on the case of the steel tooth presented in Fig. 5, with displacements $u_x = 0$ cm and $u_y = -0.01$ cm prescribed on its left edge, which is in contact with a rigid foundation where $\mu = 0.2$.

We first compare the accuracy of the results obtained using the method presented here and those obtained using the classical displacement based method, and we then compare the efficiency of each of the solvers tested.

5.1. Validity of the method

In order to check the validity of the present dual method, we compare the stress fields obtained using this method with the stress fields obtained using the classical displacement based finite element method on various homogeneous meshes.

For this purpose, we compute the relative error e, defined as:

$$e = \sqrt{\frac{\int_\Omega (\sigma - \sigma_{ref}) A^{-1} (\sigma - \sigma_{ref}) d\Omega}{\int_\Omega \sigma_{ref} A^{-1} \sigma_{ref}\, d\Omega}}, \tag{47}$$

where σ_{ref} is the reference solution's stress field, which is computed using the classical displacement finite element method on a reference refined mesh.

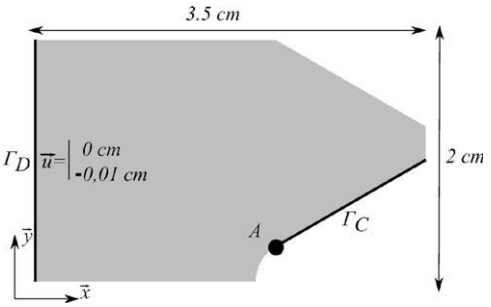

Fig. 5. First example: contact between a tooth and a rigid foundation.

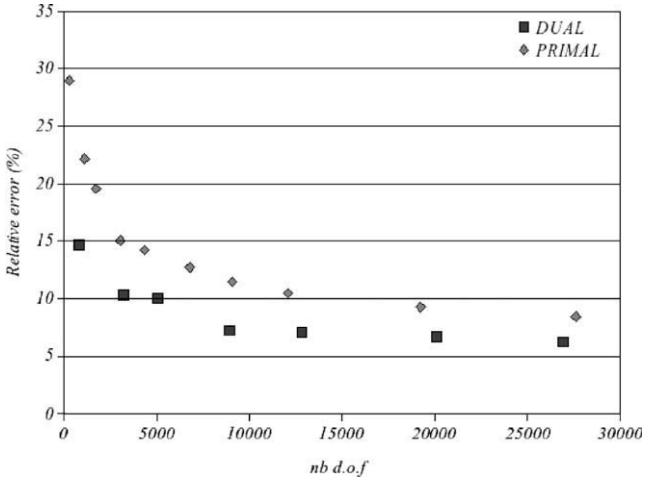

Fig. 6. Comparison between the relative errors obtained using dual and primal methods.

Here, all the displacement based finite element solutions are obtained using the computer code LMGC90, which is based on the NSCD solver [16], with linear triangular elements.

The relative error e is computed by performing numerical integration on the Gauss points of the reference mesh. The evolution of the error is shown in Fig. 6 in the case of the present method (named DUAL) and the displacement based method (named PRIMAL) with various homogeneous meshes.

These results clearly show that both methods converge towards the same reference solution. Comparisons on the relative error obtained with the DUAL and PRIMAL methods with an equivalent number of degrees of freedom show that the present method gives more accurate results in terms of the stresses in the case of the example tested.

