```
!======================================================================!
!                                                                      !
!   Software Name : HEC-MW Library for PC-cluster                      !
!         Version : 2.5                                                !
!                                                                      !
!     Last Update : 2006/06/01                                         !
!        Category : Linear Solver                                      !
!                                                                      !
!            Written by Kengo Nakajima (Univ. of Tokyo)                !
!                                                                      !
!     Contact address :  IIS,The University of Tokyo RSS21 project     !
!                                                                      !
!     ″Structural Analysis System for General-purpose Coupling         !
!      Simulations Using High End Computing Middleware (HEC-MW)″       !
!                                                                      !
!======================================================================!

      module hecmw_solver_33
      contains
!
!C***
!C*** hecmw_solve_33
!C***
!
        subroutine hecmw_solve_33 (hecMESH, hecMAT)

        use hecmw_util
        use hecmw_solver_CG_33
        use hecmw_solver_BiCGSTAB_33
        use hecmw_solver_GMRES_33
        use hecmw_solver_GPBiCG_33
        use hecmw_solver_BLCG_33
        use hecmw_solver_BLBiCGSTAB_33
        use hecmw_solver_BLGPBiCG_33
        use m_hecmw_solve_error
        use m_hecmw_comm_f
        use hecmw_matrix_ass
        use hecmw_matrix_contact

        implicit none

        type (hecmwST_matrix)     :: hecMAT
        type (hecmwST_local_mesh) :: hecMESH

        integer(kind=kint) :: ERROR
        real(kind=kreal), dimension(3,3) :: ALU
        real(kind=kreal), dimension(3)   :: PW

        integer(kind=kint) :: ITER, METHOD, PRECOND, NSET
        integer(kind=kint) :: iterPREmax, ii, i, j, k

        real(kind=kreal) :: RESID, SIGMA_DIAG, THRESH, FILTER

        integer(kind=kint) :: ITERlog, TIMElog
        real(kind=kreal) :: TIME_setup, TIME_comm, TIME_soltot, TR
        real(kind=kreal),    dimension(1) :: RHS
        integer (kind=kint), dimension(1) :: IFLAG

        integer(kind=kint) :: NREST,NPLU,iS0,iS,inum,k0
        real(kind=kreal)   :: SIGMA,S1_time,E1_time,TIME_sol

!C===
!C +------------+
!C | PARAMETERs |
!C +------------+
!C===
        ITER      = hecMAT%Iarray(1)
        METHOD    = hecMAT%Iarray(2)
        PRECOND   = hecMAT%Iarray(3)
        NSET      = hecMAT%Iarray(4)
        iterPREmax= hecMAT%Iarray(5)
        NREST     = hecMAT%Iarray(6)

        ITERlog= hecMAT%Iarray(21)
        TIMElog= hecMAT%Iarray(22)

        TIME_setup  = 0.d0
        TIME_comm   = 0.d0
        TIME_soltot = 0.d0

        RESID     = hecMAT%Rarray(1)
        SIGMA_DIAG= hecMAT%Rarray(2)
        SIGMA     = hecMAT%Rarray(3)

        THRESH= hecMAT%Rarray(4)
        FILTER= hecMAT%Rarray(5)

        if (iterPREmax.lt.1) iterPREmax= 1
        if (iterPREmax.gt.4) iterPREmax= 4

!C===
!C +-------------+
!C | ERROR CHECK |
!C +-------------+
!C===
        ERROR= 0

!C
!C-- ZERO RHS norm
        RHS(1)= 0.d0
        do i= 1, hecMAT%N
          RHS(1)= RHS(1) + hecMAT%B(3*i-2)**2 + hecMAT%B(3*i-1)**2       &
     &                   + hecMAT%B(3*i  )**2
        enddo
        if (hecMESH%mpc%n_mpc > 0) then
          do i= 1, hecMESH%mpc%n_mpc
            RHS(1)= RHS(1) + hecMESH%mpc%mpc_const(i)**2
          enddo
