```
0001: ! ********************************************************************************************
0002: ! **                    有限要素クラス（弾性体用のアイソパラメトリック1次要素）                **
0003: ! ********************************************************************************************
0004: !
0005: ! 2次元用：4節点四辺形要素（双1次） (4-node Quadrilateral element, Quad4 とも)
0006: !           ξ-η自然座標系における 要素節点番号，2x2ガウス積分点番号の定義
0007: !
0008: !       (x4,y4)          (x3,y3)       ξ：xに相当するギリシャ文字．クシーorグサイ．英語表記は xi
0009: !       (-1, 1)     η    ( 1, 1)       η：y          〃            イータorエータ．     〃     eta
0010: !           4_____|_____3             ζ：z          〃            ゼータ       .       〃     zeta
0011: !           |   4     |   3   |
0012: !           |   x     |   x   |
0013: !       ____|_____|___|___|____ξ
0014: !           |   1     |   2   |
0015: !           |   x     |   x   |
0016: !           |_____|_____|
0017: !         1           |       2
0018: !       (-1,-1)           ( 1,-1)
0019: !       (x1,y1)           (x2,y2)
0020: !
0021: ! 3次元用：8節点六面体要素（三重1次） (8-node Hexahedral element, Hex8 とも)
0022: !           ξ-η-ζ自然座標系における 要素節点番号，2x2x2ガウス積分点番号の定義
0023: !
0024: !                          (x8,y8,z8)          (x7,y7,z7)
0025: !                          (-1, 1, 1)          ( 1, 1, 1)
0026: !                               8_____________7
0027: !                              /:             /|
0028: !                   (x5,y5,z5)/ :x          x/ |
0029: !                   (-1,-1, 1)/  :8         7/  |
0030: !                           5/_____________6/   |
0031: !                           |   x:         x|   |
0032: !                           |   5:x        6|x  |
0033: !           ζ   η           |     :4........|3..|
0034: !           | /             |    ,4         |  /3
0035: !           |/___ξ          |  ,x           x| /( 1, 1,-1)
0036: !                           | ,  1          2|/  (x3,y3,z3)
0037: !                           |,_____________|/
0038: !                         1               2
0039: !                   (-1,-1,-1)          ( 1,-1,-1)
0040: !                   (x1,y1,z1)          (x2,y2,z2)
0041: !
0042: MODULE element_class
0043:
0044:    USE constants
0045:    USE material_class
0046:    IMPLICIT none
0047:    PRIVATE
0048:
0049:    ! material_class にて定義された定数を，念のため確認しておく．
0050:    PUBLIC :: ndim                          ! 次元数
0051:    PUBLIC :: nstr                          ! 応力/歪み成分数 (Num of STResses/STRains)
0052:
0053:    INTEGER, PARAMETER, PUBLIC  :: nnodee = 2**ndim         ! 頂点数 1-D: 2, 2-D: 4, 3-D: 8
0054:    INTEGER, PARAMETER, PUBLIC  :: nDOFn  = ndim            ! 節点あたりのDOF数 (Num of DOFs per Node)
0055:    INTEGER, PARAMETER, PUBLIC  :: nDOFe  = nnodee * nDOFn ! 要素DOF数          (       〃       Elm )
0056:    INTEGER, PARAMETER, PRIVATE :: nGP    = nnodee          ! ガウス積分点数    (Num of Gauss Points )
0057:
0058:    ! 自然座標系における形状関数用（計算量軽減のため，通常の定義と少し異なる）
0059:    REAL(DP),PARAMETER, PRIVATE :: hf     = 0.5_DP   ! 形状関数を N(ξ)=(1/2+ξi*ξ) と定義するため
0060:    REAL(DP),PARAMETER, PRIVATE :: xi_i(ndim ,nnodee) = RESHAPE ( &  ! 形状関数用の係数 ξi = ±1/2
0061: #ifdef TWO
0062:                                                                  & (/ -hf, -hf, &    ! 1
0063:                                                                     &  hf, -hf, &    ! 2
0064:                                                                     &  hf,  hf, &    ! 3
0065:                                                                     & -hf,  hf /),&  ! 4
0066: #else
0067:                                                                  & (/ -hf, -hf, -hf ,&    ! 1
0068:                                                                     &  hf, -hf, -hf ,&    ! 2
0069:                                                                     &  hf,  hf, -hf ,&    ! :
0070:                                                                     & -hf,  hf, -hf ,&    !
