NUMERICAL SOLUTION OF LINEAR ELASTICITY PROBLEMS WITH INTERSECTING SLIDE SURFACE CONSTRAINTS *

CHARLES TONG †

Abstract.
Incorporating slide surface constraints in implicit structural finite element analyses using Lagrange multipliers give rise to an indefinite linear system of equations which is difficult to solve by iterative methods. To improve the iterative solution process, algebraic elimination that restores positive definiteness to the discretization matrix has been shown to be effective. This paper extends the algebraic elimination techniques to applications with intersecting slide surfaces. We present a graph-theoretic algorithm for handling intersecting slide constraints and compare its performance with inexact Uzawa and block preconditioned Krylov methods.

Key words. indefinite systems, preconditioning

AMS subject classifications. 65F10, 65N20

1. Introduction. The implicit solution of linear elasticity problems using finite elements involves solving the following variational problem:

$$\mathbf{F}(\mathbf{u}) = \frac{1}{2}(\mathbf{Ku}, \mathbf{u}) - (\mathbf{f}, \mathbf{u}), \quad \mathbf{u} \subset \mathbf{R}^N \tag{1}$$

where the functional $\mathbf{F}(\mathbf{u})$ is to be minimized in $\mathbf{R}^N$ (N is the degree of freedom). Here $\mathbf{K}$, $\mathbf{u}$, and $\mathbf{f}$ are the stiffness matrix, the displacement and load vectors, respectively.

When the computational domain consists of several bodies, the "impenetrability" condition governs that two bodies cannot occupy the same space at the same time. To enforce this condition, an additional set of constraint equations

$$\mathbf{Cu} = \mathbf{g} \tag{2}$$

is introduced, where $\mathbf{C}$ is the constraint matrix and $\mathbf{g}$ is a vector containing the constraint values (for example, $\mathbf{g} = \mathbf{0}$ implies the bodies are in contact).

Using the Lagrange multiplier formulation, the modified functional that includes the constraints can be written as

$$\hat{\mathbf{F}}(\mathbf{u}) = \frac{1}{2}(\mathbf{Ku}, \mathbf{u}) - (\mathbf{f}, \mathbf{u}) + \lambda^T(\mathbf{Cu} - \mathbf{g})$$

where λ are the Lagrange multipliers.

The minimizer of this modified functional is the solution of the indefinite system

$$\begin{bmatrix} \mathbf{K} & \mathbf{C^T} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}. \tag{3}$$

While solving this indefinite system with direct methods poses few additional numerical difficulties, its computational and memory requirements becomes prohibitively large for even moderate-sized problems. For large scale applications, the use of iterative methods is inevitable. However, iterative methods suffer from significantly slower

*This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551. E-mail : chtong@llnl.gov.

convergence rates (compared to similar problems but without constraints) and thus become less attractive.

Methods for solving the indefinite systems generally fall into several categories: iterative methods applied to the original system with special preconditioning (for example, block preconditioning), iterative methods applied to reduced systems which are symmetric and positive definite, projection methods, and Uzawa methods [25]. In our previous paper [7] we have demonstrated the effectiveness of algebraic elimination techniques for problems with simple constraints. In this paper we extend the algebraic elimination techniques to handle problems with more complicated intersecting slide surface constraints.

Section 2 discusses the derivation of slide surface constraints. In Section 3 we survey solution methods for indefinite systems. Section 4 presents the new algorithm for algebraic elimination. Section 5 describes an implementation and a performance comparison of the method with the inexact Uzawa and block preconditioned iterative methods. Finally, in section 6, we conclude with a few final remarks.

2. Slide Surface Constraints.

2.1. The Physics of Slide Surfaces. Slide surfaces refer to the interface between two disjoint bodies that may come into contact in a simulation. Impact and separation of these bodies must be detected, and the bodies may slide tangentially relative to each other, with or without friction. The slide surface boundary conditions are different in each case of impact, separation, and sliding. Here we discuss only the simplest case where two bodies in contact remain in contact and slide relative to each other during the simulation (that is, $\mathbf{g} = \mathbf{0}$ in equation 2).

Two important conditions must be satisfied at the interface: conservation of momentum and an "impenetrability" constraint preventing the structures from occupying the same space at the same time [7]. These criteria can be satisfied by using Lagrange multipliers at the interface. In this case, the constraint is exactly the "impenetrability" condition, while the Lagrange multipliers themselves are the forces necessary to maintain momentum conservation.

2.2. Formulating the Slide Surface Constraints. The impenetrability condition can be prescribed by first selecting one side, the "slave" side, to apply the constraint (see Figure 1).

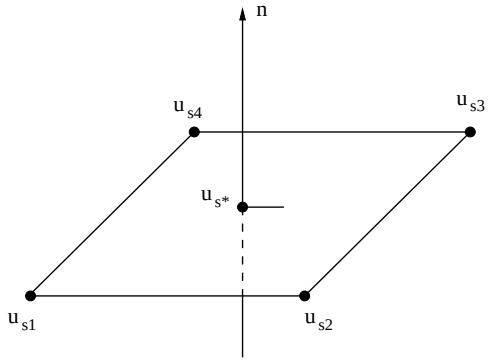

FIG. 1. *Example of a slide surface*

An iterative algorithm is then used to compute the "images" (or parametric coordinates) of the slave nodes on the other side of the interface, the "master" side.

Next, for the image of each slave node s, the nearest face of a finite element f_s on the master surface is identified. Let $\mathbf{u}_{s1}$, $\mathbf{u}_{s2}$, $\mathbf{u}_{s3}$, and $\mathbf{u}_{s4}$ be the vector displacements of the four nodes defining f_s for hexahedral elements. Then the displacement of the slave node s's image $\mathbf{u}_{s*}$ is constrained by the master side to be:

$$\mathbf{u}_{s*} = \phi_{s1}\mathbf{u}_{s1} + \phi_{s2}\mathbf{u}_{s2} + \phi_{s3}\mathbf{u}_{s3} + \phi_{s4}\mathbf{u}_{s4}$$

where ϕ_{s1}, ϕ_{s2}, ϕ_{s3}, and ϕ_{s4} are weights constructed from the parametric coordinates of node s and the four nodes of f_s using finite element shape functions.

