# An AP Program for Gaussian Elimination of Banded Complex Matrices

*Bert Jacobs*


## Abstract

High order migration schemes require solving a large number of complex, banded systems of linear equations. A program for Gaussian elimination with partial pivoting written in array processor machine language helps considerably in keeping computational costs in line when such schemes are used.


## Gaussian Elimination for Banded Matrices

The program coded up is like that discussed by Thorson in SEP-20. It is a two-part algorithm in which the first step is an in-place factorization of the matrix of coefficients into lower and upper triangular factors with row interchanges. The row interchanges are stored in a separate vector. The second step is a back substitution which leaves the factorization of the first step untouched. The two parts are separate subroutines to allow the user to solve the system of equations with a different right hand side vector without having to factor the left side more than once.

The algorithm does not look for zero pivots and will fail catastrophically if the solution of a singular system of linear equations is attempted.

Four listings are given. The first is a Fortran version of the triangularization routine like that published by Thorson. All checks for zero divisors and small pivots were eliminated to make the Fortran and the machine language versions compatible. The next listing is that of a triangularization routine written in Floating Point System's APAL Assembly Language, usable on an AP-120B with slow memory. The third and fourth listings are Fortran and APAL versions, respectively, of the back substitution routine used to solve a linear system of equations after triangularization of the matrix of coefficients.

## REFERENCES

FPS Technical Publications Staff (1978), Programmers Reference Manual (Publication No. FPS-7319), Beaverton, Oregon, Floating Point Systems, Inc.

Thorson, J., A Complex Tridiagonal Matrix Solver: SEP-15, pp. 275-82.

