```
!*****************************************************************************************
!*** 疎行列とその計算      (adj_mtx_c型のサブクラス)                                      ***
!***                   内部データ構造：「圧縮行格納(Compressed Row Storage, CRS)形式」   ***
!***                   内部アルゴリズム：直接求解もしくは反復求解                        ***
!*****************************************************************************************
!
! ○ 直接求解には，PARDISOライブラリ( http://pardiso-project.org/ ) を使用.
!
!    本コードでは，Intel MKL (Math Kernel Library)版を使用している.
!       ・コンパイルオプションに"-fpp -DMKL" を追加
!       ・Fortran90による具体例は mklディレクトリ以下を参照
!           ....¥mkl¥examples¥examples_core_f.zip¥solverf¥source¥pardiso_sym_f90.f90
!       ・詳細は「Reference Manual for Intel Math Kernel Library」を参照
!
!    もし，本家のAcademic(Free)版 PARDISO ライブラリを使う場合には以下に注意のこと.
!       ・Linux + gfortran についてのみバイナリが存在（登録のみで1年間ライセンス）
!       ・本コード中の CALL pardiso（ ）の引数と若干違うかもしれない（未確認）
!
! ○ 反復求解では，PARDISOのごく一部の引数のみだが，共有できるようにルーチンを設計した.

#ifdef MKL
INCLUDE 'mkl_pardiso.f90'    ! Intel MKL版 pardiso を使うためのヘッダファイル（intel提供）
#endif
!
```

```
0027:
0028: MODULE sp_mtx_class
0029:
0030:    USE constants
0031:    USE sort_and_search_module, ONLY: binary_search
0032: #ifdef MKL
0033:    USE mkl_pardiso                                  ! mkl_pardiso.f90 内に定義あり
0034: #else
0035:    USE PCG_CRS_module
0036: #endif
0037:    USE matvec_CRS_module
0038:    IMPLICIT none
0039:    PRIVATE
0040:                                                     ! this%state の遷移状態を列挙
0041:    INTEGER, PARAMETER :: not_yet_solved   = 1,& ! "まだ求解ルーチンsolveを呼出していない"
0042:                        & new_mtx_to_solve = 2,& ! "解くべき新規の行列"
0043:                        & solved_the_mtx   = 3   ! "その行列を解いたことあり"
0044:
0045:    TYPE, PUBLIC :: sp_mtx_c
0046:       PRIVATE
0047:       INTEGER,  POINTER     :: ind(:) => NULL() ! (n  ) adj_mtx_c型の ind を指す
0048:       INTEGER,  POINTER     :: DOF(:) => NULL() ! (nnz)       〃        DOF    〃
0049:                                                 !       FORALL( p=ind(i):ind(i+1)-1 ) DOF(p)
0050:       REAL(DP), ALLOCATABLE :: a   (:)          ! (nnz) 非ゼロ(Non-Zero)の値を格納
0051:       INTEGER               :: state &          ! 状態機械としての遷移状態フラグ (solve専用)
0052:                              & = not_yet_solved
0053: #ifdef MKL
0054:       TYPE(mkl_pardiso_handle) :: pt(64)        ! 内部アドレス格納用 (PARDISOが64を指定)
0055: #else
0056:       TYPE(PCG_M_handle) :: solver_handle
0057: #endif
0058:    END TYPE sp_mtx_c
0059:
0060:    PUBLIC :: init
0061:    INTERFACE init
0062:       MODULE PROCEDURE init_mtx_CRS
0063:    END INTERFACE
0064:
0065:    PUBLIC :: final
0066:    INTERFACE final
0067:       MODULE PROCEDURE final_mtx_CRS
0068:    END INTERFACE
0069:
0070:    PUBLIC :: clear
0071:    INTERFACE clear
0072:       MODULE PROCEDURE clear_mtx_CRS
0073:    END INTERFACE
0074:
0075:    PUBLIC :: add_clique
0076:    INTERFACE add_clique
0077:       MODULE PROCEDURE add_clique_mtx_CRS
0078:    END INTERFACE
0079:
0080:    PUBLIC :: wrt
0081:    INTERFACE wrt
0082:       MODULE PROCEDURE wrt_mtx_CRS
0083:    END INTERFACE
0084:
0085:    PUBLIC :: add
0086:    INTERFACE add
0087:       MODULE PROCEDURE add_mtx_CRS
0088:    END INTERFACE
0089:
0090:    PUBLIC :: mul
0091:    INTERFACE mul
0092:       MODULE PROCEDURE mul_mtx_CRS
0093:    END INTERFACE
0094:
0095:    PUBLIC :: solve
0096:    INTERFACE solve
0097:       MODULE PROCEDURE solve_mtx_CRS
0098:    END INTERFACE
0099:
0100: CONTAINS
0101: !
