第 4 回演習プログラム

新領域創成科学研究科　人間環境学専攻
橋本　学

第 4 回演習資料のプログラムを以下に示します.

第 3 節のプログラム : モジュール mod_elemstiffmat3d, mod_elemexforcevec3d の作成

１．要素剛性マトリックスモジュール mod_elemstiffmat3d.f90

```
 1:      MODULE mod_elemstiffmat3d
 2:!######################################################################
 3:
 4:      USE mod_nodes3d
 5:      USE mod_localelement3d
 6:      USE mod_elements3d
 7:
 8:!----------------------------------------------------------------------
 9:
10:      IMPLICIT NONE
11:
12:!----------------------------------------------------------------------
13:
14:      TYPE :: struct_elemstiffmat3d
15:
16:       !-------------------------------------------------------
17:
18:       PRIVATE
19:
20:       !-------------------------------------------------------
```

```
21:
22:       TYPE(struct_nodes3d), POINTER        :: ns3d => NULL()
23:       TYPE(struct_localelement3d), POINTER :: le3d => NULL()
24:       TYPE(struct_elements3d), POINTER     :: es3d => NULL()
25:
26:       !-------------------------------------------------------
27:       !
28:       ! k(:, :, :)
29:       ! Element stiffness matrix
30:       !
31:       !-------------------------------------------------------
32:       !
33:       ! e(:)
34:       ! Young's modulus
35:       !
36:       ! nu(:)
37:       ! Poisson's ratio
38:       !
39:       !-------------------------------------------------------
40:       !
41:       ! evec(:, :)
42:       ! Infinitesimal strain
43:       !
44:       ! svec(:, :)
45:       ! Stress
46:       !
47:       ! s_mises(:)
48:       ! Mises stress
49:       !
50:       !-------------------------------------------------------
51:
52:       REAL(8), ALLOCATABLE :: k(:, :, :)
53:       REAL(8), ALLOCATABLE :: e(:)
54:       REAL(8), ALLOCATABLE :: nu(:)
55:       REAL(8), ALLOCATABLE :: evec(:, :)
56:       REAL(8), ALLOCATABLE :: svec(:, :)
```

```
57:       REAL(8), ALLOCATABLE :: s_mises(:)
58:
59:       !-------------------------------------------------------
60:
61:      END TYPE
62:
63:!-----------------------------------------------------------------
64:
65:      CONTAINS
66:
67:
68:      ! Get element stiffness matrix
69:!#################################################################
70:      SUBROUTINE get_elemstiffmat3d_k(esm3d, k)
71:!#################################################################
72:
73:      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d
74:
75:      REAL(8), INTENT(OUT) :: k(:, :, :)
76:
77:!-----------------------------------------------------------------
78:
79:      k = esm3d%k
80:
81:!-----------------------------------------------------------------
82:
83:      RETURN
84:
85:!#################################################################
86:      END SUBROUTINE get_elemstiffmat3d_k
87:!#################################################################
88:
89:
90:      ! Set Young's modulus and Poisson's ratio
91:!#################################################################
92:      SUBROUTINE set_elemstiffmat3d_e_nu(esm3d, e, nu)
```

```
93:!######################################################################
94:
95:       TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d
96:
97:       REAL(8), INTENT(IN) :: e(:)
98:       REAL(8), INTENT(IN) :: nu(:)
99:
100:!----------------------------------------------------------------------
101:
102:      esm3d%e  = e
103:      esm3d%nu = nu
104:
105:!----------------------------------------------------------------------
106:
107:      RETURN
108:
109:!######################################################################
110:      END SUBROUTINE set_elemstiffmat3d_e_nu
111:!######################################################################
112:
113:
114:      ! Get Young's modulus and Poisson's ratio
115:!######################################################################
116:      SUBROUTINE get_elemstiffmat3d_e(esm3d, e, nu)
117:!######################################################################
118:
119:      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d
120:
121:      REAL(8), INTENT(OUT) :: e(:)
122:      REAL(8), INTENT(OUT) :: nu(:)
123:
124:!----------------------------------------------------------------------
125:
126:      e  = esm3d%e
127:      nu = esm3d%nu
128:
```

```
129:!-----------------------------------------------------------------------
130:
131:      RETURN
132:
133:!#######################################################################
134:      END SUBROUTINE get_elemstiffmat3d_e
135:!#######################################################################
136:
137:
138:      ! Get infinitesimal strain and stress
139:!#######################################################################
140:      SUBROUTINE get_elemstiffmat3d_evec_svec &
141:                  (esm3d, evec, svec, s_mises)
142:!#######################################################################
143:
144:      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d
145:
146:      REAL(8), INTENT(OUT) :: evec(:, :)
147:      REAL(8), INTENT(OUT) :: svec(:, :)
148:      REAL(8), INTENT(OUT) :: s_mises(:)
149:
150:!-----------------------------------------------------------------------
151:
152:      evec = esm3d%evec
153:      svec = esm3d%svec
154:
155:      s_mises = esm3d%s_mises
156:
157:!-----------------------------------------------------------------------
