```
c       program's elapsed time on a quiet system.
c
c       Uncomment for your corresponding platform
c
c       Note: On the SGI the resolution of etime is  1/HZ
c
c     Output
c       time: user+sytem executime time
c     ................................................................

c      SUN -Solaris
       time=etime(tar)

c      HP - HPUX
c      time=etime_(tar)              !f90
c      time=etime_(tar)              !f77

c      COMPAQ - alpha
c      time=etime(tar)

c      CRAY
c      time=tsecnd()

c      IBM
c      time=0.01*mclock()

c      SGI origin
c      time=etime(tar)

       return
       end
```

3.9 Alternative Approach for Handling Indefinite Matrix [3.5]

System of sparse, symmetrical, INDEFINITE simultaneous linear equations have arisen naturally in several important engineering and science application. Tremendous progress has been made in the past years for efficient large-scale solutions of sparse, symmetrical, definite equations. However, for sparse indefinite system of equations, only a few efficient, robust algorithms, and software are available (especially the FORTRAN versions in the public domains). These existing indefinite sparse solvers have been discussed in recent papers [1.9, 3.2, 3.5-3.7].

Major difficulties involved in developing efficient sparse indefinite solvers include the need for pivoting (or 2×2 pivoting) [3.8], criteria for when and how to switch the row(s), effective strategies to predict and to minimize the nonzero fill-in terms etc....

In this work, an alternative approach is proposed for solving system of sparse, symmetrical, indefinite equations. The key idea here is first to transform the original indefinite system into a new (modified) system of symmetrical, "definite" equations. Well-documented sparse definite solvers [1.9, 3.9-3.10] can be conveniently used to obtain the "intermediate solution" (in the "modified" system). This "intermediate" solution is then transformed back into the "original" space to obtain the "original" unknown vector.

To validate the formulas developed in our paper, and to evaluate the numerical performance of our proposed alternative approach, several NASA indefinite system

of equations have been solved (ranging from 51 to 15,637 indefinite equations) on inexpensive workstations. Our preliminary numerical results have indicated that this alterative approach may offer potential benefits in terms of accuracy, reducing incore memory requirements, and even improving the computational speed over the traditional approach when the alternative approach is implemented in a parallel computer environments.

In this section, one considers systems of indefinite equations in the form as shown in Eq. (3.99):

$$[A]\vec{x} = \vec{f} \tag{3.99}$$

where:

$$[A] = \begin{bmatrix} K & a \\ a^T & 0 \end{bmatrix} \tag{3.100}$$

In Eq. (3.100), the symmetrically indefinite matrix [A] has the dimension n × n, and [K], [a] and [o] are sub-matrices. To simplify the discussions, it is assumed (for now) that the lower right sub-matrix [o] has the dimension 1 x 1. Thus, matrix [A] has a zero on its (last) diagonal. Sub-matrix [K], shown in Eq. (3.100), is also symmetrical.

The key ideas here is to transform Eq. (3.99) into a new system, such as

$$[\bar{A}]\vec{x}^* = \vec{f} \tag{3.101}$$

where the new coefficient matrix $[\bar{A}]$ can be computed as

$$[\bar{A}] = [A] + [\Delta A] \tag{3.102}$$

Matrix $[\Delta A]$, shown in Eq. (3.102), should be selected with the following criterias:

a.) $[\Delta A]$ should be sparse, symmetrical and simple

b.) The resulting new matrix $[\overline{A}]$ will have better properties (such as positive definite) as comparing to the original coefficient matrix [A]

Thus, one selects matrix $[\Delta A]$ as :

$$[\Delta A]=\begin{bmatrix} 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ & \ddots & & & & & \\ & & 0 & \cdots & \cdots & 0 & 0 \\ & & & \ddots & & \vdots & \vdots \\ & & & & \ddots & \vdots & \vdots \\ & & & & & 0 & 0 \\ & & & & & & 1 \end{bmatrix} \tag{3.103}$$