0071:                                                                     & -hf, -hf,  hf ,&    !
0072:                                                                     &  hf, -hf,  hf ,&    !
0073:                                                                     &  hf,  hf,  hf ,&    ! :
0074:                                                                     & -hf,  hf,  hf /),& ! 8
0075: #endif
0076:                                                               & (/ndim ,nnodee/) )
```

```
0077:
0078:    ! ガウス積分点(Gauss Points) (自然座標系) ! sqrt() は定数宣言では使用できない
0079:    REAL(DP),PARAMETER, PRIVATE :: GP(ndim ,nGP) = xi_i(:,:) * 1.154700538379252_DP
0080:    ! 1.154700538379252 = ±(1/2) * 2/sqrt(3.0_DP) = 0.577350269189626
0081:
0082:    ! 有限要素物性に関する Abstract Data Type, ADT の定義
0083:    TYPE, PUBLIC :: elm_c
0084:       PRIVATE
0085:       INTEGER :: matNo              ! この要素に対する材料領域番号
0086:       INTEGER :: nodeNo ( nnodee )  ! この要素の要素節点番号から全体節点番号への変換TBL
0087:    END TYPE elm_c
0088:
0089:    PUBLIC :: init
0090:    INTERFACE init
0091:       ! 要素が接続する全体節点番号と材料領域番号の読込
0092:       MODULE PROCEDURE read_elm_c
0093:    END INTERFACE
0094:
0095:    PUBLIC :: get_nodeNo
0096:    INTERFACE get_nodeNo
0097:       MODULE PROCEDURE get_nodeNo_TBL
0098:    END INTERFACE
0099:
0100:    PUBLIC :: get_matNo
0101:    INTERFACE get_matNo
0102:       MODULE PROCEDURE get_matNo_TBL
0103:    END INTERFACE
0104:
0105:    PUBLIC :: get_DOFg
0106:    INTERFACE get_DOFg
0107:       ! 要素DOFから全体DOFへの変換TBLの作成
0108:       MODULE PROCEDURE get_DOFe2g_TBL
0109:    END INTERFACE
0110:
0111:    PUBLIC :: eval_Mass
0112:    INTERFACE eval_Mass
0113:       ! 要素整合質量行列を返す
0114:       MODULE PROCEDURE eval_Mass_elm_c
0115:    END INTERFACE
0116:
0117:    PUBLIC :: eval_Fin_and_K
0118:    INTERFACE eval_Fin_and_K
0119:       ! 要素内力ベクトル,要素剛性行列を返す
0120:       MODULE PROCEDURE eval_Fin_and_K_elm_c
0121:    END INTERFACE
0122:
0123:    PUBLIC :: wrt
0124:    INTERFACE wrt
0125:       ! ファイルに計算結果を書き込み
0126:       MODULE PROCEDURE wrt_elm_c
0127:    END INTERFACE
0128:
0129: CONTAINS
0130:
0131: !*************************************************************
0132:    SUBROUTINE read_elm_c ( this, uid, status )
0133:       ! 要素が接続する全体節点番号と材料領域番号の読込
0134:       TYPE(elm_c),        INTENT(out  ) :: this
0135:       INTEGER,            INTENT(in   ) :: uid    ! ファイルの装置番号
0136:       INTEGER, OPTIONAL,  INTENT(out  ) :: status ! I/O アクセスの結果
0137:       INTEGER :: stat
0138:       READ ( uid, *, IOSTAT=stat ) this%nodeNo(:), this%matNo
0139:       IF ( PRESENT(status) ) status = stat
0140:    END SUBROUTINE read_elm_c
0141:
0142: !*************************************************************
0143:    PURE FUNCTION get_nodeNo_TBL ( this ) RESULT ( nodeNo )
0144:       TYPE(elm_c),        INTENT(in   ) :: this
0145:       INTEGER                           :: nodeNo(nnodee)
0146:       nodeNo(:) = this%nodeNo(:)
0147:    END FUNCTION get_nodeNo_TBL
0148:
0149: !*************************************************************
0150:    PURE FUNCTION get_matNo_TBL ( this ) RESULT ( matNo   )
0151:       TYPE(elm_c),        INTENT(in   ) :: this
0152:       INTEGER                           :: matNo
```

```
0153:       matNo = this%matNo
0154:    END FUNCTION get_matNo_TBL
0155:
0156: !************************************************************
0157:    PURE FUNCTION get_DOFe2g_TBL ( this, node2DOFg_DB, edge2DOFg_DB ) RESULT ( DOFe2g )
0158:
0159:       ! 要素自由度番号から全体自由度番号への変換テーブルを返す
0160:
0161:       TYPE(elm_c),         INTENT(in   ) :: this
0162:       INTEGER,             INTENT(in   ) :: node2DOFg_DB(:,:) ! 要素節点番号から全体節点番号への変換DB
0163:       INTEGER, OPTIONAL,   INTENT(in   ) :: edge2DOFg_DB(:)   ! 要素辺番号から全体辺番号への変換DB
0164:       INTEGER                            :: DOFe2g    (nDOFe)  ! 要素DOF番号から全体DOF番号への変換TBL
0165:
0166:       ! 要素DOFの並びの定義. 質量行列, 内力ベクトル, 要素剛性の並びはこの定義に従う.
