      !| MODIFIED FROM THE NOLAN ET AL ABAQ ARTICLE INTO BLEMKER 2005 .

      !| MSA

      !| UMAT DOC http://130.149.89.49:2080/v2016/books/sub/default.htm

      SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,

     1 RPL,DDSDDT,DRPLDE,DRPLDT,

     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,

     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,

     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)

      !|:

      !|...TIME(1)  Value f step time at d beginJ f d current increment

      !|...TIME(2)  Value f total time at d beginJ f d current increment

      !|...COORDS  currCoords of ds point (Finite deform),not initCoords

      !|...DTIME  Time increment

      !|...NOEL  El number

      !|...NPT  Integration point number

      !|...JSTEP  Step number

      !|...SSE  Specific elastic strain energy

C         INCLUDE 'ABA\_PARAM.INC'

        !|:It has this content below in GitHub link:

        !|...KratosLegacyApplications/ConstitutiveLawsApplication/

        !|...custom\_external\_libraries/umat/ABA\_PARAM.INC

C       DOUBLE PRECISION STRESS,STATEV,DDSDDE,SSE,SPD,SCD,

C      1 RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,

C      2 TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,NSHR,

C      3 PROPS,COORDS,DROT,PNEWDT,CELENT,

C      4 DFGRD0,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC

C C

C       INTEGER NTENS,NDI,NSTATV,NPROPS,MATERIALNUMBER

        CHARACTER\*80 CMNAME

        DIMENSION STRESS(NTENS),STATEV(NSTATV),

     1  DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),

     2  STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),

     3  PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3),

     4  JSTEP(4) !|Added from abaq 2016 docum

        !| Nt:From UMAT doc of Abaq2016:

        !|...user coding to define DDSDDE, STRESS, STATEV, SSE, SPD, SCD

        !|...and, if necessary, RPL, DDSDDT, DRPLDE, DRPLDT, PNEWDT

        DIMENSION CAUSTR(3,3),XKIRC1(3,3), XA(3), XIF(2)

        NANISO = 1 !|Drs only 1 fibre family; so transversely isotropic.

        !|XXX> Blemker material properties

        !| Fibre vector

        XA(1) = PROPS(1)

        XA(2) = PROPS(2)

        XA(3) = PROPS(3)

        !|:PROPS(1:3) are the direction conponents.

        !|...Not yet decided how to obtain

        !|...Piecewise with manual entry is an option.

        TYPE = PROPS(4) !|Either >0 for muscle or <0 for tendon,:::+1&-1

        IF (TYPE.GT.0.0D0) THEN

        !| Muscle values

        G1    = 5.00D2 !|G1\_muscle , Pa

        G2    = 5.00D2 !|G2\_muscle , Pa

        XK    = 1.00D7 !|K\_muscle  , Pa

        XLOFL = 1.40D0 !|Stretch for optimum force

        XLSTR = 1.40D0 !|When passive muscle fibre force becomes linear

        !| Muscle specific

        P1    = 0.05D0 !|A material constant

        P2    = 6.60D0 !|A material constant

        P3    = P1\*P2\*exp(P2\*(XLSTR/XLOFL)-1)

        P4    = P1\*(exp(P2\*(XLSTR/XLOFL-1))-1) - P3\*XLSTR/XLOFL

        SIGMAX = 3.00D5 !|sig\_max , Pa , only muscle

        !| Tendon specific

        XL1    = 2.70D6 !|Pa, only tendon

        XL2    = 46.4D0 !|A material constant

        XL3    = XL1\*XL2\*exp(XL2\*(XLSTR-1))

        XL4    = XL1\*( exp(XL2\*(XLSTR-1)) - 1 ) - XL3\*XLSTR

        ELSE IF (TYPE.LT.0.0D0) THEN

        !| Tendon values

        G1    = 5.00D4 !|G1\_tendon , Pa

        G2    = 5.00D4 !|G2\_tendon , Pa

        XK    = 1.00D8 !|K\_tendon  , Pa

        XLOFL = 1.40D0 !|Stretch for optimum force

        XLSTR = 1.03D0 !|When passive muscle fibre force becomes linear

        !| Muscle specific

        P1    = 0.05D0 !|A material constant

        P2    = 6.60D0 !|A material constant

        P3    = P1\*P2\*exp(P2\*(XLSTR/XLOFL)-1)

        P4    = P1\*(exp(P2\*(XLSTR/XLOFL-1))-1) - P3\*XLSTR/XLOFL

        SIGMAX= 3.00D5 !|sig\_max , Pa , only muscle

        !| Tendon specific

        XL1    = 2.70D6 !|Pa, only tendon

        XL2    = 46.4D0 !|A material constant

        XL3    = XL1\*XL2\*exp(XL2\*(XLSTR-1))

        XL4    = XL1\*( exp(XL2\*(XLSTR-1)) - 1 ) - XL3\*XLSTR

        END IF

        !|XXX. Blemker material properties

        !|:Nt:Couldve been defined in Props & read automaticly from .inp

        !|YYY> RETURN to the solver what it needs. CALL CALL

        !| CALC STRESS

        CALL mykstress\_calc(DFGRD1,XA,G1,G2,XK,XLOFL,XLSTR,P1,P2,P3,P4,

     1                      SIGMAX,   !|Muscle extra material param

     2                      XL1,XL2,XL3,XL4, !|Tendon extra mater param

     3                      XI1,XI2,XI3,XI4,XI5,      !|Invariants

     4                      XI1B,XI2B,XI3B,XI4B,XI5B  !|Isochoric invar

     5                      CAUSTR,TIME,PROPS)    !|Cauchy stress SIGMA

        STATEV(1) = XI1

        STATEV(2) = XI2

        STATEV(3) = XI3

        STATEV(4) = XI4

        STATEV(5) = XI5

        STATEV(6) = XI1B

        STATEV(7) = XI2B

        STATEV(8) = XI3B !|Should always equal to "1.D0"

        STATEV(9) = XI4B

        STATEV(10)= XI5B

        !| Voigt vector of stress

        CALL mykmatrix2vector(CAUSTR, STRESS, nshr)

        !|STRESS here is in 6x1 vector form

        !| Tangent matrix calculation (:Material gradient/Jacobian)

        DO I=1,NTENS

          DO J=1,NTENS

            DDSDDE(I,J) = 0.D0

          END DO

        END DO

        CALL mykCTM(XKIRC1,DFGRD1,NTENS,PROPS,XA,NPT, !,ITER

     1              NANISO,DDSDDE,NPROPS,XJ1,NSHR,

     2              G1,G2,XK,XLOFL,XLSTR,P1,P2,P3,P4,

     3              SIGMAX,

     4              XL1,XL2,XL3,XL4)

        !|YYY.