Eq. (3.103) can also be expressed as

$$[\Delta A]_{n\times n} = \vec{d}_{n\times 1} * \vec{d}^{T}_{1\times n} \tag{3.104}$$

where:

$$\vec{d} = \begin{Bmatrix} 0 \\ 0 \\ \\ 0 \\ 1 \end{Bmatrix} \tag{3.105}$$

Using Eq. (3.102), Eq. (3.99) can be re-written as:

$$[\overline{A} - \Delta A]\vec{x} = \vec{f} \tag{3.106}$$

Substituting Eq. (3.104) into Eq. (3.106), one obtains:

$$\left[\overline{A} - d_i d_i^T\right]\vec{x} = \vec{f} \tag{3.107}$$

Pre-multiplying both sides of Eq. (3.107) by $\left[\overline{A}\right]^{-1}$, one has

$$\left[I - \overline{A}^{-1} d_i d_i^T\right]\vec{x} = \vec{x}^* \tag{3.108}$$

In Eq. (3.108), [I] is an n x n identity matrix, and $\vec{x}^*$ is given as

$$\vec{x}^* \equiv \left[\overline{A}\right]^{-1} * \vec{f} \tag{3.109}$$

Eq. (3.108) can also be expressed as:

$$\left[I - \vec{p}_i d_i^T\right]\vec{x} = \vec{x}^* \tag{3.110}$$

where:

$$\vec{p}_i \equiv \left[\overline{A}\right]^{-1} * d_i \tag{3.111}$$

or:

$$\left[\overline{A}\right]\ \vec{p}_i \equiv d_i \tag{3.112}$$

From Eq. (3.110), one obtains

$$\vec{x}^* = \vec{x} - p_i d_i^T \vec{x} \tag{3.113}$$

Pre-multiplying Eq. (3.113) by d_i^T, one gets

$$d_i^T \vec{x}^* = d_i^T \vec{x} - d_i^T p_i d_i^T \vec{x} \tag{3.114}$$

Since $d_i^T \vec{x}^*$, and $d_i^T \vec{x}$ are scalar quantities, hence Eq. (3.114) can be re-written as

$$d_i^T \vec{x}^* = (1 - d_i^T p_i) d_i^T \vec{x} \tag{3.115}$$

or

$$d_i^T \vec{x} = \frac{d_i^T \vec{x}^*}{1 - d_i^T p_i} \tag{3.116}$$

From Eq. (3.113), one obtains

$$\vec{x} = \vec{x}^{\,*} + p_i (d_i^T \vec{x}) \tag{3.117}$$

Substituting Eq. (3.116) into Eq. (3.117), one has

$$\vec{x} = \vec{x}^{\,*} + p_i * \frac{d_i^T \vec{x}^{\,*}}{1 - d_i^T p_i} \tag{3.118}$$

A Remark on Eq. (3.118)

Both matrices [A] and $[\bar{A}]$ are assumed to be non-singular. Then, from Eqs. (3.107,3.106,3.102), one obtains the following relationship:

$$[\bar{A}]^{-1} * [\bar{A} - d_i d_i^T] = [\bar{A}]^{-1} * [A]$$

$$[I] - [\bar{A}]^{-1} d_i d_i^T = \text{product of 2 non-singular matrices } [\bar{A}]^{-1} \text{ and } [A]$$

$$[I] - p_i d_i^T = \text{non-singular}$$

Thus, in a more general case, the denominator of Eq. (3.118) will also be NON-SINGULAR. The entire alternative formulation can be summarized in a convenient step-by-step procedure:

Step 0: Use Eq. (3.105) to form $\vec{d}_i$.

Then matrices $[\Delta A]$ and $[\bar{A}]$ can be computed according to Eqs. (3.104) and (3.102), respectively

Step 1: Use Eq. (3.101) to solve for $\vec{x}^{\,*}$

Step 2: Use Eq. (3.112) to compute $\vec{p}_i$.

Step 3: Use Eq. (3.118) to compute $\vec{x}$.

Generalized Alternative Formulation for Indefinite Systems

The alternative formulation presented in the previous section can now be generalized for cases where the original coefficient matrix [A] has multiple zero values for its diagonal terms. In this case, the sub-matrix $[\Delta A]$ will have the following form:

$$[\Delta A]=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \rightarrow i^{th} \text{ row} \\ \rightarrow j^{th} \text{ row} \end{matrix} \tag{3.119}$$

Eq. (3.119) can be expressed as:

$$[\Delta A]=[\Delta A_i]+[\Delta A_j]=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}+\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.120}$$

or

$$[\Delta A]=\sum_{i=1}^{m}[\Delta A_i] \tag{3.121}$$

In Eq. (3.121), m (<<n) represents the total number of zero diagonal terms of the original coefficient matrix $[A]$. Furthermore, Eq. (3.120) can be represented as:

$$[\Delta A]_{n\times n}=[D]_{n\times m}*[D]^T_{m\times n} \tag{3.122}$$

where:

$$[D]=\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.123}$$

Following exactly the same derivations given in the previous section, the "generalized" version of Eqs. (3.112), and (3.118) can be given as:

$$\left[\overline{A}\right]_{n\times n} * [P]_{n\times m} = [D]_{n\times m} \tag{3.124}$$

$$\vec{x} = \vec{x}^{\,*} + [P]_{n\times m} * \left[I_{m\times m} - D^T P\right]^{-1} * [D]^T_{m\times n} * \vec{x}^{\,*}_{n\times 1} \tag{3.125}$$

