```
0001: PROGRAM main_linear
0002:
0003:    USE constants
0004:    USE material_class  ! 材料クラス
0005:    USE element_class   ! 要素クラス
0006:    USE adj_mtx_class   ! 隣接行列クラス (ADJacent MaTriX)
0007:    USE sp_mtx_class    ! 疎行列クラス   (SPaRse   MaTriX)
0008:    USE timer_class
0009:    USE mesh_module     ! 結構スパゲティ
0010:
0011:    USE cond_node_class ! ほとんどスパゲティ
0012:    USE GiD_IO_module   ! 結構スパゲティ
0013: !$ USE OMP_LIB
0014:
0015:    IMPLICIT none
0016:
0017:    INTEGER                    :: nmat           ! 全材料領域数  (Num of MATerials)
0018:    TYPE(mat_c), ALLOCATABLE :: mat_DB   ( : )  ! (nmat  )       材料領域の物性値DataBase
0019:
0020:    INTEGER                    :: nnode          ! 全体節点数    (Num of NODEs    )
0021:    REAL(DP),    ALLOCATABLE :: X_DB     (:,:)  ! (ndim ,nnode) 節点座標DataBase
0022:
0023:    INTEGER                    :: nelm           ! 要素数        (Num of ELeMents )
0024:    TYPE(elm_c), ALLOCATABLE :: elm_DB   ( : )  ! (nelm  )       要素に関するDataBase
0025:
0026:    INTEGER                    :: ncndn          ! D/N境界条件を課す節点数(Num of CoND. Nodes)
0027:    TYPE(cndn_c)               :: cndn           ! 節点に関するD/N条件や従属条件の集合
0028:
0029:    INTEGER,     ALLOCATABLE :: elm2DOFg(:,:)   ! 接続行列
0030:    TYPE(jag),   ALLOCATABLE :: col2elm(:)      ! 色毎の要素
0031:    INTEGER                    :: ic, p
0032:
0033:    INTEGER,     ALLOCATABLE :: node2DOFg(:,:)  ! (nDOFn,nnode) 全体節点番号から全体DOF番号へ
0034:                                                !  nDOFn        節点あたりのDOF数
0035:    INTEGER                    :: nDOF_n         ! D条件(第1種)の最小全体DOF番号 < 0, Nega
0036:    INTEGER                    :: nDOF_p         ! N条件(第2種)  最大    〃       > 0, Posi, +
0037:    REAL(DP),    ALLOCATABLE :: U   (:)        ! (nDOF_n:nDOF_p) 既知( :-1)/未知(1: ) 変位
0038:    REAL(DP),    ALLOCATABLE :: Fin(:)         ! (     〃      ) 内力 (弾性エネルギの勾配)
0039:    REAL(DP),    ALLOCATABLE :: Fex(:)         ! (     〃      ) 未知( :-1)/既知(1: ) 外力
0040:    REAL(DP),    ALLOCATABLE :: F   (:)        ! (     1:nDOF_p) 残差
0041:
0042:    TYPE(adj_mtx_c)            :: A_pp           ! 全体DOFに関する隣接行列
0043:    TYPE(sp_mtx_c)             :: K_pp           ! 全体接線剛性行列 (弾性エネルギのヘッセ行列)
0044:
0045:    INTEGER                    :: DOFe2g   (nDOFe) ! 要素DOF番号から全体DOF番号への変換DB
0046:    REAL(DP)                   :: Ue       (nDOFe) ! 要素変位数ベクトル
0047:    REAL(DP)                   :: Fine     (nDOFe) ! 要素内力数ベクトル
0048:    REAL(DP)                   :: Ke(nDOFe,nDOFe) ! 要素接線剛性行列
0049:    REAL(DP) :: dF2ex2
0050:    INTEGER  :: step, i, k
0051:
0052:    CHARACTER(MAXCHAR) :: ID          ! モデル名 (入力ファイル群の共通名)
0053:
0054:    INTEGER  :: nThreads = 0          ! OMP_NUM_THREADS  (Open-MP並列化なき場合は 0 )
0055:    REAL(DP) :: wtime(10)
0056:
0057: !$ nThreads = OMP_GET_MAX_THREADS()
0058:    IF ( nThreads == 0 ) THEN
0059:       WRITE(*,'("逐次実行")')
0060:    ELSE
0061:       WRITE(*,'("Open-MP並列実行 ( OMP_NUM_THREADS = ",I2," )")') nThreads
0062:    END IF
0063:
0064: !   ID = "example1"
0065:    WRITE(*,'(A)',advance='no') 'モデル名 (入力ファイル群の共通名) :'
0066:    READ (*,'(A)') ID
0067:
0068:    CALL read_files_and_set
0069: !