Let $\mathbf{u}_s$ be the actual displacements of the slave node s. Then the impenetrability condition for s is given by ($\mathbf{g}_s = 0$ for objects in contact):

$$\mathbf{n}^T(\mathbf{u}_s - \mathbf{u}_{s*}) = \mathbf{g}_s = 0.$$

Physically, this constraint states that the displacement of the slave node and its image on the master surface should be identical in the direction normal to the f_s.

Substituting $\mathbf{u}_{s*}$ into the constraint equation yields:

$$\mathbf{n}^T(\mathbf{u}_s - \phi_{s1}\mathbf{u}_{s1} - \phi_{s2}\mathbf{u}_{s2} - \phi_{s3}\mathbf{u}_{s3} - \phi_{s4}\mathbf{u}_{s4}) = 0.$$

We point out that this constraint is in general nonlinear since the normal vector is a function of the slave node coordinates, and the shape functions depend on both the master and slave node coordinates. In this analysis, we linearize this constraint by evaluating the normal and the shape functions using the displacements computed in the previous time step.

Each constraint equation is thus a linear combination of the displacements of one slave node and one or more master nodes, and the number of constraint equations is the number of slave nodes to be constrained. The constraint equations can be represented in the following matrix form:

$$\mathbf{Cu} = \mathbf{0}.$$

(Again we point out that the right-hand side is nonzero if the surfaces are not originally in contact.) In three-dimensional problems, the constraint equations represents a small percentage of the total number of equations in the system, and $\mathbf{C}$ is sparse.

An important observation is that for simple slide surfaces (that is, non-intersecting ones) each column of $\mathbf{C}$ corresponding to a slave variable only has one nonzero in the column, implying that no other constraints depend on or involve this variable. This observation is the basis for the algebraic elimination described later.

2.3. Intersecting Slide Surfaces. The elegant feature of the simple slide surface constraints is that we can define one of the slave displacements (x, y, or z) as a "dependent" variable. The algebraic elimination technique can be applied in a straightforward manner once the algebraic relationships between the dependent and independent variables have been established.

For general constraints, it is not always possible to find a straightforward partitioning of the constrained variables into disjoint dependent and independent sets. An example is the intersecting slide surface problems where some nodes are master nodes for one slide surface, and slave nodes for another slide surface; or it may be slave nodes for both slide surfaces. This is illustrated in Figure 2. Arrows in the figure indicate that a slave node is dependent on a master node. In this example, node 3

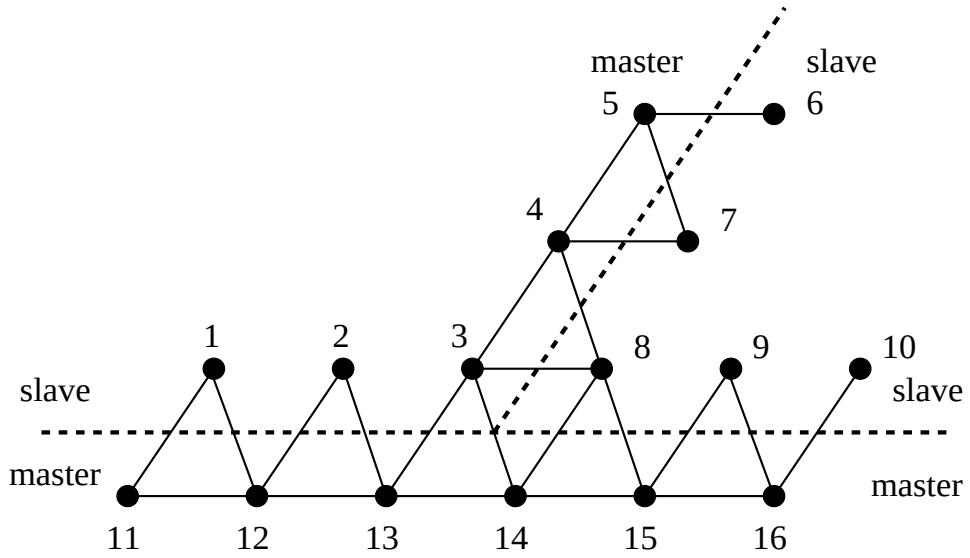

FIG. 2. *Example of an intersecting slide surface*

is a slave node on one slide surface but a master node on another, while node 8 is a slave node on both slide surfaces.

The intersecting slide surfaces in Figure 2 has 9 constraints, since there are 8 slave nodes, and one of them (node 8) is a slave to 2 master surfaces. The sparsity pattern of $\mathbf{C}$ for this set of interface nodes is (each X and 0 are row vectors of dimension 3 for three-dimensional problem):

$$(4)\ \mathbf{C} = \begin{bmatrix} X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & 0 & 0 & 0 & 0 \\ 0 & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & X & X & 0 & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

3. Solution Methods for Indefinite Systems. Indefinite systems in general are difficult to solve by iterative methods such as the class of preconditioned Krylov subspace methods. Specifically, the efficiency of Krylov methods depends on the use of effective preconditioners. Many effective preconditioners have been proposed for symmetric and positive definite system, but effective preconditioners for general indefinite systems are still relatively scarce. In this section we give a brief review of existing methods for solving indefinite systems, with emphasis given to methods that are amenable to parallel implementation.

