```
!======================================================================!
!                                                                      !
!   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)"       !
!                                                                      !
!======================================================================!

!C
!C***
!C*** module hecmw_solver_CG_33
!C***
!C
      module hecmw_solver_CG_33
      contains
!C
!C*** CG_33
!C
      subroutine hecmw_solve_CG_33( hecMESH,  hecMAT, ITER, RESID, ERROR, &
     &                              Tset, Tsol, Tcomm )

      use hecmw_util
      use m_hecmw_solve_error
      use m_hecmw_comm_f
      use hecmw_matrix_misc
      use hecmw_solver_misc
      use hecmw_solver_misc_33

      implicit none

      type(hecmwST_local_mesh) :: hecMESH
      type(hecmwST_matrix) :: hecMAT
      integer(kind=kint ), intent(inout):: ITER, ERROR
      real   (kind=kreal), intent(inout):: RESID, Tset, Tsol, Tcomm

      integer(kind=kint ) :: my_rank
      integer(kind=kint ) :: ITERlog, TIMElog
      real(kind=kreal), pointer :: B(:), X(:)

      real(kind=kreal), dimension(:,:), allocatable :: WW

      integer(kind=kint), parameter ::  R= 1
      integer(kind=kint), parameter ::  Z= 2
      integer(kind=kint), parameter ::  Q= 2
      integer(kind=kint), parameter ::  P= 3
      integer(kind=kint), parameter :: BT= 1
      integer(kind=kint), parameter ::TATX=2
      integer(kind=kint), parameter :: WK= 4

      integer(kind=kint ) :: MAXIT
      integer(kind=kint ) :: totalmpc

! local variables
      real   (kind=kreal) :: TOL
      integer(kind=kint )::i
      real   (kind=kreal)::S_TIME,S1_TIME,E_TIME,E1_TIME, START_TIME, END_TIME
      real   (kind=kreal)::BNRM2
      real   (kind=kreal)::RHO,RHO1,BETA,C1,ALPHA,DNRM2

      S_TIME= HECMW_WTIME()

!C===
!C +-------+
!C | INIT. |
!C +-------+
!C===
      my_rank = hecMESH%my_rank
      X => hecMAT%X
      B => hecMAT%B

      ITERlog = hecmw_mat_get_iterlog( hecMAT )
      TIMElog = hecmw_mat_get_timelog( hecMAT )
      MAXIT  = hecmw_mat_get_iter( hecMAT )
      TOL   = hecmw_mat_get_resid( hecMAT )

      totalmpc = hecMESH%mpc%n_mpc
      call hecmw_allreduce_I1 (hecMESH, totalmpc, hecmw_sum)

      ERROR = 0

      allocate (WW(3 * hecMAT%NP, 4))
      WW = 0.d0

      call hecmw_mpc_scale(hecMESH)

!C===
!C +----------------------------------------------+
!C | {r0}= [T'] ({b} - [A]{xg}) - [T'][A][T]{xini} |
!C +----------------------------------------------+
!C===

!C-- {bt}= [T'] ({b} - [A]{xg})
      if (totalmpc.eq.0) then
        do i = 1, hecMAT%N * 3
          WW(i,BT) = B(i)
        enddo
       else
        if (TIMElog.eq.1) then
        call hecmw_trans_b_33(hecMESH, hecMAT, B, WW(:,BT), WW(:,WK), Tcomm)
        else
        call hecmw_trans_b_33(hecMESH, hecMAT, B, WW(:,BT), WW(:,WK))
        endif
      endif

!C-- compute ||{bt}||
      if (TIMElog.eq.1) then
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,BT), WW(:,BT), BNRM2, Tcomm)
      else
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,BT), WW(:,BT), BNRM2)
      endif
      if (BNRM2.eq.0.d0) then
        iter = 0
        MAXIT = 0
        RESID = 0.d0
        X = 0.d0
      endif

!C-- {tatx} = [T’] [A] [T]{x}
      if (totalmpc.eq.0) then
        if (TIMElog.eq.1) then
        call hecmw_matvec_33(hecMESH, hecMAT, X, WW(:,TATX), Tcomm)
        else
        call hecmw_matvec_33(hecMESH, hecMAT, X, WW(:,TATX))
        endif
       else
        if (TIMElog.eq.1) then
        call hecmw_TtmatTvec_33(hecMESH, hecMAT, X, WW(:,TATX), WW(:,WK), Tcomm)