        RETURN

      ENDSUBROUTINE UMAT

      !| Called Subroutines

      INCLUDE 'mykstress\_calc.for' !|Subroutine to calc Cauchy stress

      INCLUDE 'mykpolarDecomp.for' !|Polar decomposition

      INCLUDE 'mykCTM.for' !|Subroutine to calculate the tangent matrix

      INCLUDE 'myutilities.for' !|Helper subroutines

      !| DEPOT

        !| Convert degrees to radians - IS NEEDED ?

C         XPI = 4\*ATAN(1.0D0) !|One 4th the angle whose tangent is 1.0D0

C         THETAR = THETAD\*XPI/180.D0

C         iter=0  !|?? Not used as far as I see in kstress\_calc.for

      SUBROUTINE mykmatrix2vector(XMAT, VEC, NSHR)

      !|:Matrix to Voigt vector

      !|...11 22 33 12 13 23 <SEE doc Abaq Convensions

C         INCLUDE 'ABA\_PARAM.INC'

        INTENT(in) :: XMAT, NSHR

        INTENT(out):: VEC

        DIMENSION xmat(3,3), vec(6)

        !| Normal components

        DO i=1,3

          vec(i) = xmat(i,i)

        END DO

        !| Shear components

        vec(4) = xmat(1,2)

        IF (NSHR==3) THEN

          vec(5) = xmat(1,3)

          vec(6) = xmat(2,3)

        END IF

      END SUBROUTINE mykmatrix2vector

      SUBROUTINE mykdelF(m, n, DFGRD, eps, DF) !|Delta(F) : defG perturb

        !| SEE 1996 Miehe eq2.17 for delta(F) (defGrad pertubation calc)

        !| SEE 2014 Nolan eqA.3  > ~same as above w/o g-mericTensor.

C         INCLUDE 'ABA\_PARAM.INC'

        INTENT (in) :: DFGRD, eps, m, n !|Curr DefG, incr, idx to pert

        INTENT (out):: DF !|Perturbed defGrad increment is output.

        DIMENSION dyad1(3,3),dyad2(3,3),DFGRD(3,3),DF(3,3)  ! ,DFp1(3,3)

        !| Zero the dyad matrices

        DO i = 1,3

          DO j = 1,3

            dyad1(i,j) = zero

            dyad2(i,j) = zero

          END DO

        END DO

        !| Place the 1's in the correct location

        dyad1(m,n) = 1.0;

        dyad2(n,m) = 1.0;

        DF = 0.5D0 \*( MATMUL(dyad1,DFGRD) + MATMUL(dyad2,DFGRD) ) \* eps

      END SUBROUTINE mykdelF

      SUBROUTINE myKPRINTER(TENS, m, n) !|Print out a matrix of any size.

C         INCLUDE 'ABA\_PARAM.INC'

        INTENT(in):: TENS, m, n

        DIMENSION tens(m,n)

        write(6,\*)

        DO i = 1,m

          DO j = 1,n

            ! write(6,'(e19.9)',advance='no'),tens(i,j)

            !|:Dss orig, but comma not good for some compilers.

            write(6,'(e19.9)',advance='no') tens(i,j)

          END DO

          write(6,\*)

        END DO

        write(6,\*)

        RETURN

      END SUBROUTINE myKPRINTER

      SUBROUTINE ACOSH(X,Y)

      !| INVERSE HYPERBOLIC FUNCTIONS,SEE EXPLANATION by Arnold Desitter

      !| ...computer-programming-forum.com/49-fortran/7ebc82829d81d040.htm

C         INCLUDE 'ABA\_PARAM.INC'

C         DOUBLE PRECISION :: X,Y

        ! DOUBLE PRECISION ATANH, ASINH

        INTENT(IN) :: X

        INTENT(OUT):: Y

        ! ATANH(X)=LOG((X+1.)/(-X+1.))/2

        ! ASINH(X)=LOG(X+SQRT(X\*\*2+1.))

        Y = LOG(X+SQRT((X-1.D0)\*(X+1.D0))) !|sO says dss problem fr x~=0

        !|:LOG is natural logarithm

      END SUBROUTINE ACOSH

      SUBROUTINE INVER(A,AI)

C         INCLUDE 'ABA\_PARAM.INC'

        INTENT(IN)  :: A

        INTENT(OUT) :: AI

        DIMENSION :: A(3,3),AI(3,3)

        !| TAKE THE INVERSE OF A 3x3 MATRIX

        ! Calculate the inverse determinant of the matrix

        dum1 = + A(1,1)\*A(2,2)\*A(3,3) - A(1,1)\*A(2,3)\*A(3,2)

        dum2 = - A(1,2)\*A(2,1)\*A(3,3) + A(1,2)\*A(2,3)\*A(3,1)

        dum3 = + A(1,3)\*A(2,1)\*A(3,2) - A(1,3)\*A(2,2)\*A(3,1)

        detinv = (dum1+dum2+dum3)\*\*(-1)

        ! Calculate the inverse of the matrix

        AI(1,1) = +detinv \* (A(2,2)\*A(3,3) - A(2,3)\*A(3,2))