Thorson, J., Gaussian Elimination on a Banded Matrix: SEP-20, pp.143-54.

```
"       This subroutine performs an LU decomposition on a banded matrix
"       matrix A.  A is overwritten with L,U.  L is unit lower triangular
"       and U is upper triangular.  The diagonals of A (and L,U) are
"       stored in the input array a(n,m) where n is the dimension of the
"       system and m is the bandwidth.  m must be odd.  The columns of
"       a(n,m) are assumed to be symmetrically placed about the central
"       diagonal of A.  The relation between elements of 'A' and 'a' is:
"                 A(i,j) = a(i,j-i+(m+1)/2)
"       Pivots are selected from the column below the current diagonal
"       element.  Row interchange information is stored in the integer
"       vector p(n), which is used by the subroutine solve.  The horizon-
"       tal dimension of 'a' must be at least m+(m-1)/2, where the extra
"       (m-1)/2 superdiagonals of A provide space for the interchanged
"       rows.
"
"       subroutine band(a,ml,m,n,pk)                 FORTRAN VERSION OF
"       integer g,h,i,j,k,m,n,pk(1),r                TRIANGULARIZATION
"       complex*8 a(256,7),c
"       real*4 max,d
"       r = (m + 1)/2
"       do 02 i = 1,n
"       do 04 j = m+1,m+r-1
"       a(i,j) = cmplx(0.0,0.0)
"04     continue
"02     continue
"       do 06 k = 1,n-1
"       max = 0.0
"       i = k
"       j = r
"08     if(.not.(i .le. n .and. j .ge. 1))goto 09
"       d = abs(a(i,j))
"       if(.not.(max .lt. d))goto 10
"       max = d
"       pk(k) = i
"10     continue
"       i = i + 1
"       j = j - 1
"       goto 08
"09     continue
"       if(.not.(pk(k) .ne. k))goto 14
"       i = r
"       j = r + k - pk(k)
"16     if(.not.(i .le. m+r-1 .and. i .le. n-k+r))goto 17
"       c = a(k,i)
"       a(k,i) = a(pk(k),j)
"       a(pk(k),j) = c
"       i = i + 1
"       j = j + 1
"       goto 16
"17     continue
"14     continue
"       a(k,r) = 1.0/a(k,r)
"       h = r - 1
"       i = k + 1
"18     if(.not.(h .ge. 1 .and. i .le. n))goto 19
"       a(i,h) = a(i,h)*a(k,r)
"       j = h + 1
"       g = r + 1
"20     if(.not.(g .le. m+r-1 .and. j .le. n+r-i))goto 21
"       a(i,j) = a(i,j) - a(i,h)*a(k,g)
"       j = j + 1
"       g = g + 1
"       goto 20
"21     continue
```

```
"           i = i + 1
"           h = h - 1
"           goto 18
"19         continue
"06         continue
"           a(n,r) = 1.0/a(n,r)
"           return
"           end
"
$title cband                                            "APAL VERSION OF
$entry cband,4                                          "TRIANGULARIZATION
$ext div,spmul,spflt
"Div uses sp(13-15),dpx(0,1),dpy(0),fa,fm,tm: dpx(0)=dpy(0)/dpx(0)
"Spmul uses fa,fm,dpx(0): sp(0)=sp(0)*sp(1)
"Spflt uses sp(14-15),dpx(0,1),fa,tm: dpx(1) = float(sp(15))
"Host:  call cband(aptr,n,m,pptr)

n $equ 2
aptr $equ 3
m $equ 4
r $equ 5
i $equ 6
j $equ 7
k $equ 10           "10 octal is 8
pk $equ 11          "11 octal is 9
h $equ 11
abas $equ 12        "12 octal is 10
pptr $equ 13        "13 octal is 11
l1 $equ 14          "14 octal is 12
l2 $equ 15          "15 octal is 13   crunched by div
l3 $equ 16          "16 octal is 14   crunched by div
g $equ 17           "17 octal is 15   crunched by div

cband:    mov 3,pptr
          mov 2,m
          mov 1,n
          mov 0,aptr
          mov m,r
          incr r                      "r = (m+1)/2

          mov m,j
          inc j
zl1:      mov m,l1                    "do j = m+1,m+r-1
          add r,l1
          dec l1
          sub# j,l1
          bge aa
          jmp zbr1
aa:       mov j,0
          dec 0
          mov n,1
          jsr spmul
          movl 0,l2
          add aptr,l2
          clr i
          inc i
zl2:      sub# i,n                    "do i = 1,n
          bge bb
          jmp zbr2
bb:       mov l2,l2; setma; mi<zero           "a(i,j) = 0.0
          inc l2
          incma; mi<zero; inc l2
          inc i
          jmp zl2
```

```
zbr2:     inc j
          jmp zl1

zbr1:     clr k
          inc k


kl:       sub# k,n                          "do k = 1,n-1
          bgt cc
          jmp kbr
cc:       dpx(-4)<db; db=zero; mov k,i
          mov r,j                   "initialize max <- 0.0,i <- k,j <- r
          mov j,0
          dec 0
          mov n,1
          jsr spmul
          mov 0,l2
          add i,l2
          decl l2                   "address of a(i,j) in l2
          add aptr,l2

maxl:     sub# i,n                  "while i<=n and j>=1
          bge dd
          jmp maxbr
dd:       dec# j
          bge ee
          jmp maxbr
ee:         mov l2,l2; setma          "a(i,j) in dpx(-3),dpy(-3)                    ee
          sub n,l2
          incma; sub n,l2
          dpx(-3)<md; dpy(-3)<md; inc l2
          fmul dpx(-3),dpy(-3); inc l2
          dpy(-2)<md; dpx(-2)<md; fmul
          fmul dpx(-2),dpy(-2)
          fmul; dpx(-1)<fm
          fmul; dec j
          fadd fm,dpx(-1)
          fadd
          dpy(-1)<fa
          fsub dpx(-4),dpy(-1)
          fadd
          nop
          bfge gg                              "pk(k) = i
          mov i,pk; dpx(-4)<db; db<dpy(-1)     "max= ||a(i,j)||**2
gg:       inc i
          jmp maxl

maxbr:    sub# pk,k                 "while pk(k) != k
          bne hh
          jmp swbr
hh:       mov r,i
          mov r,j
          add k,j
          sub pk,j
          mov i,0
          dec 0
          mov n,1
          jsr spmul
          mov 0,l2
          add k,l2                  "address of a(k,i) in l2
          decl l2
          add aptr,l2
          mov j,0
          dec 0
```