3.1. Algebraic Elimination (of dependent variables) Approaches. The constraint equations implicitly prescribe relationships between some of the displacement variables, making it possible to eliminating them conveniently. The idea is to identify a set of "dependent" variables which can be expressed as a linear function

of the other or "independent" variables. The elimination step gives rise to a reduced system involving only the independent variables. With algebraic elimination the system matrix is restored to be symmetric and positive definite if the stiffness matrix ($\mathbf{K}$ in equation 1) is symmetric and semi-definite. The reduced system can be solved efficiently by the conjugate gradient method with a wide selection of effective preconditioners. For general saddle point systems, this reduction is not always possible since a pure set of dependent variables (that is, in the absence of slave nodes that show up on more than one slide surfaces) may not always be found. More details of this approach will be given in Section 4.

3.2. Krylov Methods with Block Preconditioning. The linear system in equation 1 is often called a saddle point problem, and it pervades in many scientific and engineering applications such as incompressible fluid flow, structural analysis, constrained optimization, etc. One popular way of solving large-scale saddle point problems is the use of block preconditioning coupled with an indefinite Krylov solvers such as MINRES, quasi-minimal residual (QMR), or flexible generalized minimal residual (FGMRES) algorithms [22, 12, 26].

The main challenge is to develop effective preconditioners for the indefinite systems. Block preconditioning techniques have been considered by Murphy, Golub, and Wathen [21]; Little and Saad [19], Keller, Gould, and Wathen [18]; Klawonn [20], Rozložník and Simoncini [23], Rusten and Winther [24]; and many others. Specifically, block diagonal, block triangular, and block LU preconditioners in the forms of

$$\mathbf{P}_D = \begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}\hat{K}^{-1}\mathbf{C}^T \end{bmatrix}, \mathbf{P}_T = \begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{C}^T \\ \mathbf{0} & \mathbf{C}\hat{K}^{-1}\mathbf{C}^T \end{bmatrix}, \quad \text{and}$$

$$\mathbf{P}_{LU} = \begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{C} & -\mathbf{C}\hat{K}^{-1}\mathbf{C}^T \end{bmatrix} \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{K}}^{-1}\mathbf{C}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

have been investigated where $\mathbf{S} = \mathbf{C}\hat{\mathbf{K}}^{-1}\mathbf{C}^T$ is the approximate Schur complement and $\tilde{\mathbf{K}}^{-1}$ is an approximate inverse of $\mathbf{K}$ ($\hat{\mathbf{K}}^{-1}$ may be a different approximate inverse different from $\tilde{\mathbf{K}}^{-1}$). The main difficulty with this class of preconditioners is the formation or the inverse application of $\mathbf{S}$. More accurate approximation of the Schur complement incurs higher preconditioning costs, while less accurate approximation may cause convergence problems. In addition, many such techniques require the factorization of some approximates of $\mathbf{K}$, which may not be efficient nor amenable to parallel implementation.

3.3. Projection Methods. The goal of the projection methods, as with the algebraic elimination technique, is to transform the indefinite system into a positive definite system, with the advantage that the classical conjugate gradient algorithm can be applied. There are various forms of the projection method. One projection method [11, 25] computes the solution by solving the reduced system

$$\mathbf{K}_p\mathbf{u}_p = (\mathbf{P}^T\mathbf{K}\mathbf{P})\mathbf{u}_p = \mathbf{P}\mathbf{f} = \mathbf{f}_p. \tag{5}$$

One choice for $\mathbf{P}$ is the projection operator $\mathbf{P} = \mathbf{I} - \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C}$ where $\mathbf{C}$ has been defined before in 1. The procedure for this algorithm is thus

1. Compute $\mathbf{P} = \mathbf{I} - \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C}$;
2. Solve $\mathbf{K}_p\mathbf{u}_p = \mathbf{f}_p$ for $\mathbf{u}_p$;

3. Recover the solution $\mathbf{u} = \mathbf{P}^T\mathbf{u}_p$.
4. Compute the solution $\lambda = (\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C}(\mathbf{f} - \mathbf{K}\mathbf{u} + \mathbf{u})$.

There are other choices of the projectors (for example, [27], [15]), which are formed by different combinations of $\mathbf{C}$. Observe that computation using the projector above may require the factorization of $\mathbf{C}\mathbf{C}^T$, which may have efficiency problem especially on parallel computers.

3.4. Uzawa Algorithms. The exact and inexact Uzawa algorithms [2, 10] are popular for solving saddle point problems. In the following we introduce the preconditioned Uzawa iteration as well as its variants.

The Preconditioned Uzawa Algorithm : Let λ^0 be given. Compute $\mathbf{u}^n$ and λ^n using the recurrence: for $k = 0, 1, 2, \cdots, n$

1. $\mathbf{K}\mathbf{u}^{k+1} = \mathbf{f} - \mathbf{C}^T\lambda^k$
2. $\lambda^{k+1} = \lambda^k + \omega\mathbf{M}^{-1}(\mathbf{C}\mathbf{u}^{k+1} - \mathbf{g})$

It can be shown that the preconditioned Uzawa algorithm on $\mathbf{A}$ is equivalent to the classical Uzawa method applied to the Schur complement with splitting $\mathbf{C}^T\mathbf{K}^{-1}\mathbf{C} = \mathbf{M} - \mathbf{N}$ for some nonsingular $\mathbf{M}$.

The Inexact Uzawa Algorithm : Let λ^0 be given. Compute $\mathbf{u}^k$ and λ^k using the recurrence: for $k = 0, 1, 2, \cdots, n$

1. $\tilde{\mathbf{K}}\mathbf{u}^{k+1} = \mathbf{f} - \mathbf{C}^T\lambda^k$
2. $\lambda^{k+1} = \lambda^k + \omega\mathbf{M}^{-1}(\mathbf{C}\mathbf{u}^{k+1} - \mathbf{g})$

where $\tilde{\mathbf{K}}^{-1}$ is an approximate inverse of $\mathbf{K}$ and $\mathbf{M}^{-1}$ is an approximate inverse of the Schur complement $\mathbf{S}$.

