GAUSSIAN ELIMINATION ON A BANDED MATRIX

Jeff Thorson

Abstract

Two FORTRAN routines included in this paper can be used to solve banded linear systems. The routines use a Gaussian elimination algorithm tailored to the specific case of a banded matrix. Instead of the $n^3/3$ multiplies required to reduce a full matrix, a banded matrix can be reduced in about $nm^2/4$ multiplies, where n is the dimension of the matrix and m is its bandwidth. Only the nonzero diagonals of the matrix need to be stored. Algorithm 2 does no pivoting. Algorithm 3 performs partial pivoting. Partial pivoting is inherently stabler than no pivoting at all, though the difference in output between the two algorithms is probably negligible for regular wave equation operators. The algorithm listings contain all relevant documentation for their use.

Introduction

Gaussian elimination is undoubtedly the most widely used method for solving linear equations. The discussion in this section is not meant to be an analysis of the method, but a brief summary. For an excellent treatment on the subject see chapter 2 of Stewart (1973).

Consider the linear system

$$Ax = b$$

where A is an n by n matrix, b is the given right-hand side vector, and x is the unknown vector. Gaussian elimination proceeds to solve for x by first factoring A into a lower triangular matrix L and an upper triangular matrix U. Subsequently x is found by solving the much easier systems Ly = b and Ux = y. That is,

$$A = LU,$$

$$LUx = b,$$

$$Ux = L^{-1}b,$$

$$x = U^{-1}L^{-1}b.$$

The method is presented graphically in figure 1.
Notice that the right-hand side only enters into the solution in steps 2 and 3. The bulk of the work in the algorithm lies in step 1: factoring A into L and U. This takes roughly $n^3/3$ multiplies for a full matrix A. Doing steps 2 and 3 is much cheaper; together they comprise n(n-1) floating multiplies. Therefore, if the matrix A, i.e. the operator, is to be applied to a number of different right-hand sides, the LU decomposition need only be done once. The routines given later take advantage of this property. Each is divided into two subroutines: 'band,' which performs the LU decomposition, and 'solve,' which solves for x.

Pivoting

By examining the algorithm below, it can be seen that only n divisions need to be made, one for each iteration of the outermost loop. The divisors are called ***pivots***. Nominally, the pivots are the diagonal elements of A. However, in the course of the calculation the current pivot may by chance become very small with respect to the other elements. Once this happens the accuracy of the reduction degrades rapidly. To avoid this problem, the rows of the linear system can be permuted to bring another, larger element into the position of the diagonal element. Obviously, exchanging rows in A does not affect the

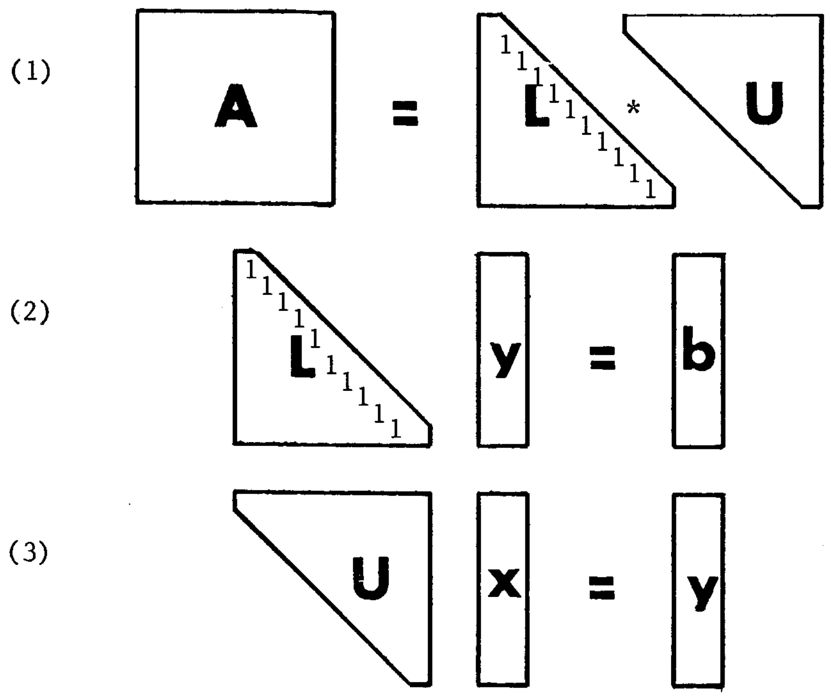

FIG. 1. The stages in solving Ax = b. By convention, L is unit lower triangular (1's on the diagonal) whereas the diagonal elements of U are arbitrary.

answer, as long as the matching rows in b are also exchanged. The banded property of A may disappear, however.