Remarks:

(a) If $\left[\overline{A}\right]$ is a symmetrical matrix (which it is!), then $\left[\overline{A}\right]^{-1}$ is also symmetrical. One starts with the following identity

$$\left[\overline{A}\right]\left[\overline{A}\right]^{-1} = [I] = \left[\overline{A}\right]^{-1}[A] \tag{3.126}$$

Transposing both sides of Eq. (3.126), one obtaines

$$\left[\overline{A}^{-1}\right]^T \left[\overline{A}\right]^T = [I] \tag{3.127}$$

Since both matrices $\left[\overline{A}\right]$ and $[I]$ are symmetrical, Eq. (3.127) becomes

$$\left[\overline{A}^{-1}\right]^T \left[\overline{A}\right] = [I] \tag{3.128}$$

or

$$\left[\overline{A}^{-1}\right]^T \left[\overline{A}\right] = \left[\overline{A}^{-1}\right]\left[\overline{A}\right] \tag{3.129}$$

Post-multiplying both sides of Eq. (3.129) by $\left[\overline{A}^{-1}\right]$, one has

$$\left[\overline{A}^{-1}\right]^T = \left[\overline{A}^{-1}\right] \tag{3.130}$$

Thus, $\left[\overline{A}^{-1}\right]$ is also a symmetrical matrix.

(b) Matrix product $[D]^T[P]$ is also symmetrical.

From Eq. (3.124), one has

$$[D]^T [P] = [D]^T \left[\overline{A}^{-1}\right][D] \tag{3.131}$$

Transposing both sides of Eq. (3.131), one obtains

$$\left[D^T P\right]^T = [D]^T \left[\overline{A}^{-1}\right]^T [D] \tag{3.132}$$

Utilizing Eq. (3.130), Eq. (3.132) becomes

$$\left[D^T P\right]^T = [D]^T \left[\overline{A}^{-1}\right]^T [D] \tag{3.133}$$

Comparing Eq. (3.131), Eq. (3.133), one concludes

$$[D]^T [P] = \left[D^T P\right]^T \tag{3.134}$$

Thus, the product of matrices $D^T P$ is also symmetrical.

(c) The orders of computations of Eq. (3.125) should be done as following

1^{st} Step: Compute $\vec{v} = [D]^T \vec{x}^*$ (3.135)

2^{nd} Step: Let $[Z] \equiv I_{m\times m} - D^T P$ (3.136)

3^{rd} Step: Let $[Z]^{-1} \vec{v} \equiv \vec{y}$ (3.137)

or $[Z]\vec{y} = \vec{v}$ (3.138)

The system of equations, shown in Eq. (3.138), in general is dense (but still is symmetrical) and can be solved for the unknown vector $\vec{y}$.

4^{th} Step: $\vec{x} = \vec{x}^* + [P]\vec{y}$ (3.139)

(d) Out-of core memory can be used to store matrix [P]. Since the matrix [P], in general, could be large (especially true when m is large), thus, block of L columns (where L<<m<<n) for matrix [P] could be solved, and stored in the out-of-core fashion from Eq. (3.124). The specific value of L can be either specified by the user, or can be automatically calculated by the computer program (based on the knowledge of available incore memory). Later on, block of L columns can be retrieved into the core memory for the computation of $\vec{x}$ in Eq. (3.125).

(e) Overhead computational costs for this alternative formulation is essentially occurred in the forward and backward solution phase to solve for matrix [P]. The number of right-hand-side columns of matrix [D] is m. In the parallel computer environments, the overhead costs can be reduced to a "single" forward and backward solution, since each processor will be assigned to solve for 1 column of the matrix [D].

(f) The computational (overhead) cost for vector $\vec{y}$ (see Eq. 3.138) is insignificant, since the dimension of matrix [Z] is small, and the right-hand-side of Eq. (3.138) is a "single" vector.

