```
1   module hecmw_solver_misc_33
2
3         contains
4
5   !C
6   !C***
7   !C*** hecmw_matvec_33
8   !C***
9   !C
10        subroutine hecmw_matvec_33 (hecMESH, hecMAT, X, Y, COMMtime)
11        use hecmw_util
12        use m_hecmw_comm_f
13        use hecmw_matrix_contact
14
15        implicit none
16        real(kind=kreal) :: X(:), Y(:)
17        type (hecmwST_local_mesh) :: hecMESH
18        type (hecmwST_matrix)     :: hecMAT
19        real(kind=kreal), optional :: COMMtime
20
21        real(kind=kreal) :: START_TIME, END_TIME
22        integer(kind=kint) :: i, j, jS, jE, in
23        real(kind=kreal) :: YV1, YV2, YV3, X1, X2, X3
24
25        if (present(COMMtime)) then
26        START_TIME= HECMW_WTIME()
27        call hecmw_update_3_R (hecMESH, X, hecMAT%NP)
28        END_TIME= HECMW_WTIME()
29        COMMtime = COMMtime + END_TIME - START_TIME
30        else
31        call hecmw_update_3_R (hecMESH, X, hecMAT%NP)
32        endif
33
34        do i= 1, hecMAT%N
35          X1= X(3*i-2)
36          X2= X(3*i-1)
```

```
37            X3= X(3*i  )
38            YV1= hecMAT%D(9*i-8)*X1 + hecMAT%D(9*i-7)*X2                  &
39          &                         + hecMAT%D(9*i-6)*X3
40            YV2= hecMAT%D(9*i-5)*X1 + hecMAT%D(9*i-4)*X2                  &
41          &                         + hecMAT%D(9*i-3)*X3
42            YV3= hecMAT%D(9*i-2)*X1 + hecMAT%D(9*i-1)*X2                  &
43          &                         + hecMAT%D(9*i  )*X3
44
45            jS= hecMAT%indexL(i-1) + 1
46            jE= hecMAT%indexL(i  )
47            do j= jS, jE
48              in  = hecMAT%itemL(j)
49              X1= X(3*in-2)
50              X2= X(3*in-1)
51              X3= X(3*in  )
52              YV1= YV1 + hecMAT%AL(9*j-8)*X1 + hecMAT%AL(9*j-7)*X2        &
53          &                                  + hecMAT%AL(9*j-6)*X3
54              YV2= YV2 + hecMAT%AL(9*j-5)*X1 + hecMAT%AL(9*j-4)*X2        &
55          &                                  + hecMAT%AL(9*j-3)*X3
56              YV3= YV3 + hecMAT%AL(9*j-2)*X1 + hecMAT%AL(9*j-1)*X2        &
57          &                                  + hecMAT%AL(9*j  )*X3
58            enddo
59            jS= hecMAT%indexU(i-1) + 1
60            jE= hecMAT%indexU(i  )
61            do j= jS, jE
62              in  = hecMAT%itemU(j)
63              X1= X(3*in-2)
64              X2= X(3*in-1)
65              X3= X(3*in  )
66              YV1= YV1 + hecMAT%AU(9*j-8)*X1 + hecMAT%AU(9*j-7)*X2        &
67          &                                  + hecMAT%AU(9*j-6)*X3
68              YV2= YV2 + hecMAT%AU(9*j-5)*X1 + hecMAT%AU(9*j-4)*X2        &
69          &                                  + hecMAT%AU(9*j-3)*X3
70              YV3= YV3 + hecMAT%AU(9*j-2)*X1 + hecMAT%AU(9*j-1)*X2        &
71          &                                  + hecMAT%AU(9*j  )*X3
72            enddo
```

```
73            Y(3*i-2)= YV1
74            Y(3*i-1)= YV2
75            Y(3*i  )= YV3
76          enddo
77
78          call hecmw_cmat_multvec_add( hecMAT%cmat, X, Y, hecMAT%NP * hecMAT%NDOF )
79
