A Code for Laminated Plate Calculations

The computational scheme for laminated plate calculations outlines in Module 15 has been coded in Fortran as listed below. A PC-executable version is also available in the readings section.

Fortran Source

```
c     plate - a prompt-driven routine for laminated plate calculations

      dimension S(3,3),Sbar(3,3),Qbar(3,3),E(6,7),kwa(6),
     *          T(3,3),Tinv(3,3),R(3,3),Rinv(3,3),Et(6,6),
     *          temp1(3,3),temp2(3,3),temp3(3,3),
     *          eps0(3),xkappa(3),sigbar(3),sig(3),vtemp1(3),
     *          vtemp2(3),E1(10),E2(10),Gnu12(10),G12(10),thk(10),
     *          z(21),mat(20),theta(20),Qsave(3,3,20),Tsave(3,3,20)
      data R/1.,3*0.,1.,3*0.,2./,Rinv/1.,3*0.,1.,3*0.,.5/,
     *     E/42*0./

c-----------------------------------------------------------------------
c     input material set selections

      i=1
10    write(6,20) i
20    format (' assign properties for lamina type ',i2,'...'/)

         write(6,*) 'enter modulus in fiber direction...'
         write(6,*) '   (enter -1 to stop): '
         read (5,*) E1(i)
         if (E1(i) .lt. 0.) go to 30
         write(6,*) 'enter modulus in transverse direction: '
         read (5,*) E2(i)
         write(6,*) 'enter principal Poisson ratio: '
         read (5,*) Gnu12(i)

c        check for isotropy
         check=abs((E1(i)-E2(i))/E1(i))
         if (check.lt.0.001) then
            G12(i)=E1(i)/(2.*(1.+Gnu12(i)))
         else
            write(6,*) 'enter shear modulus: '
            read (5,*) G12(i)
         end if

         write(6,*) 'enter ply thickness: '
         read (5,*) thk(i)
         i=i+1

      go to 10
```

```
c-----------------------------------------------------------------------
c     define layup

30    iply=1
      z(1)=0.

      write(6,*) 'define layup sequence, starting at bottom...'
      write(6,*) '   (use negative material set number to stop)'

40    write (6,50) iply
50    format (/' enter material set number for ply number',i3,': ')
      read (5,*) m
      if (m.lt.0) go to 60

         mat(iply)=m
         write(6,*) 'enter ply angle: '
         read (5,*) theta(iply)
         z(iply+1)=z(iply)+thk(m)
         iply=iply+1

      go to 40

c     compute boundary coordinates (measured from centerline)

60    thick=z(iply)
      N = iply-1
      z0 = thick/2.
      np1=N+1

      do 70 i=1,np1
         z(i)=z(i)-z0
70    continue

c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c     loop over plies, form stiffness matrix

      do 110 iply=1,N

         m=mat(iply)

c        form lamina compliance in 1-2 directions (Eqn. 3.55)


         S(1,1) = 1./E1(m)
         S(2,1) = -Gnu12(m) / E1(m)
         S(3,1) = 0.

         S(1,2) = S(2,1)
         S(2,2) = 1./E2(m)
         S(3,2) = 0.

         S(1,3) = 0.
         S(2,3) = 0.
         S(3,3) = 1./G12(m)

c-----------------------------------------------------------------------
```

```
c        transform to x-y axes
c        obtain transformation matrix T (Eqn. 3.27)


         thet = theta(iply) * 3.14159/180.

