

Algebraic elimination of slide surface constraints in implicit structural analysis*

Edmond Chow[†], Thomas A. Manteuffel[‡], Charles Tong[†], Bradley K. Wallin[§]

August 7, 2001

Abstract

Slide surface and contact boundary conditions can be implemented via Lagrange multipliers in the algebraic equations in implicit structural analysis. This indefinite set of equations is difficult to solve by iterative methods and is often too large to be solved by direct methods. When there are m constraints and there exists a set of m variables where each variable is only involved in a single constraint, we advocate a direct elimination technique which leaves a sparse, positive definite system to solve by iterative methods. We prove that the amount of “fill-in” created by this process is independent of the size of the slide surfaces. In addition, the eigenvalues of the reduced matrix do not differ significantly from the eigenvalues of the unconstrained matrix. This method can be extended to the case where constrained surfaces intersect and leads to a graph theoretic approach for determining which variables can be eliminated efficiently for constraints with more general structure.

1 Introduction

The numerical simulation of large structural deformations of materials has several applications, including the safety of high explosives, and manufacturing problems in metal forming processes. During the heating of a high explosive, for instance, the explosive expands and deforms its containment vessel. Depending on the strength of the vessel, the explosive can achieve temperatures and pressures that cause it to reach a runaway state. In modeling this phenomenon, it is important to allow relative motion between the inside surface of the container and the interface defined by the explosive. Thus, the explosive and its container are modeled as two separate bodies, and a *slide surface* boundary condition allows the bodies to slide tangentially relative to each other.

*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 (echow@llnl.gov and chtong@llnl.gov).

[‡]Applied Mathematics Department, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309 (tmanteuf@boulder.colorado.edu).

[§]B Division, Lawrence Livermore National Laboratory, L-095, Box 808, Livermore, CA 94551 (bwallin@llnl.gov).

Slide surface boundary conditions have been implemented in many explicit and implicit codes, e.g., [13, 4]. In the implicit context, slide surface boundary conditions are added as constraints to the linear algebraic equations that must be solved. The complete set of equations, unfortunately, is indefinite and is difficult to solve by iterative methods and is often too large to be solved by direct methods. The structure of the constraints, however, allows a subset of the equations to be eliminated directly, leaving a sparse, positive definite system to solve by iterative methods. This paper discusses the conditioning and sparseness of the reduced matrix and proposes a graph theoretic framework for determining which variables can be eliminated efficiently for constraints with more general structure.

In structural analysis, the “transformation method” (see, e.g., [7, 10]) and its variants [8, 1] are direct elimination methods. It seems that little attention has been given to partitioning the constrained degrees of freedom so that these methods are efficient, leading some authors to claim that these methods may be a “bottleneck” for large FE models [20]. Alternatively, Lagrange multiplier methods leading to KKT-type systems may be used, especially if direct solvers are available. If iterative solvers are used, for very large problems for example, the options are less attractive. Projection methods [11, 12, 20], Uzawa methods, e.g., [2, 9], and block preconditioners, e.g., [3, 15, 16, 14], may all suffer from slow convergence rates or high cost when many constraints are involved. When the constraints are structured such that direct elimination of some degrees of freedom is economical, then direct elimination as a preprocessing step can be effective before either a direct or iterative solution of the reduced system is carried out.

Section 2 of this paper briefly discusses the modeling of slide surfaces. Section 3 derives an elimination technique for the Lagrange multiplier method which is equivalent to the transformation method. In Section 4, the sparseness and conditioning of the reduced matrix are discussed, as well as implementation options. A few sample numerical results are given in Section 5. Section 6 shows how the elimination technique may be extended to problems with more general constraints. We close with some final remarks in Section 7.

2 Slide surface modeling

2.1 Slide surface constraints

A slide surface refers 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 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. In this paper, we will only discuss modeling the simplest case, that of two bodies in contact that remain in contact, which may slide relative to each other during the simulation.

A slide surface constraint in this case is an “impenetrability” constraint preventing structural domains from overlapping. When it is known that two bodies are in contact, this constraint can be implemented by constraining one side of the slide surface, called the slave, to the other side, called the master. Since the master side then effectively defines the surface, it is typically chosen to be that side that moves less, is denser, more rigid, or more densely gridded. Interface nodes on the slave side are called “slave nodes,” and interface nodes on the master side are

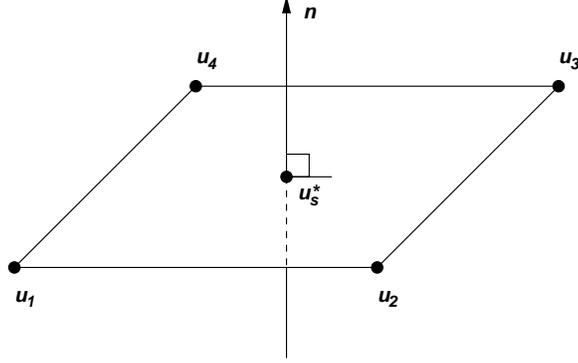


Figure 1: The normal n to the master surface and the point u_s^* . The point u_s (not shown) is constrained to lie on the surface orthogonal to n and through u_s^* , i.e., the master surface.

called “master nodes.” The surface defined by the master nodes is called the “master surface.”