5.2. Comparisons between numerical methods

5.2.1. Descriptions of benchmarking examples and algorithms parameters

In this section, we test the present methods of resolution on the examples given in Figs. 5 (Mesh0 to Mesh3) and 7 (Mesh4 to Mesh7) with the various sizes presented in Table 2. This second

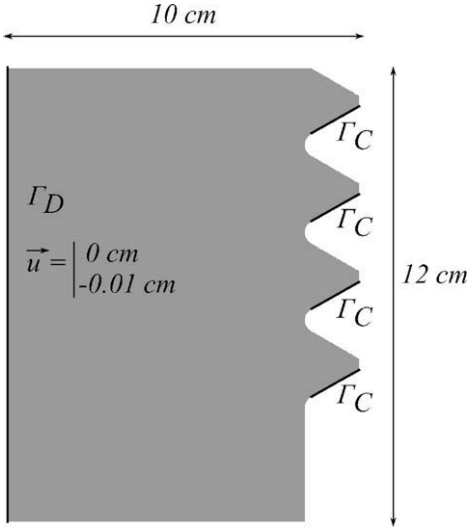

Fig. 7. An example with four teeth.

Table 2
Sizes of the examples tested.

Name	$n\phi$	n_f	n_C	Name	$n\phi$	n_f	n_C
Mesh0	761	96	38	Mesh4	8360	392	200
Mesh1	1973	144	80	Mesh5	19128	588	480
Mesh2	3105	176	120	Mesh6	26538	708	576
Mesh3	6287	232	160	Mesh7	50483	920	640

example is similar to the first one, but involves four separated contact boundaries and different ratios $\frac{n_\phi}{n_f}$ and $\frac{n_\phi}{n_C}$ when not particularly refined meshes are used. All the meshes were previously renumbered using the Cuthill McKee algorithm (see [17]) and the matrices were equilibrated using diagonal equilibration method to improve the convergence rates and the accuracy.

The following methods were then compared:

- The SYMMLQ condensation strategy.
- The three-step condensation method using two different approaches at the first level of condensation: a skyline LDL^T solver and a conjugate gradient solver. These two methods will be named 3STEPD and 3STEPCG below.
- Solving the augmented Lagrangian formulation using the Uzawa algorithm.

With the CG and SYMMLQ methods, the following approximations S^* of S were tested as preconditioners:

- The drop tolerance incomplete Cholesky factorization (DTIC) procedure with the tolerance parameter $c \in [3;6]$.
- The filling level incomplete Cholesky factorization (FLIC) procedure with the filling level factor $n_{fl} \in [1;5]$.
- The SSOR preconditioner (SS) with the relaxation parameter $\omega \in]0;2[$.
- Exact Cholesky factorization of S (SKY).

When DTIC, FLIC and SSOR preconditioners were used, the upper and lower parts of these preconditioners were stored using sparse line storage schemes. Both parts were stored in order to accelerate the matrix vector products. When the exact Cholesky factorization procedure was used, both the upper and lower parts were stored for the same reason and a skyline storage line scheme was used. When using the skyline LDL^T solver, both the lower ($-E$) and upper ($-E^T$) parts of S were stored using a skyline storage line scheme. Since the computation of line i of L does not require the $(i-1)$ lines of S, matrices L and L^T can be stored in $-E$ and $-E^T$, respectively.

To use Uzawa's algorithm, one has to choose the parameters ρ and r defined in Section 4.3. Since the parameter r defines the importance of the product matrices $C_F^T C_F$ and $C_C^T C_C$ in comparison with S, matrices C_F and C_C have been multiplied by factors p_F and p_C so that:

$$p_F = \frac{\sqrt{mean(S)}}{mean(C_F)}, \qquad p_C = \frac{\sqrt{mean(S)}}{mean(C_C)}. \tag{48}$$

This makes it possible to give any r the same weight in all the examples tested. On the other hand, since we had no rules for choosing ρ and r, we tested various combinations of these two parameters in $[1;20;40;60;80;100;200;300]$ in each case and took the best sets for the subsequent comparisons.

These computations were performed on an Intel P4 1.7 Ghz computer with 1 Gb RAM memory in order to assess the performances on a medium-range computer.