0167:       DOFe2g(:) = RESHAPE ( node2DOFg_DB(1:nDOFn, this%nodeNo(:) ), (/nDOFe/) )
0168:
0169:    END FUNCTION get_DOFe2g_TBL
0170:
0171: !************************************************************
0172:    SUBROUTINE eval_Mass_elm_c ( this, X_DB, mat_DB, Me )
0173:       ! 要素整合質量行列を返す
0174:       TYPE(elm_c),             INTENT(in   ) :: this
0175:       REAL(DP),                INTENT(in   ) :: X_DB  (:,:)    ! 要素節点の座標値(ndim, nnode)
0176:       TYPE(mat_c),             INTENT(in   ) :: mat_DB(:)      ! 各材料に関する物性値についての情報
0177:       REAL(DP),                INTENT(out  ) :: Me    (nDOFe, nDOFe)  ! 要素質量行列
0178:       REAL(DP)                               :: Ni                   ! 形状関数
0179:     ! Ni  = ( hf + GP(1,ip)*xi_i(1,i) ) * ( hf + GP(2,ip)*xi_i(2,i) )
0180:       WRITE(*,*) "要素整合質量行列は実装していません"
0181:       Me(:,:) = 0.0_DP
0182:    END SUBROUTINE eval_Mass_elm_c
0183:
0184: !************************************
0185:    SUBROUTINE eval_Dump_elm_c( this, X_DB, mat_DB, Ce )
0186:       ! 要素減衰行列を返す
0187:       TYPE(elm_c),             INTENT(in   ) :: this
0188:       REAL(DP),                INTENT(in   ) :: X_DB  (:,:)    ! 要素節点の座標値(ndim, nnode)
0189:       TYPE(mat_c),             INTENT(in   ) :: mat_DB(:)      ! 各材料に関する物性値についての情報
0190:       REAL(DP),                INTENT(out  ) :: Ce    (nDOFe, nDOFe)  ! 要素減衰行列
0191:       WRITE(*,*) '減衰行列は実装していない. '
0192:       Ce(:,:) = -9.9999_DP
0193:    END SUBROUTINE eval_Dump_elm_c
0194:
0195: !************************************************************
0196:    SUBROUTINE update_elm_c ( this, X_DB, mat_DB, Ue )
0197:
0198:       TYPE(elm_c),             INTENT(inout) :: this
0199:       REAL(DP),    OPTIONAL,   INTENT(in   ) :: X_DB  (:,:)    ! 要素節点の座標値(ndim, nnode)
0200:       TYPE(mat_c),OPTIONAL,    INTENT(in   ) :: mat_DB(:)      ! 材料領域
0201:       REAL(DP),    OPTIONAL,   INTENT(in   ) :: Ue    (nDOFe)  ! 要素変位
0202:
0203:       ! 今回は, 何もしなくて良い. すなわち, ゴミルーチン.
0204:
0205:    END SUBROUTINE update_elm_c
0206: !