        AI(2,1) = -detinv \* (A(2,1)\*A(3,3) - A(2,3)\*A(3,1))

        AI(3,1) = +detinv \* (A(2,1)\*A(3,2) - A(2,2)\*A(3,1))

        AI(1,2) = -detinv \* (A(1,2)\*A(3,3) - A(1,3)\*A(3,2))

        AI(2,2) = +detinv \* (A(1,1)\*A(3,3) - A(1,3)\*A(3,1))

        AI(3,2) = -detinv \* (A(1,1)\*A(3,2) - A(1,2)\*A(3,1))

        AI(1,3) = +detinv \* (A(1,2)\*A(2,3) - A(1,3)\*A(2,2))

        AI(2,3) = -detinv \* (A(1,1)\*A(2,3) - A(1,3)\*A(2,1))

        AI(3,3) = +detinv \* (A(1,1)\*A(2,2) - A(1,2)\*A(2,1))

      END SUBROUTINE INVER

      SUBROUTINE TRACE(A,TR\_A)

C         INCLUDE 'ABA\_PARAM.INC'

        DIMENSION :: A(3,3)

        OUTVAL = A(1,1) + A(2,2) + A(3,3)

      END SUBROUTINE TRACE

      SUBROUTINE DEVCUR(CMAT,AB,DEV\_AB) !|B for Bar.

        !| DEVCUR : Deviatoric operator in current config, not reference

        !| AB is the isochoric one, B:bar

        !| A is the full one

C         INCLUDE 'ABA\_PARAM.INC'

        DIMENSION :: CMAT(3,3),CMATI(3,3),AB(3,3),DEV\_AB(3,3)

        CALL TRACE(MATMUL(TRANSPOSE(AB),CMAT),TR\_)

        CALL INVER(CMAT,CMATI)

        DEV\_AB = AB -1/3\*TR\_\*CMATI !|Deviatoric part

      END SUBROUTINE

      !| DEPOT

C       subroutine kdotprod(A, B, dotp, n) !|MATMUL() available fortran 90

C       !|:Dot product of two vectors

C         INCLUDE 'ABA\_PARAM.INC'

C         INTENT(in) :: A, B, n

C         INTENT(out):: dotp

C         DIMENSION A(n), B(n)

C         dotp = 0.0

C         DO i = 1,n

C           dotp = dotp + A(i)\*B(i)

C         END DO

C       end subroutine kdotprod

C       SUBROUTINE KMTMS (M, N, L, A, KA, B, KB, C, KC) !|MATMUL available

C       !|:Multiply two real matrices

C         INCLUDE 'ABA\_PARAM.INC'

C         !| Because IMPLICITE NONE is not used here,

C         !| ...I,J,K,L,M,N (and what starts with them are integers;

C         !| ...and others are real (single precision)

C         INTENT(in) :: M, N, L, A, KA, B, KB, KC

C         !|:So here KA is an implied integer since it starts with a "K".

C         INTENT(out):: C

C         DIMENSION A(KA,N), B(KB,L), C(KC,L)

C         !|:A(M,N),B(N,L),C(M,L)

C         !|::KA,KB,KC are they even needed??

C         DOUBLE PRECISION W

C         DO 30 J = 1,L

C           DO 20 I = 1,M

C             W = 0.D0 !|Dummy var t store C(I,J)=A(I,:).B(:,J) multiplic.

C             DO 10 K = 1,N

C               W = W + A(I,K) \* B(K,J)

C    10       CONTINUE  !|C(I,J) el is done

C             C(I,J) = W

C    20     CONTINUE

C    30   CONTINUE

C         RETURN !|You can RETURN if you like, it is assigned to var RETURN.

C         !|:SEE the website ibm.com/support/knowledgecenter/SSGMCP\_5.4.0/

C         !|...applications/developing/rexx/subrou.html

C       END SUBROUTINE KMTMS

C     Subroutine to calculate the consistent tangent matrix using the

C     perturbation method outlined by Miehe and Sun

C

C     15-Aug-2018

C     David Nolan

C     Adapted from the above by me.

      !|Nt:Var names dt start with ijklmn are implied integers.

      !|...others are implied real's (single precision).

      !|::Source : https://www.ibm.com/

      SUBROUTINE mykCTM(XKIRC1,DFGRD1,NTENS,PROPS,XAMAT,NPT,

     1                NANISO,DDSDDE,NPROPS,XJ1,NSHR,

     2                G1,G2,XK,XLOFL,XLSTR,P1,P2,P3,P4,

     3                SIGMAX,

     4                XL1,XL2,XL3,XL4) !,ITER

C         INCLUDE 'ABA\_PARAM.INC'

        DIMENSION XKIRC1(3,3),DF\_IJ(3,3),DFGRDP(3,3),

     1  XKIRCP(3,3),CTM\_IJ(3,3),CTMCOL(NTENS),ilist(6),jlist(6),

     2  XAMAT(3,1),XI4(2),DFGRD1(3,3),DDSDDE(NTENS,NTENS),PROPS(NPROPS)

        eps = 1.0e-08 !|Perturbation parameter

        XKIRC1 = XJ1\*CAUSTR !|kirchhoffStr(tau) = J \* cauchyStr(sigma)

        !| Nt:Material props are already in XAMAT, and type is not used

        !| The Kirchhoff stress is perturbed six times over the

        !| ...i, j'th components of the deformation gradient. This

        !| ...array lists each of the componets to be perturbed over.

        ilist(1) = 1; jlist(1) = 1; !|11

        ilist(2) = 2; jlist(2) = 2; !|22

        ilist(3) = 3; jlist(3) = 3; !|33

        ilist(4) = 1; jlist(4) = 2; !|12

        ilist(5) = 1; jlist(5) = 3; !|13

        ilist(6) = 2; jlist(6) = 3; !|23 |See doc Abaq Conventions.

        !| CREATE THE PERTURBED DEFORMATION GRADIENT

        Perturbation: do iter = 1,NTENS

        !|:Loop over each of the six components of the deformation

        !|...gradient which are to be perturbed, each time calculating

        !|...a new column of the tangent matrix.