5.2.2. Influence of the preconditioner

Before comparing these methods, we analyzed the influence of preconditioners on the SYMMLQ and 3STEPCG methods. Since similar behaviour was found to occur in each case and with both methods, we focused on results obtained with the SYMMLQ method on the Mesh3 example given in Table 3. This table gives the name of each preconditioner, the value of the parameter (c with DTIC, n_{fl} with FLIC and ω with SSOR), the time taken to compute the preconditioner matrix, the time required to perform the condensation, the time required to complete the entire resolution (condensation and Gauss–Seidel methods), the size that would be used by the lower part of the full Cholesky factorization preconditioner, the size of the lower part of the preconditioner, and the maximum RAM memory used.

First, if we look at the DTIC, it can be seen that DTIC3 gives a good compromise between memory requirements and speed, and that DTIC6 gives the best results, since it is similar, in terms of size, to the full Cholesky factorization. It is worth noting that the RAM value of DTIC6 is greater than the one of SKY because that the skyline storage scheme is lighter than the sparse storage scheme, when dealing with the same number of components.

Table 3
Results obtained using various preconditioners when the SYMMLQ method was applied to case of Mesh3 (6287 dof).

Name	Parameter	Tprec (s)	Tcond (s)	Ttotal (s)	Full size (nb terms)	Lprec size (nb terms)	Maxmem (Mb)	δ
DTIC2	2	5.92	258.66	258.72	1264705	171548	31.84	
DTIC3	3	6.61	91.50	91.56	1264705	353749	36.66	
DTIC4	4	9.73	58.54	58.60	1264705	655518	43.58	
DTIC5	5	13.45	43.50	43.56	1264705	1005769	51.59	
DTIC6	6	15.52	41.10	41.16	1264705	1204640	56.14	
FLIC1	1	5.99	546.54	546.60	1264705	80018	29.74	3.5×10^{-2}
FLIC2	2	5.98	292.95	293.01	1264705	159908	31.64	5×10^{-3}
FLIC3	3	13.11	258.99	259.05	1264705	239097	33.31	5×10^{-3}
FLIC4	4	19.23	285.38	285.44	1264705	317324	35.85	5×10^{-3}
FLIC5	5	24.65	315.80	315.86	1264705	393895	37.60	5×10^{-3}
SKY		6.91	14.50	14.56	1264705	1264705	47.87	
SS02	0.2	2.46	1571.28	1571.34	1264705	80018	30.67	
SS04	0.4	2.61	1258.93	1258.99	1264705	80018	30.67	
SS06	0.6	2.64	1081.68	1081.74	1264705	80018	30.67	
SS08	0.8	2.62	975.84	975.90	1264705	80018	30.67	
SS10	1	2.60	921.31	921.37	1264705	80018	30.67	
SS12	1.2	2.65	926.26	926.32	1264705	80018	30.67	
SS14	1.4	2.70	984.78	984.84	1264705	80018	30.67	
SS16	1.6	2.78	1136.87	1136.93	1264705	80018	30.66	
SS18	1.8	2.88	1457.94	1458.00	1264705	80018	30.66	

The last column gives the value δ added to the diagonal of S when the FLIC preconditioner was used. This procedure involves adding $\Delta\delta$ to the diagonal when the algorithm fails to compute the preconditioner and the computation has to be restarted. The times given in Table 3 take these restarts into account and $\Delta\delta = 0.005$. It can be seen that the strategy which consists in taking the $(i-1)$ first lines of L^* to compute the ith line usually involves restarts and therefore does not make for any computational time savings.

The optimum parameter ω used for the SSOR preconditioner was 1 in all the cases tested here. This preconditioner gives a good approximation for S, but it is less efficient than the incomplete Cholesky preconditioner when working on problems of this kind.

The SKY preconditioner stored on a skyline scheme eventually gave the best results. First since choosing the exact factorization procedure for S gives an excellent approximation of P, the block preconditioner; the SYMMLQ algorithm converges in 1 or 2 iterations with this preconditioner. Secondly, the skyline storage scheme plays an important role: to confirm this point, the DTIC preconditioner was tested with a threshold value $c = 10^{-21}$; this preconditioner therefore gives very similar results to those obtained with the full Cholesky preconditioner. But due to the sparse storage scheme involved, the SYMMLQ method takes longer with the DTIC preconditioner than with the SKY preconditioner.