```

```
0207:
0208: !*************************************************************
0209:    SUBROUTINE eval_Fin_and_K_elm_c ( this, X_DB, mat_DB, Ue, Fine, Ke )
0210:
0211:       ! 要素変位ベクトルUe を入力し，静的要素内力ベクトルFine と要素接線剛性行列Ke を返す
0212:
0213:       TYPE(elm_c),           INTENT(in   ) :: this
0214:       REAL(DP),              INTENT(in   ) :: X_DB  (:,:)    ! 要素節点の座標値(ndim, nnode)
0215:       TYPE(mat_c),           INTENT(in   ) :: mat_DB(:)      ! 各材料に関する物性値についての情報
0216:       REAL(DP),              INTENT(in   ) :: Ue    (nDOFe)        ! 要素変位＆ベクトルポテンシャル
0217:       REAL(DP),              INTENT(out  ) :: Fine  (nDOFe)        ! 要素内力ベクトル
0218:       REAL(DP),    OPTIONAL, INTENT(out  ) :: Ke    (nDOFe, nDOFe) ! 要素剛性行列
0219:
0220:       REAL(DP) :: Xe    (ndim  ,nnodee)  ! 要素節点座標
0221:       REAL(DP) :: Be    (nstr , nDOFe)   ! 変位ー歪み変換行列
0222:       REAL(DP) :: BeT   (nDOFe, nstr )   ! = Be^T
0223:       REAL(DP) :: stress(nstr )
0224:       REAL(DP) :: strain(nstr )
0225:       REAL(DP) :: cD    (nstr , nstr )   ! 弾性係数行列
0226:       REAL(DP) :: det_Je                 ! あるガウス積分点におけるヤコビ行列Jeの行列式
0227:       integer  :: ip
0228:
0229:       Xe(:,:) = X_DB(:,this%nodeNo(:))
0230:       ! ガウス求積法により要素内力ベクトルと要素接線剛性行列を評価
0231:       Fine(:  ) = 0.0_DP
0232:       Ke  (:,:) = 0.0_DP
0233:       DO ip = 1, nGP     ! ガウス積分点GP(:,ip) における関数値に 重み(w=1)を掛けつつ加算
0234:
0235:          CALL eval_Be_and_det_Je ( Xe, GP(:,ip), Be, det_Je )
0236:          strain = MATMUL( Be, Ue )
0237:          BeT    = TRANSPOSE( Be )
0238:
0239:          IF ( PRESENT( Ke ) ) THEN
0240:             CALL eval_Fin_and_K ( mat_DB(this%matNo), strain, stress , cD )
0241:             Ke(:,:) = Ke(:,:) + MATMUL( BeT, MATMUL( cD, Be) ) * det_Je ! *w(=1)
0242:          ELSE
0243:             CALL eval_Fin_and_K ( mat_DB(this%matNo), strain, stress      )
0244:          END IF
0245:
0246:          Fine(:) = Fine(:) + MATMUL( BeT, stress ) * det_Je !*w(=1)
0247:
0248:        ! ! 等価要素節点フラックス(荷重)の作成 (各積分点での等価要素節点荷重の足し込み)
0249:        ! DO i = 1, nnodee  ! 形状関数に関しては，マトリクスを作らず逐一計算していく.
0250:        !    Ni  = ( hf + GP(1,ip)*xi_i(1,i) ) * ( hf + GP(2,ip)*xi_i(2,i) )
0251:        !    fe(ndim*i-1:ndim*i ) =   fe(ndim*i-1:ndim*i ) &
0252:        !                                & + this%bdy_per_A(:) * Ni *det_Je      ! *w(=1)
0253:        ! END DO
0254:
0255:       END DO
0256:
0257:    END SUBROUTINE eval_Fin_and_K_elm_c
0258:
0259: !*************************************************************************
0260:    SUBROUTINE wrt_elm_c ( this, X_DB, mat_DB, Ue ,uid, status )
0261:
0262:       ! ファイルに計算結果を書き込み
0263:
0264:       TYPE(elm_c),           INTENT(in   ) :: this
0265:       REAL(DP),              INTENT(in   ) :: X_DB  (:,:)   ! (ndim, nnode)
0266:       TYPE(mat_c),           INTENT(in   ) :: mat_DB(:)     ! 各材料に関する物性値についての情報