...ee

```
          mov n,1
          jsr spmul
          mov 0,l3
          add pk,l3                 "address of a(pk(k),j) in l3
          decl l3
          add aptr,l3

swl:      mov m,l1
          add r,l1
          dec l1
          sub# i,l1                 "while i <= m+r-1
          bge ii
          jmp swbr
ii:       mov n,l1
          sub k,l1
          add r,l1
          sub# i,l1                 "while i <= n-k+r
          bge jj
          jmp swbr
jj:       mov l2,l2; setma
          nop
          incma
          dpx(-4)<md                "a(k,i) in dpx(-4),dpy(-4)
          mov l3,l3; setma
          dpy(-4)<md
          incma
          dpx(-3)<md; inc j         "a(pk,j) in dpx(-3),dpy(-3)
          mov l3,l3; setma; mi<dpx(-4)
          dpy(-3)<md; add n,l3      "swap a(k,i) and a(pk,j)
          incma; mi<dpy(-4); add n,l3
          inc i
          mov l2,l2; setma; mi<dpx(-3)
          add n,l2
          incma; mi<dpy(-3); add n,l2
          jmp swl

swbr:     mov r,0
          dec 0
          mov n,1
          jsr spmul
          mov 0,l1
          add k,l1                  "address of a(k,r) in l1
          decl l1                   "store in dpx(-4),dpy(-4)
          add aptr,l1; setma        "  and in dpy(-3),dpx(-3)
          nop
          incma
          dpx(-4)<md; dpy(-3)<md
          fmul dpx(-4),dpy(-3)
          fmul; dpy(-4)<md; dpx(-3)<md
          fmul dpy(-4),dpx(-3)
          fmul; dpx(0)<fm
          fmul
          fadd fm,dpx(0)
          fadd                               "|a(k,r)|**2 in dpx(0)
          dpx(0)<fa; ldspi 0; db=!one        "1.0 in dpy(0) from table memory
          settma
          nop
          dpy(0)<tm
          jsr div                   "1/(|a(k,r)|**2) in dpx(0)
          fmul dpx(0),dpy(-4)
          fmul dpx(0),dpy(-3)
          fmul
          fmul; fsubr fm,zero
          fadd; mov l1,l1; setma; mi<fm; dpx(-4)<fm
```

...ee

```
          dpy(-4)<fa
          incma; mi<dpy(-4)          "1/a(k,r) in memory
                                     "new a(k,r) in dpx(-4),dpy(-4)

          mov pk,17                  "store pk(k) in memory
          jsr spflt
          mov k,l1
          dec l1
          add pptr,l1; setma; mi<dpx(1)

          mov r,h                    "S-pad pk free
          dec h
          mov k,i
          inc i
mull:     sub# i,n                   "while h >= 1 and i <= n
          bge kk
          jmp mulbr
kk:       dec# h
          bge ll
          jmp mulbr
ll:       mov h,0
          dec 0
          mov n,1
          jsr spmul
          mov 0,l1
          add i,l1                   "a(k,r) still in dpx(-4),dpy(-4)
          decl l1                    "a(i,h) in dpx(-3),dpy(-3)
          add aptr,l1; setma         "  and in dpy(-2),dpx(-2)
          nop
          incma
          dpx(-3)<md; dpy(-2)<md
          fmul dpy(-2),dpx(-4)
          dpy(-3)<md; dpx(-2)<md; fmul dpx(-3),dpy(-4)
          fmul dpx(-2),dpy(-4)
          fmul dpy(-3),dpx(-4); dpx(0)<fm
          fmul; dpy(0)<fm
          fmul; fsubr fm,dpx(0); mov h,j
          fadd fm,dpy(0); inc j
          dpx(-3)<fa; dpy(-2)<fa; fadd; mov r,g
          dpy(-3)<fa; dpx(-2)<fa; inc g
          mov l1,l1; setma; mi<dpx(-3)
          mov g,0
          dec 0; incma; mi<dpy(-3)
          mov n,1
          jsr spmul
          mov 0,l3
          add k,l3
          decl l3
          add aptr,l3                "address of a(k,g) in l3
          mov j,0
          dec 0
          mov n,1
          jsr spmul
          mov 0,l2
          add i,l2
          decl l2
          add aptr,l2; setma         "address of a(i,j) in l2
          nop                        "a(i,j) in dpx(-1),dpy(-1)
          incma
          dpx(-1)<md
          nop
          dpy(-1)<md
hl:       mov m,l1
          add r,l1
```