5.2.3. Comparisons between algorithms

In the previous section, efficient preconditioners were selected in order to reduce the number of methods compared. In this part, we use only the DTIC and SKY preconditioners with iterative methods. The results obtained with the following methods on all the examples are compared in Figs. 8 and 9:

- The SYMMLQ condensation method with the DTIC3, DTIC6 and SKY preconditioners, which was named SDTIC3, SDTIC6 and SSKY, respectively.
- The 3STEP condensation method with the direct LDL^T and the conjugate gradient solver, using the DTIC3 and DTIC6 preconditioners, which was named 3STEPD, 3STEPCG3 and 3STEPCG6, respectively.
- The solution of the augmented Lagrangian formulation using the Uzawa algorithm, which was named ALM.

The computational times are summarized in Fig. 8.

The memory requirements of all the methods tested here are summarized in Fig. 9.

First, by comparing SDTIC3 with 3STEPCG3 and SDTIC6 with 3STEPCG6, it can be seen that the SYMMLQ method and the 3STEPCG method are both equivalent when equivalent preconditioners are used, but these four strategies are less efficient than the others. As a general rule, iterative methods can only be as efficient as the 3STEPD and ALM methods when highly efficient preconditioners are used, as in the case of the SSKY strategy. But these methods involve the storage of both the global matrix and the preconditioner; they therefore require a greater amount of memory than the 3STEPD and ALM methods.

The 3STEPD and ALM methods therefore seem to be the most efficient of these methods. In all the examples tested, these two methods required similar computational times, although the ALM method was found to be slightly faster.