         sc = sin(thet)*cos(thet)
         s2 = (sin(thet))**2
         c2 = (cos(thet))**2

         T(1,1) = c2
         T(2,1) = s2
         T(3,1) = -1.*sc

         T(1,2) = s2
         T(2,2) = c2
         T(3,2) = sc

         T(1,3) = 2.*sc
         T(2,3) = -2.*sc
         T(3,3) = c2 - s2

c        inverse transformation matrix

         Tinv(1,1) = c2
         Tinv(2,1) = s2
         Tinv(3,1) = sc

         Tinv(1,2) = s2
         Tinv(2,2) = c2
         Tinv(3,2) = -1.*sc

         Tinv(1,3) = -2.*sc
         Tinv(2,3) = 2.*sc
         Tinv(3,3) = c2 - s2

c        transformation [Sbar] = [R][T]-1[R]-1[S][T] (Eqn. 3.56)

         call matmul (3,3,3,3,3,3,R,Tinv,temp1)
         call matmul (3,3,3,3,3,3,temp1,Rinv,temp2)
         call matmul (3,3,3,3,3,3,temp2,S,temp3)
         call matmul (3,3,3,3,3,3,temp3,T,Sbar)

c---------------------------------------------------------------------
c        invert Sbar (transformed compliance matrix) to obtain
c               Qbar (transformed stiffness matrix)
c        start by setting Qbar = Sbar, then call inversion routine

         do 80 i=1,3
         do 80 j=1,3
            Qbar(i,j)=Sbar(i,j)
80       continue

         call matinv(isol,idsol,3,3,Qbar,3,kwa,det)

c        save Qbar and Tinv matrices
```

```
         do 90 i=1,3
         do 90 j=1,3
            Qsave(i,j,iply)=Qbar(i,j)
            Tsave(i,j,iply)=Tinv(i,j)
90       continue

c        add to laminate stiffness matrix

         ip1=iply+1
         z1=        (z(ip1)   -z(iply)   )
         z2=    0.5*(z(ip1)**2-z(iply)**2)
         z3=(1./3.)*(z(ip1)**3-z(iply)**3)
         do 100 i=1,3
         do 100 j=1,3
            E(i,j)    =  E(i,j) +     Qbar(i,j)*z1
            xx        =               Qbar(i,j)*z2
            E(i+3,j)  =  E(i+3,j) +   xx
            E(i,j+3)  =  E(i,j+3) +   xx
            E(i+3,j+3)=  E(i+3,j+3) + Qbar(i,j)*z3
100      continue

c     end loop over plies; stiffness matrix now formed
110   continue

c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c     output stiffness matrix

      write(6,120)
120   format(/' laminate stiffness matrix:',/)
      do 140 i=1,6
         write(6,130) (e(i,j),j=1,6)
130      format (4x,3e12.4,2x,3d12.4)
         if (i.eq.3) write(6,*)
140   continue

c-----------------------------------------------------------------------
c     obtain and print laminate compliance matrix

c      do 300 i=1,6
c      do 300 j=1,6
c         Et(i,j)=E(i,j)
c300   continue

c      call matinv(isol,idsol,6,6,Et,6,kwa,det)

c      write(6,310)
c310   format(/' laminate compliance matrix:',/)
c      do 320 i=1,6
c         write(6,130) (Et(i,j),j=1,6)
c         if (i.eq.3) write(6,*)
c320   continue

c-----------------------------------------------------------------------
c     obtain traction-moment vector

      write(6,*)
```

```
      write(6,*) 'input tractions and moments...'
      write(6,*)
      write(6,*) '   Nx: '
      read (5,*) e(1,7)
      write(6,*) '   Ny: '
      read (5,*) e(2,7)
      write(6,*) '  Nxy: '
      read (5,*) e(3,7)
      write(6,*) '   Mx: '
      read (5,*) e(4,7)
      write(6,*) '   My: '
      read (5,*) e(5,7)
      write(6,*) '  Mxy: '
      read (5,*) e(6,7)

c-----------------------------------------------------------------------
c     solve resulting system; print strains and rotations

      call matinv(isol,idsol,6,7,e,6,kwa,det)
      write(6,150) (e(i,7),i=1,6)
150   format(/' midplane strains:',//3x,'eps-xx =',e12.4,
     *   /3x,'eps-yy =',e12.4,/3x,'eps-xy =',e12.4,
     *   //' rotations:',//3x,'kappa-xx =',e12.4,
     *   /3x,'kappa-yy= ',e12.4,/3x,'kappa-xy =',e12.4//)

c-----------------------------------------------------------------------
c     compute ply stresses

      write(6,160)
160   format (/' stresses:',/2x,'ply',5x,'sigma-1',
     *        5x,'sigma-2',4x,'sigma-12'/)

      do 210 iply=1,N

         do 180 i=1,3
            eps0(i)=e(i,7)
            xkappa(i)=e(i+3,7)
            do 170 j=1,3
               Qbar(i,j)=Qsave(i,j,iply)
               Tinv(i,j)=Tsave(i,j,iply)
170         continue
180      continue

         call matmul (3,3,3,3,3,1,Qbar,eps0,vtemp1)
         call matmul (3,3,3,3,3,1,Qbar,xkappa,vtemp2)

         zctr=(z(iply)+z(iply+1))/2.
         do 190 i=1,3
            sigbar(i) = vtemp1(i) + zctr*vtemp2(i)
190      continue

         call matmul (3,3,3,3,3,1,Tinv,sigbar,sig)
         write(6,200) iply,sig
200      format (3x,i2,3e12.4)

210   continue
```

```
      stop
      end


c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c --------  library routines for matrix operations -----------

      subroutine matmul(lra,lrb,lrc,i,j,k,a,b,c)
c
c     this subroutine performs the multiplication of two
c     two-dimensional matrices (a(i,j)*b(j,k) = c(i,k)).