...ee

```
        dec l1
        sub# g,l1
        bge mm
        jmp hbr
mm:     mov n,l1            "while g<=m+r-1 and j<=n+r-i
        add r,l1
        sub i,l1
        sub# j,l1
        bge nn
        jmp hbr
nn:     mov l3,l3; setma
        add n,l3
        incma; add n,l3
        dpx(0)<md           "a(k,g) in dpx(0),dpy(0)
        mov l2,l2; setma; fmul dpx(0),dpy(-2)
        dpy(0)<md; fmul dpx(0),dpy(-3)
        incma; fmul dpy(0),dpx(-2)
        dpx(-1)<md; fmul dpy(0),dpx(-3); dpy(1)<fm
        fmul; dpx(1)<fm               "a(i,j) in dpx(-1),dpy(-1)
        dpy(-1)<md; fmul; fsubr fm,dpy(1)
        fadd fm,dpx(1)
        fsub dpx(-1),fa; inc j
        fsub dpy(-1)<fa; inc g
        fadd; mov l2,l2; setma; mi<fa
        dpy(1)<fa; add n,l2
        incma; mi<dpy(1); add n,l2
        jmp hl

hbr:    inc i
        dec h
        jmp mull

mulbr:  inc k
        jmp kl

kbr:    mov r,0
        dec 0
        mov n,1
        jsr spmul
        mov 0,l1
        add n,l1
        decl l1
        add aptr,l1; setma                "a(n,r) in dpx(-4),dpy(-4)
        nop                               "  and dpy(-3),dpx(-3)
        incma
        dpx(-4)<md; dpy(-3)<md
        fmul dpx(-4),dpy(-3)
        fmul; dpy(-4)<md; dpx(-3)<md
        fmul dpy(-4),dpx(-3)
        fmul; dpx(-2)<fm
        fmul
        fadd fm,dpx(-2)
        fadd
        dpx(0)<fa
        ldspi 0; db=!one
        settma
        nop
        dpy(0)<tm
        jsr div
        fmul dpy(-4),dpx(0)
        fmul dpy(-3),dpx(0)
        fmul
        fsubr fm,zero; fmul
        mov l1,l1; setma; mi<fm
```