        endif
        call hecmw_allreduce_R (hecMESH, RHS, 1, hecmw_sum)

        if (RHS(1).eq.0.d0) then
          ERROR= 2002
          call hecmw_solve_error (hecMESH, ERROR)
        endif

!C
!C-- ZERO DIAGONAL component
        IFLAG(1)= 0
        do i= 1, hecMAT%N
          if (dabs(hecMAT%D(9*i-8)).eq.0.d0) IFLAG(1)= 1
          if (dabs(hecMAT%D(9*i-4)).eq.0.d0) IFLAG(1)= 1
          if (dabs(hecMAT%D(9*i  )).eq.0.d0) IFLAG(1)= 1
        enddo

        call hecmw_allreduce_I (hecMESH, IFLAG, 1, hecmw_sum)
        if (IFLAG(1).ne.0) then
          ERROR= 2001
          call hecmw_solve_error (hecMESH, ERROR)
        endif

!C
!C-- INCONSISTENT SOLVER/PRECONDITIONING
        IFLAG(1)= 0
        if (METHOD.le.0 .or. METHOD.ge.5)    IFLAG(1)= 1
        if (PRECOND.le.0 .or. PRECOND.gt.21) IFLAG(1)= 1

        call hecmw_allreduce_I (hecMESH, IFLAG, 1, hecmw_sum)
        if (IFLAG(1).ne.0) then
          ERROR= 1001
          call hecmw_solve_error (hecMESH, ERROR)
        endif

        IFLAG(1)= 1
```

```
      if (METHOD.eq.1) then
        if (PRECOND.eq. 1) IFLAG(1)= 0
        if (PRECOND.eq. 2) IFLAG(1)= 0
        if (PRECOND.eq. 3) IFLAG(1)= 0
        if (PRECOND.eq.10) IFLAG(1)= 0
        if (PRECOND.eq.11) IFLAG(1)= 0
        if (PRECOND.eq.12) IFLAG(1)= 0
      endif

      if (METHOD.eq.2) then
        if (PRECOND.eq. 1) IFLAG(1)= 0
        if (PRECOND.eq. 2) IFLAG(1)= 0
        if (PRECOND.eq. 3) IFLAG(1)= 0
        if (PRECOND.eq.10) IFLAG(1)= 0
        if (PRECOND.eq.11) IFLAG(1)= 0
        if (PRECOND.eq.12) IFLAG(1)= 0
      endif

      if (METHOD.eq.3) then
        if (PRECOND.eq. 1) IFLAG(1)= 0
        if (PRECOND.eq. 2) IFLAG(1)= 0
        if (PRECOND.eq. 3) IFLAG(1)= 0
      endif

      if (METHOD.eq.4) then
        if (PRECOND.eq. 1) IFLAG(1)= 0
        if (PRECOND.eq. 2) IFLAG(1)= 0
        if (PRECOND.eq. 3) IFLAG(1)= 0
        if (PRECOND.eq.10) IFLAG(1)= 0
        if (PRECOND.eq.11) IFLAG(1)= 0
        if (PRECOND.eq.12) IFLAG(1)= 0
      endif

      if (IFLAG(1).ne.0) then
        ERROR= 1001
        call hecmw_solve_error (hecMESH, ERROR)
        endif

!C===
!C +-----------+
!C | BLOCK LUs |
!C +-----------+
!C===
        if (.not.associated(hecMAT%ALU).and.PRECOND.lt.10) NSET=  0
        if (     associated(hecMAT%ALU).and.PRECOND.lt.10) NSET= -1

        S1_time= HECMW_WTIME()
        if (NSET.eq.0 .and. PRECOND.lt.10) then
          allocate (hecMAT%ALU(9*hecMAT%N))
        endif
        if (NSET.le.0 .and. PRECOND.lt.10) then
          hecMAT%ALU  = 0.d0

          do ii= 1, hecMAT%N
            hecMAT%ALU(9*ii-8) = hecMAT%D(9*ii-8)
            hecMAT%ALU(9*ii-7) = hecMAT%D(9*ii-7)
            hecMAT%ALU(9*ii-6) = hecMAT%D(9*ii-6)
            hecMAT%ALU(9*ii-5) = hecMAT%D(9*ii-5)
            hecMAT%ALU(9*ii-4) = hecMAT%D(9*ii-4)
            hecMAT%ALU(9*ii-3) = hecMAT%D(9*ii-3)
            hecMAT%ALU(9*ii-2) = hecMAT%D(9*ii-2)
            hecMAT%ALU(9*ii-1) = hecMAT%D(9*ii-1)
            hecMAT%ALU(9*ii  ) = hecMAT%D(9*ii  )
          enddo

          if (hecMAT%cmat%n_val.gt.0) then
            do k= 1, hecMAT%cmat%n_val
              if (hecMAT%cmat%pair(k)%i.ne.hecMAT%cmat%pair(k)%j) cycle
              ii = hecMAT%cmat%pair(k)%i
              hecMAT%ALU(9*ii-8) = hecMAT%ALU(9*ii-8) + hecMAT%cmat%pair(k)%val(1, 1)
              hecMAT%ALU(9*ii-7) = hecMAT%ALU(9*ii-7) + hecMAT%cmat%pair(k)%val(1, 2)
              hecMAT%ALU(9*ii-6) = hecMAT%ALU(9*ii-6) + hecMAT%cmat%pair(k)%val(1,
3)
              hecMAT%ALU(9*ii-5) = hecMAT%ALU(9*ii-5) + hecMAT%cmat%pair(k)%val(2,
1)