158:
159:      RETURN
160:
161:!#######################################################################
162:      END SUBROUTINE get_elemstiffmat3d_evec_svec
163:!#######################################################################
164:
```

```
165:
166:!######################################################################
167:      SUBROUTINE init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
168:!######################################################################
169:
170:      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d
171:
172:      TYPE(struct_nodes3d), TARGET, INTENT(IN)        :: ns3d
173:      TYPE(struct_localelement3d), TARGET, INTENT(IN) :: le3d
174:      TYPE(struct_elements3d), TARGET, INTENT(IN)     :: es3d
175:
176:!----------------------------------------------------------------------
177:
178:      INTEGER :: le3d_nnodes
179:      INTEGER :: es3d_n
180:
181:!----------------------------------------------------------------------
182:
183:      esm3d%ns3d => ns3d
184:      esm3d%le3d => le3d
185:      esm3d%es3d => es3d
186:
187:!----------------------------------------------------------------------
188:
189:      CALL get_localelement3d_nnodes(esm3d%le3d, le3d_nnodes)
190:
191:      CALL get_elements3d_n(esm3d%es3d, es3d_n)
192:
193:!----------------------------------------------------------------------
194:
195:      ALLOCATE( esm3d%k(3*le3d_nnodes, 3*le3d_nnodes, es3d_n) )
196:
197:      esm3d%k = 0.0D0
198:
199:      !-------------------------------------------------------------
200:
```

```
201:      ALLOCATE( esm3d%e(es3d_n)  )
202:      ALLOCATE( esm3d%nu(es3d_n) )
203:
204:      esm3d%e  = 0.0D0
205:      esm3d%nu = 0.0D0
206:
207:      !-------------------------------------------------------------
208:
209:      ALLOCATE( esm3d%evec(6, es3d_n) )
210:
211:      esm3d%evec = 0.0D0
212:
213:      ALLOCATE( esm3d%svec(6, es3d_n) )
214:
215:      esm3d%svec = 0.0D0
216:
217:      ALLOCATE( esm3d%s_mises(es3d_n) )
218:
219:      esm3d%s_mises = 0.0D0
220:
221:!-----------------------------------------------------------------------
222:
223:      RETURN
224:
225:!#######################################################################
226:      END SUBROUTINE init_elemstiffmat3d
227:!#######################################################################
228:
229:
230:!#######################################################################
231:      SUBROUTINE cal_elemstiffmat3d(esm3d)
232:!#######################################################################
233:
234:      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d
235:
236:!-----------------------------------------------------------------------
```

```
237:
238:      INTEGER :: ns3d_n
239:      INTEGER :: le3d_nnodes
240:      INTEGER :: le3d_nqps
241:      INTEGER :: es3d_n
242:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
243:      INTEGER :: nqps_tot
244:      INTEGER :: i, j, k
245:      INTEGER :: id
246:      INTEGER :: na, nb
247:      INTEGER :: ie
248:      INTEGER :: idof
249:      INTEGER :: isize, jsize
250:      INTEGER :: jsize1, jsize2, jsize3
251:      INTEGER :: ijk
252:
253:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
254:      REAL(8), ALLOCATABLE :: ns3d_u(:)
255:      REAL(8), ALLOCATABLE :: le3d_xi_qp(:, :)
256:      REAL(8), ALLOCATABLE :: le3d_w_qp(:, :)
257:      REAL(8), ALLOCATABLE :: le3d_n_qp(:, :)
258:      REAL(8), ALLOCATABLE :: le3d_dndxi_qp(:, :, :)
259:      REAL(8), ALLOCATABLE :: x_local(:, :)
260:      REAL(8), ALLOCATABLE :: u_local(:)
261:      REAL(8) :: nqps_tot_inv
262:      REAL(8) :: g1(3), g2(3), g3(3)
263:      REAL(8) :: det_j, det_j_inv
264:      REAL(8) :: w_w_w_det_j
265:      REAL(8) :: cg1(3), cg2(3), cg3(3)
266:      REAL(8), ALLOCATABLE :: dndx(:, :)
267:      REAL(8), ALLOCATABLE :: bmat(:, :)
268:      REAL(8) :: evec(6)
269:      REAL(8) :: lambda, mu
270:      REAL(8) :: dmat(6, 6)
271:      REAL(8), ALLOCATABLE :: cmat(:, :)
272:      REAL(8) :: svec(6)
```

```
273:
274:!------------------------------------------------------------------
275:
276:      CALL get_nodes3d_n(esm3d%ns3d, ns3d_n)
277:      ALLOCATE( ns3d_x(3, ns3d_n) )
278:      CALL get_nodes3d_x(esm3d%ns3d, ns3d_x)
279:      ALLOCATE( ns3d_u(3*ns3d_n) )
280:      CALL get_nodes3d_u(esm3d%ns3d, ns3d_u)
281:
282:      CALL get_localelement3d_nnodes(esm3d%le3d, le3d_nnodes)
283:      CALL get_localelement3d_nqps(esm3d%le3d, le3d_nqps)
284:      nqps_tot = le3d_nqps*le3d_nqps*le3d_nqps
285:      nqps_tot_inv = 1.0D0/DFLOAT( nqps_tot )
286:      ALLOCATE( le3d_xi_qp(3, nqps_tot) )
287:      ALLOCATE( le3d_w_qp(3, nqps_tot) )
288:      CALL get_localelement3d_xi_w_qp          &
289:           (esm3d%le3d, le3d_xi_qp, le3d_w_qp)
290:      ALLOCATE( le3d_n_qp(le3d_nnodes, nqps_tot) )
291:      ALLOCATE( le3d_dndxi_qp(3, le3d_nnodes, nqps_tot) )
292:      CALL get_localelement3d_n_qp               &
293:           (esm3d%le3d, le3d_n_qp, le3d_dndxi_qp)
294:
295:      CALL get_elements3d_n(esm3d%es3d, es3d_n)
296:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
297:      CALL get_elements3d_connectivity(esm3d%es3d, es3d_connectivity)
298:
299:      ALLOCATE( x_local(3, le3d_nnodes) )
300:      ALLOCATE( u_local(3*le3d_nnodes) )
301:      ALLOCATE( dndx(3, le3d_nnodes) )
302:      ALLOCATE( bmat(6, 3*le3d_nnodes) )