...ee

```
          fadd
          incma; mi<fa

          return
$end
"
"
"         This subroutine performs partial pivoting back
"         substitution.  It solves the system Ax=b given a right-
"         hand side b.  The solution x is overwritten on the
"         vector b.  a(n,m) contains the LU factored form of A
"         generated by the subroutine band.  p(n) contains the
"         pivoting information returned by subroutine band.
"
"         subroutine solve(a,b,m,n,pk)              FORTRAN VERSION OF
"         integer i,j,k,m,n,pk(1),r                 BACK-SUBSTITUTION
"         complex*8 a(256,7),b(1),c
"         r = (m+1)/2
"         do 24 k = 1,n-1
"         if(.not.(pk(k) .ne. k))goto 26
"         c = b(k)
"         b(k) = b(pk(k))
"         b(pk(k)) = c
"26       continue
"         i = k + 1
"         j = r - 1
"28       if(.not.(j .ge. 1 .and. i .le. n))goto 29
"         b(i) = b(i) - a(i,j)*b(k)
"         i = i + 1
"         j = j - 1
"         goto 28
"29       continue
"24       continue
"         do 30 k = n,1,-1
"         i = k + 1
"         j = r + 1
"32       if(.not.(j .le. m+r-1 .and. i .le. n))goto 33
"         b(k) = b(k) - a(k,j)*b(i)
"         i = i + 1
"         j = j + 1
"         goto 32
"33       continue
"         b(k) = b(k)*a(k,r)
"30       continue
"         return
"         end
"
$title cbsolv                                       "APAL VERSION OF
$entry cbsolv,5                                     "BACK-SUBSTITUTION
$ext spmul,spflt
"Spmul uses fa,fm,dpx(0): sp(0)=sp(0)*sp(1)
"Spflt uses sp(14-15),dpx(0,1),fa,tm:  dpx(1) = float(sp(15))
"Host:  call cbsolv(aptr,n,m,bptr,pptr)

n $equ 2
aptr $equ 3
m $equ 4
bptr $equ 5
pptr $equ 6
r $equ 7
i $equ 10          "10 octal is 8
j $equ 11          "11 octal is 9
k $equ 12          "12 octal is 10
pk $equ 13         "13 octal is 11
```

...ee

```
l1 $equ 14          "14 octal is 12
b1 $equ 15          "15 octal is 13
l2 $equ 16          "16 octal is 14

cbsolv:  mov 4,pptr
         mov 3,bptr
         mov 2,m
         mov 1,n
         mov 0,aptr
         mov m,r
         incr r                    "r = (m+1)/2

         clr k
         inc k
kl:      mov n,l1                  "do k=1,n-1
         dec l1
         sub# k,l1
         bge aa
         jmp kbr
aa:      mov k,l1
         dec l1
         add pptr,l1; setma        "read in pk(k) and
         nop                       "   convert to an integer
         nop
         fixt md
         fadd
         dpx(0)<fa
         ldspi pk; db<dpx(0)
         mov k,b1
         decl b1
         add bptr,b1; setma
         nop
         incma
         dpx(-4)<md                "b(k) in dpx(-4),dpy(-4)
         nop
         dpy(-4)<md
         sub# k,pk
         bne bb
         jmp swbr
bb:      mov pk,l1                 "while pk(k) != k
         decl l1
         add bptr,l1; setma
         nop
         incma
         dpx(-3)<md                "b(pk(k)) in dpx(-3),dpy(-3)
         mov b1,b1; setma; mi<dpx(-3)
         dpy(-3)<md
         incma; mi<dpy(-3)
         nop
         mov l1,l1; setma; mi<dpx(-4)
         dpx(-4)<dpx(-3)
         incma; mi<dpy(-4)
         dpy(-4)<dpy(-3)
swbr:    mov k,i
         inc i
         mov r,j
         dec j
         mov j,0
         dec 0
         mov n,1
         jsr spmul
         mov 0,l1
         add i,l1
         decl l1
```