The rate of convergence of this algorithm depends on how accurately the inner iterations are solved. Specifically, the error at the $(i+1)$-th iteration $\mathbf{e}_{i+1}$ is related to $\mathbf{e}_i$ by

$$\mathbf{e}_{i+1} = \begin{bmatrix} \mathbf{e}^x_{i+1} \\ \mathbf{e}^y_{i+1} \end{bmatrix} \begin{bmatrix} \mathbf{I} - \tilde{\mathbf{K}}^{-1}\mathbf{K} & \tilde{\mathbf{K}}^{-1}\mathbf{C}^T \\ \mathbf{M}^{-1}\mathbf{C}(\mathbf{I} - \tilde{\mathbf{K}}^{-1}\mathbf{K}) & \mathbf{I} - \mathbf{M}^{-1}\mathbf{C}\tilde{\mathbf{K}}^{-1}\mathbf{C}^T \end{bmatrix} \begin{bmatrix} \mathbf{e}^x_i \\ \mathbf{e}^y_i \end{bmatrix} = \mathbf{G}\mathbf{e}_i. \quad (6)$$

Convergence of the method requires that the magnitude of all eigenvalues of the iteration matrix $\mathbf{G}$ be smaller than 1. Many convergence proofs have been given previously for the inexact Uzawa algorithm and its variants [3, 4, 28, 10, 6]. Practical application of the Uzawa algorithm often requires the approximation of the extreme eigenvalues of the Schur complement, which may be expensive to compute. Other approaches solves the exact Schur complement with the conjugate gradient method to a prescribed accuracy, and convergence proofs have been established [10, 17]. However, each conjugate gradient iteration for solving the Schur complement requires an exact solve with $\mathbf{K}$, and thus the overall cost may become prohibitive. Next, we describe a modified Uzawa algorithm as a preconditioner to FGMRES, in which case convergence criterion is less stringent.

Krylov Method with Modified Uzawa Algorithm : use FGMRES with the following preconditioning step: [17]

1. $\mathbf{u}^{k+1/2} = \mathbf{u}^k + \tilde{\mathbf{K}}^{-1}(\mathbf{f} - \mathbf{A}\mathbf{u}_i - \mathbf{C}^T\lambda^k)$,
2. $\lambda^{k+1/2} = \lambda^k + \mathbf{M}^{-1}(\mathbf{C}\mathbf{u}^{k+1/2} - \mathbf{g})$,
3. $\mathbf{u}^{k+1} = \mathbf{u}^k + \tilde{\mathbf{K}}^{-1}(\mathbf{f} - \mathbf{A}\mathbf{u}_i - \mathbf{C}^T\lambda^{k+1/2})$,
4. $\lambda^{k+1} = \lambda^{k+1/2} + \mathbf{M}^{-1}(\mathbf{C}\mathbf{u}^{k+1} - \mathbf{g})$.

It was observed that using this modified Uzawa preconditioner gives faster convergence for our test problems. It can be shown that this preconditioner is

equivalent to the following block preconditioner (with the Schur complement approximated by a scaled identity $\alpha\mathbf{I}$):

$$(7)\quad \mathbf{P}^{-1}_{uzawa} = \begin{bmatrix} \tilde{\mathbf{K}}^{-1}(\mathbf{I} + \alpha\mathbf{C}^T\mathbf{C}\tilde{\mathbf{K}}^{-1}) & -\alpha\tilde{\mathbf{K}} \\ \alpha\mathbf{C}\tilde{\mathbf{K}}^{-1}(2\mathbf{I} - \alpha\mathbf{C}^T\mathbf{C}\tilde{\mathbf{K}}^{-1}) & \alpha^2\mathbf{C}\tilde{\mathbf{K}}^{-1}\mathbf{C}^T - (1+\alpha)\mathbf{I}. \end{bmatrix}$$

4. Algebraic Eliminations for Slide Surface Constraints. With the slide surface constraints, the linear system to be solved in each implicit solution step is:

$$(8)\quad \mathbf{Au} = \begin{bmatrix} \mathbf{K} & \mathbf{C^T} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} = \mathbf{b}.$$

Since this paper concerns the simple case where the bodies are always in contact, the right hand side for the constraint equations $\mathbf{g} = \mathbf{0}$. Since $\mathbf{A}$ is indefinite, this linear system is in general difficult to solve using iterative methods. In this section we first review an algebraic solution method for reducing equation 8 into a positive definite system. More details can be found in [7]. The main focus, however, is on how to apply the algebraic elimination technique to problems with intersecting slide surfaces.

4.1. Algebraic Elimination for Simple Slide Surfaces. We define a set of M dependent variables consisting of one displacement variable (x, y, or z) from each slave node by $\mathbf{u}_s$. All the other variables are denoted $\mathbf{u}_m$. We then partition $\mathbf{A}$ into three blocks containing the set of independent variables, the set of dependent variables, and the Lagrange multipliers variables, respectively, giving

$$(9)\quad \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{ms} & \mathbf{C_m^T} \\ \mathbf{K}_{sm} & \mathbf{K}_{ss} & \mathbf{C_s^T} \\ \mathbf{C}_m & \mathbf{C}_s & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_m \\ \mathbf{u}_s \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f}_m \\ \mathbf{f}_s \\ \mathbf{0} \end{bmatrix}.$$

Using the constraint equations, the slave variables can be expressed in terms of the master variables by

$$\mathbf{u}_s = -\mathbf{C}_s^{-1}\mathbf{C}_m\mathbf{u}_m.$$

Substituting this relationship into the master and the slave equations and then eliminating the Lagrange multiplier variables yields:

$$\begin{aligned} \mathbf{A}_m\mathbf{u}_m &= [\mathbf{K}_{mm} - \mathbf{K}_{ms}\mathbf{C}_s^{-1}\mathbf{C}_m - \mathbf{C}_m^T\mathbf{C}_s^{-T}\mathbf{K}_{sm} + \mathbf{C}_m^T\mathbf{C}_s^{-T}\mathbf{K}_{ss}\mathbf{C}_s^{-1}\mathbf{C}_m] \\ &= \mathbf{f}_m - \mathbf{C}_m^T\mathbf{C}_s^{-T}\mathbf{f}_s \\ &= \mathbf{b}_m. \end{aligned}$$