80          end subroutine hecmw_matvec_33
81
82    !C
83    !C***
84    !C*** hecmw_matresid_33
85    !C***
86    !C
87          subroutine hecmw_matresid_33 (hecMESH, hecMAT, X, B, R, COMMtime)
88          use hecmw_util
89
90          implicit none
91          real(kind=kreal) :: X(:), B(:), R(:)
92          type (hecmwST_matrix)     :: hecMAT
93          type (hecmwST_local_mesh) :: hecMESH
94          real(kind=kreal), optional :: COMMtime
95
96          integer(kind=kint) :: i
97
98          if (present(COMMtime)) then
99          call hecmw_matvec_33 (hecMESH, hecMAT, X, R, COMMtime)
100         else
101         call hecmw_matvec_33 (hecMESH, hecMAT, X, R)
102         endif
103         do i = 1, hecMAT%N * 3
104           R(i) = B(i) - R(i)
105         enddo
106
107         end subroutine hecmw_matresid_33
108
```

```
109   !C
110   !C***
111   !C*** hecmw_rel_resid_L2_33
112   !C***
113   !C
114         function hecmw_rel_resid_L2_33 (hecMESH, hecMAT, COMMtime)
115         use hecmw_util
116         use hecmw_solver_misc
117
118         implicit none
119         real(kind=kreal) :: hecmw_rel_resid_L2_33
120         type ( hecmwST_local_mesh ), intent(in) :: hecMESH
121         type ( hecmwST_matrix     ), intent(in) :: hecMAT
122         real(kind=kreal), optional :: COMMtime
123
124         real(kind=kreal), allocatable :: r(:)
125         real(kind=kreal) :: bnorm2, rnorm2
126
127         allocate(r(hecMAT%NDOF*hecMAT%NP))
128
129         if (present(COMMtime)) then
130         call  hecmw_InnerProduct_R(hecMESH,  hecMAT%NDOF,  hecMAT%B,  hecMAT%B,  bnorm2,
131   COMMtime)
132         call hecmw_matresid_33(hecMESH, hecMAT, hecMAT%X, hecMAT%B, r, COMMtime)
133         call hecmw_InnerProduct_R(hecMESH, hecMAT%NDOF, r, r, rnorm2, COMMtime)
134         else
135         call hecmw_InnerProduct_R(hecMESH, hecMAT%NDOF, hecMAT%B, hecMAT%B, bnorm2)
136         call hecmw_matresid_33(hecMESH, hecMAT, hecMAT%X, hecMAT%B, r)
137         call hecmw_InnerProduct_R(hecMESH, hecMAT%NDOF, r, r, rnorm2)
138         endif
139         hecmw_rel_resid_L2_33 = sqrt(rnorm2 / bnorm2)
140
141         deallocate(r)
142         end function hecmw_rel_resid_L2_33
143
144   !C
```

```
145   !C***
146   !C*** hecmw_Tvec_33
147   !C***
148   !C
149         subroutine hecmw_Tvec_33 (hecMESH, X, Y, COMMtime)
150         use hecmw_util
151         use m_hecmw_comm_f
152
153         implicit none
154         real(kind=kreal) :: X(:), Y(:)
155         type (hecmwST_local_mesh) :: hecMESH
156         real(kind=kreal), optional :: COMMtime
157
158         real(kind=kreal) :: START_TIME, END_TIME
159         integer(kind=kint) :: i, j, jj, k, kk
160
161         if (present(COMMtime)) then
162         START_TIME= HECMW_WTIME()
163         call hecmw_update_3_R (hecMESH, X, hecMESH%n_node)
164         END_TIME= HECMW_WTIME()
165         COMMtime = COMMtime + END_TIME - START_TIME
166         else
167         call hecmw_update_3_R (hecMESH, X, hecMESH%n_node)
168         endif
169
170         do i= 1, hecMESH%nn_internal * hecMESH%n_dof
171           Y(i)= X(i)
172         enddo
173