              hecMAT%ALU(9*ii-4) = hecMAT%ALU(9*ii-4) + hecMAT%cmat%pair(k)%val(2,
2)
              hecMAT%ALU(9*ii-3) = hecMAT%ALU(9*ii-3) + hecMAT%cmat%pair(k)%val(2,
3)
              hecMAT%ALU(9*ii-2) = hecMAT%ALU(9*ii-2) + hecMAT%cmat%pair(k)%val(3,
1)
              hecMAT%ALU(9*ii-1) = hecMAT%ALU(9*ii-1) + hecMAT%cmat%pair(k)%val(3,
2)
              hecMAT%ALU(9*ii  ) = hecMAT%ALU(9*ii  ) + hecMAT%cmat%pair(k)%val(3,
3)
            enddo

            call hecmw_cmat_LU( hecMAT )

          endif

          do ii= 1, hecMAT%N
            ALU(1,1)= hecMAT%ALU(9*ii-8)
            ALU(1,2)= hecMAT%ALU(9*ii-7)
            ALU(1,3)= hecMAT%ALU(9*ii-6)
            ALU(2,1)= hecMAT%ALU(9*ii-5)
            ALU(2,2)= hecMAT%ALU(9*ii-4)
            ALU(2,3)= hecMAT%ALU(9*ii-3)
            ALU(3,1)= hecMAT%ALU(9*ii-2)
            ALU(3,2)= hecMAT%ALU(9*ii-1)
            ALU(3,3)= hecMAT%ALU(9*ii  )
            do k= 1, 3
              ALU(k,k)= 1.d0/ALU(k,k)
              do i= k+1, 3
                ALU(i,k)= ALU(i,k) * ALU(k,k)
                do j= k+1, 3
                  PW(j)= ALU(i,j) - ALU(i,k)*ALU(k,j)
                enddo
```

```
        do j= k+1, 3
         ALU(i,j)= PW(j)
        enddo
       enddo
      enddo
      hecMAT%ALU(9*ii-8)= ALU(1,1)
      hecMAT%ALU(9*ii-7)= ALU(1,2)
      hecMAT%ALU(9*ii-6)= ALU(1,3)
      hecMAT%ALU(9*ii-5)= ALU(2,1)
      hecMAT%ALU(9*ii-4)= ALU(2,2)
      hecMAT%ALU(9*ii-3)= ALU(2,3)
      hecMAT%ALU(9*ii-2)= ALU(3,1)
      hecMAT%ALU(9*ii-1)= ALU(3,2)
      hecMAT%ALU(9*ii  )= ALU(3,3)
     enddo
    endif

    E1_time= HECMW_WTIME()
    TIME_setup= TIME_setup + E1_time - S1_time

!C===
!C +------------------+
!C | ITERATIVE solver |
!C +------------------+
!C===

!C
!C-- CG
   if (METHOD.eq.1 .and. PRECOND.lt.10) then
    if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
     if (PRECOND.eq.1) write (*,'(a,i3)') '### 3x3 B-IC-CG  (0)',  &
  &                                       iterPREmax
     if (PRECOND.eq.2) write (*,'(a,i3)') '### 3x3 B-SSOR-CG(0)',  &
  &                                       iterPREmax
     if (PRECOND.eq.3) write (*,'(a,i3)') '### 3x3 B-scale-CG  ',  &
  &                                       iterPREmax
```

```
        endif

        call hecmw_solve_CG_33( hecMESH,  hecMAT, ITER, RESID, ERROR,   &
     &                          TIME_setup, TIME_sol, TIME_comm )
      endif

!C
!C-- BiCGSTAB
      if (METHOD.eq.2 .and. PRECOND.lt.10) then
        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.1) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-ILU-BiCGSTAB (0)', iterPREmax
          if (PRECOND.eq.2) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-SSOR-BiCGSTAB(0)', iterPREmax
          if (PRECOND.eq.3) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-scale-BiCGSTAB  ', iterPREmax
        endif

        call hecmw_solve_BiCGSTAB_33( hecMESH,  hecMAT, ITER, RESID, ERROR, &
     &                                TIME_setup, TIME_sol, TIME_comm )
      endif

!C
!C-- GMRES
      if (METHOD.eq.3 .and. PRECOND.lt.10) then
        ! imposing MPC by penalty
        call hecmw_mat_ass_equation ( hecMESH, hecMAT )

        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.1) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-ILU-GMRES (0)', iterPREmax
          if (PRECOND.eq.2) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-SSOR-GMRES(0)', iterPREmax
          if (PRECOND.eq.3) write (*,'(a,i3)')                          &
     &                  '### 3x3 B-scale-GMRES  ', iterPREmax
        endif

        call hecmw_solve_GMRES_33                                       &