The impenetrability constraint for a particular slave node constrains its displacement u_s to lie on the master surface. Given a displacement u_s^* that lies on the master surface near u_s , the impenetrability constraint states that

$$n^T(u_s - u_s^*) = 0 \quad (1)$$

where n is the normal to the master surface at u_s^* (see Figure 1).

The parametric coordinates of u_s^* are found by an iterative procedure that projects the current u_s along the normal n onto the closest element face on the master surface. The actual displacement of this point u_s^* is then interpolated using the finite element shape functions. For linear hexahedral elements, the interpolation on the element face reduces to

$$u_s^* = \phi_1 u_1 + \phi_2 u_2 + \phi_3 u_3 + \phi_4 u_4$$

where u_1 , u_2 , u_3 , and u_4 are the displacements of the four nodes defining the master face, and ϕ_1 , ϕ_2 , ϕ_3 , and ϕ_4 are the corresponding shape function values for the parametric coordinates of u_s^* . Combining this with (1) yields

$$n^T(u_s - \phi_1 u_1 - \phi_2 u_2 - \phi_3 u_3 - \phi_4 u_4) = 0.$$

This constraint is nonlinear; the normal vector and the shape functions depend on both the master and the slave node displacements. We linearize this constraint by evaluating the normal and the shape functions using the current estimates for the displacements, while u_s , u_1 , u_2 , u_3 , and u_4 represent the new displacements.

For another treatment of slide surface constraints, see, for example [18].

2.2 Structure of the constraints

In matrix form, the set of constraints for all slave nodes is $G^T u = 0$, where u is the vector of nodal displacements. Assuming that the nodes are numbered such that all slave nodes follow

all master nodes which follow all other nodes, the matrix G^T has a structure of the form

$$\left[\begin{array}{ccc|cccc} 0 & \dots & 0 & -\phi_{11}n_1^T & -\phi_{12}n_1^T & -\phi_{13}n_1^T & -\phi_{14}n_1^T & & n_1^T & & \\ \vdots & & \vdots & & -\phi_{22}n_2^T & -\phi_{23}n_2^T & -\phi_{24}n_2^T & -\phi_{25}n_2^T & & n_2^T & \\ 0 & \dots & 0 & \dots & \dots & \dots & \dots & \dots & & & \dots \end{array} \right] \quad (2)$$

where $n_i^T = (n_i^x, n_i^y, n_i^z)$ is the unit normal for the i th slave node, and ϕ_{ij} are shape function values for the four master nodes (indexed by j) associated with the same slave node. We abbreviate the above structure by $[0, -N^T W, N^T]$ where W is a matrix containing the positive shape function values ϕ_{ij} . Since no node depends on any slave nodes in a slide surface constraint, the matrix N^T contains at most one nonzero in each column. The number of constraints is equal to the number of slave nodes on the slide surface and is typically much smaller than the total number of nodes.

3 Algebraic elimination for constrained problems

The discrete and linear constraints $G^T u = 0$ may be incorporated into the finite element equations $Au = f$ via the transformation method or the Lagrange multiplier method. The latter leads to a system that is indefinite and generally difficult to solve by iterative methods. However, like the transformation method, solving this system can be reduced to solving a sparse, positive definite system. The economy of this transformation depends on the constraints having the specific structure described below.

3.1 Transformation method

In the transformation method [7, 10], the variables are partitioned into two sets: *independent* and *dependent*. The number of dependent variables must equal the number of constraint equations, m . Let u_1 denote the independent variables and u_2 denote the dependent variables. The system $Au = f$ may now be partitioned symmetrically into block form as

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

while $G^T u = 0$ can be partitioned as

$$\begin{bmatrix} B^T & D^T \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0.$$

The transformation matrix T is defined so that $u = Tu_1$, hence

$$T = \begin{bmatrix} I \\ -D^{-T} B^T \end{bmatrix}.$$

The transformed system to be solved is

$$T^T A T u_1 = T^T f$$

the solution of which may be substituted into $u = Tu_1$ to recover the full solution. The transformed coefficient matrix is

$$T^T AT = A_{11} - A_{12}D^{-T}B^T - BD^{-1}A_{21} + BD^{-1}A_{22}D^{-T}B^T \quad (3)$$

which is symmetric positive definite by construction, provided A is symmetric positive definite.

We define a *purely dependent* variable to be a variable that is only involved in one constraint. If all the dependent variables u_2 are purely dependent, then D is composed of one nonzero per column and one nonzero per row (it is a reordered diagonal matrix) and its inverse is sparse. The transformed matrix (3) then would be sparse and economical to construct and factorize. In many problems it is possible to choose m purely dependent variables. This is often the case in structural analysis since slave variables are often defined so that they are purely dependent.

For the constraint matrix (2), each column of N^T has at most one nonzero entry, and since each row of N^T represents a unit vector, each row contains at least one nonzero entry. Choosing the m purely dependent variables amounts to choosing one of the components of $n_i^T = (n_i^x, n_i^y, n_i^z)$ for each constraint i . Thus we can reorder and partition N^T by columns into $[N_1^T, N_2^T]$ such that N_2^T is a diagonal matrix.