          !| Pick out the (i,j)th component f d defGrad to be perturbed

          ii = ilist(iter) ; jj = jlist(iter)

          CALL kdelF(ii, jj, DFGRD1, eps, DF\_IJ) !|Delta(F)\_i\_j : DF\_IJ

          DFGRDP = DFGRD1 + DF\_IJ !|Perturbed defGrad

          CALL kstress\_calc(DFGRDP,G1,G2,XK,XLOFL,XLSTR,P1,P2,P3,P4,

     1                      SIGMAX,   !|Muscle extra material param

     2                      XL1,XL2,XL3,XL4, !|Tendon extra mater param

     3                      XI1,XI2,XI3,XI4,XI5,      !|Invariants

     4                      XI1B,XI2B,XI3B,XI4B,XI5B  !|Isochoric invar

     5                      CAUSTP,XAMAT)    !|PertD Cauchy stress SIGMA

          XKIRCHP = XJ1\*CAUSTP !|kirch(tau) = J \* cauchy(sigma)

          CTM\_IJ = (XKIRCP - XKIRC1)/XJ1/eps !|cauchy(sig)=kirch(tau)/J

          !|:SEE 1996 Miehe eq2.10>so this is its Eulerian form

          !|:SEE 2014 Nolan eqA.4 > ~same as 1996 Miehe.

          !|FFF FORM THE CONSISTENT TANGENT MATRIX

          CALL kmatrix2vector(CTM\_IJ, CTMCOL, NSHR) !|3x3 Tens > 6x1 vec

          !| Insert this vector into the iter'th column of DDSDDE

          DO insert = 1,NTENS

            DDSDDE(insert,iter) = CTMCOL(insert)

          END DO

          !|FFF END

        END DO Perturbation

      END SUBROUTINE mykCTM

      !| DEPOT - Local arrays

C XKIRC1         - True Kirchhoff stress (i.e. not based on perturbed def. grad.)

C DFGRD1          - True deformation gradient for this increment

C DF\_IJ             - Increment of the perturbed deformation gradient

C DFGRDP      - Perturbed def. grad.

C XKIRCP     - Perturbed Kirchhoff stress

C CTM\_IJ             - (:,:,I,J) Components of material jacobian

C CTMCOL          - Above in vector form

C ILIST, JLIST    - Set of index components to be perturbed upon

C DUMSTRSS        - DUMMY STRESS TENSOR

C XAMAT           - array containing reference configuration fibre vectors

C XI4             - array storing the fibre invariant (will not be used in this subroutine)

C DDSDDE          - Voigt notation matrix to store the tangent matrix

C PROPS           - Properties read in from Abaqus .inp file

      !| MODIFIED FROM THE NOLAN ET AL ABAQ ARTICLE INTO BLEMKER 2005 .

      !| MSA

      !| Nt:In comments b 2000 Holzepfel syntax is used ::: I\_4=A.C A

      SUBROUTINE mykstress\_calc(DFGRD,XA,G1,G2,XK,XLOFL,XLSTR,

     1                    P1,P2,P3,P4,SIGMAX,

     2                    XL1,XL2,XL3,XL4,

     3                    XJ,XI1,XI2,XI3,XI4,XI5,

     4                    XI1B,XI2B,XI3B,XI4B,XI5B,

     5                    CAUSTR,TIME,PROPS)

      !| call kstress\_calc(DFGRD1,XA,G1,G2,XK,XLOFL,XLSTR,

      !|1                    P1,P2,P3,P4,SIGMAX, !|Muscle extra mat para

      !|2                    XL1,XL2,XL3,XL4) !|Tendon extra mater param

      !|3                    XI1,XI2,XI3,XI4,XI5,      !|Invariants

      !|4                    XI1B,XI2B,XI3B,XI4B,XI5B) !|Isochoric invar

      !|5                    CAUSTR,TIME,PROPS) !|Cauchy stress SIGMA

C         INCLUDE 'ABA\_PARAM.INC'

        !|:It has this content below in GitHub link:

        !|...KratosLegacyApplications/ConstitutiveLawsApplication/

        !|...custom\_external\_libraries/umat/ABA\_PARAM.INC

C       DOUBLE PRECISION STRESS,STATEV,DDSDDE,SSE,SPD,SCD,

C      1 RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,

C      2 TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,NSHR,

C      3 PROPS,COORDS,DROT,PNEWDT,CELENT,

C      4 DFGRD0,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC

C C

C       INTEGER NTENS,NDI,NSTATV,NPROPS,MATERIALNUMBER

        INTENT(IN) :: DFGRD, XA

        INTENT(IN) :: G1,G2,XK,XLOFL,XLSTR

        INTENT(IN) :: P1,P2,P3,P4,SIGMAX

        INTENT(IN) :: XL1,XL2,XL3,XL4

        INTENT(OUT):: XJ,XI1,XI2,XI3,XI4,XI5

        INTENT(OUT):: XI1B,XI2B,XI3B,XI4B,XI5B

        INTENT(OUT):: CAUSTR

        DIMENSION :: TIME(2) !|NEEDED becoz not in ABA\_PARAM.INC

        DIMENSION :: XA(3,1),aCUR(3,1)