174         do i= 1, hecMESH%mpc%n_mpc
175           k = hecMESH%mpc%mpc_index(i-1) + 1
176           kk = 3 * (hecMESH%mpc%mpc_item(k) - 1) + hecMESH%mpc%mpc_dof(k)
177           Y(kk) = 0.d0
178           do j= hecMESH%mpc%mpc_index(i-1) + 2, hecMESH%mpc%mpc_index(i)
179             jj = 3 * (hecMESH%mpc%mpc_item(j) - 1) + hecMESH%mpc%mpc_dof(j)
180             Y(kk) = Y(kk) - hecMESH%mpc%mpc_val(j) * X(jj)
```

```
181           enddo
182         enddo
183
184         end subroutine hecmw_Tvec_33
185
186   !C
187   !C***
188   !C*** hecmw_Ttvec_33
189   !C***
190   !C
191         subroutine hecmw_Ttvec_33 (hecMESH, X, Y, COMMtime)
192         use hecmw_util
193         use m_hecmw_comm_f
194
195         implicit none
196         real(kind=kreal) :: X(:), Y(:)
197         type (hecmwST_local_mesh) :: hecMESH
198         real(kind=kreal), optional :: COMMtime
199
200         real(kind=kreal) :: START_TIME, END_TIME
201         integer(kind=kint) :: i, j, jj, k, kk
202
203         if (present(COMMtime)) then
204         START_TIME= HECMW_WTIME()
205         call hecmw_update_3_R (hecMESH, X, hecMESH%n_node)
206         END_TIME= HECMW_WTIME()
207         COMMtime = COMMtime + END_TIME - START_TIME
208         else
209         call hecmw_update_3_R (hecMESH, X, hecMESH%n_node)
210         endif
211
212         do i= 1, hecMESH%nn_internal * hecMESH%n_dof
213           Y(i)= X(i)
214         enddo
215
216         do i= 1, hecMESH%mpc%n_mpc
```

```
217           k = hecMESH%mpc%mpc_index(i-1) + 1
218           kk = 3 * (hecMESH%mpc%mpc_item(k) - 1) + hecMESH%mpc%mpc_dof(k)
219           Y(kk) = 0.d0
220           do j= hecMESH%mpc%mpc_index(i-1) + 2, hecMESH%mpc%mpc_index(i)
221             jj = 3 * (hecMESH%mpc%mpc_item(j) - 1) + hecMESH%mpc%mpc_dof(j)
222             Y(jj) = Y(jj) - hecMESH%mpc%mpc_val(j) * X(kk)
223           enddo
224         enddo
225
226         end subroutine hecmw_Ttvec_33
227
228   !C
229   !C***
230   !C*** hecmw_TtmatTvec_33
231   !C***
232   !C
233         subroutine hecmw_TtmatTvec_33 (hecMESH, hecMAT, X, Y, W, COMMtime)
234         use hecmw_util
235
236         implicit none
237         real(kind=kreal) :: X(:), Y(:), W(:)
238         type (hecmwST_local_mesh) :: hecMESH
239         type (hecmwST_matrix)     :: hecMAT
240         real(kind=kreal), optional :: COMMtime
241
242         if (present(COMMtime)) then
243         call hecmw_Tvec_33(hecMESH, X, Y, COMMtime)
244         call hecmw_matvec_33 (hecMESH, hecMAT, Y, W, COMMtime)
245         call hecmw_Ttvec_33(hecMESH, W, Y, COMMtime)
246         else
247         call hecmw_Tvec_33(hecMESH, X, Y)
248         call hecmw_matvec_33 (hecMESH, hecMAT, Y, W)
249         call hecmw_Ttvec_33(hecMESH, W, Y)
250         endif
251
252         end subroutine hecmw_TtmatTvec_33
```

```
253
254   !C
255   !C***
256   !C*** hecmw_precond_33
257   !C***
258   !C
259         subroutine hecmw_precond_33(hecMESH, hecMAT, R, Z, ZP, COMMtime)
260         use hecmw_util
261         use hecmw_matrix_misc
262
263         implicit none
264         real(kind=kreal) :: R(:), Z(:), ZP(:)