3.2 Lagrange multiplier method

When the variables are partitioned into slave nodes, master nodes, and interior (all other) nodes, the Lagrange multiplier method leads to an equation with the structure

$$\left[\begin{array}{ccc|c} A_i & A_{im} & A_{is} & -W^T N \\ A_{mi} & A_m & & N \\ A_{si} & & A_s & 0 \\ \hline & -N^T W & N^T & 0 \end{array} \right] \begin{bmatrix} u_i \\ u_m \\ u_s \\ \lambda \end{bmatrix} = \begin{bmatrix} f_i \\ f_m \\ f_s \\ 0 \end{bmatrix}. \quad (4)$$

where λ is the vector of Lagrange multipliers and can be interpreted as the normal force at each slave node necessary to conserve momentum. There is no interaction in the stiffness matrix A between the slave nodes and the master nodes.

A technique equivalent to the transformation method may be derived for matrices in this form. As before, we partition the constrained variables into independent and dependent sets. For our slide surface constraints, we partition G^T into $[0, -N^T W, N_1^T, N_2^T]$ where N_2^T is diagonal as above. We can now further partition (4) as

$$\left[\begin{array}{ccc|c} X & X & X & X \\ X & X & & -W^T N \\ X & & X & N_1 \\ \hline X & & X & N_2 \\ \hline & -N^T W & N_1^T & N_2^T \\ & & & 0 \end{array} \right] \begin{bmatrix} u_i \\ u_m \\ u_{s_1} \\ u_{s_2} \\ \lambda \end{bmatrix} = \begin{bmatrix} f_i \\ f_m \\ f_{s_1} \\ f_{s_2} \\ 0 \end{bmatrix} \quad (5)$$

where X represents a nonzero block in the matrix and u_{s_1} and u_{s_2} are slave variables. We will use the following simplified notation for (5):

$$\begin{bmatrix} A_{11} & A_{12} & B \\ A_{21} & A_{22} & D \\ B^T & D^T & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ 0 \end{bmatrix}. \quad (6)$$

The Schur complement of the matrix in (6) with respect to $\begin{bmatrix} A_{22} & D \\ D^T & 0 \end{bmatrix}$ is

$$S = A_{11} - \begin{bmatrix} A_{12} & B \end{bmatrix} \begin{bmatrix} A_{22} & D \\ D^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} A_{21} \\ B^T \end{bmatrix}. \quad (7)$$

Since D^{-1} is sparse, the inverse

$$\begin{bmatrix} A_{22} & D \\ D^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & D^{-T} \\ D^{-1} & -D^{-1}A_{22}D^{-T} \end{bmatrix} \quad (8)$$

is sparse, and thus the Schur complement is sparse in general. The Schur complement is positive definite since it is equivalent to the transformed matrix (3).

The block system (6) can be solved by solving the reduced system

$$Su_1 = f_1 - BD^{-1}f_2 \quad (9)$$

and substituting u_1 into

$$u_2 = -D^{-T}B^T u_1 \quad (10)$$

$$\lambda = D^{-1}(f_2 - A_{21}u_1) + D^{-1}A_{22}D^{-T}B^T u_1. \quad (11)$$

4 Properties of the reduced matrix

4.1 Eigenvalues of S

The convergence rate of the conjugate gradient method for solving systems with S depends on the eigenvalues of S , see, e.g., [19]. If A , the stiffness matrix without the constraints, is symmetric positive definite, the next lemma bounds the eigenvalues of S away from the origin.

Lemma 4.1 *If A is symmetric, the smallest eigenvalue of S , $\lambda_{\min}(S)$, is larger than the smallest eigenvalue of A , $\lambda_{\min}(A)$.*

PROOF. Choose $x_1 \neq 0$ such that

$$\frac{x_1^T S x_1}{x_1^T x_1} = \lambda_{\min}(S)$$

and let $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $x_2 = -D^{-T}B^T x_1$. Then,

$$\frac{x_1^T S x_1}{x_1^T x_1} \geq \frac{x_1^T S x_1}{x_1^T x_1 + x_2^T x_2} = \frac{x^T A x}{x^T x} \geq \lambda_{\min}(A).$$

□

The behavior of the other eigenvalues can be determined by applying well-known theorems in eigenvalue sensitivity to the reduced matrix in the form (7). Consider the symmetric matrix

$$A = \begin{bmatrix} A_{11} & \bar{B} \\ \bar{B}^T & W \end{bmatrix}, \quad \text{with} \quad W = \begin{bmatrix} a & d \\ d & 0 \end{bmatrix}, \quad a > 0, \quad d \neq 0$$

and the pairwise elimination of a single constraint and its corresponding slave equation to get the Schur complement S . Eliminating m constraints and their corresponding slave equations is simply a sequence of these pairwise eliminations.