```

```
0102:
0103: !*****************************************************************************************
0104:    SUBROUTINE init_mtx_CRS ( this ,adj )
0105:
0106:       USE adj_mtx_class
0107:
0108:       TYPE(sp_mtx_c),  INTENT(inout) :: this
0109:       TYPE(adj_mtx_c), INTENT(inout) :: adj
0110:
0111:       INTEGER :: nnz        ! Num. of Non-Zeros
0112:
0113:       CALL final_mtx_CRS( this )               ! フェイルセイフ
0114:
0115:       CALL associate( adj, this%ind, this%DOF ) ! adj_mtx_c型の変数adjを結合（ダウンキャスト）
0116:       IF (      .NOT. ASSOCIATED(this%ind) &
0117:          & .OR. .NOT. ASSOCIATED(this%DOF) ) THEN
0118:          STOP "init エラー：adj_mtx_c型の変数が初期化されていない. "
0119:       END IF
0120:       nnz = SIZE( this%DOF )
0121:       ALLOCATE  ( this%a(nnz) )
0122:       CALL clear_mtx_CRS ( this )
0123:
0124:       this%state = not_yet_solved
0125:
0126:    END SUBROUTINE init_mtx_CRS
0127:
0128: !*****************************************************************************************
0129:    SUBROUTINE final_mtx_CRS ( this )
0130:
0131:       TYPE(sp_mtx_c), INTENT(inout) :: this
0132:
0133:       INTEGER  :: error
0134:       INTEGER  :: phase
0135:       INTEGER  :: idum(1)
0136:       REAL(DP) :: rdum(1)
0137:
0138:       IF ( this%state /= not_yet_solved ) THEN
0139: #ifdef MKL
0140:          ! PARDISOの引数や値は, solve_mtx_CRS やマニュアルを参照のこと
0141:          phase = -1                              ! PARDISO: 内部メモリを解放
0142:          CALL pardiso ( this%pt, idum(1), idum(1), idum(1),&
0143:                       & phase,   idum(1), rdum,    idum,   &
0144:                       & idum,    idum,    idum(1), idum,   &
0145:                       & idum(1), rdum,    rdum,    error   )
0146:          IF (error /= 0) STOP "PARDISOエラー : 内部メモリの解放に失敗"
0147: #else
0148:          CALL final_PCG_CRS ( this%solver_handle )
0149: #endif
0150:       END IF
0151:
0152:       IF ( ASSOCIATED( this%ind ) ) this%ind => NULL()
0153:       IF ( ASSOCIATED( this%DOF ) ) this%DOF => NULL()
0154:       IF ( ALLOCATED ( this%a   ) ) DEALLOCATE( this%a )
0155:
0156:       this%state = not_yet_solved
0157:
0158:    END SUBROUTINE final_mtx_CRS
0159:
0160: !*****************************************************************************************
0161:    SUBROUTINE clear_mtx_CRS ( this, B_, beta_ )
0162:
0163:       ! ゼロ, B_, β_* B_ のいずれかで クリア
0164:       ! 用途) 動的解析において, K22 + Cγ/(β Δt) + M/(β Δt^2) を計算
0165:       ! CALL clear ( K22, M, beta_dt2       )  ! K22 ←       M/(β Δt^2)
0166:       ! CALL add   ( K22, C, beta_gamma_dt )  ! K22 ← K22 + Cγ/(β Δt)