     &     ( hecMAT%N, hecMAT%NP, hecMAT%NPL, hecMAT%NPU,               &
     &       hecMAT%D, hecMAT%AL, hecMAT%indexL, hecMAT%itemL,          &
     &                 hecMAT%AU, hecMAT%indexU, hecMAT%itemU,          &
     &       hecMAT%B, hecMAT%X,  hecMAT%ALU, RESID, ITER,              &
     &       ERROR,    hecMESH%my_rank,                                 &
     &       hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,                &
     &       hecMESH%import_index, hecMESH%import_item,                 &
     &       hecMESH%export_index, hecMESH%export_item,                 &
     &       hecMESH%MPI_COMM, PRECOND, iterPREmax, NREST,              &
     &       TIME_setup, TIME_sol, TIME_comm, ITERlog)
      endif

!C
!C-- GPBiCG
      if (METHOD.eq.4 .and. PRECOND.lt.10) then
        ! imposing MPC by penalty
        call hecmw_mat_ass_equation ( hecMESH, hecMAT )

        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.1) write (*,'(a,i3)')                          &
     &                   '### 3x3 B-ILU-GPBiCG   (0)', iterPREmax
          if (PRECOND.eq.2) write (*,'(a,i3)')                          &
     &                   '### 3x3 B-SSOR-GPBiCG  (0)', iterPREmax
          if (PRECOND.eq.3) write (*,'(a,i3)')                          &
     &                   '### 3x3 B-scale-GPBiCG    ', iterPREmax
        endif

        call hecmw_solve_GPBiCG_33                                      &
     &     ( hecMAT%N, hecMAT%NP, hecMAT%NPL, hecMAT%NPU,               &
     &       hecMAT%D, hecMAT%AL, hecMAT%indexL, hecMAT%itemL,          &
     &                 hecMAT%AU, hecMAT%indexU, hecMAT%itemU,          &
     &       hecMAT%B, hecMAT%X,  hecMAT%ALU, RESID, ITER,              &
     &       ERROR,    hecMESH%my_rank,                                 &
     &       hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,                &
     &       hecMESH%import_index, hecMESH%import_item,                 &
     &       hecMESH%export_index, hecMESH%export_item,                 &
     &       hecMESH%MPI_COMM, PRECOND, iterPREmax,                     &
     &       TIME_setup, TIME_sol, TIME_comm, ITERlog)
      endif

!C
!C-- CG-1/2
      if (METHOD.eq.1 .and. (PRECOND.ge.10.and.PRECOND.le.20)) then
        ! imposing MPC by penalty
        call hecmw_mat_ass_equation ( hecMESH, hecMAT )

        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.10) write (*,'(a)') '### 3x3 B-IC-CG  (0)'
          if (PRECOND.eq.11) write (*,'(a)') '### 3x3 B-IC-CG  (1)'
          if (PRECOND.eq.12) write (*,'(a)') '### 3x3 B-IC-CG  (2)'
        endif

        SIGMA     = 1.d0
        SIGMA_DIAG= 1.d0
        call hecmw_solve_BLCG_33                                        &
     &     ( hecMAT%N, hecMAT%NP, hecMAT%NPL, hecMAT%NPU,               &
     &       hecMAT%D, hecMAT%AL, hecMAT%indexL, hecMAT%itemL,          &
     &                 hecMAT%AU, hecMAT%indexU, hecMAT%itemU,          &
     &       hecMAT%B, hecMAT%X,  RESID, SIGMA, SIGMA_DIAG,             &
     &       ITER, ERROR,  hecMESH%my_rank,                             &
     &       hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,                &
     &       hecMESH%import_index, hecMESH%import_item,                 &
     &       hecMESH%export_index, hecMESH%export_item,                 &
     &       hecMESH%MPI_COMM, PRECOND, iterPREmax,                     &
     &       TIME_setup, TIME_sol, TIME_comm, ITERlog)
      endif

!C
!C-- BiCGSTAB-1/2
      if (METHOD.eq.2 .and. (PRECOND.ge.10.and.PRECOND.le.20)) then