```

```
0070:
0071:    CALL tic()
0072:    ! 要素のマルチカラー化
0073:    ALLOCATE( elm2DOFg(nDOFe,nelm) )
0074:    DO k = 1, nelm
0075:       elm2DOFg(:,k) = get_DOFg( elm_DB(k), node2DOFg )
0076:    END DO
0077:    CALL mk_col2elm_from_elm2DOFg( col2elm, elm2DOFg )
0078:    DEALLOCATE( elm2DOFg )
0079:    wtime(1) = toc()
0080:
0081:    CALL tic()
0082:    ! 隣接行列 A_pp の作成
0083:    CALL init( A_pp, nDOF_p )
0084:    !$OMP PARALLEL
0085:    DO ic = LBOUND(col2elm,dim=1), UBOUND(col2elm,dim=1)
0086:    !$OMP DO PRIVATE( p, k, DOFe2g )
0087:       DO p = 1, SIZE(col2elm(ic)%c)
0088:          k = col2elm(ic)%c(p)
0089:          DOFe2g(:) = get_DOFg( elm_DB(k), node2DOFg )
0090:          CALL add_clique( A_pp, DOFe2g )
0091:       END DO
0092:       !$OMP END DO
0093:    END DO
0094:    !$OMP END PARALLEL
0095: !   CALL wrt( A_pp, 'A_pp_mtx_org.dat')! 隣接行列を出力
0096:    wtime(2) = toc()
0097:
0098:    CALL tic()
0099: !   ! リナンバリング
0100:    CALL renumber( A_pp, node2DOFg )
0101: !   CALL wrt( A_pp, 'A_pp_mtx_new.dat')! 隣接行列を出力
0102:    wtime(3) = toc()
0103:
0104:    ! 接線剛性行列 K_pp と節点内力 Fin の作成
0105:    CALL tic()
0106:    CALL init( K_pp, A_pp )
0107:    wtime(4) = toc()
0108:
0109:    ALLOCATE ( U  (nDOF_n:nDOF_p), &
0110:             & Fin(nDOF_n:nDOF_p), &
0111:             & Fex(nDOF_n:nDOF_p), &
0112:             & F  (1     :nDOF_p)  )
0113:
0114:    !$OMP PARALLEL WORKSHARE
0115:    U  (:) = 0.0_DP
0116:    Fex(:) = 0.0_DP
0117:    Fin(:) = 0.0_DP
0118:    !$OMP END PARALLEL WORKSHARE
0119:    dF2ex2 = 0.0_DP
0120:
0121:    ! D/N 境界条件の設定（コードはスパゲティ）
0122:    CALL proceed_next_U_and_Fex( cndn, node2DOFg, nDOF_n, nDOF_p, U, Fex, dF2ex2 )
0123:
0124:    CALL tic()
0125:    !$OMP PARALLEL
0126:    DO i = LBOUND(col2elm,DIM=1), UBOUND(col2elm,DIM=1)
0127:       !$OMP DO PRIVATE( p, k, DOFe2g, Ue, Fine, Ke )
0128:       DO p = 1, SIZE(col2elm(i)%c)
0129:          k = col2elm(i)%c(p)
0130:          DOFe2g(:) = get_DOFg( elm_DB(k), node2DOFg )
0131:          Ue(:)     = U(DOFe2g(:))
0132:          CALL eval_Fin_and_K( elm_DB(k), X_DB, mat_DB, Ue , Fine, Ke )
0133:          Fin(DOFe2g(:)) = Fin(DOFe2g(:)) + Fine(:)
0134:          CALL add_clique( K_pp, Ke, DOFe2g )
0135:       END DO
0136:       !$OMP END DO
0137:    END DO
0138:    !$OMP END PARALLEL
0139:    wtime(5) = toc()
0140:
0141:    ! 求解
0142:    !$OMP PARALLEL WORKSHARE
0143:    F(:) = Fex(1: ) - Fin(1: )
0144:    !$OMP END PARALLEL WORKSHARE
0145:
```

```
0146:     CALL tic()
0147:     CALL solve( K_pp, F, U(1: ) )
0148:     wtime(6) = toc()
0149:
0150:     CALL final( K_pp )