Define $E \equiv -\bar{B}W^{-1}\bar{B}^T$ so that $S = A_{11} + E$. The eigenvalues of $-W^{-1}$ are

$$\frac{a \pm \sqrt{a^2 + 4d^2}}{2d^2}$$

which shows that one eigenvalue is positive and the other is negative. From the Sylvester law of inertia, assuming \bar{B} has full rank, E has one positive eigenvalue and one negative eigenvalue; the other eigenvalues are zero.

For a matrix S , let $\lambda_k(S)$ denote the k th largest eigenvalue and let n denote the dimension of S . Two corollaries of the Courant-Fischer minimax theorem apply. First, the extremal eigenvalues of A_{11} are bounded by the extremal eigenvalues of A (interlacing property). Second, we have (see [21])

$$\begin{aligned} \lambda_{r+s-1}(S) &\leq \lambda_r(A_{11}) + \lambda_s(E), \quad r + s - 1 \leq n \\ \text{and} \quad \lambda_{r-s+1}(S) &\geq \lambda_r(A_{11}) + \lambda_{n-s+1}(E), \quad s \leq r \leq n. \end{aligned}$$

In particular, if the positive eigenvalue of E is bounded, then the largest eigenvalue of S is bounded. In addition, if E has a very large positive eigenvalue, the bound on the largest eigenvalue of S increases by the size of this eigenvalue, but since E has many zero eigenvalues, the other eigenvalues of S are still bounded by the eigenvalues of A_{11} ,

$$\begin{aligned} \lambda_{r+1}(S) &\leq \lambda_r(A_{11}), \quad 1 \leq r \leq n - 1 \\ \lambda_{r+1}(S) &\geq \lambda_{r+2}(A_{11}), \quad 0 \leq r \leq n - 2. \end{aligned}$$

Small eigenvalues of E can generally be avoided by choosing to eliminate a slave variable that corresponds to large magnitude values of d , and, less importantly, small values of a . (This assumes that the nonsquare matrix \bar{B} is not too poorly conditioned.) Small magnitude values of d can easily be avoided for slide surface constraints since $|d| \geq 1/\sqrt{3}$ can always be chosen.

Instead of being concerned by the eigenvalues of E , we can guarantee that the spectrum of S does not differ too much from the spectrum of A_{11} by bounding the size of the entries in E , or more precisely, the Frobenius norm of E . This is equivalent to bounding the sum of the squares of the eigenvalues of E and is the result of the Wielandt-Hoffmann theorem,

$$\sum_{i=1}^n (\lambda_i(S) - \lambda_i(A_{11}))^2 \leq \|E\|_F^2.$$

Again, the norm of E can be roughly controlled by not choosing excessively small magnitude values of d .

It is possible in some cases to choose variables to eliminate such that the conditioning of S is better than the conditioning of A_{11} . This is very difficult to do in general, however.

In conclusion, we try to control the spectrum of S by guaranteeing that it does not differ too much from the spectrum of A_{11} . Very small magnitudes of d can seriously harm convergence and can easily be avoided by simply choosing larger components in $n_i^T = (n_i^x, n_i^y, n_i^z)$ for each constraint row i when partitioning N^T . With this strategy, we find in practice that solving with S is only slightly more difficult than solving with A . Section 5 will show a typical example of this.

4.2 Sparseness of S

In this section we show that the number of nonzeros in each row of S is bounded, and is not dependent on the number of slave variables or constraints. Thus the algebraic elimination method can be used for very large problems. In the following, we will use the graph theoretic representation of a sparse matrix. To analyze the sparsity pattern of S , “fill-paths” and the graph interpretation of Gaussian elimination [17] give an over-estimate of the fill-in that occurs, since they assume that the inverse (8) is dense. Hence, we proceed as follows.

The variables in u can be partitioned into interior, master, slave, uneliminated slave, and constraint (Lagrange multiplier), and these sets may be represented by the symbols i , m , s , t , and c , respectively. Uneliminated slave variables are those variables at the same grid point as an eliminated slave variable. We then label the relevant blocks in (5) as follows:

$$\left[\begin{array}{ccc|c|c} X & X & X & A_{is} & \\ X & X & & & A_{mc} \\ X & & X & A_{ts} & A_{tc} \\ \hline A_{si} & & A_{st} & A_{ss} & A_{sc} \\ \hline & A_{cm} & A_{ct} & A_{cs} & 0 \end{array} \right]. \quad (12)$$

The sparsity pattern of a matrix may be described by its *graph*. For a symmetric matrix A of order n , its graph $G(A)$ is composed of the vertices $\{1, \dots, n\}$ and the edges $\{(i, j), i \neq j \text{ and } A_{ij} \neq 0\}$. When the context is clear, we will not distinguish between vertices and variables, and between edges and nonzeros. The graph of a matrix is often related to its discretization grid. An example graph (for a matrix without uneliminated slave variables, i.e., a scalar problem) is shown in Figure 2.