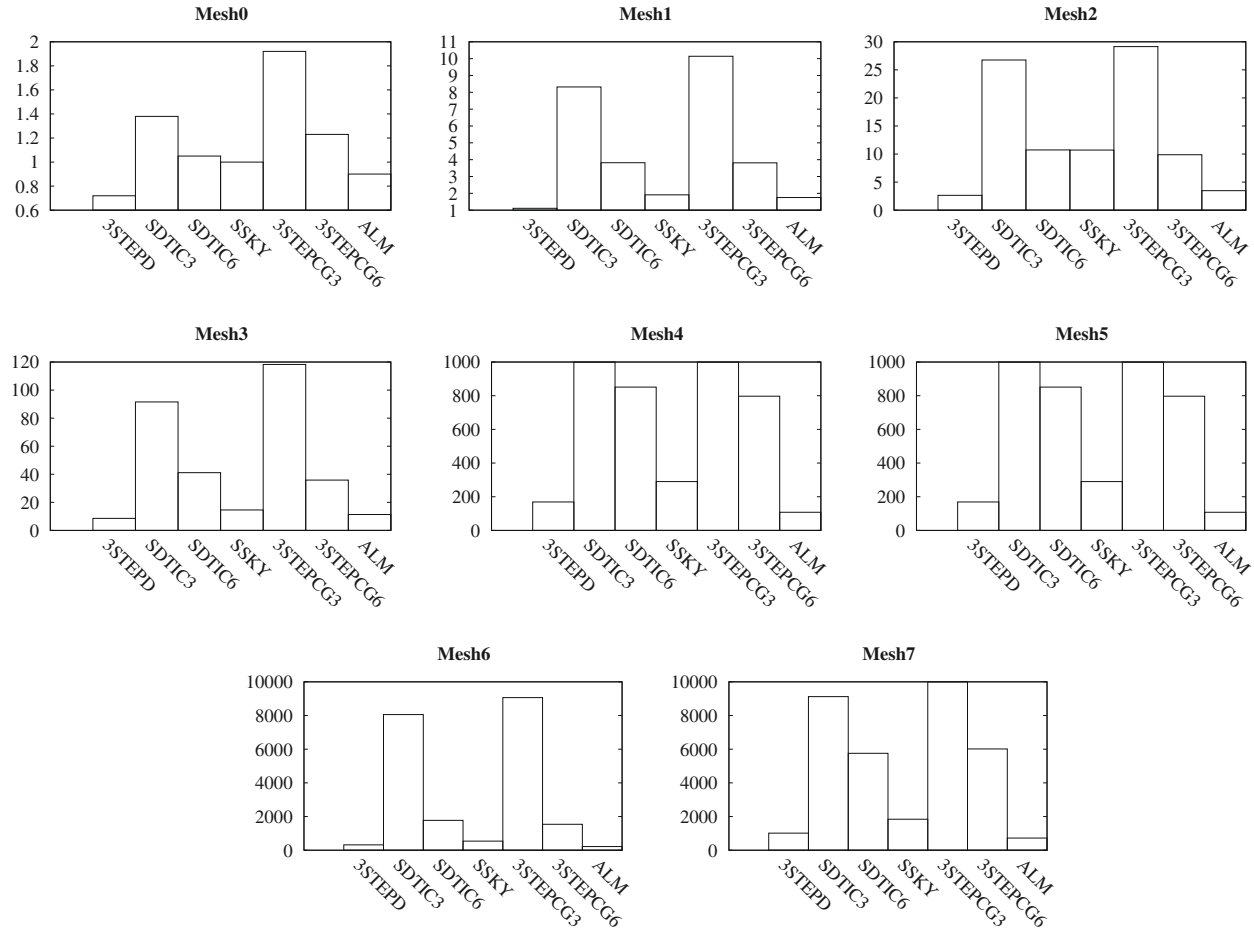

Fig. 8. Comparisons between computational times (in s).