303:      ALLOCATE( cmat(6, 3*le3d_nnodes) )
304:
305:!------------------------------------------------------------------
306:
307:      esm3d%k = 0.0D0
308:
```

```
309:      DO ie = 1, es3d_n
310:
311:       !-----------------------------------------------------
312:
313:       DO na = 1, le3d_nnodes
314:
315:        id = es3d_connectivity(na, ie)
316:
317:        DO i = 1, 3
318:
319:         idof  = 3*(id-1)+i
320:         isize = 3*(na-1)+i
321:
322:         x_local(i, na) = ns3d_x(i, id)
323:         u_local(isize) = ns3d_u(idof)
324:
325:        END DO
326:
327:       END DO
328:
329:       !-----------------------------------------------------
330:
331:       DO ijk = 1, nqps_tot
332:
333:        !-------------------------------------------------
334:
335:        ! Covariant basis vector
336:        DO i = 1, 3
337:
338:         g1(i) = 0.0D0
339:         g2(i) = 0.0D0
340:         g3(i) = 0.0D0
341:
342:         DO na = 1, le3d_nnodes
343:
344:          g1(i) = g1(i)+le3d_dndxi_qp(1, na, ijk)*x_local(i, na)
```

```
345:           g2(i) = g2(i)+le3d_dndxi_qp(2, na, ijk)*x_local(i, na)
346:           g3(i) = g3(i)+le3d_dndxi_qp(3, na, ijk)*x_local(i, na)
347:
348:         END DO
349:
350:       END DO
351:
352:       !-------------------------------------------------
353:
354:       ! Jacobian
355:       det_j = g1(1)*( g2(2)*g3(3)-g2(3)*g3(2) ) &
356:             +g1(2)*( g2(3)*g3(1)-g2(1)*g3(3) ) &
357:             +g1(3)*( g2(1)*g3(2)-g2(2)*g3(1) )
358:
359:       det_j_inv = 1.0D0/det_j
360:
361:       w_w_w_det_j                                              &
362:       = le3d_w_qp(1, ijk)*le3d_w_qp(2, ijk)*le3d_w_qp(3, ijk) &
363:         *det_j
364:
365:       !-------------------------------------------------
366:
367:       ! Contravariant basis vector
368:       cg1(1) = det_j_inv                    &
369:                *( g2(2)*g3(3)-g2(3)*g3(2) )
370:       cg1(2) = det_j_inv                    &
371:                *( g2(3)*g3(1)-g2(1)*g3(3) )
372:       cg1(3) = det_j_inv                    &
373:                *( g2(1)*g3(2)-g2(2)*g3(1) )
374:       cg2(1) = det_j_inv                    &
375:                *( g3(2)*g1(3)-g3(3)*g1(2) )
376:       cg2(2) = det_j_inv                    &
377:                *( g3(3)*g1(1)-g3(1)*g1(3) )
378:       cg2(3) = det_j_inv                    &
379:                *( g3(1)*g1(2)-g3(2)*g1(1) )
380:       cg3(1) = det_j_inv                    &
```

```
381:                 *( g1(2)*g2(3)-g1(3)*g2(2) )
382:         cg3(2) = det_j_inv                    &
383:                 *( g1(3)*g2(1)-g1(1)*g2(3) )
384:         cg3(3) = det_j_inv                    &
385:                 *( g1(1)*g2(2)-g1(2)*g2(1) )
386:
387:         !-----------------------------------------------
388:
389:         DO na = 1, le3d_nnodes
390:
391:          dndx(1, na)                          &
392:          = cg1(1)*le3d_dndxi_qp(1, na, ijk) &
393:           +cg2(1)*le3d_dndxi_qp(2, na, ijk) &
394:           +cg3(1)*le3d_dndxi_qp(3, na, ijk)
395:          dndx(2, na)                          &
396:          = cg1(2)*le3d_dndxi_qp(1, na, ijk) &
397:           +cg2(2)*le3d_dndxi_qp(2, na, ijk) &
398:           +cg3(2)*le3d_dndxi_qp(3, na, ijk)
399:          dndx(3, na)                          &
400:          = cg1(3)*le3d_dndxi_qp(1, na, ijk) &
401:           +cg2(3)*le3d_dndxi_qp(2, na, ijk) &
402:           +cg3(3)*le3d_dndxi_qp(3, na, ijk)
403:
404:         END DO
405:
406:         !-----------------------------------------------
407:
408:         ! B matrix
409:
410:         bmat = 0.0D0
411:
412:         DO nb = 1, le3d_nnodes
413:
414:          jsize1 = 3*(nb-1)+1
415:          jsize2 = 3*(nb-1)+2
416:          jsize3 = 3*(nb-1)+3
```

```
417:
418:         bmat(1, jsize1) = dndx(1, nb)
419:         bmat(4, jsize1) = dndx(2, nb)
420:         bmat(6, jsize1) = dndx(3, nb)
421:         bmat(2, jsize2) = dndx(2, nb)
422:         bmat(4, jsize2) = dndx(1, nb)
423:         bmat(5, jsize2) = dndx(3, nb)
424:         bmat(3, jsize3) = dndx(3, nb)
425:         bmat(5, jsize3) = dndx(2, nb)
426:         bmat(6, jsize3) = dndx(1, nb)
427:
428:        END DO
429:
430:        !-----------------------------------------------
431:
432:        evec = 0.0D0
433:
434:        DO i = 1, 6
435:
436:         DO jsize = 1, 3*le3d_nnodes
437:
438:          evec(i) = evec(i)+bmat(i, jsize)*u_local(jsize)
439:
440:         END DO
441:
442:        END DO
443:
444:        !-----------------------------------------------
445:
446:        lambda                                                  &
447:        = ( esm3d%e(ie)*esm3d%nu(ie) )                          &
448:          /( ( 1.0D0+esm3d%nu(ie) )*( 1.0D0-2.0D0*esm3d%nu(ie) ) )
449:        mu = esm3d%e(ie)/( 2.0D0*( 1.0D0+esm3d%nu(ie) ) )
450:
451:        !-----------------------------------------------
452:
```

```
453:        ! D matrix
454:
455:        dmat = 0.0D0
456:
457:        dmat(1, 1) = lambda+2.0D0*mu
458:        dmat(2, 2) = lambda+2.0D0*mu
459:        dmat(3, 3) = lambda+2.0D0*mu
460:        dmat(1, 2) = lambda
461:        dmat(1, 3) = lambda
462:        dmat(2, 1) = lambda
463:        dmat(2, 3) = lambda
464:        dmat(3, 1) = lambda
465:        dmat(3, 2) = lambda
466:        dmat(4, 4) = mu
467:        dmat(5, 5) = mu
468:        dmat(6, 6) = mu
469:
470:        !-----------------------------------------------
471:
472:        DO i = 1, 6
473:
474:         DO jsize = 1, 3*le3d_nnodes
475:
476:          cmat(i, jsize) = 0.0D0
477:
478:          DO k = 1, 6
479:
480:           cmat(i, jsize) = cmat(i, jsize)+dmat(i, k)*bmat(k, jsize)
481:
482:          END DO
483:
484:         END DO
485:
486:        END DO
487:
488:        DO isize = 1, 3*le3d_nnodes
```

```
489:
490:         DO jsize = 1, 3*le3d_nnodes
491:
492:          DO k = 1, 6
493:
494:           esm3d%k(isize, jsize, ie)                    &
495:           = esm3d%k(isize, jsize, ie)                  &
496:            +w_w_w_det_j*bmat(k, isize)*cmat(k, jsize)
497:
498:          END DO
499:
500:         END DO
501:
502:        END DO
503:
504:        !-----------------------------------------------
505:
506:        svec = 0.0D0
507:
508:        DO i = 1, 6
509:
510:         DO j = 1, 6
511:
512:          svec(i) = svec(i)+dmat(i, j)*evec(j)
513:
514:         END DO
515:
516:        END DO
517:
518:        !-----------------------------------------------
519:
520:        DO i = 1, 6
521:
522:         esm3d%evec(i, ie) = esm3d%evec(i, ie)+evec(i)
523:         esm3d%svec(i, ie) = esm3d%svec(i, ie)+svec(i)
524:
```

```
525:        END DO
526:
527:        !-------------------------------------------------
528:
529:       END DO
530:
531:       !-------------------------------------------------------
532:
533:       DO I = 1, 6
534:
535:        esm3d%evec(i, ie) = nqps_tot_inv*esm3d%evec(i, ie)
536:        esm3d%svec(i, ie) = nqps_tot_inv*esm3d%svec(i, ie)
537:
538:       END DO
539:
540:       esm3d%s_mises(ie)                                          &
541:       = DSQRT( 0.5D0*( ( esm3d%svec(1, ie)-esm3d%svec(2, ie) )**2 &
542:                       +( esm3d%svec(2, ie)-esm3d%svec(3, ie) )**2 &
543:                       +( esm3d%svec(3, ie)-esm3d%svec(1, ie) )**2 &
544:                       +6.0D0                                      &
545:                        *( esm3d%svec(4, ie)**2                    &
546:                          +esm3d%svec(5, ie)**2                    &
547:                          +esm3d%svec(6, ie)**2 ) ) )
548:
549:       !-------------------------------------------------------
550:
551:      END DO
552:
553:!-----------------------------------------------------------------------
554:
555:      DEALLOCATE( ns3d_x )
556:      DEALLOCATE( ns3d_u )
557:
558:      DEALLOCATE( le3d_xi_qp )
559:      DEALLOCATE( le3d_w_qp )
560:      DEALLOCATE( le3d_n_qp )
```

```
561:      DEALLOCATE( le3d_dndxi_qp )
562:
563:      DEALLOCATE( es3d_connectivity )
564:
565:      DEALLOCATE( x_local )
566:      DEALLOCATE( u_local )
567:      DEALLOCATE( dndx )
568:      DEALLOCATE( bmat )
569:      DEALLOCATE( cmat )
570:
571:!---------------------------------------------------------------------
572:
573:      RETURN
574:
575:!#####################################################################
576:      END SUBROUTINE cal_elemstiffmat3d
577:!#####################################################################
578:
579:
580:!#####################################################################
581:      SUBROUTINE del_elemstiffmat3d(esm3d)
582:!#####################################################################
583:
584:      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d
585:
586:!---------------------------------------------------------------------
587:
588:      NULLIFY( esm3d%ns3d )
589:      NULLIFY( esm3d%le3d )
590:      NULLIFY( esm3d%es3d )
591:
592:!---------------------------------------------------------------------
593:
594:      DEALLOCATE( esm3d%k )
595:
596:      !-------------------------------------------------------------
```

```
597:
598:      DEALLOCATE( esm3d%e  )
599:      DEALLOCATE( esm3d%nu )
600:
601:      !-------------------------------------------------------------
602:
603:      DEALLOCATE( esm3d%evec )
604:      DEALLOCATE( esm3d%svec )
605:      DEALLOCATE( esm3d%s_mises )
606:
607:!-----------------------------------------------------------------------
608:
609:      RETURN
610:
611:!#######################################################################
612:      END SUBROUTINE del_elemstiffmat3d
613:!#######################################################################
614:
615:
616:!#######################################################################
617:      END MODULE mod_elemstiffmat3d
```

２．要素外力ベクトルモジュール mod_elemexforcevec3d.f90

```
 1:      MODULE mod_elemexforcevec3d
 2:!#######################################################################
 3:
 4:      USE mod_nodes3d
 5:      USE mod_localelement3d
 6:      USE mod_elements3d
 7:
 8:!-----------------------------------------------------------------------
 9:
10:      IMPLICIT NONE
11:
12:!-----------------------------------------------------------------------
```

```
13:
14:     TYPE :: struct_elemexforcevec3d
15:
16:      !-------------------------------------------------------
17:
18:      PRIVATE
19:
20:      !-------------------------------------------------------
21:
22:      TYPE(struct_nodes3d), POINTER        :: ns3d => NULL()
23:      TYPE(struct_localelement3d), POINTER :: le3d => NULL()
24:      TYPE(struct_elements3d), POINTER     :: es3d => NULL()
25:
26:      !-------------------------------------------------------
27:      !