For the matrix (12), the “fill-in matrix,” $S - A_{11}$, after eliminating the s and c variables is

$$\left[\begin{array}{c|c|c} 0 & A_{is}A_{sc}^{-1}A_{cm} & A_{is}A_{sc}^{-1}A_{ct} \\ \hline A_{mc}A_{cs}^{-1}A_{si} & -A_{mc}A_{cs}^{-1}A_{ss}A_{sc}^{-1}A_{cm} & A_{mc}A_{cs}^{-1}A_{st} \\ & & -A_{mc}A_{cs}^{-1}A_{ss}A_{sc}^{-1}A_{ct} \\ \hline A_{tc}A_{cs}^{-1}A_{si} & A_{ts}A_{sc}^{-1}A_{cm} & A_{ts}A_{sc}^{-1}A_{ct} \\ & -A_{tc}A_{cs}^{-1}A_{ss}A_{sc}^{-1}A_{cm} & +A_{tc}A_{cs}^{-1}A_{st} \\ & & -A_{tc}A_{cs}^{-1}A_{ss}A_{sc}^{-1}A_{ct} \end{array} \right]. \quad (13)$$

In the graph of this matrix, we first note that there is no fill-in between any interior vertices. The (1, 2) block $A_{is}A_{sc}^{-1}A_{cm}$ and the (2, 1) block $A_{mc}A_{cs}^{-1}A_{si}$ correspond to fill-in between

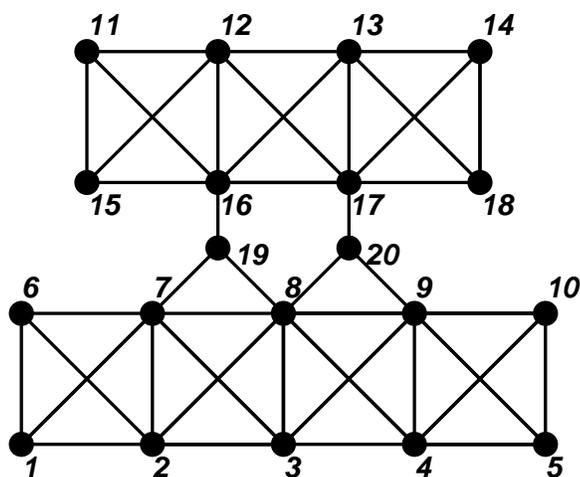


Figure 2: Graph of an example problem a two side surface. The top side is the slave side and the bottom side is the master side. Vertices 16 and 17 are slaves, vertices 7, 8 and 9 are masters, and vertices 19 and 20 are constraint (Lagrange multiplier) vertices.

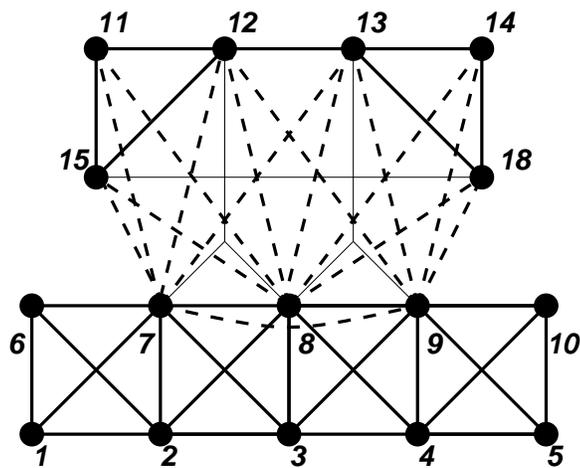


Figure 3: Graph of the reduced matrix after the slave and constraint vertices have been eliminated. Dotted lines indicate the fill-in edges. Note that there is no fill-in edge between nodes 9 and 11 as would be predicted by the fill-path theorem for Gaussian elimination.

an interior vertex and a master vertex when there is path between these vertices through a slave vertex and a constraint vertex in $G(A)$. These paths may be abbreviated (i, s, c, m) and (m, c, s, i) . The $(2, 2)$ block $-A_{mc}A_{cs}^{-1}A_{ss}A_{sc}^{-1}A_{cm}$ corresponds to fill-in between master vertices that are connected via a path (m, c, s, s, c, m) .

The fill-in caused by the other blocks can be interpreted similarly. None of these blocks cause fill-in between vertices that are more than a small fixed number of path lengths apart. Thus the fill-in in a row of S is bounded.

Figure 3 shows the graph of S for the problem of Figure 2. The fill-in is shown with dotted edges. The edge (7,9) is caused by a path of the type (m, c, s, s, c, m) . The fill-in generally involves vertices on or adjacent to the slide surface.

Counting paths in the graph of (13) gives the number of fill-ins that will occur for a given slide surface geometry.

4.3 Alternative implementations

To solve a system with S by iterative methods, it is not necessary to form S but only to apply its action to a vector. If S does need to be formed, for example for use by a preconditioner, this cost is low since S is sparse as shown above.

Besides forming S directly via (3), it is conceivable to adapt a direct solver to construct S in order to save programming effort. If the matrix in (6) is symmetrically reordered such that the first $2m$ equations are the individual slave equations to be eliminated followed *immediately* by their corresponding constraint equations (i.e., equations corresponding to slaves and constraints are *interlaced*), then S is the Schur complement that remains after the first $2m$ variables are eliminated via either an LU factorization or symmetric indefinite factorization with 2-by-2 blocks, both *without* pivoting. This ordering reduces intermediate fill-in in the LU factorization.