Partial pivoting involves examining the elements in the column beneath the current diagonal, choosing the largest one, and interchanging the two rows to bring it into position as the new pivot. If the largest element found happens to be near zero, it indicates the original matrix is nearly singular.

Algorithm 1, written in pseudo-FORTRAN, outlines Gaussian elimination on a full matrix with partial pivoting applied. L and U are overwritten onto A. L fits in the lower triangular half of A, while U fits into the upper triangular half. L has an implicit unit diagonal (which is not stored). The diagonal of A holds 1/U(k,k). P(n) is an integer vector that holds pivoting indices. Finally, the solution x overwrites the initial right-hand side b.

```
Algorithm 1: GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
-------------------------------------------------------

Given A(n,n)
LU Decomposition:
For k = 1,n
        Find p(k) >= k such that
        abs(A(p(k),k)) = max { abs(A(i,k)) : i >= k }
        If A(p(k),k) = 0, A is singular. Exit.
        For j = k,n
                Interchange A(k,j), A(p(k),j)
        Continue
        A(k,k) = 1/A(k,k)
        For i = k+1,n
                A(i,k) = A(i,k) * A(k,k)
                For j = k+1,n
                        A(i,j) = A(i,j) - A(i,k)*A(k,j)
                Continue
        Continue
Continue

Given b(n)
Solve:
For k = 1,n-1
        Interchange b(k), b(p(k))
        For i = k+1,n
                b(i) = b(i) - A(i,k)*b(k)
        Continue
Continue
For k = n,1 (decrement)
        For i = k+1,n
                b(k) = b(k) - A(k,i)*b(i)
        Continue
        b(k) = b(k) * A(k,k)
Continue
End
```

Banded Matrices

One can take advantage of the distribution of zeros in a banded matrix in two ways. First, index limits can be shortened, ensuring that iterations are only made over nonzero terms. Second, only nonzero diagonals of A need be stored. The two FORTRAN programs below, algorithms 2 and 3, are designed to solve banded matrix systems. They follow the logic of algorithm 1 closely, but with different indexing and storage. Algorithm 2 performs Gaussian elimination without pivoting. A is stored as a series of diagonal vectors, and its LU decomposition fits nicely into the original storage for A. Algorithm 3 is a version implementing partial pivoting. Unlike the non-pivoting version, the LU decomposition may not fit in the original space allocated to A -- the row switching done by pivoting introduces nonzero elements into A that lie outside the band. Figure 2 illustrates this. An additional space of n by half the bandwidth must be allocated to A to hold these nonzero elements.

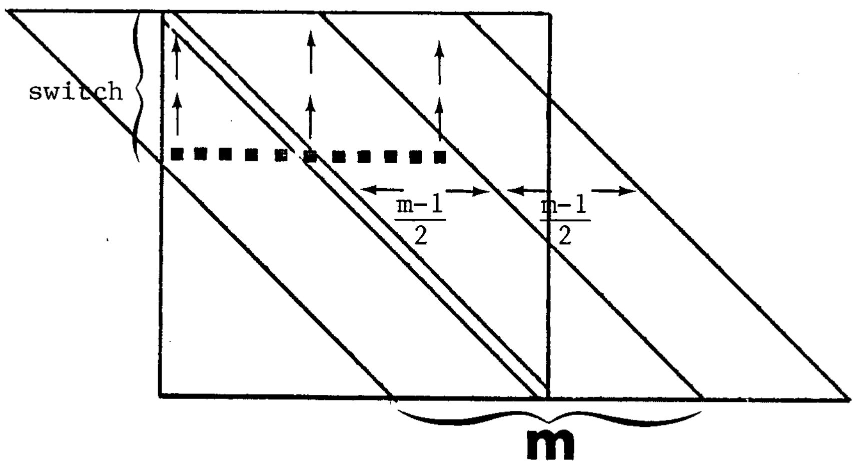

FIG. 2. An extra (m-1)/2 vectors are required to store the permuted rows of A. Two rows no farther apart than (m-1)/2 can be interchanged. Any rows farther apart than this can't be interchanged, for the corresponding pivot is always zero. Notice if m > 2(n-1)/3 + 1, the banded form requires more storage than the original matrix!

Algorithm 2 (without pivoting) requires (m+1)n complex storage points, and performs approximately $n(m+1)^2/4$ multiplications. In comparison, algorithm 3 (with pivoting) requires 3/2 as much storage, and performs 9/4 as many multiplications.