265         type (hecmwST_local_mesh) :: hecMESH
266         type (hecmwST_matrix)     :: hecMAT
267         real(kind=kreal), optional :: COMMtime
268
269         integer(kind=kint ) :: PRECOND, iterPREmax
270         integer(kind=kint ) :: i, j, k, iterPRE, isL, ieL, isU, ieU
271         real   (kind=kreal) ::X1,X2,X3
272         real   (kind=kreal) ::SW1,SW2,SW3
273
274         PRECOND = hecmw_mat_get_precond( hecMAT )
275         iterPREmax = hecmw_mat_get_iterpremax( hecMAT )
276
277         if (iterPREmax.le.0) then
278           do i= 1, hecMAT%N * 3
279             Z(i)= R(i)
280           enddo
281           return
282         endif
283
284   !C===
285   !C +----------------+
286   !C | {z}= [Minv]{r} |
287   !C +----------------+
288   !C===
```

```
289
290   !C
291   !C== Block SSOR
292         if (PRECOND.le.2) then
293
294         do i= 1, hecMAT%NP * 3
295           ZP(i)= R(i)
296         enddo
297         do i= 1, hecMAT%NP * 3
298           Z(i)= 0.d0
299         enddo
300
301         do iterPRE= 1, iterPREmax
302
303   !C-- FORWARD
304
305           do i= 1, hecMAT%N
306             SW1= ZP(3*i-2)
307             SW2= ZP(3*i-1)
308             SW3= ZP(3*i  )
309             isL= hecMAT%indexL(i-1)+1
310             ieL= hecMAT%indexL(i)
311             do j= isL, ieL
312                 k= hecMAT%itemL(j)
313               X1= ZP(3*k-2)
314               X2= ZP(3*k-1)
315               X3= ZP(3*k  )
316               SW1= SW1 - hecMAT%AL(9*j-8)*X1 - hecMAT%AL(9*j-7)*X2 - hecMAT%AL(9*j-6)*X3
317               SW2= SW2 - hecMAT%AL(9*j-5)*X1 - hecMAT%AL(9*j-4)*X2 - hecMAT%AL(9*j-3)*X3
318               SW3= SW3 - hecMAT%AL(9*j-2)*X1 - hecMAT%AL(9*j-1)*X2 - hecMAT%AL(9*j  )*X3
319             enddo
320
321             if (hecMAT%cmat%n_val.ne.0) then
322               isL= hecMAT%indexCL(i-1)+1
323               ieL= hecMAT%indexCL(i)
324               do j= isL, ieL
```

```
                  k= hecMAT%itemCL(j)
                 X1= ZP(3*k-2)
                 X2= ZP(3*k-1)
                 X3= ZP(3*k  )
                SW1=   SW1   -   hecMAT%CAL(9*j-8)*X1   -   hecMAT%CAL(9*j-7)*X2   -
hecMAT%CAL(9*j-6)*X3
                SW2=   SW2   -   hecMAT%CAL(9*j-5)*X1   -   hecMAT%CAL(9*j-4)*X2   -
hecMAT%CAL(9*j-3)*X3
                SW3=   SW3   -   hecMAT%CAL(9*j-2)*X1   -   hecMAT%CAL(9*j-1)*X2   -
hecMAT%CAL(9*j  )*X3
              enddo
            endif

            X1= SW1
            X2= SW2
            X3= SW3
            X2= X2 - hecMAT%ALU(9*i-5)*X1
            X3= X3 - hecMAT%ALU(9*i-2)*X1 - hecMAT%ALU(9*i-1)*X2
            X3= hecMAT%ALU(9*i  )*  X3
            X2= hecMAT%ALU(9*i-4)*( X2 - hecMAT%ALU(9*i-3)*X3 )
            X1= hecMAT%ALU(9*i-8)*( X1 - hecMAT%ALU(9*i-6)*X3 - hecMAT%ALU(9*i-7)*X2)
            ZP(3*i-2)= X1
            ZP(3*i-1)= X2
            ZP(3*i  )= X3
          enddo

!C-- BACKWARD

          do i= hecMAT%N, 1, -1
            isU= hecMAT%indexU(i-1) + 1
            ieU= hecMAT%indexU(i)
            SW1= 0.d0
            SW2= 0.d0
            SW3= 0.d0
            do j= ieU, isU, -1