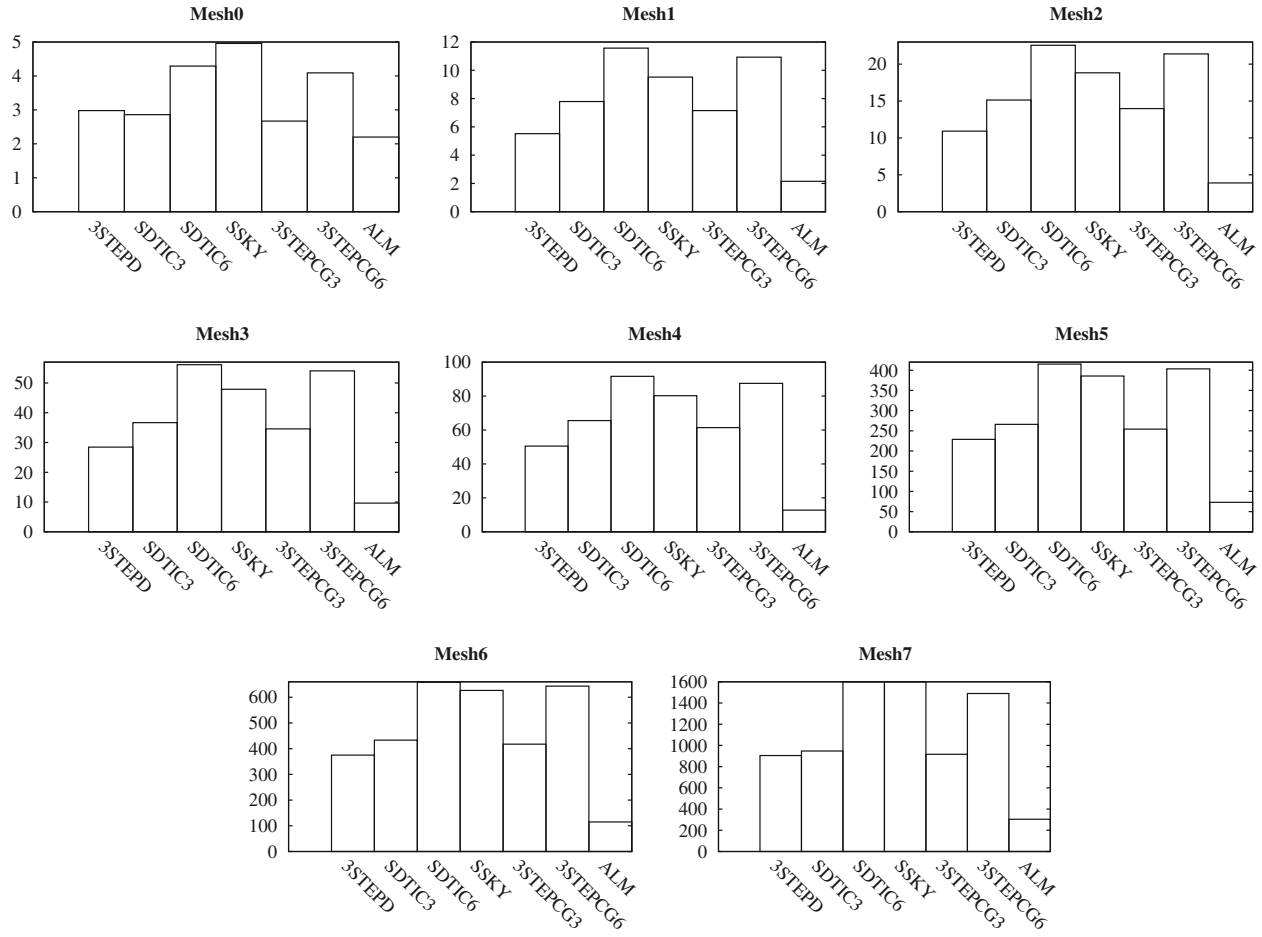

Fig. 9. Comparisons between memory requirements (in Mb).

The main advantage of the ALM method is that it requires very little memory, whereas the 3STEPD methods involve the storage of intermediate matrices at each of the condensation steps. This drawback of the 3STEPD method could be overcome using an appropriate file storage strategy.

The main advantage of the 3STEPD method is its robustness, since unlike the ALM method, it is not parameter dependent. The computation times are therefore predictable, since they depend mainly on the profile of the flexibility matrix. On the other hand, this method can easily be parallelized.

6. Conclusion

In this paper, an adapted version of the equilibrium finite element method is presented for solving contact and friction problems. With the element presented, the interpolated stress fulfills the equilibrium conditions *a priori*.

The use of equilibrium finite elements requires the prescribed and contact forces to be introduced into the matrix system using Lagrangian multipliers. The matrix system is thus non-positive definite and the global system is non-linear and constrained. Various solution algorithms based on condensation and on an augmented Lagrangian formulation have been proposed to solve this particular system.

The validity of the equilibrium finite element method was confirmed in the case of various examples. Results obtained in the case of a tooth using both the classical displacement based finite element method and that proposed here were compared in the last section. The finite element method presented here gives more accurate results in terms of the stresses than the classical one. This can be explained by the fact that the contact and friction conditions are directly imposed on the stress field in the case of the present method.

Lastly, all the numerical methods developed so far for solving the global system were compared. Two of these methods turned out to be particularly efficient. The first one, which is based on the use of a suitable condensation strategy and of direct solvers, is very robust but needs some memory management. The second method, which is based on the Uzawa algorithm applied to an augmented Lagrangian formulation requires little memory, but some difficult parameters have to be determined to make it efficient.

Appendix A. Some convergence results

In this paragraph, some convergence results obtained using the finite elements method in a simplified framework are presented. We introduce the space

$$W = H(div;\Omega) = \{\tau \in H; div\,\tau \in [L^2(\Omega)]^2\}. \tag{49}$$

To simplify the problem, the domain Ω is assumed to be polygonal. We take $\mathbb{T}_h$ to denote a triangulation of Ω such that $\Omega = \cup_{T\in\mathbb{T}_h} T$. $\mathbb{T}_h$ is assumed to be regular, so that all the angles of the element $T \in \mathbb{T}_h$ are bounded far from zero and there exists a positive constant α such that the length of any side of any $T \in \mathbb{T}_h$ is at least αh, where h is the characteristic length of the elements.