Is there any advantage in using one algorithm over using the other? Pivoting is numerically more stable, and will fail only when the matrix itself is close to singularity. When does the non-pivoting case break down? The answer is given in a theorem in Stewart (1973, p. 120), which states that a pivot A(k,k) will be nonzero if and only if the leading principle submatrices of A are nonsingular. In many applications the matrix A represents an operator derived by some finite differencing scheme. This theorem implies that if the operator is well defined no matter how coarse the discrete grid on which it is applied, a non-pivoting algorithm should never break down. Most wave equation operators fall into this category: they are diagonally dominant and "look the same" on any grid scale. The time factor saved in applying the non-pivoting algorithm will be approximately 2 to 1. One should be aware, however, that if the conditions of the theorem above do not hold, elimination without pivoting will definitely produce garbage.

REFERENCE

Stewart, G.W., 1973, Introduction to Matrix Computations: New York, Academic Press.

Algorithm 2: GAUSSIAN ELIMINATION ON A BANDED MATRIX

```
c       -------------------------------------------
c       Subroutine band
c       This subroutine performs an LU decomposition
c       on  a banded matrix A. A is overwritten with
c       L,U. L is unit lower triangular, U is  upper
c       triangular.
c       A(i,j) <-- U(i,j) for j .gt. i,
c       A(i,i) <-- Inverse of U(i,i),
c       A(i,j) <-- L(i,j) for j .lt. i.
c       (The unit diagonal of L is implicit.)
c       The accompanying subroutine 'solve' uses the
c       output  of  this routine to solve for a par-
c       ticular right-hand side.
c
c       Storage:
c       The diagonals of A (and L,U) are  stored  in
c       the input array a(n,m) where n is the dimen-
c       sion of the system and m is the bandwidth. m
c       must  be  odd. The columns of a(n,m) are as-
c       sumed to be symmetrically placed  about  the
c       central diagonal of A.  The relation between
c       elements of 'A' and 'a' is:
c                A(i,j) = a(i,j-i+(m+1)/2).
c
c       Pivots always lie on the main diagonal. If a
c       very  small diagonal element is encountered,
c       m is returned zero.  Caution:  small  pivots
c       may give inaccurate results.
c       -------------------------------------------
        subroutine band(a,m,n)
        integer g,h,i,j,k,m,n,r
        complex a(n,m)
        real eps

        r = (m+1)/2
        eps = 1.0e-6

        do 20 k = 1,n
        if( cabs(a(k,r)) .le. eps ) goto 99
        a(k,r) = 1.0/a(k,r)
            h = r-1
            i = k+1
10          if(h.lt.1 .or. i.gt.n) goto 20
            a(i,h) = a(i,h) * a(k,r)
                j = h+1
                g = r+1
30              if(g.gt.m .or. j.gt.(r+n-i)) goto 40
                a(i,j) = a(i,j) - a(i,h)*a(k,g)
                j = j+1
                g = g+1
                goto 30
40            continue
              i = i+1
              h = h-1
              goto 10
20        continue
          return

99        m = 0
          return
          end

c         ------------------------------------------
c         Subroutine solve
c         Solves the system Ax = b  given a right-hand
c         side b.  Solution x is overwritten onto vec-
c         tor b.  a(n,m) contains the LU factored form
c         of A generated by subroutine band.
c         ------------------------------------------
          subroutine solve(a,b,m,n)
          integer i,j,k,m,n,r
          complex a(n,m),b(n)

          r = (m+1)/2

c         Forward elimination

          do 100 k = 1,n-1
              i = k+1
              j = r-1
110           if(j.lt.1 .or. i.gt.n) goto 100
              b(i) = b(i) - a(i,j)*b(k)
              i = i+1
              j = j-1
              goto 110
100       continue

c         Back substitution

          do 120 k = n,1,-1
              i = k+1
              j = r+1
130           if(j.gt.m .or. i.gt.n) goto 140
              b(k) = b(k) - a(k,j)*b(i)
              i = i+1
              j = j+1
              goto 130
140       continue
          b(k) = b(k) * a(k,r)
```

```
120     continue
        return
        end
```