This reduced system has the property that $\mathbf{A}_m$ is symmetric and positive definite if $\mathbf{K}$ is symmetric and positive definite. Another property is that $\mathbf{A}_m$ is relatively sparse if $\mathbf{A}$ is sparse and $\mathbf{C}_s^{-1}$ is also sparse. For linear systems arising from simple slide surfaces, $\mathbf{C}_s$ is a diagonal matrix, and we have shown [7] that the corresponding reduced system is only slightly denser than $\mathbf{A}$.

In matrix notation, this reduction is equivalent to forming the Schur complement with respect to $\begin{bmatrix} \mathbf{K}_{ss} & \mathbf{C}_s \\ \mathbf{C}_s^T & \mathbf{0} \end{bmatrix}$

$$\mathbf{A}_m = \mathbf{K}_{mm} - \begin{bmatrix} \mathbf{K}_{ms} & \mathbf{C}_m \end{bmatrix} \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{C}_s \\ \mathbf{C}_s^T & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K}_{sm} \\ \mathbf{C}_m^T \end{bmatrix}. \tag{10}$$

A critical observation is that the matrix inverse in equation 10 can be formed analytically by

$$\begin{bmatrix} \mathbf{K}_{ss} & \mathbf{C}_s \\ \mathbf{C}_s^T & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_s^{-T} \\ \mathbf{C}_s^{-1} & -\mathbf{C}_s^{-1}\mathbf{K}_{ss}\mathbf{C}_s^{-T} \end{bmatrix},$$

and, with $\mathbf{C}_s$ a diagonal matrix for simple constraints, this inverse is both easy to compute and sparse. After the reduced system is solved for $\mathbf{u}_m$, $\mathbf{u}_s$ and λ can be recovered by back substitution using

$$\mathbf{u}_s = -\mathbf{C}_s^{-T}\mathbf{C}_m\mathbf{u}_m, \quad \text{and}$$

$$\lambda = \mathbf{C}_s^{-1}(\mathbf{f}_s - \mathbf{K}_{sm}\mathbf{u}_m) + \mathbf{C}_s^{-1}\mathbf{K}_{ss}\mathbf{C}_s^{-T}\mathbf{C}_m\mathbf{u}_m.$$

An important step in the algebraic elimination is to identify a set of slave variables to be eliminated along with the Lagrange multiplier variables. While this information is available from a finite element software package, it is often not available when only the matrix and right hand side vector are provided to the solver package. Fortunately, using the fact that the slave candidates can be selected from the columns of $\mathbf{C}$ having only one nonzero entry, we can easily identify M slave variables.

4.2. Algebraic Elimination for Intersecting Slide Surfaces. In cases where slide surface intersect, we can no longer identify a set of slave variables

$$\mathbf{u}_s = -\mathbf{C}_s^{-1}\mathbf{C}_m\mathbf{u}_m$$

such that $\mathbf{C}_s$ is a diagonal matrix. In the following we will show that it is possible to identify a set of dependent variables such that $\mathbf{C}_s$ is block diagonal with small block sizes. In this case, the inverse and therefore the Schur complement in equation 10 are still sparse, although they will be denser than if $\mathbf{C_s}$ is diagonal.

Several implementation details of slide surfaces contribute to the feasibility of this approach. First, in many practical problems, the number of slide surfaces that intersect each other is no more than three so that each slave variable appears in at most two constraints. This will ensure that the long-chained master-slave relationship, which can result in large connected sets of slave variables and thus large blocks, cannot occur (in fact, it can be shown that this will also imply that each subgraph consisting of a connected set of selected variables will have the longest path length between any two vertices to be no more than 3). Secondly, the "master" side is typically chosen to be the side that is more rigid or more densely gridded. This selection criterion has the advantage that the number of slave variables that depend on the same master variable is small (at most four in our implementation) and bounded, implying that relatively small block sizes can be achieved. Finally, for three-dimensional problems, each slave node has three displacement variables, allowing more flexibility in selecting slave variables.

In the following subsections we address the problem of identifying a set of M slave variables given a $(N+M)\times(N+M)$ matrix A with M constraint equations at the bottom of the matrix.

4.2.1. A Graph Theoretic Algorithm for Slave Selections. In this section, we develop a graph theoretic framework for partitioning $\mathbf{C}$ such that $\mathbf{C}_s$ is block diagonal with small blocks.

Define a $M \times N$ binary matrix $\mathbf{Q}$ such that $(\mathbf{Q})_{ij} = 1$ if $\mathbf{C}_{ij} \neq 0$ (that is, for the example given in equation 4, form $\mathbf{Q}$ by replacing all X's with 1's). Next construct a binary graph $\mathbf{G} = \mathbf{Q}^T\mathbf{Q}$ (that is, $\mathbf{G}_{ij} = 1$ if $\sum_{k=1}^{n} \mathbf{Q}(k,i)\mathbf{Q}(k,j) \neq 0$.) Several remarks can be made about $\mathbf{G}$.

1. $\mathbf{G}_{ij} \neq 0$ if and only if vertex i and j are involved in the same constraint.
2. There are M cliques in $\mathbf{G}$, each representing one constraint.
3. If an independent set of M vertices exists in $\mathbf{G}$, then $\mathbf{Q}$ can be partitioned into $[\mathbf{Q}_1\mathbf{Q}_2]$ such that $\mathbf{Q}_2$ is a diagonal matrix.