0151:     CALL final( A_pp )
0152:     CALL final( cndn )
0153:
0154:     ! ポストプロセス（GiD用の出力）
0155:     step = 1
0156:     CALL init_GiD_postfile( ID,        X_DB, elm_DB )
0157:     CALL  wrt_GiD_postfile( ID, step, X_DB, elm_DB, mat_DB,         &
0158:                           & nDOF_n, nDOF_p, U, Fin, Fex, node2DOFg )
0159:
0160:     WRITE(*,'("---- 計算時間の内訳（秒）  OMP_NUM_THREADS = ",I2,"----")'), nThreads
0161:     WRITE(*,'("   マルチカラー化: ",F10.3)') wtime(1)
0162:     WRITE(*,'("   隣接行列の作成: ",F10.3)') wtime(2)
0163:     WRITE(*,'("   自由度番号付替: ",F10.3)') wtime(3)
0164:     WRITE(*,'("   ダウンキャスト: ",F10.3)') wtime(4)
0165:     WRITE(*,'("   剛性行列の作成: ",F10.3)') wtime(5)
0166:     WRITE(*,'("   方程式の求解:   ",F10.3)') wtime(6)
0167: !
```

```
0168:
0169: CONTAINS
0170: !***********************************************************************
0171:    SUBROUTINE read_files_and_set ( )
0172:
0173:       INTEGER :: uid = 50  ! 装置番号
0174:       INTEGER :: i, dummy, nbelm
0175:       LOGICAL :: exist
0176:
0177:       ! まずは, 配列に動的にメモリを割り当てるために必要となる値を読み込む
0178:       OPEN (uid,file=TRIM(ID)//'.size.dat', action='read')
0179:       READ (uid,*) nnode
0180:       READ (uid,*) nelm
0181:       READ (uid,*) nmat
0182:       READ (uid,*) nbelm
0183:       READ (uid,*) dummy  ! dummy (読み込みを変えないためだけ)
0184:       READ (uid,*) ncndn  ! これは size.dat に入れるメリットなし. *.cndファイルに入れるべき？
0185:       CLOSE(uid)
0186:
0187:       ! 節点座標値の読み込み
0188:       ALLOCATE( X_DB(ndim  ,nnode ) )
0189:       OPEN( uid, file=TRIM(ID)//'.node.dat', action='read')
0190:       DO i = 1, nnode
0191:          READ (uid,*) X_DB(:,i)
0192:       END DO
0193:       CLOSE(uid)
0194:
0195:       ! 要素データ (構造型) の読み込み
0196:       ALLOCATE( elm_DB( nelm ) )
0197:       OPEN( uid, file=TRIM(ID)//'.elm.dat', action='read')
0198:       DO i = 1, nelm
0199:          CALL init( elm_DB(i), uid )
0200:       END DO
0201:       CLOSE(uid)
0202:
0203:       ! 材料領域データ (構造型) の読み込み
0204:       ALLOCATE( mat_DB( nmat ) )
0205:       OPEN( uid, file=TRIM(ID)//'.mat.dat', action='read')
0206:       DO i = 1, nmat
0207:          CALL init( mat_DB(i), uid )
0208:       END DO
0209:       CLOSE(uid)
0210:
0211:       ! 節点に関する D/N境界条件を読み込むとともに, 全体DOF番号(仮)を割り振る.
0212:       nDOF_n = 0
0213:       nDOF_p = 0
0214:       ALLOCATE( node2DOFg( nDOFn, nnode ) )
0215:       OPEN( uid, file=TRIM(ID)//'.cndn.dat', action='read')
0216:       CALL init( cndn, ncndn, uid, node2DOFg, nDOF_n, nDOF_p )
0217:       CLOSE(uid)
0218:
0219:    END SUBROUTINE read_files_and_set
0220:
0221: END PROGRAM main_linear
```