(g) In practice, the value of max. A_{ii} (for i = 1,2,..., n) should be used in Eq. (3.119), instead of the value. 1. Furthermore, the alternative formulation can be made even more stable by using the Gerschgorin's theorem to determine the precise (diagonal) locations and the precise values to be used for diagonals terms of matrix $[\Delta A]$ in Eq. (3.119). Let Z_i denote the circle in the complex plane with center a_{ii} (= diagonal terms of matrix [A] whose eigenvalues are sought) and radius r_i. Thus, the eigenvalues of [A] will fall into those circles

$$Z_i = \{z \in C | z - a_{ii}| \le r_i\} \tag{3.140}$$

where

$$r_i = \sum_{j=1;\, j\neq i}^{n} |a_{ij}| \quad \text{for I=1,2,…,n} \tag{3.141}$$

Hence:

$$a_{ii} - r_i \le Z_i \le a_{ii} + r_i \tag{3.142}$$

From Eq. (3.142), one can estimate the number of NEGATIVE and/or POSITIVE eigenvalues of matrix [A]. For example, if $a_{ii} - r_i > 0$, then the eigenvalues associated with the circle Z_i must be positive. Adding a positive number δ to the diagonal term of [A] will shift the value of the eigenvalue associated with the circle Z_i. For example, adding δ to make

$$a_{ii} - r_i + \delta_i > 0 \tag{3.143}$$

will ensure that the eigenvalue associated with the circle Z_i to be positive. From Eq. (3.143), one may add

$$\delta_i = r_i - a_{ii} > 0 \tag{3.144}$$

or, to be even safer

$$\delta_i = r_i - a_{ii} > \in \quad \text{where } \in \text{ is a small positive value} \tag{3.145}$$

to the diagonal term a_{ii} so that all eigenvalues will be positive (and therefore, the new matrix $|\overline{A}|$ will be positive definite).

A simple example is given in the following paragraphs to clarify the above discussions. The matrix [A] is given as

$$[A] = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & -4 \end{bmatrix} \tag{3.146}$$

From the data shown in Eq. (3.146), one computes:

$$\left.\begin{aligned} a_{11} = 4; r_1 = 1 \text{and} \delta_1 = r_1 - a_{11} = -3 \\ a_{22} = 0; r_2 = 2 \text{and} \delta_2 = r_2 - a_{22} = 2 \\ a_{33} = -4; r_3 = 2 \text{and} \delta_3 = r_3 - a_{33} = 6 \end{aligned}\right\} \tag{3.147}$$

Let $\in$ =1, it is then suggested to add positive values $\delta_2 = 2, \text{and} \delta_3 = 6$ to the second, and third diagonal terms of [A]. The new matrix $|\overline{A}|$ is therefore defined as

$$[\overline{A}] = [A] + [\Delta A]$$

$$[\overline{A}] = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & -4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix} \tag{3.148}$$

$$[\overline{A}] = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

The 3 eigenvalues associated with matrices [A] and $|\overline{A}|$ are λ_A = {-4.2146, -0.0556, 4.2702} and $\lambda_{\overline{A}}$ = {4.618, 2.382, 1.000}, respectively.

(h) Major advantages of the generalized alternative formulation are:
- Pivoting strategies are NOT required, hence the algorithm should be more robust and better solution accuracy can be expected.
- Incore memory requirements can be reduced, and can be predicted before entering the numerical factorization phase.
- Any positive definite sparse solvers can be integrated into this alternative formulation.

(i) In actual computer coding, there is no need to "push" all zero diagonal terms of [A] to the bottom right of [A]. Furthermore, the sub-matrix [o] shown in Eq. (3.100) does NOT have to be completely zero. In other words, the off-diagonal terms of this sub-matrix [o] may, or may not be zero (only the diagonal terms of sub-matrix [o] are zero).

Numerical Applications

Based upon the proposed alternative formulation presented in previous sections, several bench-mark indefinite system of equations (obtained from NASA Langley Research Center) have been used to validate the proposed algorithms. The sizes (= Neq = Number of equations), the original number of non-zero, off-diagonal coefficients of matrix [A] (= Ncoff) are given in Table 3.12. In the partitioned form, Eq. (3.99) can also be expressed as (please refer to Eq. 3.100):

$$\begin{bmatrix} K & a \\ a^T & 0 \end{bmatrix} \begin{Bmatrix} \vec{x}_u \\ \vec{x}_\lambda \end{Bmatrix} = \begin{Bmatrix} \vec{f}_u \\ \vec{f}_\lambda \end{Bmatrix} \tag{3.149}$$