The use of pivoting may disrupt the above ordering, and the use of any other ordering may in addition require pivoting so that the factorization is stable. For example, in the equation below we show a matrix in a given ordering and its factorization after the first block of variables have been eliminated:

$$\left[\begin{array}{c|cc} A_{22} & D & A_{21} \\ \hline D^T & 0 & B^T \\ \hline A_{12} & B & A_{11} \end{array} \right] = \left[\begin{array}{c|cc} L & 0 & 0 \\ \hline D^T U^{-1} & -D^T A_{22}^{-1} D & B^T - D^T A_{22}^{-1} A_{21} \\ \hline A_{12} U^{-1} & B - A_{12} A_{22}^{-1} D & A_{11} - A_{12} A_{22}^{-1} A_{21} \end{array} \right] \left[\begin{array}{c|cc} U & L^{-1} D & L^{-1} A_{21} \\ \hline 0 & I & 0 \\ \hline 0 & 0 & I \end{array} \right]$$

where $LU = A_{22}$. The Schur complement will be dense if A_{22}^{-1} is dense, as is typically the case. After the next block is eliminated, the final reduced matrix is sparse.

Forming the Schur complement directly via (3) is stable in the sense that the process will not break down. The size of the largest entries in S is controlled insofar as the smallest entries in D are controlled. Since we can choose components in D as the largest component of a unit vector, these components are at least $1/\sqrt{3}$ in magnitude.

5 Numerical Tests

An octant of three concentric spherical shells is shown in Figure 4. The outer shells are composed of steel, and the inner shell is composed of lucite. Slide surface boundary conditions are used

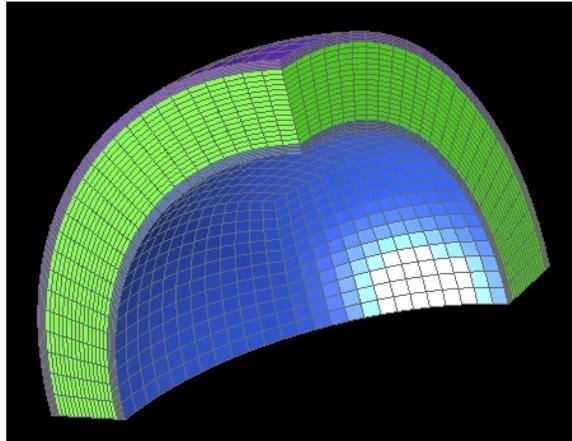


Figure 4: Octant of three concentric spherical shells.

between the steel and lucite shells. This test problem has 11400 elements and 13832 nodes, with 3 displacement variables at each node. The same problem without modeling the slide surfaces has 12844 nodes.

The algebraic elimination technique was applied to the linear systems that arise and the resulting reduced systems were solved with the preconditioned conjugate gradient method with factorized sparse approximate inverse preconditioning [5, 6]. A zero initial guess was used, and the residual norm was reduced by six orders of magnitude. Four processors of an IBM SP computer were used in the computation, with interprocessor communication implemented with MPI. In the following table, we compare a few performance metrics between problems with and without slide surface constraints.

test problem	with slide surfaces	no slide surfaces
nonzeros in S or A	1.64 million	1.37 million
nonzeros in preconditioner	0.95 million	0.91 million
algebraic elimination time	0.9 seconds	N/A
preconditioner setup time	9.3 seconds	6.0 seconds
iteration counts	646	646
total solution time	33.6 seconds	26.7 seconds

Table 1: Comparison between problems with and without slide surfaces.

The table shows that the reduced matrix S contains only slightly more nonzeros than the matrix A for a problem that does not model slide surfaces. The iteration counts are also similar (in this case they happen to be the same). In addition, the time for the elimination step is only a small fraction of the overall solution time.

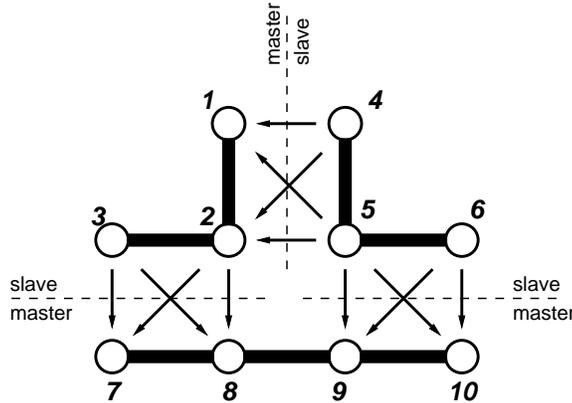


Figure 5: Intersecting slide surfaces.

6 Extension to more general constraints

6.1 Intersecting constrained surfaces

For general constraints, it is not always possible to find a partitioning of the constrained variables into independent and dependent sets such that the dependent variables are purely dependent. Algebraically, this means that it may be impossible to partition G^T into $[B^T, D^T]$ such that D is diagonal. However, the algebraic elimination procedure can still be economical if the inverse of D is sparse.

