```
      subroutine hecmw_solve_CG_33( hecMESH,  hecMAT, ITER, RESID, ERROR, &
     &                              Tset, Tsol, Tcomm )

      type(hecmwST_local_mesh) :: hecMESH
      type(hecmwST_matrix) :: hecMAT
      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 :: WK= 4

!C | INIT. |
      call hecmw_precond_33_setup(hecMAT, hecMESH, 1)
!C | {r0}= {b} - [A]{x0} |
      call hecmw_matresid_33(hecMESH, hecMAT, X, B, WW(:,R), Tcomm)
!C | compute ||{b}||
      call hecmw_InnerProduct_R(hecMESH, NDOF, B, B, BNRM2, Tcomm)
!C
!C***** Conjugate Gradient Iteration start *****

      do iter = 1, MAXIT

!C | {z}= [Minv]{r} |
      call  hecmw_precond_33_apply(hecMESH,  hecMAT,  WW(:,R),  WW(:,Z),  WW(:,WK),
Tcomm)
!C | {RHO}= {r}{z} |
      call hecmw_InnerProduct_R(hecMESH, NDOF, WW(:,R), WW(:,Z), RHO, Tcomm)

!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
      if ( ITER.eq.1 ) then
        do i = 1, NNDOF
          WW(i,P) = WW(i,Z)
        enddo
```

```
1           else
2             BETA = RHO / RHO1
3             do i = 1, NNDOF
4               WW(i,P) = WW(i,Z) + BETA*WW(i,P)
5             enddo
6           endif
7
8     !C | {q}= [A] {p} |
9           call hecmw_matvec_33(hecMESH, hecMAT, WW(:,P), WW(:,Q), Tcomm)
10
11    !C | ALPHA= RHO / {p}{q} |
12          call hecmw_InnerProduct_R(hecMESH, NDOF, WW(:,P), WW(:,Q), C1, Tcomm)
13
14          ALPHA= RHO / C1
15
16    !C | {x}= {x} + ALPHA*{p} |
17    !C | {r}= {r} - ALPHA*{q} |
18          do i = 1, NNDOF
19             X(i)  = X(i)    + ALPHA * WW(i,P)
20            WW(i,R)= WW(i,R) - ALPHA * WW(i,Q)
21          enddo
22
23          call hecmw_InnerProduct_R(hecMESH, NDOF, WW(:,R), WW(:,R), DNRM2, Tcomm)
24
25          RESID= dsqrt(DNRM2/BNRM2)
26          if ( RESID.le.TOL   ) exit
27          if ( ITER .eq.MAXIT ) ERROR = HECMW_SOLVER_ERROR_NOCONV_MAXIT
28
29          enddo
30    !C
31    !C***** Conjugate Gradient Iteration end *****
32    !C
33          end subroutine hecmw_solve_CG_33
34
```