        DIMENSION :: DFGRD(3,3),DFGRDB(3,3),DFGRDI(3,3),

     1  A0A(3,3),aoa(3,3),DIAG\_A(3,3),

     2  BMAT(3,3),BMAT2(3,3),BMATB(3,3),BMATB2(3,3),

     3  CMAT(3,3),CMAT2(3,3),CMATB(3,3),CMATB2(3,3),CMATI(3,3),

     4  C\_A0A(3,3),CB\_A0A(3,3),C2\_A0A(3,3),CB2A0A(3,3),

     5  DI1BCB(3,3),DI2BCB(3,3),DI3BCB(3,3),DI4BCB(3,3),DI5BCB(3,3),

     7  DW1CB(3,3),S1B(3,3),DEVS1B(3,3),S1(3,3),

     6  DW2CB(3,3),S2B(3,3),DEVS2B(3,3),S2(3,3),

     6  CAUSTR(3,3),

     7  SVOL(3,3),SIG(3,3),SIG1(3,3),SIG2(3,3),SIG3(3,3),SIGVOL(3,3),

     7  EYE(3,3)

        EYE = 0.D0

        DO 30 k = 1,3

          EYE(k,k) = 1.D0

   30   CONTINUE

        DFGRDB = XJ\*\*(-1.D0/3.D0)\*DFGRD !|Isochoric defGrad, B for bar

        CALL INVER(DFGRD,DFGRDI) !|Inverse deformation gradient

        BMAT  = MATMUL(DFGRD ,TRANSPOSE(DFGRD) ) !|Left Cauch Green DefT

        BMAT2 = MATMUL(BMAT,BMAT)

        BMATB = MATMUL(DFGRDB,TRANSPOSE(DFGRDB)) !|Left Cauch Greed DefT

        BMATB2= MATMUL(BMATB,BMATB)

C         TRB   = BMAT(1,1)+BMAT(2,2)+BMAT(3,3)

        CALL TRACE(BMAT,TRB)

        TRB2  = BMAT2(1,1)+BMAT2(2,2)+BMAT2(3,3)

        TRBB  = BMATB(1,1)+BMATB(2,2)+BMATB(3,3)

        TRBB2 = BMATB2(1,1)+BMATB2(2,2)+BMATB2(3,3)

        !

        CMAT  = MATMUL(TRANSPOSE(DFGRD) ,DFGRD) !|Right Cauch Green DefT

        CMAT2 = MATMUL(CMAT,CMAT)

        CMATB = MATMUL(TRANSPOSE(DFGRDB),DFGRDB)

        CMATB2= MATMUL(CMATB,CMATB)

        TRC   = CMAT(1,1)+CMAT(2,2)+CMAT(3,3)

        TRC2  = CMAT2(1,1)+CMAT2(2,2)+CMAT2(3,3)

        TRCB  = CMATB(1,1)+CMATB(2,2)+CMATB(3,3)

        TRCB2 = CMATB2(1,1)+CMATB2(2,2)+CMATB2(3,3)

        CALL INVER(CMAT,CMATI)

        !|AAA> Fibre vector - related calculations

        A0A = MATMUL(XA,TRANSPOSE(XA)) !|Structural Tensor in ref config !TRANSPOSE DOEES NOT WORK with rank 1

C         DO i=1,3 !|IF YOU DEFINE XA WITH RANK 1 : XA(3)

C           DO j=1,3

C             A0A(i,j) = XA(i) \* XA(j)

C           END DO

C         END DO

        C\_A0A  = MATMUL(CMAT,A0A)

        CB\_A0A = MATMUL(CMATB,A0A)

        C2\_A0A = MATMUL(CMAT2,A0A)

        CB2A0A = MATMUL(CMATB2,A0A)

        TRCA0A = C\_A0A(1,1)+C\_A0A(2,2)+C\_A0A(3,3)

        TCBA0A = CB\_A0A(1,1)+CB\_A0A(2,2)+CB\_A0A(3,3)

        TC2A0A = C2\_A0A(1,1)+C2\_A0A(2,2)+C2\_A0A(3,3)

        TCB2AA = CB2A0A(1,1)+CB2A0A(2,2)+CB2A0A(3,3)

        aCUR  = MATMUL(DFGRD,XA) !|Fibre vector in current config

        !|:aCUR is not normalised to 1 !; so it s not aCUR = aCUR/|aCUR|

        aoa = MATMUL(aCUR,TRANSPOSE(aCUR)) !|Structural tensor, cur conf !TRANSPOSE DOEES NOT WORK with rank 1

C         DO i=1,3 !|IF YOU DEFINE aCUR WITH RANK 1 : aCUR(3)

C           DO j=1,3

C             aoa(i,j) = aCUR(i) \* aCUR(j)

C           END DO

C         END DO

        TRaoa = aoa(1,1)+aoa(2,2)+aoa(3,3)

        !|AAA. Fibre vector - related calculations

        !|III Invariant calculations

        XJ = +DFGRD(1, 1)\*DFGRD(2, 2)\*DFGRD(3, 3)

     1       +DFGRD(1, 2)\*DFGRD(2, 3)\*DFGRD(3, 1)

     2       +DFGRD(1, 3)\*DFGRD(2, 1)\*DFGRD(3, 2)

     3       -DFGRD(1, 2)\*DFGRD(2, 1)\*DFGRD(3, 3)

     4       -DFGRD(1, 3)\*DFGRD(2, 2)\*DFGRD(3, 1)

     5       -DFGRD(1, 1)\*DFGRD(2, 3)\*DFGRD(3, 2)  !|Jacobian J=det(F)

        XJ\_13 = XJ\*\*(-1.D0/3.D0)

        XJ\_23 = XJ\*\*(-2.D0/3.D0)

        XJ\_43 = XJ\*\*(-4.D0/3.D0)

        XI1  = TRC

C         XI1  = TRB

C         XI1B = TRCB

C         XI1B = TRBB

        XI1B = XJ\_13\*XI1

        XI2  = 0.5D0\*(TRC\*\*2.0-TRC2)

C         XI2  = 0.5D0\*(TRB\*\*2.0-TRB2)

C         XI2B = 0.5D0\*(TRCB\*\*2.0-TRCB2)

C         XI2B = 0.5D0\*(TRBB\*\*2.0-TRBB2)

        XI2B = XJ\_43\*XI2

        XI3 = +CMAT(1, 1)\*CMAT(2, 2)\*CMAT(3, 3)

     1        +CMAT(1, 2)\*CMAT(2, 3)\*CMAT(3, 1)