                k= hecMAT%itemU(j)
```

```
                X1= ZP(3*k-2)
                X2= ZP(3*k-1)
                X3= ZP(3*k  )
              SW1= SW1 + hecMAT%AU(9*j-8)*X1 + hecMAT%AU(9*j-7)*X2 + hecMAT%AU(9*j-6)*X3
              SW2= SW2 + hecMAT%AU(9*j-5)*X1 + hecMAT%AU(9*j-4)*X2 + hecMAT%AU(9*j-3)*X3
              SW3= SW3 + hecMAT%AU(9*j-2)*X1 + hecMAT%AU(9*j-1)*X2 + hecMAT%AU(9*j  )*X3
            enddo

            if (hecMAT%cmat%n_val.gt.0) then
              isU= hecMAT%indexCU(i-1) + 1
              ieU= hecMAT%indexCU(i)
              do j= isU, ieU
                  k= hecMAT%itemCU(j)
                X1= ZP(3*k-2)
                X2= ZP(3*k-1)
                X3= ZP(3*k  )
                SW1=   SW1   +   hecMAT%CAU(9*j-8)*X1   +   hecMAT%CAU(9*j-7)*X2   +
hecMAT%CAU(9*j-6)*X3
                SW2=   SW2   +   hecMAT%CAU(9*j-5)*X1   +   hecMAT%CAU(9*j-4)*X2   +
hecMAT%CAU(9*j-3)*X3
                SW3=   SW3   +   hecMAT%CAU(9*j-2)*X1   +   hecMAT%CAU(9*j-1)*X2   +
hecMAT%CAU(9*j  )*X3
              enddo
            endif

            X1= SW1
            X2= SW2
            X3= SW3
            X2= X2 - hecMAT%ALU(9*i-5)*X1
            X3= X3 - hecMAT%ALU(9*i-2)*X1 - hecMAT%ALU(9*i-1)*X2
            X3= hecMAT%ALU(9*i  )*  X3
            X2= hecMAT%ALU(9*i-4)*( X2 - hecMAT%ALU(9*i-3)*X3 )
            X1= hecMAT%ALU(9*i-8)*( X1 - hecMAT%ALU(9*i-6)*X3 - hecMAT%ALU(9*i-7)*X2)
            ZP(3*i-2)=  ZP(3*i-2) - X1
            ZP(3*i-1)=  ZP(3*i-1) - X2
            ZP(3*i  )=  ZP(3*i  ) - X3
```

```
397           enddo
398   !C
399   !C-- additive Schwartz
400   !C
401           do i= 1, hecMAT%N * 3
402             Z(i)= Z(i) + ZP(i)
403           enddo
404
405           if (iterPRE.eq.iterPREmax) exit
406
407   !C--    {ZP} = {R} - [A] {Z}
408
409           if (present(COMMtime)) then
410           call hecmw_matresid_33 (hecMESH, hecMAT, Z, R, ZP, COMMtime)
411           else
412           call hecmw_matresid_33 (hecMESH, hecMAT, Z, R, ZP)
413           endif
414
415         enddo
416         endif
417
418   !C
419   !C== Block SCALING
420         if (PRECOND.eq.3) then
421
422         do i= 1, hecMAT%N * 3
423           Z(i)= R(i)
424         enddo
425
426         do i= 1, hecMAT%N
427           X1= Z(3*i-2)
428           X2= Z(3*i-1)
429           X3= Z(3*i  )
430           X2= X2 - hecMAT%ALU(9*i-5)*X1
431           X3= X3 - hecMAT%ALU(9*i-2)*X1 - hecMAT%ALU(9*i-1)*X2
432           X3= hecMAT%ALU(9*i  )*  X3
```

```
433           X2= hecMAT%ALU(9*i-4)*( X2 - hecMAT%ALU(9*i-3)*X3 )
434           X1= hecMAT%ALU(9*i-8)*( X1 - hecMAT%ALU(9*i-6)*X3 - hecMAT%ALU(9*i-7)*X2)
435           Z(3*i-2)= X1
436           Z(3*i-1)= X2
437           Z(3*i  )= X3
438         enddo
439         endif
440
441         end subroutine hecmw_precond_33
442
443   !C
444   !C***
445   !C*** hecmw_mpc_scale
446   !C***
447   !C
448         subroutine hecmw_mpc_scale(hecMESH)
449         use hecmw_util
450
451         implicit none
452         type (hecmwST_local_mesh) :: hecMESH
453         integer(kind=kint) :: i, j, k
454         real(kind=kreal) :: WVAL