        else
        call hecmw_TtmatTvec_33(hecMESH, hecMAT, X, WW(:,TATX), WW(:,WK))
        endif
      endif

!C-- {r} = {bt} - {tatx}
      do i = 1, hecMAT%N * 3
        WW(i,R) = WW(i,BT) - WW(i,TATX)
      enddo

      E_TIME = HECMW_WTIME()
      Tset = Tset + E_TIME - S_TIME

      Tcomm = 0.d0
      S1_TIME = HECMW_WTIME()
!C
!C************************************************ Conjugate Gradient Iteration start
!C
      do iter = 1, MAXIT

!C===
!C +---------------+
!C | {z}= [Minv]{r} |
!C +---------------+
!C===
      if (TIMElog.eq.1) then
      call hecmw_precond_33(hecMESH, hecMAT, WW(:,R), WW(:,Z), WW(:,WK), Tcomm)
      else
      call hecmw_precond_33(hecMESH, hecMAT, WW(:,R), WW(:,Z), WW(:,WK))
      endif

!C===
!C +--------------+
!C | {RHO}= {r}{z} |
!C +--------------+
!C===
      if (TIMElog.eq.1) then
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,R), WW(:,Z), RHO, Tcomm)
      else
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,R), WW(:,Z), RHO)
      endif

!C===
!C +----------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-----------------------------+
!C===
      if ( ITER.eq.1 ) then
        do i = 1, hecMAT%N * 3
          WW(i,P) = WW(i,Z)
        enddo
       else
         BETA = RHO / RHO1
         do i = 1, hecMAT%N * 3
           WW(i,P) = WW(i,Z) + BETA*WW(i,P)
         enddo
      endif

!C===
!C +---------------------+
!C | {q}= [T'][A][T] {p} |
!C +---------------------+
!C===
      if (totalmpc.eq.0) then
        if (TIMElog.eq.1) then
        call hecmw_matvec_33(hecMESH, hecMAT, WW(:,P), WW(:,Q), Tcomm)
        else
        call hecmw_matvec_33(hecMESH, hecMAT, WW(:,P), WW(:,Q))
        endif
       else
        if (TIMElog.eq.1) then
        call hecmw_TtmatTvec_33(hecMESH, hecMAT, WW(:,P), WW(:,Q), WW(:,WK), Tcomm)
        else
        call hecmw_TtmatTvec_33(hecMESH, hecMAT, WW(:,P), WW(:,Q), WW(:,WK))
        endif
      endif

!C===
!C +---------------------+
!C | ALPHA= RHO / {p}{q} |
!C +---------------------+
!C===
      if (TIMElog.eq.1) then
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,P), WW(:,Q), C1, Tcomm)
      else
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,P), WW(:,Q), C1)
      endif

      ALPHA= RHO / C1

!C===
!C +----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +----------------------+
!C===
      do i = 1, hecMAT%N * 3
         X(i)  = X(i)    + ALPHA * WW(i,P)
        WW(i,R)= WW(i,R) - ALPHA * WW(i,Q)
      enddo

      if (TIMElog.eq.1) then
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,R), WW(:,R), DNRM2, Tcomm)
      else
      call hecmw_InnerProduct_R(hecMESH, 3, WW(:,R), WW(:,R), DNRM2)
      endif

      RESID= dsqrt(DNRM2/BNRM2)

!C##### ITERATION HISTORY
      if (my_rank.eq.0.and.ITERLog.eq.1) write (*,'(i7, 1pe16.6)') ITER, RESID
!C#####

      if ( RESID.le.TOL   ) exit
      if ( ITER .eq.MAXIT ) ERROR = -300

      RHO1 = RHO

      enddo
!C
!C************************************************* Conjugate Gradient Iteration end
!C
      if (totalmpc.ne.0) then
        if (TIMElog.eq.1) then
        call hecmw_tback_x_33(hecMESH, X, WW(:,WK), Tcomm)
        else
        call hecmw_tback_x_33(hecMESH, X, WW(:,WK))
        endif
      endif
!C
!C-- INTERFACE data EXCHANGE
!C
      START_TIME= HECMW_WTIME()
      call hecmw_update_3_R (hecMESH, X, hecMAT%NP)
      END_TIME = HECMW_WTIME()
      Tcomm = Tcomm + END_TIME - START_TIME

      E1_TIME = HECMW_WTIME()
      Tsol = E1_TIME - S1_TIME

      deallocate (WW)

      end subroutine hecmw_solve_CG_33
      end module     hecmw_solver_CG_33

```