     2        +CMAT(1, 3)\*CMAT(2, 1)\*CMAT(3, 2)

     3        -CMAT(1, 3)\*CMAT(3, 1)\*CMAT(2, 2)

     4        -CMAT(1, 2)\*CMAT(2, 1)\*CMAT(3, 3)

     5        -CMAT(1, 1)\*CMAT(3, 2)\*CMAT(2, 3)

C         XI3 = +BMAT(1, 1)\*BMAT(2, 2)\*BMAT(3, 3)

C      1        +BMAT(1, 2)\*BMAT(2, 3)\*BMAT(3, 1)

C      2        +BMAT(1, 3)\*BMAT(2, 1)\*BMAT(3, 2)

C      3        -BMAT(1, 3)\*BMAT(2, 2)\*BMAT(3, 1)

C      4        -BMAT(1, 2)\*BMAT(2, 1)\*BMAT(3, 3)

C      5        -BMAT(1, 1)\*BMAT(3, 2)\*BMAT(2, 3)

C         XI3 = XJ\*\*2

C         XI3B= +CMATB(1,1)\*CMATB(2,2)\*CMATB(3,3)

C      1        +CMATB(1,2)\*CMATB(2,3)\*CMATB(3,1)

C      2        +CMATB(1,3)\*CMATB(2,1)\*CMATB(3,2)

C      3        -CMATB(1,3)\*CMATB(2,2)\*CMATB(3,1)

C      4        -CMATB(1,2)\*CMATB(2,1)\*CMATB(3,3)

C      5        -CMATB(1,1)\*CMATB(3,2)\*CMATB(2,3) !| = 1.D0

C         XI3B= +BMATB(1,1)\*BMATB(2,2)\*BMATB(3,3)

C      1        +BMATB(1,2)\*BMATB(2,3)\*BMATB(3,1)

C      2        +BMATB(1,3)\*BMATB(2,1)\*BMATB(3,2)

C      3        -BMATB(1,3)\*BMATB(2,2)\*BMATB(3,1)

C      4        -BMATB(1,2)\*BMATB(2,1)\*BMATB(3,3)

C      5        -BMATB(1,1)\*BMATB(2,3)\*BMATB(3,2) !| = 1.D0

        XI3B= 1.D0 !|since isochoric : J=1. So J^2=0

        XI4  = TRCA0A !|I\_4 = tr(C A0A) = A.C A  (:= A\_i C\_ij A\_j)

C         XI4  = TRaoa  !|I\_4 = A.F^T F A = a o a

C         XI4B = TCBA0A !|I\_4\_bar = tr(C\_bar A0A) = A.C\_bar A

        XI4B = XJ\_23\*I4

        XI5  = TC2A0A !|I\_5 = tr(C^2 A0A) = A.C^2 A  (:= A\_i C^2\_ij A\_j)

C         XI5B = TCB2AA !|I\_5\_bar = tr(C\_bar^2 A0A) = A.C\_bar^2 A

        XI5B = XJ\_43\*I5

        XLAM = SQRT(XI4)

        LAMBDB = XJ\_13\*XLAM

        !|III END Invariant calculations

        !|SSS> STRESS

        DI1BCB = EYE

        DI2BCB = XI1B\*EYE - CMATB

        DI3BCB = 0.0D0

        DI4BCB = A0A

        DI5BCB = MATMUL(XA,TRANSPOSE(MATMUL(CMATB,XA)))

     +          +MATMUL(MATMUL(TRANSPOSE(CMATB),XA),TRANSPOSE(XA))

C    +          +MATMUL(MATMUL(CMATB,XA),TRANSPOSE(XA)) !|CB symmetric

        !|:??

        !|: Ni.Cjm.Nm  is\_elements\_of  [N] . [[C] . [N]]^T

        !|: Nm.Cmi.Nj  is\_elements\_of  [C]^T . [N] . [N]^T

        !|S\_111> SIG1

        DW1I4B = -G1\*XI5B/(XI4B\*\*2)

        DW1I5B = +G1/XI4B

        DW1CB = DW1I4B\*DI4BCB + DW1I5B\*DI5BCB

        S1B = 2.D0\*DW1CB

        CALL DEVCUR(CMAT,S1B,DEVS1B)

        S1 = XJ\_23\*DEVS1B

        SIG1 = XJ\*\*(-1)\*DFGRD\*S1\*TRANSPOSE(DFGRD)

        !|S\_111. SIG1

        !|S\_222> SIG2

        C\_W2 = (XI1B\*XI4B-XI5B)/(2\*SQRT(XI4B))

        CALL ACOSH(C\_W2,ACOSH1)

        C\_W2\_2 = 2\*G2\*ACOSH1/SQRT(C\_W2\*\*2 - 1)

        DW2I1B = +C\_W2\_2\*(SQRT(XI4B)/2)

        DW2I4B = +C\_W2\_2\*(XI1B\*XI4B+XI5B)/(2\*XI4B\*(3/2))

        DW2I5B = -C\_W2\_2\*(SQRT(XI4B\*\*(3/2)))/2

        DW2CB = DW2I1B\*DI1BCB + DW2I4B\*DI4BCB + DW2I5B\*DI5BCB

        S2B = 2.D0\*DW2CB

        CALL DEVCUR(CMAT,S2B,DEVS2B)

        S2 = XJ\_23\*DEVS2B

        SIG2 = XJ\*\*(-1)\*DFGRD\*S2\*DFGRDI

        !|S\_222. SIG2

        !|S\_333 SIG3

        !| Piece-wise stretch dependent parameters, Page b15

        IF (PROPS(1).LT.0.0D0) THEN  !|MUSCLE

          !| MUSCLE - Passive part

          IF (XLAM.LE.XLOFL) THEN

            FFPAS = 0.D0 !|f\_fibre\_passive

          ELSE IF (XLAM.GT.XLOFL .AND. XLAM.LT.XLSTR) THEN

            FFPAS = P1\*( exp( P2\*(XLAM/XLOFL-1) ) - 1 )