455
456         do i = 1, hecMESH%mpc%n_mpc
457           k = hecMESH%mpc%mpc_index(i-1)+1
458           WVAL = 1.d0 / hecMESH%mpc%mpc_val(k)
459           hecMESH%mpc%mpc_val(k) = 1.d0
460           do j = hecMESH%mpc%mpc_index(i-1)+2, hecMESH%mpc%mpc_index(i)
461             hecMESH%mpc%mpc_val(j) = hecMESH%mpc%mpc_val(j) * WVAL
462           enddo
463           hecMESH%mpc%mpc_const(i) = hecMESH%mpc%mpc_const(i) * WVAL
464         enddo
465
466         end subroutine hecmw_mpc_scale
467
468   !C
```

```
469   !C***
470   !C*** hecmw_trans_b_33
471   !C***
472   !C
473         subroutine hecmw_trans_b_33(hecMESH, hecMAT, B, BT, W, COMMtime)
474         use hecmw_util
475
476         implicit none
477         type (hecmwST_local_mesh) :: hecMESH
478         type (hecmwST_matrix)     :: hecMAT
479         real(kind=kreal) :: B(:), W(:)
480         real(kind=kreal), target :: BT(:)
481         real(kind=kreal), optional :: COMMtime
482
483         real(kind=kreal), pointer :: XG(:)
484         integer(kind=kint) :: i, k, kk
485
486   !C===
487   !C +---------------------------+
488   !C | {bt}= [T'](({b} - [A]{xg}) |
489   !C +---------------------------+
490   !C===
491         XG => BT
492         XG = 0.d0
493
494   !C-- Generate {xg} from mpc_const
495         do i = 1, hecMESH%mpc%n_mpc
496           k = hecMESH%mpc%mpc_index(i-1) + 1
497           kk = 3 * hecMESH%mpc%mpc_item(k) + hecMESH%mpc%mpc_dof(k) - 3
498           XG(kk) = hecMESH%mpc%mpc_const(i)
499         enddo
500
501   !C-- {w} = {b} - [A]{xg}
502         if (present(COMMtime)) then
503         call hecmw_matresid_33 (hecMESH, hecMAT, XG, B, W, COMMtime)
504         else
```

```
505         call hecmw_matresid_33 (hecMESH, hecMAT, XG, B, W)
506         endif
507
508   !C-- {bt} = [T'] {w}
509         if (present(COMMtime)) then
510         call hecmw_Ttvec_33(hecMESH, W, BT, COMMtime)
511         else
512         call hecmw_Ttvec_33(hecMESH, W, BT)
513         endif
514
515         end subroutine hecmw_trans_b_33
516
517   !C
518   !C***
519   !C*** hecmw_tback_x_33
520   !C***
521   !C
522         subroutine hecmw_tback_x_33(hecMESH, X, W, COMMtime)
523         use hecmw_util
524
525         implicit none
526         type (hecmwST_local_mesh) :: hecMESH
527         real(kind=kreal) :: X(:), W(:)
528         real(kind=kreal), optional :: COMMtime
529
530         integer(kind=kint) :: i, k, kk
531
532   !C-- {tx} = [T]{x}
533         if (present(COMMtime)) then
534         call hecmw_Tvec_33(hecMESH, X, W, COMMtime)
535         else
536         call hecmw_Tvec_33(hecMESH, X, W)
537         endif
538
539   !C-- {x} = {tx} + {xg}
540
```

```
541         do i= 1, hecMESH%nn_internal * 3
542           X(i)= W(i)
543         enddo
544
545         do i = 1, hecMESH%mpc%n_mpc
546           k = hecMESH%mpc%mpc_index(i-1) + 1
547           kk = 3 * hecMESH%mpc%mpc_item(k) + hecMESH%mpc%mpc_dof(k) - 3
548           X(kk) = X(kk) + hecMESH%mpc%mpc_const(i)
549         enddo
550
551         end subroutine hecmw_tback_x_33
552
553   end module hecmw_solver_misc_33
554
```