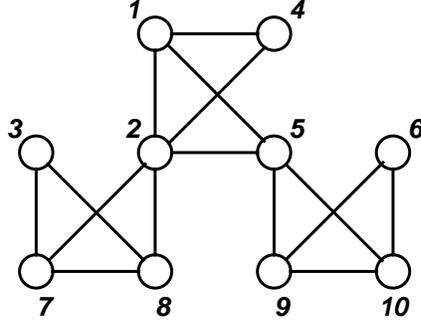
A partition of purely dependent variables cannot be found, for example, when two slide surfaces intersect at a T. Figure 5 shows this case, which is actually treated as three slide surfaces. Master and slave sides are chosen for each slide surface, and these are marked in the figure. Arrows in the figure indicate that a slave node is dependent on a master node. Node 2 is a slave node for one slide surface, but is a master node for another. Also, node 5 is a slave node for two slide surfaces. Nodes 2 and 5 are not purely dependent nodes.

For intersecting slide surfaces, however, it is possible to reorder and partition G^T such that D is *block* diagonal. We start by choosing all slave nodes to be in the dependent set. Then the two cases above must be considered:

1. a node is a slave in one constraint and a master in one or more other constraints,
2. a node is a slave node in more than one constraint.

In the first case, if a slave node is a master for k other nodes, then D will contain a block of size $(k + 1)$ -by- $(k + 1)$ since the column of G^T corresponding to the slave variable has $k + 1$ nonzeros.

In the second case, there are more constraints than slave nodes and one or more master nodes must be selected to be eliminated. Let s_1 be a slave node that participates in constraints c_1 and c_2 . A master node must be chosen from the master nodes in c_1 or c_2 to help guarantee that D will be nonsingular. If the master node is a node that participates in k constraints, then D will contain block of size k -by- k ; master nodes with small k should be preferred. If s_1 is a slave node

Figure 6: Graph of QQ^T .

in more than two constraints, then two or more master nodes need to be selected simultaneously. Here, master nodes should be selected so that the total number of new constraints involved is minimized.

An algorithm for the intersecting slide surface problem, then, would first select all slave nodes, and then add master nodes as described in the second case. The blocks formed may overlap so that the actual blocks may be larger than the sizes mentioned. This procedure is effective, but because of possible overlapping blocks, it is not optimal in the sense that the smallest blocks are found.

6.2 General constraints

In the case where the constraints are even more general, a block diagonal D can still often be found. In this section, we develop a graph theoretic framework for partitioning G^T so that D is block diagonal with small blocks.

Define the binary matrix Q^T such that $\{Q^T\}_{ij} = 1$ if and only if constraint i involves node j (which is either a master or slave node). The matrix Q^T contains m rows and n columns, where m is the number of constraints and n is the total number of slave and master nodes. For the geometry in Figure 5, we have the matrix

$$Q^T = \begin{bmatrix} 1 & 1 & . & 1 & . & . & . & . & . & . \\ 1 & 1 & . & . & 1 & . & . & . & . & . \\ . & . & . & . & 1 & . & . & . & 1 & 1 \\ . & . & . & . & . & 1 & . & . & 1 & 1 \\ . & 1 & . & . & . & . & 1 & 1 & . & . \\ . & . & 1 & . & . & . & 1 & 1 & . & . \end{bmatrix}. \quad (14)$$

The matrix QQ^T contains a nonzero at (i, j) if and only if nodes i and j are involved in the same constraint. The graph of QQ^T will be helpful to understand the partitioning problem at hand. The graph of QQ^T for matrix (14) is shown in Figure 6.

A clique is a subgraph such that every vertex in the subgraph has an edge to every other vertex in the subgraph. A maximal clique is a clique such that no vertex can be added to the clique to form a larger clique. The graph of QQ^T contains m maximal cliques, each corresponding

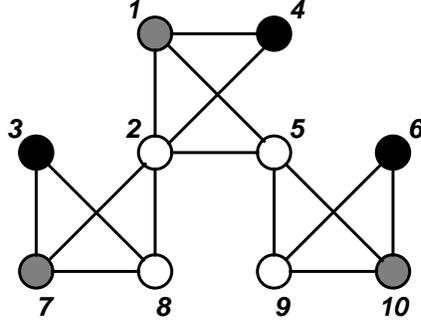


Figure 7: Selected nodes, forming three blocks of 2-by-2 in D .

to a constraint, and the vertices of each clique correspond to the nodes involved in the constraint. The partitioning problem is to select m nodes such that D is block diagonal.

Fact 6.1 *If an independent set of m vertices exists in the graph of QQ^T , then Q^T can be partitioned into $[Q_1^T, Q_2^T]$ such that Q_2^T is a diagonal matrix.*

An independent set of size m may not exist. Finding this independent set is equivalent to the *maximum* independent set problem, which is NP-hard.

Fact 6.2 *A node from each clique must be selected, otherwise D will be singular.*

Fact 6.3 *A connected set of k selected nodes will form a block of size exactly k -by- k in D .*

Fact 6.4 *Let C_j denote the number of cliques that node j participates in. Selecting this node as a dependent node will form a block of size at least C_j -by- C_j in D .*

Algorithms may be constructed that try to select nodes with small C_j . For example, m nodes with the smallest C_j may be selected to form the dependent set. Figure 7 is a possible solution for the graph of Figure 6. Three nodes with $C_j = 1$ are selected (solid circles) and three nodes with $C_j = 2$ are selected (gray circles). Figure 8 shows in comparison the situation when all the slave nodes (solid circles) are selected first, as suggested in the previous subsection. Here, the solution is worse because D contains a large block.