Algorithm 3: GAUSSIAN ELIMINATION ON A BANDED MATRIX
             PARTIAL PIVOTING

```
-----------------------------------------------------

c       --------------------------------------------
c       Subroutine band --  Partial pivoting version
c
c       This subroutine performs an LU decomposition
c       on  a banded matrix A. A is overwritten with
c       L,U. L is unit lower triangular, U is  upper
c       triangular.
c       A(i,j) <-- U(i,j) for j .gt. i,
c       A(i,i) <-- Inverse of U(i,i),
c       A(i,j) <-- L(i,j) for j .lt. i.
c       (The unit diagonal of L is implicit.)
c       The accompanying subroutine 'solve' uses the
c       output  of  this routine to solve for a par-
c       ticular right-hand side.
c
c       Storage:
c       The diagonals of A (and L,U) are  stored  in
c       the input array a(n,m) where n is the dimen-
c       sion of the system and m is the bandwidth. m
c       must  be  odd. The columns of a(n,m) are as-
c       sumed to be symmetrically placed  about  the
c       central diagonal of A.  The relation between
c       elements of 'A' and 'a' is:
c                A(i,j) = a(i,j-i+(m+1)/2).
c
c       Pivoting:
c       Pivots are selected from  the  column  below
c       the  current  diagonal  element.  Row inter-
c       change information is stored in the  integer
c       vector  p(n),  which  is  used by subroutine
c       solve. The horizontal dimension of 'a'  must
c       be  at least m + (m-1)/2 : the extra (m-1)/2
c       superdiagonals of A provide  space  for  the
c       interchanged  rows. If a very small pivot is
c       encountered,  m  is  returned  zero.   Small
c       pivots indicate a near singular matrix.
c       --------------------------------------------
        subroutine band(a,m,n,p)
        integer g,h,i,j,k,m,n,p(n),r
        complex a(n,1),c
        real eps,max,d
```

```
              r = (m+1)/2
              eps = 1.0e-10
              do 10 i = 1,n
              do 10 j = m+1,m+r-1
              a(i,j) = cmplx(0.0,0.0)
      10      continue

              do 20 k = 1,n

      c       Find pivots

              max = 0.0
                  i = k
                  j = r
      25          if(i.gt.n .or. j.lt.1) goto 30
                  d = cabs(a(i,j))
                  if( max .ge. d ) goto 35
                  max = d
                  p(k) = i
      35          i = i+1
                  j = j-1
                  goto 25
      30      continue
              if( max .le. eps ) goto 99

      c       Switch pivot rows

              if( p(k) .eq. k ) goto 40
                  i = r
                  j = r+k-p(k)
      50          if(i.gt.m+r-1 .or. i.gt.n-k+r) goto 40
                  c = a(k,i)
                  a(k,i) = a(p(k),j)
                  a(p(k),j) = c
                  i = i+1
                  j = j+1
                  goto 50
      40      continue

      c       Decompose A

              a(k,r) = 1.0/a(k,r)
                  h = r-1
                  i = k+1
      60          if(h.lt.1 .or. i.gt.n) goto 20
                  a(i,h) = a(i,h) * a(k,r)
                      j = h+1
                      g = r+1
      70              if(g.gt.m+r-1 .or. j.gt.n+r-i) goto 80
                      a(i,j) = a(i,j) - a(i,h)*a(k,g)
                      j = j+1
```

```
                g = g+1
                goto 70
80          continue
            i = i+1
            h = h-1
            goto 60
20      continue
        return

99      m = 0
        return
        end

c       ---------------------------------------------
c       Subroutine solve -- Partial pivoting version
c
c       Solves the system Ax = b given a  right-hand
c       side b.  Solution x is overwritten onto vec-
c       tor b.  a(n,m) contains the LU factored form
c       of A generated by subroutine band. p(n) con-
c       tains the pivoting information  returned  by
c       band.
c       ---------------------------------------------
        subroutine solve(a,b,m,n,p)
        integer i,j,k,m,n,p(n),r
        complex a(n,1),b(n),c

        r = (m+1)/2

c       Forward elimination

        do 100 k = 1,n-1
        c = b(k)
        b(k) = b(p(k))
        b(p(k)) = c

            i = k+1
            j = r-1
110         if(j.lt.1 .or. i.gt.n) goto 100
            b(i) = b(i) - a(i,j)*b(k)
            i = i+1
            j = j-1
            goto 110
100     continue

c       Back substitution

        do 120 k = n,1,-1
            i = k+1
            j = r+1
130         if(j.gt.m+r-1 .or. i.gt.n) goto 140
```

```
              b(k) = b(k) - a(k,j)*b(i)
              i = i+1
              j = j+1
              goto 130
140       continue
          b(k) = b(k) * a(k,r)
120       continue
          return
          end
```