The objective is to select a total of M vertices, and at least one vertex from each clique as slaves, while minimizing the sizes of each connected set of selected vertices. This can be achieved by solving an 0-1 nonlinear integer programming problem which is both complicated to formulate and solve. Fortunately, the problem can be much simplified by incorporating one realistic assumption given before, namely that each selected subtree has the longest path between any two vertices of no more than 3. This simplified case can be cast into the following integer programming problem: let $\mathbf{v}$ be a binary vector of size N. Find $\mathbf{v} \in \{0,1\}^N$ such that

$$\min_{\mathbf{v}} \| \mathbf{Gv} \|_\infty \tag{11}$$

$$\text{subject to} \quad \sum_{i=1}^{N} \mathbf{v}_i = M, \quad \text{and}$$

$$\sum_{j=1}^{N} \mathbf{Q}_{ij}\mathbf{v}_j \geq 1, \quad i = 1, \cdots, M$$

This simplified problem is still expensive to solve. In the following subsections we use heuristics to find an approximate solution to this problem by incrementally search the solution space for a suboptimal solution. We first present the overall algorithm.

Algorithm findSlaves

isSize = 1, initial independent set $\mathbf{I} = \emptyset$
while cardinality(I) $< M$ do
    FindConstrMaximalIndepSet($\mathbf{G}, \mathbf{I}, M$,isSize++)
end while

Here the function FindConstrMaximalIndepSet takes the graph $\mathbf{G}$ and an existing independent set $\mathbf{I}$, and attempt to find or modify $\mathbf{I}$ so that the maximum value of $\mathbf{Gv}$ is no more than isSize. The arithmetic complexity of the algorithm is $O(N)$.

4.2.2. Selection of a Constrained Maximally Independent Set. The first step in the algorithm is the selection of a constrained maximally independent set (MIS), meaning a maximal set consisting of vertices that are not adjacent to each other in $\mathbf{G}$ and one vertex per constraint. This can be achieved by first identifying all columns of $\mathbf{Q}$ that have a single nonzero entry in the corresponding columns as candidates, and then examine each candidate to determine if it should be added to the set $\mathbf{I}$. It is straightforward to verify that with this choice the entries of the vector $\mathbf{Gv}$ has a maximum value of 1. In the case of a single slide surface, this step would have been enough to find an independent set of cardinality M.

To maintain numerical stability, the criterion for determining the slave variable to a given constraint is that its corresponding matrix entry has the largest magnitude among all slave candidates available for the constraint.

4.2.3. Modification of the Constrained MIS. Subsequent steps create maximally independent sets of connected vertices such that the size of each connected set is bounded (by isSize). This can be achieved by first identifying all columns of $\mathbf{Q}$ that have isSize or less nonzero entries and which are adjacent to some selected vertices. Each unprocessed constraint will have its slave candidates examined to see if its inclusion into existing connected sets will keep sizes of the set to be bounded by isSize. This process goes on until one slave vertex is identified for each constraint.

Again numerical stability is an important consideration here. Therefore, when a slave candidate is examined for inclusion into the selected set, the resulting block diagonal matrix is examined to see if it is singular or near singular, and the candidate that gives the largest departure from singularity is selected.

4.2.4. Hybrid Approach to Saddle Point Problems. It will be demonstrated algebraic elimination gives much faster convergence than the block methods applied directly to the indefinite system. However, we recognize that this algebraic elimination has its own limitations. Specifically, for more general constraints it may not be possible to form block diagonal $\mathbf{C_s}$ with small block size. In such cases algebraic elimination can be combined with other approaches (for example, block-precondtioned Krylov method) to achieve better performance.

5. Numerical Experiments. We have implemented a sequential version of our algebraic elimination method. In this section we present numerical results demonstrating the effectiveness of this method compared to the block-preconditioned and Uzawa-preconditioned Krylov methods. We have not included the projection algorithms in our experiment due to the difficulties in constructing efficient preconditioners for the composite operator $\mathbf{K}_p$ (equation 5).

5.1. Implementation Details. To take advantage of the fast multigrid methods we replace the eliminated variables with identity rows and columns so that the degree of freedom at each grid node is preserved. Our implementation sets the default maximum block size to be 100, but allows users to adjust it. The algorithm declares a failure if one or more constraint-slave pairs cannot be found with a given maximum block size. To maintain numerical stability in the elimination process, diagonal blocks in $\mathbf{C}_s$ are allowed only if the conditioning of the blocks are within a prescribed tolerance, which is 10^5 in our implementation.

For the block-preconditioned Krylov methods, we use the flexible generalized minimal residual method (FGMRES) since using Krylov methods for inner iterations gives a variable preconditioner, and the preconditioners are in general nonsymmetric. For the inner iterations, a wide selection of preconditioners are available for the $(1,1)$ block. The convergence tolerance can be relaxed for the inner solves (we use a relative tolerance of 10^{-2} to 10^{-1}). Since the Schur complement $\mathbf{C}\mathbf{K}^{-1}\mathbf{C}^T$ is too expensive to form explicitly, we approximate $\mathbf{K}^{-1}$ by the inverse of its diagonal. We have also implemented the capability to compute an approximate inverse of $\mathbf{K}$ with different sparsity patterns, but our numerical experience points to diagonal approximation as often being the most efficient.

For the Uzawa-preconditioned Krylov methods, we also need to use the FGMRES method. Similar considerations in prescribing looser convergence tolerance for the inner iterations, using different preconditioners, and approximating the Schur

complement are needed here. We use a scaled identity matrix as an approximation to the Schur complement, and we vary the scaling factor to obtain better overall efficiency.

5.2. Test Problems. The test problem used in our experiment is a three-dimensional elasticity problem generated by an implicit hydrodynamics code at LLNL. The domain has 3 blocks which are in contact, as depicted in Figure 3. There are three slide surfaces intersecting each other at an edge. We vary the size of each block in our numerical investigations. All experiments are run on a Compaq alpha-based machine running at 1 GHz.