28:      ! f(:, :)
29:      ! Element external force vector
30:      !
31:      !-------------------------------------------------------
32:
33:      ! rho(:)
34:      ! Density
35:      !
36:      ! g
37:      ! Gravitational acceleration
38:      !
39:      !-------------------------------------------------------
40:
41:      REAL(8), ALLOCATABLE :: f(:, :)
42:      REAL(8), ALLOCATABLE :: rho(:)
43:      REAL(8) :: g
44:
45:      !-------------------------------------------------------
46:
47:     END TYPE struct_elemexforcevec3d
48:
```

```
49:!-------------------------------------------------------------------
50:
51:      CONTAINS
52:
53:
54:      ! Get element external force vector
55:!#####################################################################
56:      SUBROUTINE get_elemexforcevec3d_f(efv3d, f)
57:!#####################################################################
58:
59:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
60:
61:      REAL(8), INTENT(OUT) :: f(:, :)
62:
63:!-------------------------------------------------------------------
64:
65:      f = efv3d%f
66:
67:!-------------------------------------------------------------------
68:
69:      RETURN
70:
71:!#####################################################################
72:      END SUBROUTINE get_elemexforcevec3d_f
73:!#####################################################################
74:
75:
76:      ! Set density
77:!#####################################################################
78:      SUBROUTINE set_elemexforcevec3d_rho(efv3d, rho)
79:!#####################################################################
80:
81:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
82:
83:      REAL(8), INTENT(IN) :: rho(:)
84:
```

```
 85:!-----------------------------------------------------------------------
 86:
 87:      efv3d%rho = rho
 88:
 89:!-----------------------------------------------------------------------
 90:
 91:      RETURN
 92:
 93:!#######################################################################
 94:      END SUBROUTINE set_elemexforcevec3d_rho
 95:!#######################################################################
 96:
 97:
 98:      ! Get density
 99:!#######################################################################
100:      SUBROUTINE get_elemexforcevec3d_rho(efv3d, rho)
101:!#######################################################################
102:
103:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
104:
105:      REAL(8), INTENT(OUT) :: rho(:)
106:
107:!-----------------------------------------------------------------------
108:
109:      rho = efv3d%rho
110:
111:!-----------------------------------------------------------------------
112:
113:      RETURN
114:
115:!#######################################################################
116:      END SUBROUTINE get_elemexforcevec3d_rho
117:!#######################################################################
118:
119:
120:      ! Set gravitational acceleration
```

```
121:!########################################################################
122:      SUBROUTINE set_elemexforcevec3d_g(efv3d, g)
123:!########################################################################
124:
125:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
126:
127:      REAL(8), INTENT(IN) :: g
128:
129:!------------------------------------------------------------------------
130:
131:      efv3d%g = g
132:
133:!------------------------------------------------------------------------
134:
135:      RETURN
136:
137:!########################################################################
138:      END SUBROUTINE set_elemexforcevec3d_g
139:!########################################################################
140:
141:
142:      ! Get gravitational acceleration
143:!########################################################################
144:      SUBROUTINE get_elemexforcevec3d_g(efv3d, g)
145:!########################################################################
146:
147:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
148:
149:      REAL(8), INTENT(OUT) :: g
150:
151:!------------------------------------------------------------------------
152:
153:      g = efv3d%g
154:
155:!------------------------------------------------------------------------
156:
```

```
157:      RETURN
158:
159:!######################################################################
160:      END SUBROUTINE get_elemexforcevec3d_g
161:!######################################################################
162:
163:
164:!######################################################################
165:      SUBROUTINE init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
166:!######################################################################
167:
168:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
169:
170:      TYPE(struct_nodes3d), TARGET, INTENT(IN)        :: ns3d
171:      TYPE(struct_localelement3d), TARGET, INTENT(IN) :: le3d
172:      TYPE(struct_elements3d), TARGET, INTENT(IN)     :: es3d
173:
174:!----------------------------------------------------------------------
175:
176:      INTEGER :: le3d_nnodes
177:      INTEGER :: es3d_n
178:
179:!----------------------------------------------------------------------
180:
181:      efv3d%ns3d => ns3d
182:      efv3d%le3d => le3d
183:      efv3d%es3d => es3d
184:
185:!----------------------------------------------------------------------
186:
187:      CALL get_localelement3d_nnodes(efv3d%le3d, le3d_nnodes)
188:
189:      CALL get_elements3d_n(efv3d%es3d, es3d_n)
190:
191:!----------------------------------------------------------------------
192:
```

```
193:       ALLOCATE( efv3d%f(3*le3d_nnodes, es3d_n) )
194:
195:       efv3d%f = 0.0D0
196:
197:       !-------------------------------------------------------------
198:
199:       ALLOCATE( efv3d%rho(es3d_n) )
200:
201:       efv3d%rho = 0.0D0
202:
203:       efv3d%g = 0.0D0
204:
205:!---------------------------------------------------------------------
206:
207:       RETURN
208:
209:!#####################################################################
210:       END SUBROUTINE init_elemexforcevec3d
211:!#####################################################################
212:
213:
214:!#####################################################################
215:       SUBROUTINE cal_elemexforcevec3d(efv3d)
216:!#####################################################################
217:
218:       TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
219:
220:!---------------------------------------------------------------------
221:
222:!---------------------------------------------------------------------
223:
224:       RETURN
225:
226:!#####################################################################
227:       END SUBROUTINE cal_elemexforcevec3d
228:!#####################################################################
```

```
229:
230:
231:!######################################################################
232:      SUBROUTINE del_elemexforcevec3d(efv3d)
233:!######################################################################
234:
235:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
236:
237:!----------------------------------------------------------------------
238:
239:      NULLIFY( efv3d%ns3d )
240:      NULLIFY( efv3d%le3d )
241:      NULLIFY( efv3d%es3d )
242:
243:!----------------------------------------------------------------------
244:
245:      DEALLOCATE( efv3d%f )
246:
247:      !----------------------------------------------------------------
248:
249:      DEALLOCATE( efv3d%rho )
250:
251:      efv3d%g = 0.0D0
252:
253:!----------------------------------------------------------------------
254:
255:      RETURN
256:
257:!######################################################################
258:      END SUBROUTINE del_elemexforcevec3d
259:!######################################################################
260:
261:
262:!######################################################################
263:      END MODULE mod_elemexforcevec3d
```

第 4 節のプログラム：モジュール mod_fem3d の作成

３．有限要素法モジュール mod_fem3d.f90

```
 1:      MODULE mod_fem3d
 2:!########################################################################
 3:
 4:      USE mod_nodes3d
 5:      USE mod_localelement3d
 6:      USE mod_elements3d
 7:      USE mod_elemstiffmat3d
 8:      USE mod_elemexforcevec3d
 9:
10:!------------------------------------------------------------------------
11:
12:      IMPLICIT NONE
13:
14:!------------------------------------------------------------------------
15:
16:      TYPE :: struct_fem3d
17:
18:       !-----------------------------------------------------
19:
20:       PRIVATE
21:
22:       !-----------------------------------------------------
23:
24:       TYPE(struct_nodes3d), POINTER          :: ns3d  => NULL()
25:       TYPE(struct_localelement3d), POINTER   :: le3d  => NULL()
26:       TYPE(struct_elements3d), POINTER       :: es3d  => NULL()
27:       TYPE(struct_elemstiffmat3d), POINTER   :: esm3d => NULL()
28:       TYPE(struct_elemexforcevec3d), POINTER :: efv3d => NULL()
29:
30:       !-----------------------------------------------------
31:       !
```

```
32:       ! ndofs
33:       ! The total number of DOFs
34:       !
35:       ! k(:, :)
36:       ! Stiffness matrix
37:       !
38:       ! f(:)
39:       ! External force vector
40:       !
41:       !-------------------------------------------------------
42:       !
43:       ! nnodes_loaded
44:       ! The total number of nodes with an equivalent nodal force
45:       !
46:       ! id_loaded(:)
47:       ! Node no. with an equivalent nodal force
48:       !
49:       ! f_loaded(:, :)
50:       ! Equivalent nodal force
51:       !
52:       !-------------------------------------------------------
53:
54:       INTEGER :: ndofs
55:       INTEGER :: nnodes_loaded
56:       INTEGER, ALLOCATABLE :: id_loaded(:)
57:
58:       REAL(8), ALLOCATABLE :: k(:, :)
59:       REAL(8), ALLOCATABLE :: f(:)
60:       REAL(8), ALLOCATABLE :: f_loaded(:, :)
61:
62:       !-------------------------------------------------------
63:
64:     END TYPE struct_fem3d
65:
66:!---------------------------------------------------------------------
67:
```

```
68:       PRIVATE :: cal_fem3d_stiffmat
69:       PRIVATE :: cal_fem3d_rhs
70:       PRIVATE :: cal_fem3d_dirichletbc
71:       PRIVATE :: cal_fem3d_linearsolver
72:
73:!-------------------------------------------------------------------
74:
75:       CONTAINS
76:
77:
78:       ! Get the total number of DOFs
79:!#####################################################################
80:       SUBROUTINE get_fem3d_ndofs(fem3d, ndofs)
81:!#####################################################################
82:
83:       TYPE(struct_fem3d), INTENT(IN) :: fem3d
84:
85:       INTEGER, INTENT(OUT) :: ndofs
86:
87:!-------------------------------------------------------------------
88:
89:       ndofs = fem3d%ndofs
90:
91:!-------------------------------------------------------------------
92:
93:       RETURN
94:
95:!#####################################################################
96:       END SUBROUTINE get_fem3d_ndofs
97:!#####################################################################
98:
99:
100:      ! Get stiffness matrix
101:!#####################################################################
102:      SUBROUTINE get_fem3d_k(fem3d, k)
103:!#####################################################################
```

```
104:
105:      TYPE(struct_fem3d), INTENT(IN) :: fem3d
106:
107:      INTEGER, INTENT(OUT) :: k(:, :)
108:
109:!-----------------------------------------------------------------------