For structural mechanic applications, the vector $\vec{x}_u$ may represent the displacements, where-as the vector $\vec{x}_\lambda$ may represent the Lagrange multipliers associated with the interfaced problems. Solution accuracy of the proposed algorithms can be established by comparing the following quantities with Boeing's sparse indefinite equation solver:

1. Maximum absolute displacement (of $\vec{x}_u$, shown in Eq. 3.149)

2. Summation of absolute displacements (of $\vec{x}_u$, shown in Eq. 3.149)

3. Relative error norm (Rel Err = $\dfrac{\left\| A\vec{x} - \vec{f} \right\|}{\left\| \vec{f} \right\|}$, shown in Eq. (3.99)

The above quantities are also included in Table 3.12. It should be emphasized that CPU time comparisons are NOT yet included in this study, due to the following reasons:

(a) Boeing's sparse indefinite solver timing has been implemented earlier on the Cray-YMP supercomputer. However, the author' FORTRAN code has been recently tested on Old Dominion University (ODU) Sun (= Norfolk) workstation, since the author currently has no access to the Cray-YMP,nor Boeing's source code.
(b) The reordering algorithms, such as Modified Minimum Degree (MMD) or Nested Disection (ND) have NOT yet been integrated into the current version of the author' FORTRAN code.
(c) Parallel processing for $[\overline{A}][P] = [D]$ = multiple RHS has not been done.

Table 3.12 Comparison of ODU and Boeing Indefinite Sparse Solvers

Neq	Ncoff	$\sum_i \lvert x_{u_i} \rvert$	$\text{Max} \lvert x_{u_i} \rvert$	Rel Err	Time (ODU-Norfolk)
51(Boeing)	218	$2.265*10^{-2}$ ($2.265*10^{-2}$)	$2.000*10^{-3}$ ($1.999*10^{-3}$)	$4.68*10^{-6}$ ($7.0*10^{-14}$)	0.0sec N/A
247(Boeing)	2009	3.16 (3.16)	0.1525 (0.1525)	$2.63*10^{-10}$ ($4.03*10^{-10}$)	0.1sec N/A
1440(Boeing)	22137	29.68 (29.68)	0.20289 (0.20289)	$4.27*10^{-11}$ ($3.26*10^{-10}$)	8.7sec N/A
7767(Boeing)	76111	113.71 (113.71)	0.1610576 (0.1610576)	$5.31*10^{-8}$ ($6.00*10^{-8}$)	42.7sec N/A
15367(Boeing)	286044	512.35 (512.35)	0.205696 (0.205696)	$9.22*10^{-10}$ ($4.38*10^{-11}$)	5400sec N/A

Conclusions

Alternative formulations and algorithms for solving sparse system of equations have been developed. The proposed numerical algorithms have been implemented and validated through several bench-mark NASA applications. Preliminary results have indicated that the proposed alternative algorithms are quite robust and are in excellent agreements with the Boeing's commercial sparse indefinite solver. Detailed analysis of the proposed sparse indefinite algorithms have suggested that:

(a) The proposed algorithms are quire robust and accurate
(b) The additional (overhead) costs for the proposed algorithms is mainly occurred in the forward and backward solution phase (of the associated positive definite system). This overhead costs can be easily and drastically reduced by performing the forward and backward solution phase (of multiple-right-hand-side vectors) in the parallel computer environments.
(c) In the proposed formulation, one only deals with "positive-definite" sparse systems. Thus, complex and expensive pivoting strategies are NOT required. As the consequences of these desired features, several important advantages can be realized, such as:
    - Incore memory requirements can be easily and efficiently predicted (before entering the sparse numerical factorization phase).
    - The amount of non-zero "fill-in" (and hence, the number of floating operations) can be reduced.
    - Any efficient sparse "positive definite" solvers can be conveniently integrated into the proposed formulations

Efforts are underway to incorporate various reordering algorithms (such as MMD, ND, etc...) into the proposed algorithms and to implement the entire procedure in the MPI parallel computer environments. Additional results will be reported in a near future.

3.10 Unsymmetrical Matrix Equation Solver

Let's consider the following system of unsymmetrical linear equations

$$Ax = b \tag{3.150}$$

where the coefficient matrix A is unsymmetrical and the vectors x and b represent the unknown vector (nodal displacement) and the right-hand side (known nodal load) vector, respectively. In this section, a solver for unsymmetrical matrices where the upper and lower triangular portions of the matrix are symmetrical in locations but unsymmetrical in values (see Figure 3.6) will be developed.