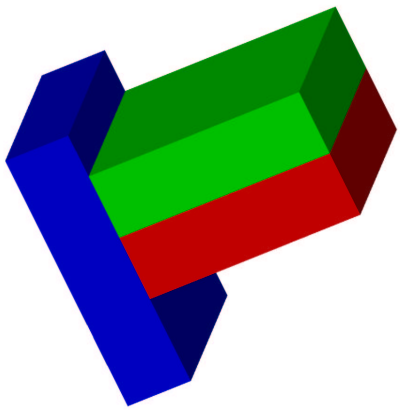

FIG. 3. *3D Elasticity Problem with Intersecting Slide Surfaces*

5.3. Numerical Comparisons. The objective of the first set of experiment is to investigate the relative performance of different methods on a moderate-size problem. The best performing methods will be used subsequently for problems of various sizes. In all numerical tests, a relative tolerance of 10^{-6} is used in the convergence criterion.

5.3.1. Comparisons of Different Methods. For solving the reduced system from algebraic elimination, we use the classical conjugate gradient method with different preconditioners. Specifically, we use diagonal, sparse approximate inverse (the *ParaSails* software package [8]), the classical algebraic multigrid (the *BoomerAMG* software package [16]), and the smoothed aggregation algebraic multigrid (*SAAMG* [29] in the HYPRE [16] linear solver library). The parameters used for *ParaSails* are *Nlevels*= 1, *Threshold*= $0.1 - 0.3$, and *Filter*= 0.1 and 0.2 (denoted, for example, *ParaSails*(1,0.1,0.1)). The parameters used for *BoomerAMG* are 2 sweeps of symmetric Gauss Seidel smoothings, different coarsening threshold ranging from 0.5 to 0.99, and a system size of 3 (denoted, for example, *BoomerAMG*(SGS,2,0.5,3)). The parameters used for *SAAMG* are 1 or 2 sweeps of symmetric Gauss Seidel smoothings and a null space dimension of 3 or 6 (for example, *SAAMG*(SGS,2,3)).

For the unreduced system, we use FGMRES with Uzawa and block preconditioners. The conjugate gradient method is used to solve the (1,1) block (namely, $\mathbf{K}$), and smoothed aggregation multigrid with the same parameter settings as above is used as preconditioner. The convergence criterion for the (1,1) block solve is a relative tolerance of 10^{-1} or 10^{-2}. The approximate Schur complement for the Uzawa pre-

conditioner is the scaled identity with scaling factor 0.01 and $\mathbf{C}[\mathrm{diag}(\mathbf{K})]^{-1}\mathbf{C}^T$ for the block triangular (*blockT*) preconditioner.

Numerical results are given in Table 1. Here we observe that algebraic elimination is significantly more efficient (a factor of more than 10 is observed here) than Krylov methods applied directly to the indefinite system. In addition, less parameter selection is needed to tune its performance. The algebraically reduced system is relatively easy to solve so that diagonal preconditioner gives very good overall timings. Smoothed aggregation multigrid gives the best timings. The classical algebraic multigrid gives the best iteration counts, but the computational cost per iteration is very high so that the overall cost is relatively high. Reducing the number of relaxations from 2 to 1 results in much higher iteration counts. The best sparse approximate inverse uses a relatively high threshold of 0.3. As the threshold gets closer to 1.0, it becomes closer to a diagonal preconditioner. This implies that the simple diagonal scaling may already be a very efficient preconditioner.

TABLE 1
Numerical Results for the 18K Element Problem

Reduction	Krylov method	Preconditioner	Iterations	Time
No	FGMRES	*Uzawa*(itol=10^{-1})	39	470 sec
No	FGMRES	*Uzawa*(itol=10^{-2})	32	597 sec
No	FGMRES	*blockT*(itol=10^{-1})	48	222 sec
No	FGMRES	*blockT*(itol=10^{-2})	34	270 sec
Yes	CG	diagonal	594	27 sec
Yes	CG	*ParaSails*(1,0.1,0.1)	330	77 sec
Yes	CG	*ParaSails*(1,0.1,0.2)	663	93 sec
Yes	CG	*ParaSails*(1,0.2,0.1)	599	42 sec
Yes	CG	*ParaSails*(1,0.3,0.1)	648	40 sec
Yes	CG	*BoomerAMG*(SGS,2,0.5,3)	17	278 sec
Yes	CG	*BoomerAMG*(SGS,2,0.75,3)	28	148 sec
Yes	CG	*BoomerAMG*(SGS,2,0.99,3)	71	159 sec
Yes	CG	*SAAMG*(SGS,1,3)	63	21 sec
Yes	CG	*SAAMG*(SGS,2,3)	51	27 sec
Yes	CG	*SAAMG*(SGS,1,6)	59	24.5 sec
Yes	CG	*SAAMG*(SGS,2,6)	48	32 sec
note: itol - tolerance for the (1,1) block solve				

5.3.2. Different preconditoners on the reduced systems. Having demonstrated the effectiveness of our algebraic elimination method, we investigate the efficiency of different preconditioners on the algebraic reduced systems in this section. Only the more efficient preconditioners such as diagonal and smoothed aggregation multigrid preconditioners are studied here. The numerical results are given in Table 2.

We observe that the reduction time can be as much as 50% of the overall solution time. The irregularities in the reduction time with different problem size is due to the different maximum block sizes generated in each case. We found that the maximum block sizes required for algebraic elemination are 11 for the largest and smallest (66K and 452K) problems, but 60 for the other two. Further study into how to better control the maximum block size and how to improve the algebraic elimination time

TABLE 2
Comparison of Different Preconditioners

Problem Information				diag precond		SAAMG(SGS,1,3)	
Size	NNZ	NNZ ratio	RTime	iter	time	iter	time
66404	4.5 M	1.14	8 sec	594	27 sec	63	21 sec
104780	7.2 M	1.14	38 sec	848	62 sec	122	63 sec
235053	13.3 M	1.11	110 sec	951	175 sec	93	113 sec
452166	32.9 M	1.08	54 sec	1197	385 sec	109	251 sec

NNZ ratio - ratio of NNZ for reduced system versus **A**
RTime - algebraic elimination time

are needed.