0167:
0168:       TYPE(sp_mtx_c),           INTENT(inout) :: this
0169:       TYPE(sp_mtx_c), OPTIONAL, INTENT(in   ) :: B_
0170:       REAL(DP),       OPTIONAL, INTENT(in   ) :: beta_
0171:
0172:       INTEGER :: i
0173:
0174:       IF ( PRESENT( B_ ) ) THEN
0175:          IF ( .NOT. ASSOCIATED( this%DOF, B_%DOF ) ) THEN
0176:             STOP "clear エラー：異なる疎構造の行列では初期化できない"
0177:          END IF
```

```
0178:          IF ( PRESENT( beta_ ) ) THEN
0179:             !$OMP PARALLEL DO
0180:             DO i = 1, SIZE(this%a)
0181:                this%a(i) = beta_ * B_%a(i)
0182:             END DO
0183:          ELSE
0184:             !$OMP PARALLEL DO
0185:             DO i = 1, SIZE(this%a)
0186:                this%a(i) =         B_%a(i)
0187:             END DO
0188:          END IF
0189:       ELSE
0190:          !$OMP PARALLEL DO
0191:          DO i = 1, SIZE(this%a)
0192:             this%a(i) = 0.0_DP
0193:          END DO
0194:       END IF
0195:
0196:       IF ( this%state /= not_yet_solved ) this%state = new_mtx_to_solve
0197:
0198:    END SUBROUTINE clear_mtx_CRS
0199:
0200:
0201:    SUBROUTINE clear_diag_mtx_CRS ( this, D )
0202:
0203:       ! 対角項を D(:)で, 非対角項をゼロでクリア
0204:
0205:       TYPE(sp_mtx_c), INTENT(inout) :: this
0206:       REAL(DP),       INTENT(in   ) :: D(:)  ! (nDOF_p) 対角項
0207:
0208:       INTEGER :: nDOF_p
0209:       INTEGER :: i, loc, p
0210:       LOGICAL :: exist
0211:
0212:       nDOF_p = SIZE(this%ind) -1
0213:
0214:       IF ( nDOF_p /= SIZE(D) ) STOP "clear エラー：配列の大きさが一致しない"
0215:
0216:       !$OMP PARALLEL DO
0217:       DO i = 1, SIZE(this%a)
0218:          this%a(i) = 0.0_DP
0219:       END DO
0220:
0221:       !$OMP PARALLEL DO
0222:       DO i = 1, nDOF_p
0223: #ifdef SYM_MTX
0224:          p = this%ind(i)
0225: #else
0226:          loc = binary_search( i, this%DOF(this%ind(i):this%ind(i+1)-1), exist )
0227:          IF ( .NOT. exist ) STOP "clear エラー：疎行列に対角成分がない"
0228:          p = this%ind(i) -1 + loc
0229: #endif
0230:          this%a(p) = D(i)
0231:       END DO
0232:
0233:       IF ( this%state /= not_yet_solved ) this%state = new_mtx_to_solve
0234:
0235:    END SUBROUTINE clear_diag_mtx_CRS
0236:
0237: !*******************************************************************************************
```

```
0238:
0239:    SUBROUTINE add_clique_mtx_CRS ( this, Ke, DOFe2g )
0240:
0241:       ! 要素DOF番号から全体DOF番号への変換テーブル DOFe2g(:) に基づき,
0242:       ! 全体疎行列 this に 要素行列 Ke(:,:) を加算
0243:
0244:       ! 2分探索(binary_search)で逐一探索する. 計算効率は気にせず, 念のための確認も行う.