0267:       REAL(DP),              INTENT(in   ) :: Ue    (nDOFe) ! 要素変位
0268:       INTEGER,               INTENT(in   ) :: uid           ! ファイルの装置番号
0269:       INTEGER,     OPTIONAL, INTENT(out  ) :: status        ! I/O アクセスの結果
0270:
0271:       REAL(DP) :: Xe    (ndim  ,nnodee)  ! 要素節点座標
0272:       REAL(DP) :: Be  ( nstr, nDOFe )  ! 変位ー歪み変換行列
0273:       REAL(DP) :: stress (nstr)
0274:       REAL(DP) :: strain (nstr)
0275:       INTEGER  :: stat
0276:
0277:       REAL(DP) :: det_Je     ! あるガウス積分点におけるヤコビ行列Jeの行列式
0278:       integer  :: ip
0279:
0280:       Xe(:,:) = X_DB(:,this%nodeNo(:))
0281:       DO ip = 1, nGP     ! ガウス積分点GP(:,ip)
0282:          CALL eval_Be_and_det_Je ( Xe, GP(:,ip), Be, det_Je )
```

```
0283:          strain = MATMUL( Be, Ue )
0284:          CALL eval_Fin_and_K ( mat_DB(this%matNo), strain, stress      )
0285:          strain(ndim+1:nstr) = strain(ndim+1:nstr) * 0.5_DP  ! GiD可視化用にテンソル歪み成分に変換
0286:          WRITE( uid, '(22'//FP_FMT//')', IOSTAT=stat ) strain(:), stress(:)
0287:       END DO
0288:       IF ( PRESENT(status) ) status = stat
0289:
0290:    END SUBROUTINE wrt_elm_c
0291:
0292: !************************************************************
0293:    SUBROUTINE eval_Be_and_det_Je ( Xe, xi, Be, det_Je )
0294:
0295:       ! 自然座標系のある点 (ξ,η,ζ) (=xi(:)) における
0296:       !  Bマトリクス(変位ー歪み変換行列)と ヤコビ行列 Je の det(Je) を評価
0297:
0298:       REAL(DP), INTENT(in   ) :: Xe    (ndim  ,nnodee)  ! 要素節点座標 (物理座標)
0299:       REAL(DP), INTENT(in   ) :: xi    (ndim         )  ! 自然座標系のある点ξの値 (専ら積分点)
0300:       REAL(DP), INTENT(out  ) :: Be    (nstr  ,nDOFe )  ! その点における Bマトリクス
0301:       REAL(DP), INTENT(out  ) :: det_Je                 !       〃       ヤコビ行列Jeの行列式
0302:
0303:       REAL(DP)                :: Je    (ndim  ,ndim  )  !       〃      ヤコビ行列(Jacobian)
0304:       REAL(DP)                :: inv_Je(ndim  ,ndim  )  !       〃      ヤコビ行列Jeの逆行列
0305:       REAL(DP)                :: dN_xi (ndim  ,nnodee)  ! 形状関数の自然座標系による微分
0306:       REAL(DP)                :: dN_x  (ndim         )  !     〃    物理座標系による微分
0307:       INTEGER                 :: i, j, k
0308:
0309:       ! ある点 ξ=xi(:) におけるヤコビ行列(Jacobian) [∂(x,y)/∂(ξ,η)] の値を評価
0310:       Je(:,:) = 0.0_DP
0311:       DO i  = 1, nnodee
0312: #ifdef TWO
0313:          dN_xi(1,i) =     xi_i(1,i)        *(hf+xi_i(2,i)*xi(2)) ! = ∂N/∂ξ  (where ξi = ±1/2)
0314:          dN_xi(2,i) = (hf+xi_i(1,i)*xi(1)) *    xi_i(2,i)        ! = ∂N/∂η
0315: #else
0316:          dN_xi(1,i) =     xi_i(1,i)        *(hf+xi_i(2,i)*xi(2)) *(hf+xi_i(3,i)*xi(3)) ! = ∂N/∂ξ
0317:          dN_xi(2,i) = (hf+xi_i(1,i)*xi(1)) *    xi_i(2,i)        *(hf+xi_i(3,i)*xi(3)) ! = ∂N/∂η
0318:          dN_xi(3,i) = (hf+xi_i(1,i)*xi(1)) *(hf+xi_i(2,i)*xi(2)) *    xi_i(3,i)        ! = ∂N/∂ζ
0319: #endif
0320:          DO k = 1, ndim
0321:             DO j = 1, ndim
0322:                Je(j,k) = Je(j,k) + dN_xi(j,i)*Xe(k,i)