In a general framework, spaces W_h and V_h are chosen so that:

if $\tau \in W_h$ and $(div\,\tau, v_h) = 0 \quad \forall v_h \in V_h$ then $div\,\tau = 0$ in Ω. (50)

In addition, we assume that it is possible to construct an interpolation operator $\pi_h : W \to W_h$ fulfilling:

$$(div\,\pi_h \sigma v_h) = (div\,\sigma, v_h) \quad \forall v_h \in V_h. \tag{51}$$

An element is defined in keeping with the above hypothesis. For all $T \in \mathbb{T}_h$, the following finite dimensional space of piecewise linear stress tensors is defined:

$$W_T \subset H(div; T) \tag{52}$$

and we write

$$W_h = \{\tau \in \hat{W}_h; div\,\tau \in [L^2(\Omega)]^2\}, \tag{53}$$

where

$$\hat{W}_h = \{\tau \in H; \tau/_T \in W_T \ \forall T \in \mathbb{T}_h\}. \tag{54}$$

We now adopt some restrictive hypotheses. F (the weight, for example) is assumed to be constant and f (the linear force distribution) to be linear. Here we take the case of Tresca friction: in problems (P_s) and (P_d^*), the sets $\Sigma(\sigma)$ and $\Sigma^*(\sigma_n)$ are replaced by the sets $\Sigma(\beta)$ and $\Sigma^*(\beta_n)$, respectively, where β and β_n are given. β (resp. β_n) are assumed to be concave or piecewise linear. Using a classical result obtained by Ciarlet [10], one can obtain.

Elasticity: there exists a constant C independent of h such that:

$$\|\sigma - \sigma_h\|_W \leqslant Ch\|\sigma\|. \tag{55}$$

Tresca (P_s) there exists a constant C independent of h such that:

$$\|\sigma - \sigma_h\|_W \leqslant Ch^{1/2}\|\sigma\|_2. \tag{56}$$

Tresca (P_d^*) there exists a constant C independent of h such that:

$$\|\sigma - \sigma_h\|_{-1/2,\Gamma_2} \leqslant Ch^{1/2}\|\sigma\|_2. \tag{57}$$

These results are proved in [11].

Comment: In the case of the Signorini–Coulomb problem, the hypotheses adopted in theorem (7.1) are generally not satisfied. It is therefore necessary to use external approximations [10]. This problem still remains to be solved.

A.1. Accuracy of the Block preconditioner

In this section, we explain why the Block preconditioner presented in Section 4.1.2 is very accurate when the full Cholesky factorization method is used as the first block of the preconditioner. Let us first recall the condensation principle. To obtain expression (31), one has to solve a series of systems:

$M_{11}\tilde{U}_i = U_i$and
$M_{11}X = M_{12}$ treated by n_C operations :
do $i = 1$ to n_C :
$M_{11}X(:,i) = M_{12}(:,i)$ (where $X(:,i)$ denotes the column i of X)
enddo

(58)

These $(1 + n_C)$ systems can be solved using the SYMMLQ method with the block preconditioner (35). When the full Cholesky factorization procedure is used as the first block of the preconditioner, one of the n_C systems solved by applying the preconditioner (35) is:

$$\begin{bmatrix} S & 0 \\ 0 & -T \end{bmatrix} \begin{bmatrix} X_1(:,i) \\ X_2(:,i) \end{bmatrix} = \begin{bmatrix} M_{12}^1(:,i) \\ M_{12}^2(:,i) \end{bmatrix}, \tag{59}$$

which is equivalent to:

$$\begin{cases} SX_1(:,i) = M_{12}^1(:,i), \\ -CS^{-1}C^TX_2(:,i) = M_{12}^2(:,i). \end{cases} \tag{60}$$