0245:
0246:       TYPE(sp_mtx_c), INTENT(inout) :: this
0247:       REAL(DP),       INTENT(in   ) :: Ke  (:,:)
0248:       INTEGER,        INTENT(in   ) :: DOFe2g(:) ! SIZE(DOFe) <= SIZE(Ke,dim=*)
0249:
0250:       INTEGER :: nDOFe   ! 要素DOF数
0251:       INTEGER :: ie, je  ! 要素DOF番号
0252:       INTEGER :: i,  j   ! 全体DOF番号
0253:       INTEGER :: loc, p
0254:       LOGICAL :: exist
0255:
0256:       nDOFe = SIZE(DOFe2g)
0257:
0258:       IF ( nDOFe > SIZE(Ke,dim=1) .OR. &
0259:          & nDOFe > SIZE(Ke,dim=2)      ) STOP "add_clique: 配列寸法がおかしい"
0260:
0261:       DO ie = 1, nDOFe
0262:          i = DOFe2g(ie)                          ! 従属自由度があっても問題なし
0263:          IF ( i > 0 ) THEN
0264:             DO je = 1, nDOFe
0265:                j = DOFe2g(je)
0266:                ! --- プリプロセッサは SJIS の 5C文字 に対応していないので要注意 ---
0267: #ifdef SYM_MTX
0268:                IF ( i <= j ) THEN
0269: #else
0270:                IF ( j >  0 ) THEN
0271: #endif
0272:                   ! 全体行列の第(i,j)成分の格納位置p を探索し, 該当成分 ke(ie,je) を加算
0273:                   loc = binary_search( j, this%DOF(this%ind(i):this%ind(i+1)-1), exist )
0274:                   IF ( .NOT. exist ) STOP "add_clique: 疎行列に該当成分の格納場所がない"
0275:                   p = this%ind(i) -1 + loc
0276:                   this%a(p) = this%a(p) + Ke(ie,je)
0277:                END IF
0278:             END DO
0279:          END IF
0280:       END DO
0281:
0282:    END SUBROUTINE add_clique_mtx_CRS
0283:
0284: !*******************************************************************************************
0285:    SUBROUTINE wrt_mtx_CRS (this, fl_name )
0286:
0287:       ! 疎行列this を Coodinate(COO)形式 ( ija形式 ) で出力
0288:
0289:       TYPE(sp_mtx_c), INTENT(in   ) :: this
0290:       CHARACTER(*),   INTENT(in   ) :: fl_name
0291:
0292:       INTEGER :: uid = 30
0293:       INTEGER :: i, j, p, nDOF_p
0294:
0295:       OPEN (uid, FILE=fl_name )
0296:       nDOF_p = SIZE(this%ind) -1
0297:       DO i = 1, nDOF_p
0298:          DO p = this%ind(i), this%ind(i+1)-1
0299:             j = this%DOF(p)
0300:             WRITE(uid,'(2I10,ES30.15)') i, j, this%a(p)
0301:          END DO
0302:       END DO
0303:       CLOSE(uid)
0304:
0305:       WRITE(*,'("Coodinate(COO)形式の疎行列をMATLABで可視化するには, 次のコマンド:")')
0306:       WRITE(*,'("ija  = load(''",A,"'');")') fl_name
0307:       WRITE(*,'("K_pp = sparse(ija(:,1), ija(:,2), ija(:,3) );")')
0308:
0309:    END SUBROUTINE wrt_mtx_CRS
0310:
0311: !*******************************************************************************************
0312:    SUBROUTINE add_mtx_CRS ( this, B, beta_ )
0313:
```

```
0314:       ! this ← this + B  もしくは  this ← this + β * B
0315:
0316:       TYPE(sp_mtx_c),      INTENT(inout) :: this
0317:       TYPE(sp_mtx_c),      INTENT(inout) :: B
0318:       REAL(DP), OPTIONAL,  INTENT(in   ) :: beta_   ! スカラ値β
0319:
0320:       INTEGER :: i
0321:
0322:       IF ( .NOT. ASSOCIATED( this%DOF, B%DOF ) ) THEN
0323:          STOP "add エラー：異なる疎構造の行列同士を足そうとしている"
0324:       END IF
0325:
0326:       IF ( PRESENT( beta_ ) ) THEN
0327:          !$OMP PARALLEL DO
0328:          DO i = 1, SIZE(this%a)
0329:             this%a(i) = this%a(i) + beta_ * B%a(i)
0330:          END DO
0331:       ELSE
0332:          !$OMP PARALLEL DO
0333:          DO i = 1, SIZE(this%a)
0334:             this%a(i) = this%a(i) +         B%a(i)