6. Concluding remarks. Algebraic elimination presents a promising technique to handle linear elasticity problems with simple and intersecting slide surface constraints. The elimination is efficient to perform and only increases the sparsity of the matrix slightly. The restoration of positive definiteness greatly improves the efficiency of the solution process, as demonstrated in our numerical experiments. Further work includes its optimization, its parallel implementation, and its extension to more general constraints.

REFERENCES

[1] J. F. Abel and M. S. Shephard, *An Algorithm for Multipoint Constraints in Finite Element Analysis,* Int. J. Num. Meth. Engrg. 14, 464-467, 1979.

[2] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Nonlinear Programming,* Stanford University Press, Stanford, CA 1958.

[3] R. E. Bank, B. D. Welfert, and H. Yserentant, *A Class of Iterative Methods for Solving Saddle Point Problems,* Numer. Math. 56, pp. 645-666, 1990.

[4] J. Bramble, J. Pasciak, and Vassilev, *Analysis of the Inexact Uzawa Algorithm for Saddle Point Problems,* SIAM J. Numer. Anal. 32, No. 3, pp. 1072-1092, June 1997.

[5] N. J. Carpenter and R. L. Taylor and M. G. Katona, *Lagrange Constraints for Transient Finite Element Surface Contact,* Int. J. Num. Meth. Engrg. 32, 103-128, 1991.

[6] X. L. Cheng and J. Zou, *An Inexact Uzawa-Type Iterative Method for Solving Saddle Point Problems,* manuscript, Mathematics Department, Chinese University of Hong Kong.

[7] E. Chow, T. A. Manteuffel, C. Tong, and B. K. Wallin, *Algebraic Elimination of Slide Surface Constraints in Implicit Structural Analysis,* manuscript.

[8] E. Chow, T. *ParaSails User's Guide,* Tech. Report UCRL-MA-137863, Lawrence Livermore National Laboratory, Livermore, CA, 2000.

[9] J. I Curiskis and S. Valliappan, *A Solution Algorithm for Linear Constraint Equations in Finite Element Analysis,* Computers and Structures, 8, 117-124, 1978.

[10] H. C. Elman and G. H. Golub, *Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems,* SIAM J. Numer. Anal. 31, pp. 1645-1661, 1994.

[11] R. Fletcher and C. M. Reeves, *Function Minimization by Conjugate Gradients,* Comput. J. 7, pp. 149-154, 1964.

[12] R. W. Freund and N. M. Nachtigal, *QMR: a Quasi-minimal Residual Method for non-Hermitian Linear Systems,* Numer. Math. 60, pp. 315-339, 1991.

[13] P. E. Gill and W. Murray, *Numerical Methods for Constrained Optimization,* Academic Press, London, 1974.

[14] G. H. Golub and A. J. Wathen, *An Iteration For Indefinite Systems and Its Application to the Navier Stokes Equations,* SIAM J. Sci. Comput. 21, No. 6, pp. 1969-1972.

[15] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization,* Technical Report RAL-TR-1998-069, Rutherford Appleton Laboratory, 1998.

[16] Henson, V.E. and U.M. Yang, *BoomerAMG: a Parallel Algebraic Multigrid Solver and Pre-*

conditioner, Technical report UCRL-JC-141495, Lawrence Livermore National Laboratory, 2000.

[17] Q. Y. Hu and J. Zou, *Two New Variants of Nonlinear Inexact Uzawa Algorithms for Saddle Point Problems,* to appear in Numerische Mathematik.

[18] C. Keller, N. I. M. Gould, and a. J. Wathen, Constraint Preconditioning for Indefinite Linear Systems, SIAM J. Matrix Anal. Appl. 21, pp. 1300-1317, 2000.

[19] L. Little and Y. Saad, *Block LU Preconditioners for Symmetric and Nonsymmetric Saddle Point Problems,* Technical report 99/104, Minnesota Supercomputing Institute, 1999.

[20] A. Klawonn, *preconditioners for Indefinite Problems,* PhD thesis, Westfälischen Wilhelms-Universität Münster, 1995.

[21] M. F. Murphy, G. H. Golub, and A. Wathen, *A Note on Preconditioning for Indefinite Linear Systems,* SIAM J. Sci. Comput. 21, No. 6, pp. 1969-1972.

[22] C. Paige and M. Saunders, *Solution of Sparse Indefinite Systems of Linear Equations,* SIAM J. Numer. Anal. 12, pp. 617-629, 1975.

[23] M. Rozložník and V. Simoncini, *Krylov Subspace methods for Saddle Point Problems with Indefinite Preconditioning,* To appear in SIMAX.

[24] T. Rusten and R. Winther, *A Preconditioned Iterative Method for Saddle Point Problems,* SIAM J. Matrix Anal. Appl. 13, No. 3, pp. 887-904, 1992.

[25] P. Saint-Georges and Y. Notay and G. Warzée, *Efficient iterative solution of constrained finite element analyses,* Comput. Meth. Appl. Mech. Engrg., 160, 101-114, 1998.

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems,* PWS Publishing Company, 1995.

[27] V. Sarin and A. Sameh, *An Efficient Iterative Method for the Generalized Stokes Problem,* SIAM J. Sci. Comput. 19, No. 1, pp. 206-226, January 1998.

[28] Z. Tong and A. Sameh, *On an Iterative Method for Saddle Point Problems,* Numer. Math. 79, pp. 643-646, 1998.

[29] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing 56, pp. 179-196, 1996.