...ee

```
        add aptr,l1           "address of a(i,j) in l1
        mov i,b1
        decl b1
        add bptr,b1           "address of b(i) in b1
mull:   dec# j                "while j >= 1 and i <= n
        bge cc
        jmp mulbr
cc:     sub# i,n
        bge dd
        jmp mulbr
dd:       mov l1,l1; setma                  "b(k) in dpx(-4),dpy(-4)
        sub n,l1
        incma; sub n,l1                     "a(i,j) in dpx(-3),dpy(-3)
        dpx(-3)<md; dpy(-2)<md; inc l1      "  and dpy(-2),dpx(-2)
        mov b1,b1; setma; fmul dpx(-4),dpy(-2)
        dpy(-3)<md; dpx(-2)<md; inc l1; fmul dpy(-4),dpx(-3)
        incma; fmul dpy(-4),dpx(-2)         "b(i) in dpx(-1),dpy(-1)
        dpx(-1)<md; fmul dpx(-4),dpy(-3); dpy(0)<fm
        fmul; dpx(0)<fm
        dpy(-1)<md; fmul; fsubr fm,dpy(0)
        fadd fm,dpx(0)
        fsub dpx(-1),fa
        fsub dpy(-1),fa; dec j
        fadd; dpx(-1)<fa; inc i
        dpy(-1)<fa; mov b1,b1; setma; mi<dpx(-1)
        inc b1
        incma; mi<dpy(-1); inc b1
        jmp mull

mulbr:  inc k
        jmp kl

kbr:    mov n,k
        inc k

bkl:    dec k                       "do k = n,1,-1
        bgt ee
        jmp bkbr
ee:     mov k,i               "initialize i and j
        inc i
        mov r,j
        inc j
        mov j,0
        dec 0
        mov n,1
        jsr spmul
        mov 0,l1
        add k,l1
        decl l1
        add aptr,l1                 "address of a(k,j) in l1
        mov k,b1
        decl b1
        add bptr,b1; setma
        nop
        incma
        dpx(-4)<md                  "b(k) in dpx(-4),dpy(-4)
        mov i,b1
        dpy(-4)<md; decl b1
        add bptr,b1                 "address b(i) in b1

pml:    mov m,l2
        add r,l2
        dec l2
        sub# j,l2                   "while j<=m+r-1 and i <= n
```

dd

...*dd*

```
        bge ff
        jmp pmbr
ff:     sub# i,n
        bge gg
        jmp pmbr
gg:     mov b1,b1; setma
        nop
        incma
        dpx(-3)<md; dpy(-2)<md                 "b(i) in dpx(-3),dpy(-3)
        mov l1,l1; setma                       "  and dpy(-2),dpx(-2)
        dpy(-3)<md; dpx(-2)<md                 "a(i,j) in dpx(-1),dpy(-1)
        incma
        dpx(-1)<md
        fmul dpx(-1),dpy(-2)
        dpy(-1)<md; fmul dpx(-1),dpy(-3)
        fmul dpy(-1),dpx(-2)
        fmul dpy(-1),dpx(-3); dpx(0)<fm
        fmul; dpy(0)<fm
        fmul; fsubr fm,dpx(0); inc j
        fadd fm,dpy(0); inc i
        fsub dpx(-4),fa; inc b1
        fsub dpy(-4),fa; inc b1
        fadd; dpx(-4)<fa; add n,l1
        dpy(-4)<fa; add n,l1
        jmp pml

pmbr:   mov r,0
        dec 0
        mov n,1
        jsr spmul
        add k,0
        decl 0
        add aptr,0; setma                      "a(k,r) in dpx(-3),dpy(-3)
        nop                                    "    and dpy(-2),dpx(-2)
        incma
        dpx(-3)<md; dpy(-2)<md
        fmul dpx(-4),dpy(-2)
        dpy(-3)<md; dpx(-2)<md; fmul dpy(-4),dpx(-3)
        fmul dpy(-4),dpx(-2)
        fmul dpx(-4),dpy(-3); dpx(0)<fm
        fmul; dpy(0)<fm
        fmul; fsubr fm,dpx(0); mov k,b1
        fadd fm,dpy(0); decl b1
        fadd; dpx(-4)<fa; add bptr,b1
        dpy(-4)<fa; mov b1,b1; setma; mi<dpx(-4)
        nop
        incma; mi<dpy(-4)
        jmp bkl
bkbr:   nop
        return
$end
```