          ELSE IF (XLAM.GE.XLSTR) THEN

            FFPAS = P3\*XLAM/XLOFL + P4

          END IF

          !| MUSCLE - Active part

          IF (XLAM.LE.(0.6D0\*XLOFL)) THEN

            FFACT = 9\*(XLAM/XLOFL-0.4D0)\*\*2 !|f\_fibre\_active

          ELSE IF (XLAM.GT.(0.6D0\*XLOFL).AND.XLAM.LT.(1.4D0\*XLOFL)) THEN

            FFACT = 1-4\*(1-XLAM/XLOFL)\*\*2

          ELSE IF (XLAM.GE.(1.4D0\*XLOFL)) THEN

            FFACT =  9\*(XLAM/XLOFL-1.6D0)\*\*2

          END IF

          !| Activation level: TONUS is %1 of total activation

          ALPHA = 0.01 !|Preset alpha, to be tried first.

          ! ALPHA = TIME(1)\*0.01 !|Gradual alpha

          !|:See initial stress in Abaq doc, whether it is useful,

          !|...instead of tonus definition with alpha.

          FFTOT = FFPAS + ALPHA\*FFACT  !|f\_fibre\_total

          !| Total muscle fibre stress value, IN FIBRE DIRECTION

          SIGFIB = SIGMAX\*FFTOT\*(XLAM/XLOFL)

        ELSE IF (PROPS(1).LT.0.0D0) THEN !|TENDON

          !| TENDON - direct total response

          IF (XLAM.LE.1.D0) THEN

            SIGTEN = 0.0D0

          ELSE IF (XLAM.GT.1.0D0.AND.XLAM.LT.XLSTR) THEN

            SIGTEN = XL1\*( EXP(XL2\*(XLAM-1)) - 1 )

          ELSE IF (XLAM.GE.XLSTR) THEN

            SIGTEN = XL3\*XLAM+XL4

          END IF

        END IF

        !| Make the fibre stress value a tensor

        DIAG\_A = 0.0D0

        DIAG\_A(1,1) = XA(1,1)

        DIAG\_A(2,2) = XA(2,1)

        DIAG\_A(3,3) = XA(3,1)

        IF (PROPS(1).GT.0.0D0) THEN !|MUSCLE

          SIG3 = SIGFIB\*DIAG\_A

        ELSE IF (PROPS(1).LT.0.0D0) THEN !|TENDON

          SIG3 = SIGTEN\*DIAG\_A

        END IF

        !|333. SIG3

        !|S\_VVV> SIGVOL

        PRESS = XK\*LOG(XJ)/XJ

        SVOL = XJ\*PRESS\*CMATI

        SIGVOL = XJ\*\*(-1)\*DFGRD\*SVOL\*DFGRDI

        !|S\_VVV.

        !| TOTAL SIGMA

        SIG = SIG1+SIG2+SIG3+SIGVOL

        CAUSTR = SIG

        !|SSS. END STRESS

        RETURN

      END SUBROUTINE mykstress\_calc

      INCLUDE 'myutilities.for'

      !| DEPOT

      !|PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0)

      !| LOCAL ARRAYS

      !| DFGRD     - the deformation gradient

      !| BMAT      - left Cauchy-Green deformation tensor

      !| XKIRCH    - Kirchhoff stress

      !| XA     - array containing fibre vectors

      !| B2MAT     - square of the B tensor

      !| XA        - fibre vector in the reference configuration

      !| a\_curnt   - fibre vector in the current configuration

      !| aoa       - fibre structural tensor

      !| XANISOK   - anisotropic Kirchhoff stress for a given fibre family

      !| XANISOTOT - total anisotropic Kirchhoff stress

      !| XI4       - array storing the fibre invariant

C     Subroutine to perform a polar decomposition on the deformation gradient

C

C     Adapted from Sommer for an Abaqus subroutine. Downloaded from the

C     International Society for Biomechanics' website. This subroutine is

C     free for non-commercial use.

C     https://isbweb.org/software/movanal/sommer.txt

C

C     H.J. Sommer III, Professor of Mechanical Engineering, 207 Reber Building

C     The Pennsylvania State University, University Park, PA  16802

C     (814)865-2519, FAX (814)863-4848, Bitnet HJS@PSUECL, Internet HJS@ECL.PSU.EDU

C

c\*\*\*\*\*\*\*\*1\*\*\*\*\*\*\*\*\*2\*\*\*\*\*\*\*\*\*3\*\*\*\*\*\*\*\*\*4\*\*\*\*\*\*\*\*\*5\*\*\*\*\*\*\*\*\*6\*\*\*\*\*\*\*\*\*7\*\*

c

c kpolarDecomp(G,R,S)

c

c positive polar decomposition of 3x3 matrix  G = R \* S

c forcing positive orthonormal rotation matrix R

c

c INPUTS

c G = 3x3 general matrix

c

c OUTPUTS

c R = 3x3 positive orthonormal matrix

c S = 3x3 symmetric matrix

C| In our case F = R.U , R-rotation tensor, U-Right stretch tensor

C| ...This means, first stretch then rotate. BN

C| ...Here it is G = R.S

c

c PRECISION:  single !| !!!

c COMMONS:  none

c CALLS:    none

c FUNCTIONS:  ABS, SQRT

c REFERENCE:  Veldpaus, F.E., H.J. Woltring, and L.J.M.G. Dortmans,

c       A Least-Squares Algorithm for the Equiform

c       Transformation from Spatial Marker Coordinates,

c       J. Biomechanics, 21(1):45-54 (1988).

c DATE:   10/8/92 - HJSIII

c

c

      SUBROUTINE mykpolarDecomp(G,S,R)

c

c declarations

C       INCLUDE 'ABA\_PARAM.INC'

      dimension G(3,3),R(3,3),S(3,3),COG(3,3),XP(3,3),ADP(3,3),

     1 XPBI(3,3)

c

c constants

      EPS=1.0E-5

c

c cofactors and determinant of g

      COG(1,1)=G(2,2)\*G(3,3)-G(2,3)\*G(3,2)