6.3 Dense rows in G^T

Dense rows in G^T occur when constraints involve every variable of the problem. Constraints that involve many nodes make the algebraic elimination technique very costly. As an alternative, these rows may be omitted from the elimination. Consider the partitioning of the matrix in (6) into

$$\begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & 0 \end{bmatrix}$$

where \tilde{B}^T represents dense or very full rows, and \tilde{A} represents the remainder of the problem with sparse constraints. The algebraic elimination technique is only applied to \tilde{A} . If the number of

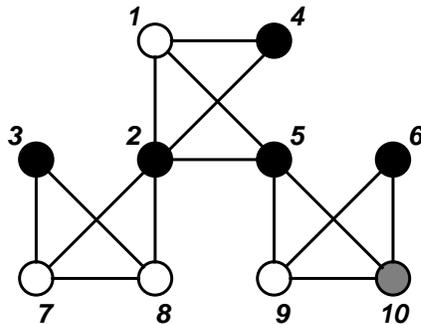


Figure 8: Selected nodes, forming two blocks of 1-by-1 and one block of 4-by-4 in D .

rows in \tilde{B}^T is small, the reduced system based on the matrix $\tilde{B}^T \tilde{A}^{-1} \tilde{B}$ may be solved iteratively. The reduced matrix is not formed, and an iterative method for indefinite systems is required. Each iteration involves a solve with \tilde{A} which in turn involves a solve with a positive definite matrix. If the number of rows in \tilde{B}^T is very small as in many applications, then very few iterations may be required.

7 Concluding remarks

We have been regularly using the algebraic elimination technique described in this paper for symmetric problems, and have not encountered any difficulties with very poorly conditioned reduced matrices. The extension to more general constraints in Section 6 may be very valuable, and in future research we intend to develop and test algorithms based on these ideas.

Acknowledgments

The authors are grateful to Ivan Otero, who originally formulated implicit slide surface constraints, and John Ruge, who suggested revisiting the use of Lagrange multipliers for the sake of generating symmetric linear systems. The authors also wish to thank Colin Aro and Juliana Hsu, who participated in testing our original implementation of the software, and Mark Adams, David Hysom, Ali Pinar, Panayot Vassilevski, and Alan Williams for helpful discussions.

References

- [1] J. F. Abel and M. S. Shephard. An algorithm for multipoint constraints in finite element analysis. *Intl. 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.*, 56:645–666, 1990.

- [4] N. J. Carpenter, R. L. Taylor, and M. G. Katona. Lagrange constraints for transient finite element surface contact. *Intl. J. Num. Meth. Engrg.*, 32:103–128, 1991.
- [5] E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21:1804–1822, 2000.
- [6] E. Chow. Parallel implementation and practical use of sparse approximate inverses with a priori sparsity patterns. *Intl. J. High Perf. Comput. Appl.*, 15:56–74, 2001.
- [7] R. D. Cook, D. S. Malkus, and M. E. Plesha. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, New York, 3rd edition, 1989.
- [8] J. I Curiskis and S. Valliappan. A solution algorithm for linear constraint equations in finite element analysis. *Comput. Structures*, 8:117–124, 1978.
- [9] H. C. Elman and G. H. Golub. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM J. Numer. Anal.*, 31:1645–1661, 1994.
- [10] R. H. Gallagher. *Finite Element Analysis Fundamentals*. Prentice-Hall, Englewood Cliffs, NJ, 1975.
- [11] P. E. Gill and W. Murray. *Numerical Methods for Constrained Optimization*. Academic Press, London, 1974.
- [12] 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, Chilton, England, 1998.
- [13] J. O. Hallquist, G. L. Goudreau, and D. J. Benson. Sliding interfaces with contact-impact in large-scale lagrangian computations. *Comput. Meth. Appl. Mech. Engrg.*, 51:107–137, 1985.
- [14] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM J. Matrix Anal. Appl.*, 21:1300–1317, 2000.
- [15] Axel Klawonn. *Preconditioners for Indefinite Problems*. PhD thesis, Westfälischen Wilhelms-Universität Münster, 1995.
- [16] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Statist. Comput.*, 21:1969–1972, 2000.
- [17] S. V. Parter. The use of linear graphs in Gauss elimination. *SIAM Rev.*, 3:119–130, 1961.
- [18] G. Rebel, K. C. Park, and C. A. Felippa. A contact-impact formulation based on localised lagrange multipliers. Technical Report CU-CAS-00-18, Center for Aerospace Structures, University of Colorado, Boulder, CO, 2000.
- [19] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Co., Boston, MA, 1996.

- [20] P. Saint-Georges, Y. Notay, and G. Warzée. Efficient iterative solution of constrained finite element analyses. *Comput. Meth. Appl. Mech. Engrg.*, 160:101–114, 1998.
- [21] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, 1965.