110:
111:      k = fem3d%k
112:
113:!-----------------------------------------------------------------------
114:
115:      RETURN
116:
117:!#######################################################################
118:      END SUBROUTINE get_fem3d_k
119:!#######################################################################
120:
121:
122:      ! Get external force vector
123:!#######################################################################
124:      SUBROUTINE get_fem3d_f(fem3d, f)
125:!#######################################################################
126:
127:      TYPE(struct_fem3d), INTENT(IN) :: fem3d
128:
129:      INTEGER, INTENT(OUT) :: f(:)
130:
131:!-----------------------------------------------------------------------
132:
133:      f = fem3d%f
134:
135:!-----------------------------------------------------------------------
136:
137:      RETURN
138:
139:!#######################################################################
```

```
140:      END SUBROUTINE get_fem3d_f
141:!######################################################################
142:
143:
144:      ! Set equivalent nodal force
145:!######################################################################
146:      SUBROUTINE set_fem3d_f_loaded(fem3d, id_loaded, f_loaded)
147:!######################################################################
148:
149:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
150:
151:      INTEGER, INTENT(IN) :: id_loaded(:)
152:      REAL(8), INTENT(IN) :: f_loaded(:, :)
153:
154:!----------------------------------------------------------------------
155:
156:      fem3d%id_loaded = id_loaded
157:      fem3d%f_loaded  = f_loaded
158:
159:!----------------------------------------------------------------------
160:
161:      RETURN
162:
163:!######################################################################
164:      END SUBROUTINE set_fem3d_f_loaded
165:!######################################################################
166:
167:
168:      ! Get equivalent nodal force
169:!######################################################################
170:      SUBROUTINE get_fem3d_f_loaded(fem3d, id_loaded, f_loaded)
171:!######################################################################
172:
173:      TYPE(struct_fem3d), INTENT(IN) :: fem3d
174:
175:      INTEGER, INTENT(OUT) :: id_loaded(:)
```

```
176:      REAL(8), INTENT(OUT) :: f_loaded(:, :)
177:
178:!-----------------------------------------------------------------
179:
180:      id_loaded = fem3d%id_loaded
181:      f_loaded  = fem3d%f_loaded
182:
183:!-----------------------------------------------------------------
184:
185:      RETURN
186:
187:!#################################################################
188:      END SUBROUTINE get_fem3d_f_loaded
189:!#################################################################
190:
191:
192:      ! Initialize fem3d
193:!#################################################################
194:      SUBROUTINE init_fem3d                                &
195:                 (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
196:                  nnodes_loaded)
197:!#################################################################
198:
199:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
200:
201:      TYPE(struct_nodes3d), TARGET, INTENT(IN)          :: ns3d
202:      TYPE(struct_localelement3d), TARGET, INTENT(IN)   :: le3d
203:      TYPE(struct_elements3d), TARGET, INTENT(IN)       :: es3d
204:      TYPE(struct_elemstiffmat3d), TARGET, INTENT(IN)   :: esm3d
205:      TYPE(struct_elemexforcevec3d), TARGET, INTENT(IN) :: efv3d
206:
207:      INTEGER, INTENT(IN) :: nnodes_loaded
208:
209:!-----------------------------------------------------------------
210:
211:      INTEGER :: ns3d_n
```

```
212:
213:!-----------------------------------------------------------------------
214:
215:      fem3d%ns3d => ns3d
216:      fem3d%le3d => le3d
217:      fem3d%es3d => es3d
218:      fem3d%esm3d => esm3d
219:      fem3d%efv3d => efv3d
220:
221:!-----------------------------------------------------------------------
222:
223:      CALL get_nodes3d_n(fem3d%ns3d, ns3d_n)
224:
225:!-----------------------------------------------------------------------
226:
227:      fem3d%ndofs = 3*ns3d_n
228:
229:      !-------------------------------------------------------------
230:
231:      ALLOCATE( fem3d%k(fem3d%ndofs, fem3d%ndofs) )
232:
233:      fem3d%k = 0.0D0
234:
235:      ALLOCATE( fem3d%f(fem3d%ndofs) )
236:
237:      fem3d%f = 0.0D0
238:
239:      !-------------------------------------------------------------
240:
241:      fem3d%nnodes_loaded = nnodes_loaded
242:
243:      ALLOCATE( fem3d%id_loaded(nnodes_loaded) )
244:
245:      fem3d%id_loaded = 0
246:
247:      ALLOCATE( fem3d%f_loaded(3, nnodes_loaded) )
```

```
248:
249:      fem3d%f_loaded = 0.0D0
250:
251:!---------------------------------------------------------------------
252:
253:      RETURN
254:
255:!#####################################################################
256:      END SUBROUTINE init_fem3d
257:!#####################################################################
258:
259:
260:      ! Calculate fem3d
261:!#####################################################################
262:      SUBROUTINE cal_fem3d(fem3d)
263:!#####################################################################
264:
265:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
266:
267:!---------------------------------------------------------------------
268:
269:      ! Stiffness matrix
270:      CALL cal_fem3d_stiffmat(fem3d)
271:
272:!---------------------------------------------------------------------
273:
274:      ! RHS vector
275:      CALL cal_fem3d_rhs(fem3d)
276:
277:!---------------------------------------------------------------------