0335:          END DO
0336:       END IF
0337:
0338:    END SUBROUTINE add_mtx_CRS
0339:
0340: !*******************************************************************************************
0341:    SUBROUTINE mul_mtx_CRS (this, x, y )
0342:
0343:       ! 行列・数ベクトル積  y ← Ax
0344:
0345:       TYPE(sp_mtx_c), INTENT(inout) :: this
0346:       REAL(DP),       INTENT(in   ) :: x(:)
0347:       REAL(DP),       INTENT(out  ) :: y(:)
0348:
0349:       INTEGER :: nDOF_p
0350:
0351:       nDOF_p = SIZE(this%ind(:)) -1
0352:       IF ( SIZE(x) /= nDOF_p ) STOP "mul エラー：連立1次方程式の次元が一致しない"
0353:
0354:       CALL matvec_F77 ( this%a, this%DOF, this%ind, x, y, nDOF_p )
0355:       ! 構造体を使うと最適化しない可能性があるので F77へのバックドア
0356:
0357:    END SUBROUTINE mul_mtx_CRS
0358:
0359: !*******************************************************************************************
```

```
0360:
0361:    SUBROUTINE solve_mtx_CRS ( this, b, x )
0362:
0363:       ! Ax = b を x について解く
0364:
0365:       TYPE(sp_mtx_c), INTENT(inout) :: this
0366:       REAL(DP),       INTENT(inout) :: b(:)  ! (inout指定はPardiso由来)
0367:       REAL(DP),       INTENT(inout) :: x(:)  ! (          〃          )
0368:
0369:       INTEGER  :: n, nnz
0370:
0371:       ! Pardisoに必要なパラメータ
0372:       INTEGER, PARAMETER :: maxfct =  1      ! ユーザーがメモリー中に同時に保持する
0373:                                              !   同じ非ゼロのスパース構造を持つ最大の係数
0374:       INTEGER, PARAMETER :: mnum   =  maxfct ! 解フェーズの実際の行列
0375:
0376: #ifdef SYM_MTX
0377:       INTEGER, PARAMETER :: mtype  =  2      ! 正定値行列の場合
0378: #else
0379:       INTEGER, PARAMETER :: mtype  = 11      ! 非対称行列の場合
0380: #endif
0381:          !   1  real and structurally symmetric
0382:          !   2  real and symmetric positive definite
0383:          !  -2  real and symmetric indefinite
0384:          !   3  complex and structurally symmetric
0385:          !   4  complex and Hermitian positive definite
0386:          !  -4  complex and Hermitian indefinite
0387:          !   6  complex and symmetric
0388:          !  11  real and nonsymmetric
0389:          !  13  complex and nonsymmetric
0390:
0391:       INTEGER  :: phase                      ! ソルバーの実行を制御
0392:          !  11  Analysis
0393:          !  12  Analysis, numerical factorization
0394:          !  13  Analysis, numerical factorization, solve, iterative refinement
0395:          !  22  Numerical factorization
0396:          !  23  Numerical factorization, solve, iterative refinement
0397:          !  33  Solve, iterative refinement
0398:          !  331 like phase=33, but only forward substitution
0399:          !  332 like phase=33, but only diagonal substitution (if available)
0400:          !  333 like phase=33, but only backward substitution
0401:          !  0   Release internal memory for L and U matrix number mnum
0402:          ! -1   Release all internal memory for all matrices
0403:
0404:       INTEGER, PARAMETER :: nrhs   =  1      ! 解を求める右辺の数
0405:       INTEGER  :: iparm(64) = 0              ! pardisoのパラメタ格納用配列
0406:                                              ! iparm(1)=0で iparm(2:64)はデフォルト値
0407:       INTEGER  :: idummy(1)                  ! ダミー配列 (Permutation用途)
0408:       INTEGER  :: msglvl                     ! 1: 求解に関する情報を出力, 0: 出力なし
0409:       INTEGER  :: error
0410:       INTEGER  :: iter