c

c     lra - row dimension of "a" (multiplier) matrix
c     lrb - row dimension of "b" (multiplicand) matrix
c     lrc - row dimension of "c" (product) matrix
c     i   - actual number of rows in "a"
c     j   - actual number of columns in "a"
c     k   - actual number of columns in "b"
c     a   - multiplier
c     b   - multiplicand
c     c   - product

      dimension a(1), b(1), c(1)
      do 20 l = 1,i
      nm1 = 0
      lm = l
      do 20 m = 1,k
      c(lm) = 0.0
      nm = nm1
      ln = l
      do 10 n = 1,j
      nm = nm + 1
      c(lm) = c(lm) + a(ln)*b(nm)
 10   ln = ln + lra
      nm1 = nm1 + lrb
 20   lm = lm + lrc
      return
      end



      subroutine matinv(isol,idsol,nr,nc,a,mra,kwa,det)
c
c     this subroutine finds the inverse and/or solves
c     simultaneous equations, or neither, and
c     calculates a determinant of a real matrix.
c

c     isol - communications flag (output)
c        1 - inverse found or equations solved
c        2 - unable to solve
c        3 - input error
c     idsol - determinant calculation flag (output)
c        1 - did not overflow
c        2 - overflow
```

```
c     nr - number of rows in input matrix "a"
c     nc - number of columns in "a"
c     a  - input matrix, first "nr" columns will be inverted
c             on output, "a" is converted to a-inverse
c     mra - row dimension of "a" matrix
c     kwa - work array
c     det - value of determinant (if idsol = 1)

      dimension a(1), kwa(1)
      ir = nr
      isol = 1
      idsol = 1
      if(nr.le.0) go to 330
      if((ir-mra).gt.0) go to 330
      ic = iabs(nc)
      if ((ic - ir).lt.0) ic = ir
      ibmp = 1
      jbmp = mra
      kbmp = jbmp + ibmp
      nes = ir*jbmp
      net = ic*jbmp
      if(nc) 10,330,20
 10   mdiv = jbmp + 1
      iric = ir - ic
      go to 30
 20   mdiv = 1
 30   mad = mdiv
      mser = 1
      kser = ir
      mz = 1
      det = 1.0
 40   piv = 0.
      i = mser
 50   if (( i - kser).gt.0) go to 70
      if((abs(a(i))-piv).le.0.) go to 60
      piv = abs(a(i))
      ip = i
 60   i = i + ibmp
      go to 50
 70   if(piv.eq.0.) go to 340
      if(nc.lt.0) go to 80
      i = ip-((ip - 1)/jbmp)*jbmp
      j = mser - ((mser - 1)/jbmp)*jbmp
      jj = mser/kbmp + 1
      ii = jj + (ip -mser)
      kwa(jj) = ii
      go to 90
 80   i = ip
      j = mser
 90   if (ip - mser) 330,120,100
 100  if ((j - net).gt.0) go to 110
      psto = a(i)
      a(i) = a(j)
      a(j) = psto
      i = i + jbmp
      j = j + jbmp
      go to 100
```

```
 110  det = - det
 120  psto = a(mser)
      det = det*psto
 130  if (det.eq.0.) goto 150
 140  psto = 1./psto
      go to 160
 150  idsol = 1
      isol = 2
      return
 160  continue
      a(mser) = 1.0
      i = mdiv
 170  if((i - net).gt.0) go to 180
      a(i) = a(i)*psto
      i = i + jbmp
      go to 170
 180  if((mz - kser).gt.0) go to 210
      if((mz-mser).eq.0) go to 200
      i = mad
      j = mdiv
      psto = a(mz)
      if(psto.eq.0.) go to 200
      a(mz) = 0.
 190  if((j-net).gt.0) go to 200
      a(i) = a(i) - a(j)*psto
      j = j + jbmp
      i = i + jbmp
      go to 190
 200  mad = mad + ibmp
      mz = mz + ibmp
      go to 180
  210 continue
c 210  need a test here.....call overfl(ivf)
c      go to (350,220),ivf
ccccccc  need at test here, anyhow
 220  kser = kser + jbmp
      if ((kser-nes).gt.0) go to 260
      mser = mser + kbmp
      if(nc.lt.0) go to 230
      mdiv = mdiv + ibmp
      mz = ((mser - 1)/jbmp)*jbmp + 1
      mad = 1
      go to 40
 230  mdiv = mdiv + kbmp
      if(iric.ne.0) go to 240
      mz = mser + ibmp
      go to 250
 240  mz = ((mser - 1)/jbmp)*jbmp + 1
 250  mad = mz + jbmp
      go to 40
 260  if(nc.lt.0) return
      jr = ir
 270  if(jr) 330,360,280
 280  if(kwa(jr) - jr) 330,320,290
 290  k = (jr - 1)*jbmp
      j = k + ir
      l = (kwa(jr) - 1)*jbmp + ir
```

```
300  if(j - k) 330,320,310
310  psto = a(l)
     a(l) = a(j)
     a(j) = psto
     j = j - ibmp
     l = l - ibmp
     go to 300
320  jr = jr - 1
     go to 270
330  isol = 3
     return
340  det = 0.
     isol = 2
     idsol = 1
     return
350  isol = 2
     idsol = 2
360  return
     end
```