      COG(2,1)=G(1,3)\*G(3,2)-G(1,2)\*G(3,3)

      COG(3,1)=G(1,2)\*G(2,3)-G(1,3)\*G(2,2)

      COG(1,2)=G(2,3)\*G(3,1)-G(2,1)\*G(3,3)

      COG(2,2)=G(1,1)\*G(3,3)-G(1,3)\*G(3,1)

      COG(3,2)=G(1,3)\*G(2,1)-G(1,1)\*G(2,3)

      COG(1,3)=G(2,1)\*G(3,2)-G(2,2)\*G(3,1)

      COG(2,3)=G(1,2)\*G(3,1)-G(1,1)\*G(3,2)

      COG(3,3)=G(1,1)\*G(2,2)-G(1,2)\*G(2,1)

      G3=G(1,1)\*COG(1,1)+G(2,1)\*COG(2,1)+G(3,1)\*COG(3,1)

c

c P = trans(G) \* G = S \* S

      DO 10000 I=1,3

      XP(I,1)=G(1,I)\*G(1,1)+G(2,I)\*G(2,1)+G(3,I)\*G(3,1)

      XP(I,2)=G(1,I)\*G(1,2)+G(2,I)\*G(2,2)+G(3,I)\*G(3,2)

      XP(I,3)=G(1,I)\*G(1,3)+G(2,I)\*G(2,3)+G(3,I)\*G(3,3)

10000 CONTINUE

c

c adjoint of P

      ADP(1,1)=XP(2,2)\*XP(3,3)-XP(2,3)\*XP(3,2)

      ADP(2,2)=XP(1,1)\*XP(3,3)-XP(1,3)\*XP(3,1)

      ADP(3,3)=XP(1,1)\*XP(2,2)-XP(1,2)\*XP(2,1)

c

c G invariants

      G1SQ=XP(1,1)+XP(2,2)+XP(3,3)

      G1=SQRT(G1SQ)

      G2SQ=ADP(1,1)+ADP(2,2)+ADP(3,3)

      G2=SQRT(G2SQ)

c

c initialize iteration

      H1=G2/G1SQ

      H2=G3\*G1/G2SQ

      X=1.0

      Y=1.0

c

c iteration loop

10001 CONTINUE

      DEN=2.0\*(X\*Y-H1\*H2)

      RES1=1.0-X\*X+2.0\*H1\*Y

      RES2=1.0-Y\*Y+2.0\*H2\*X

      DX=(Y\*RES1+H1\*RES2)/DEN

      DY=(H2\*RES1+X\*RES2)/DEN

      X=X+DX

      Y=Y+DY

      IF(ABS(DX/X).GT.EPS.OR.ABS(DY/Y).GT.EPS)GO TO 10001

c

c BETA invariants

      BETA1=X\*G1

      BETA2=Y\*G2

c

c invert ( trans(G) \* G + BETA2 \* identity )

      XP(1,1)=XP(1,1)+BETA2

      XP(2,2)=XP(2,2)+BETA2

      XP(3,3)=XP(3,3)+BETA2

      XPBI(1,1)=XP(2,2)\*XP(3,3)-XP(2,3)\*XP(3,2)

      XPBI(1,2)=XP(1,3)\*XP(3,2)-XP(1,2)\*XP(3,3)

      XPBI(1,3)=XP(1,2)\*XP(2,3)-XP(1,3)\*XP(2,2)

      XPBI(2,1)=XP(2,3)\*XP(3,1)-XP(2,1)\*XP(3,3)

      XPBI(2,2)=XP(1,1)\*XP(3,3)-XP(1,3)\*XP(3,1)

      XPBI(2,3)=XP(1,3)\*XP(2,1)-XP(1,1)\*XP(2,3)

      XPBI(3,1)=XP(2,1)\*XP(3,2)-XP(2,2)\*XP(3,1)

      XPBI(3,2)=XP(1,2)\*XP(3,1)-XP(1,1)\*XP(3,2)

      XPBI(3,3)=XP(1,1)\*XP(2,2)-XP(1,2)\*XP(2,1)

      DETPBI=XP(1,1)\*XPBI(1,1)+XP(2,1)\*XPBI(1,2)+XP(3,1)\*XPBI(1,3)

c

c R = (cofac(G)+BETA1\*G) \* inv(trans(G)\*G+BETA2\*identity)

      DO 10002 I=1,3

      R(I,1)=((COG(I,1)+BETA1\*G(I,1))\*XPBI(1,1)

     1       +(COG(I,2)+BETA1\*G(I,2))\*XPBI(2,1)

     2       +(COG(I,3)+BETA1\*G(I,3))\*XPBI(3,1))/DETPBI

      R(I,2)=((COG(I,1)+BETA1\*G(I,1))\*XPBI(1,2)

     1       +(COG(I,2)+BETA1\*G(I,2))\*XPBI(2,2)

     2       +(COG(I,3)+BETA1\*G(I,3))\*XPBI(3,2))/DETPBI

      R(I,3)=((COG(I,1)+BETA1\*G(I,1))\*XPBI(1,3)

     1       +(COG(I,2)+BETA1\*G(I,2))\*XPBI(2,3)

     2       +(COG(I,3)+BETA1\*G(I,3))\*XPBI(3,3))/DETPBI

10002 CONTINUE

c

c S = trans(R) \* G

      DO 10003 I=1,3

      S(I,1)=R(1,I)\*G(1,1)+R(2,I)\*G(2,1)+R(3,I)\*G(3,1)

      S(I,2)=R(1,I)\*G(1,2)+R(2,I)\*G(2,2)+R(3,I)\*G(3,2)

      S(I,3)=R(1,I)\*G(1,3)+R(2,I)\*G(2,3)+R(3,I)\*G(3,3)

10003 CONTINUE

c

c done

      RETURN

      END