```

```
        ! imposing MPC by penalty
        call hecmw_mat_ass_equation ( hecMESH, hecMAT )

        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.10) write (*,'(a)')                            &
     &                       '### 3x3 B-IIU-BiCGSTAB (0)'
          if (PRECOND.eq.11) write (*,'(a)')                            &
     &                       '### 3x3 B-IIU-BiCGSTAB (1)'
          if (PRECOND.eq.12) write (*,'(a)')                            &
     &                       '### 3x3 B-IIU-BiCGSTAB (2)'
        endif

        SIGMA     = 1.d0
        SIGMA_DIAG= 1.d0
        call hecmw_solve_BLBiCGSTAB_33                                  &
     &     ( hecMAT%N, hecMAT%NP, hecMAT%NPL, hecMAT%NPU,               &
     &       hecMAT%D, hecMAT%AL, hecMAT%indexL, hecMAT%itemL,          &
     &                 hecMAT%AU, hecMAT%indexU, hecMAT%itemU,          &
     &       hecMAT%B, hecMAT%X,  RESID, SIGMA, SIGMA_DIAG,             &
     &       ITER, ERROR,  hecMESH%my_rank,                             &
     &       hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,                &
     &       hecMESH%import_index, hecMESH%import_item,                 &
     &       hecMESH%export_index, hecMESH%export_item,                 &
     &       hecMESH%MPI_COMM, PRECOND, iterPREmax,                     &
     &       TIME_setup, TIME_sol, TIME_comm, ITERlog)
      endif

!C
!C-- GPBiCG-1/2
      if (METHOD.eq.4 .and. (PRECOND.ge.10.and.PRECOND.le.20)) then
        ! imposing MPC by penalty
        call hecmw_mat_ass_equation ( hecMESH, hecMAT )

        if (hecMESH%my_rank.eq.0 .and. (ITERlog.eq.1 .or. TIMElog.eq.1)) then
          if (PRECOND.eq.10) write (*,'(a)') '### 3x3 B-ILU-GPBiCG (0)'
          if (PRECOND.eq.11) write (*,'(a)') '### 3x3 B-ILU-GPBiCG (1)'
          if (PRECOND.eq.12) write (*,'(a)') '### 3x3 B-ILU-GPBiCG (2)'
        endif

        SIGMA     = 1.d0
        SIGMA_DIAG= 1.d0
        call hecmw_solve_BLGPBiCG_33                                     &
     &     ( hecMAT%N, hecMAT%NP, hecMAT%NPL, hecMAT%NPU,                 &
     &       hecMAT%D, hecMAT%AL, hecMAT%indexL, hecMAT%itemL,            &
     &                 hecMAT%AU, hecMAT%indexU, hecMAT%itemU,            &
     &       hecMAT%B, hecMAT%X,  RESID, SIGMA, SIGMA_DIAG,               &
     &       ITER, ERROR,  hecMESH%my_rank,                               &
     &       hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,                  &
     &       hecMESH%import_index, hecMESH%import_item,                   &
     &       hecMESH%export_index, hecMESH%export_item,                   &
     &       hecMESH%MPI_COMM, PRECOND, iterPREmax,                       &
     &       TIME_setup, TIME_sol, TIME_comm, ITERlog)
      endif

      if (RESID.gt.hecMAT%Rarray(1)) then
        call hecmw_solve_error (hecMESH, 3001)
      endif

      if (hecMESH%my_rank.eq.0 .and. TIMElog.eq.1) then
        TR= (TIME_sol-TIME_comm)/(TIME_sol+1.d-24)*100.d0
        write (*,'(/a)')          '### summary of linear solver'
        write (*,'(i10,a, 1pe16.6)')      ITER, ' iterations  ', RESID
        write (*,'(a, 1pe16.6 )') '    set-up time      : ', TIME_setup
        write (*,'(a, 1pe16.6 )') '    solver time      : ', TIME_sol
        write (*,'(a, 1pe16.6 )') '    solver/comm time : ', TIME_comm
        write (*,'(a, 1pe16.6/)') '    work ratio (%)   : ', TR
      endif

      if (hecMAT%cmat%n_val.gt.0) then
        call hecmw_cmat_LU_free( hecMAT )
      endif

```

```
end subroutine hecmw_solve_33
end module hecmw_solver_33
```