0323:             END DO
0324:          END DO
0325:       END DO
0326:
0327:       ! ヤコビ行列の行列式と逆行列 Je^(-1) = [∂(ξ,η)/∂(x,y)] を計算
0328: #ifdef TWO
0329:       CALL inv_2x2( Je, inv_Je, det_Je )
0330: #else
0331:       CALL inv_3x3( Je, inv_Je, det_Je )
0332: #endif
0333:
0334:       ! Bマトリクスを評価
0335:       DO i = 1, nnodee
0336:          ! 微分の連鎖則 : {∂N/∂(x,y)} = [∂(ξ,η)/∂(x,y)] {∂N/∂(ξ,η)}
0337:          dN_x = MATMUL(inv_Je, dN_xi(:,i))
0338: #ifdef TWO
0339:          Be(:,ndim*i-1) = (/  dN_x(1),  0.0_DP, dN_x(2) /)
0340:          Be(:,ndim*i  ) = (/  0.0_DP, dN_x(2), dN_x(1) /)
0341: #else
0342:          Be(:,ndim*i-2) = (/ dN_x(1),  0.0_DP,  0.0_DP, dN_x(2),  0.0_DP, dN_x(3) /)
0343:          Be(:,ndim*i-1) = (/  0.0_DP, dN_x(2),  0.0_DP, dN_x(1), dN_x(3),  0.0_DP /)
0344:          Be(:,ndim*i  ) = (/  0.0_DP,  0.0_DP, dN_x(3),  0.0_DP, dN_x(2), dN_x(1) /)
0345: #endif
0346:       END DO
0347:
0348:    END SUBROUTINE eval_Be_and_det_Je
0349:
0350: !*****************************************************************************
0351:    SUBROUTINE inv_2x2( A, invA, detA )
0352:
0353:       ! 公式による逆行列の求解 (2x2行列限定)
0354:
0355:       REAL(DP), INTENT(in   ) :: A   (2,2)
0356:       REAL(DP), INTENT(out  ) :: invA(2,2) ! 逆行列
0357:       REAL(DP), INTENT(out  ) :: detA      ! ヤコビアン
0358:
```

```
0359:       REAL(DP) :: div_detA
0360:
0361:       detA = A(1,1)*A(2,2) -A(1,2)*A(2,1)
0362:       div_detA  = 1.0_DP/detA
0363:       invA(1,1) =  A(2,2)*div_detA
0364:       invA(2,1) = -A(2,1)*div_detA
0365:       invA(1,2) = -A(1,2)*div_detA
0366:       invA(2,2) =  A(1,1)*div_detA
0367:    END SUBROUTINE inv_2x2
0368:
0369:
0370:    SUBROUTINE inv_3x3( A, invA, detA )
0371:
0372:       ! 公式による逆行列の求解（3x3行列限定）
0373:
0374:       REAL(DP), INTENT(in   ) :: A    (3,3)
0375:       REAL(DP), INTENT(out  ) :: invA(3,3) ! 逆行列
0376:       REAL(DP), INTENT(out  ) :: detA      ! ヤコビアン
0377:
0378:       REAL(DP) :: div_detA
0379:
0380:       detA =  A(1,1)*A(2,2)*A(3,3) +A(2,1)*A(3,2)*A(1,3) +A(3,1)*A(1,2)*A(2,3) & ! サラス
0381:            & -A(3,1)*A(2,2)*A(1,3) -A(2,1)*A(1,2)*A(3,3) -A(1,1)*A(3,2)*A(2,3)   ! 公式
0382:       div_detA  = 1.0_DP/detA
0383:       invA(1,1) = ( A(2,2)*A(3,3) -A(2,3)*A(3,2) ) * div_detA
0384:       invA(2,1) = ( A(2,3)*A(3,1) -A(2,1)*A(3,3) ) * div_detA
0385:       invA(3,1) = ( A(2,1)*A(3,2) -A(2,2)*A(3,1) ) * div_detA
0386:       invA(1,2) = ( A(3,2)*A(1,3) -A(3,3)*A(1,2) ) * div_detA
0387:       invA(2,2) = ( A(3,3)*A(1,1) -A(3,1)*A(1,3) ) * div_detA
0388:       invA(3,2) = ( A(3,1)*A(1,2) -A(3,2)*A(1,1) ) * div_detA
0389:       invA(1,3) = ( A(1,2)*A(2,3) -A(1,3)*A(2,2) ) * div_detA
0390:       invA(2,3) = ( A(1,3)*A(2,1) -A(1,1)*A(2,3) ) * div_detA
0391:       invA(3,3) = ( A(1,1)*A(2,2) -A(1,2)*A(2,1) ) * div_detA
0392:
0393:    END SUBROUTINE inv_3x3
0394:
0395: END MODULE element_class
```