The exact system is:

$$\begin{bmatrix} S & C^T \\ C^T & 0 \end{bmatrix} \begin{bmatrix} X_1(:,i) \\ X_2(:,i) \end{bmatrix} = \begin{bmatrix} M_{12}^1(:,i) \\ M_{12}^2(:,i) \end{bmatrix}, \tag{61}$$

which is equivalent to:

$$\begin{cases} SX_1(:,i) = M_{12}^1(:,i) - C^TX_2(:,i), \\ -CS^{-1}C^TX_2(:,i) = M_{12}^2(:,i). \end{cases} \tag{62}$$

Note that this could be a two-step condensation strategy.

With this choice of preconditioner, systems (60) and (62) can be seen to be very similar; the SYMMLQ method therefore converges after only one or two iterations when this preconditioner is used.

References

[1] Kuss F, Lebon F. Dual methods for unilateral contact problems, Proceedings of the Fifth International Conference on Engineering Computational Technology. In: Topping BHV, Montero G, Montenegro R, editors. Proceedings of the fifth international conference on engineering computational technology. Stirlingshire, United Kingdom: Civil-Comp Press; 2006. paper 106.
[2] Wriggers P. Computational contact mechanics. Wiley; 2002.
[3] Telega JJ. Quasi-static Signorini's contact problem with friction and duality. Int Series Numer Math 1991;101:199–214.
[4] Fraeijs de Veubeke B. Displacement and equilibrium models in the finite element method. Int J Numer Meth Eng 2001;52:287–342.
[5] Kempeneers M, Beckers P, Moitinho de Almeida JP, Almeida Pereira OJB. Modéles équilibre pour l'analyse duale. Revue européenne des éléments finis 2003;12:737–60 [in french].
[6] Zavelani-Rossi A. An equilibrium approach to plane problems. Comput Struct 2001;79:1877–95.
[7] Wieckowski Z, Youn SK, Moon BS. Stress based finite element analysis of plane plasticity problems. Int J Numer Meth Eng 1999;44:1505–25.
[8] Bisegna P, Lebon F, Maceri F. Relaxation procedures for solving Signorini–Coulomb contact problems. Adv Eng Software 2004;35:595–600.
[9] Bisegna P, Lebon F, Maceri F. D-PANA: a convergent block-relaxation solution method for the discretized dual formulation of the Signorini–Coulomb contact problem. Comptes Rendus Mathématiques Académie des Sciences Paris 2001;333:1053–8.
[10] Ciarlet PG. The finite element method for elliptic problems. North Holland Publishing Company; 1979.
[11] Capatina A, Lebon F. editor Mihailescu-Suliciu M, Remarks on the equilibrium finite element method for frictional contact problems, New Trends in Continuum Mechanics, Theta, 25-33, 2005.
[12] Paige CC, Saunders MA. Solution of sparse indefinite systems of linear equations. SIAM J Numer Anal 1975;12:617–29.
[13] Corral C, Giménez I, Tur M, Ródenas JJ. Iterative preconditioned methods for the solution of contact problems by the finite element method. In: Topping BHV, Montero G, Montenegro R, editors. Proceedings of the fifth international conference on engineering computational technology. Stirlingshire, United Kingdom: Civil-Comp Press; 2006. paper 87.
[14] Jones MT, Plassmann PE. An improved incomplete Cholesky factorization. ACM Trans Math Software 1995;21:5–17.
[15] Glowinski R, Lions JL, Tremoliéres R. Numerical analysis of variational inequalities. North-Holland; 1981.
[16] Jean M. The non-smooth contact dynamics method. Comput Meth Appl Mech Eng 1999;177:235–57.
[17] Lascaux P., Théodor R. Analyse numérique matricielle appliquée á l'art de l'ingénieur, tomes 1 et 2, Dunod, 2004.