0411:
0412:       n   = SIZE(this%ind) -1
0413:       nnz = SIZE(this%DOF)
0414:       IF ( SIZE(b)/=n .OR. SIZE(x)/=n ) STOP "solve エラー：疎行列と数ベクトルの次元が一致せず"
0415:
0416:       msglvl =  0                            ! 「init 後初回」状態以外では画面出力なし
0417:       IF ( this%state == not_yet_solved ) msglvl = 1
0418:
0419: #ifdef MKL
0420:       SELECT CASE ( this%state )
0421:          CASE ( not_yet_solved   )
0422:             this%pt(:)%dummy = 0             ! pardiso: 初回だけ必須
0423:             phase  = 13                      ! Pardiso: シンボリック解析(1)から求解(3)まで
0424:          CASE ( new_mtx_to_solve )
0425:             phase  = 23                      ! Pardiso: 数値分解(2)から求解(3)まで
0426:          CASE ( solved_the_mtx   )
0427:             phase  = 33                      ! Pardiso: 求解(3)のみ
0428:          CASE default
0429:             STOP "solve: この状態に遷移することはない"
0430:       END SELECT
0431:       CALL pardiso( this%pt  ,maxfct  ,mnum     ,mtype    ,&
0432:                   & phase    ,n       ,this%a   ,this%ind ,&
0433:                   & this%DOF ,idummy  ,nrhs     ,iparm    ,&
0434:                   & msglvl   ,b       ,x        ,error     )
0435:       IF (error /= 0) CALL pardiso_error_message ( error )
```

```
0436: #else
0437:       IF ( this%state == not_yet_solved   ) THEN
0438:          CALL  init_PCG_CRS ( this%solver_handle ,this%a ,this%DOF ,this%ind )
0439:          this%state = new_mtx_to_solve
0440:       END IF
0441:       IF ( this%state == new_mtx_to_solve ) THEN
0442:          CALL set_M_PCG_CRS ( this%solver_handle ,this%a ,this%DOF ,this%ind )
0443:       END IF
0444:       CALL    solve_PCG_CRS ( this%solver_handle ,this%a ,this%DOF ,this%ind ,  &
0445:                               & msglvl,    b,     x  )
0446: #endif
0447:       this%state = solved_the_mtx
0448:
0449:    CONTAINS
0450:
0451:       SUBROUTINE pardiso_error_message ( error )
0452:          INTEGER, INTENT(in   ) :: error
0453:          SELECT CASE ( error )
0454:           ! CASE (   0 ); WRITE(*,*) "Pardiso 正常終了"
0455:             CASE (  -1 )
0456:                STOP "input inconsistent"
0457:             CASE (  -2 )
0458:                STOP "not enough memory"
0459:             CASE (  -3 )
0460:                STOP "reordering problem"
0461:             CASE (  -4 )
0462:                STOP "zero pivot, numerical factorization or iterative refinement problem"
0463:             CASE (  -5 )
0464:                STOP "unclassified (internal) error"
0465:             CASE (  -6 )
0466:                STOP "reordering failed (nonsymmetric matrix only)"
0467:             CASE (  -7 )
0468:                STOP "diagonal matrix is singular"
0469:             CASE (  -8 )
0470:                STOP "32-bit integer overflow problem"
0471:             CASE (  -9 )
0472:                STOP "not enough memory for Out-Of-Core"
0473:             CASE ( -10 )
0474:                STOP "error opening Out-of-Core files"
0475:             CASE ( -11 )
0476:                STOP "read/write error with OOC files"
0477:             CASE ( -12 )
0478:                STOP "(pardiso_64 only) pardiso_64 called from 32-bit library"
0479:          END SELECT
0480:       END SUBROUTINE pardiso_error_message
0481:
0482:    END SUBROUTINE solve_mtx_CRS
0483:
0484: END MODULE sp_mtx_class
```