278:
279:      ! Dirichlet boundary condition
280:      CALL cal_fem3d_dirichletbc(fem3d)
281:
282:!---------------------------------------------------------------------
283:
```

```
284:      ! Linear solver
285:      CALL cal_fem3d_linearsolver(fem3d)
286:
287:!----------------------------------------------------------------------
288:
289:      ! Strain and stress
290:      CALL cal_elemstiffmat3d(fem3d%esm3d)
291:
292:!----------------------------------------------------------------------
293:
294:      RETURN
295:
296:!######################################################################
297:      END SUBROUTINE cal_fem3d
298:!######################################################################
299:
300:
301:      ! Calculate fem3d (stiffness matrix)
302:!######################################################################
303:      SUBROUTINE cal_fem3d_stiffmat(fem3d)
304:!######################################################################
305:
306:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
307:
308:!----------------------------------------------------------------------
309:
310:      INTEGER :: le3d_nnodes
311:      INTEGER :: es3d_n
312:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
313:      INTEGER :: i, j
314:      INTEGER :: id, jd
315:      INTEGER :: na, nb
316:      INTEGER :: ie
317:      INTEGER :: idof, jdof
318:      INTEGER :: isize, jsize
319:
```

```
320:      REAL(8), ALLOCATABLE :: esm3d_k(:, :, :)
321:      REAL(8), ALLOCATABLE :: efv3d_f(:, :)
322:
323:!-----------------------------------------------------------------
324:
325:      ! Element stiffness matrix
326:      CALL cal_elemstiffmat3d(fem3d%esm3d)
327:
328:!-----------------------------------------------------------------
329:
330:      CALL get_localelement3d_nnodes(fem3d%le3d, le3d_nnodes)
331:
332:      CALL get_elements3d_n(fem3d%es3d, es3d_n)
333:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
334:      CALL get_elements3d_connectivity(fem3d%es3d, es3d_connectivity)
335:
336:      ALLOCATE( esm3d_k(3*le3d_nnodes, 3*le3d_nnodes, es3d_n) )
337:      CALL get_elemstiffmat3d_k(fem3d%esm3d, esm3d_k)
338:
339:      ! Stiffness matrix
340:
341:      fem3d%k = 0.0D0
342:
343:      DO ie = 1, es3d_n
344:
345:       DO na = 1, le3d_nnodes
346:
347:        id = es3d_connectivity(na, ie)
348:
349:        DO i = 1, 3
350:
351:         idof  = 3*(id-1)+i
352:         isize = 3*(na-1)+i
353:
354:         DO nb = 1, le3d_nnodes
355:
```

```
356:          jd = es3d_connectivity(nb, ie)
357:
358:          DO j = 1, 3
359:
360:           jdof  = 3*(jd-1)+j
361:           jsize = 3*(nb-1)+j
362:
363:           fem3d%k(idof, jdof)                              &
364:           = fem3d%k(idof, jdof)+esm3d_k(isize, jsize, ie)
365:
366:          END DO
367:
368:         END DO
369:
370:        END DO
371:
372:       END DO
373:
374:      END DO
375:
376:!-----------------------------------------------------------------
377:
378:      DEALLOCATE( es3d_connectivity )
379:
380:      DEALLOCATE( esm3d_k )
381:
382:!-----------------------------------------------------------------
383:
384:      RETURN
385:
386:!#################################################################
387:      END SUBROUTINE cal_fem3d_stiffmat
388:!#################################################################
389:
390:
391:      ! Calculate fem3d (RHS vector)
```

```
392:!######################################################################
393:      SUBROUTINE cal_fem3d_rhs(fem3d)
394:!######################################################################
395:
396:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
397:
398:!----------------------------------------------------------------------
399:
400:      INTEGER :: le3d_nnodes
401:      INTEGER :: es3d_n
402:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
403:      INTEGER :: i
404:      INTEGER :: id
405:      INTEGER :: id_l
406:      INTEGER :: na
407:      INTEGER :: ie
408:      INTEGER :: idof
409:      INTEGER :: isize
410:
411:      REAL(8), ALLOCATABLE :: efv3d_f(:, :)
412:
413:!----------------------------------------------------------------------
414:
415:      ! Element external force vector
416:      CALL cal_elemexforcevec3d(fem3d%efv3d)
417:
418:!----------------------------------------------------------------------
419:
420:      CALL get_localelement3d_nnodes(fem3d%le3d, le3d_nnodes)
421:
422:      CALL get_elements3d_n(fem3d%es3d, es3d_n)
423:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
424:      CALL get_elements3d_connectivity(fem3d%es3d, es3d_connectivity)
425:
426:      ALLOCATE( efv3d_f(3*le3d_nnodes, es3d_n) )
427:      CALL get_elemexforcevec3d_f(fem3d%efv3d, efv3d_f)
```

```
428:
429:      !------------------------------------------------------------
430:
431:      ! External force vector
432:
433:      fem3d%f = 0.0D0
434:
435:      DO id_l = 1, fem3d%nnodes_loaded
436:
437:       id = fem3d%id_loaded(id_l)
438:
439:       DO i = 1, 3
440:
441:        idof = 3*(id-1)+i
442:
443:        fem3d%f(idof) = fem3d%f_loaded(i, id_l)
444:
445:       END DO
446:
447:      END DO
448:
449:      DO ie = 1, es3d_n
450:
451:       DO na = 1, le3d_nnodes
452:
453:        id = es3d_connectivity(na, ie)
454:
455:        DO i = 1, 3
456:
457:         idof  = 3*(id-1)+i
458:         isize = 3*(na-1)+i
459:
460:         fem3d%f(idof) = fem3d%f(idof)+efv3d_f(isize, ie)
461:
462:        END DO
463:
```

```
464:       END DO
465:
466:      END DO
467:
468:!-----------------------------------------------------------------
469:
470:      DEALLOCATE( es3d_connectivity )
471:
472:      DEALLOCATE( efv3d_f )
473:
474:!-----------------------------------------------------------------
475:
476:      RETURN
477:
478:!#################################################################
479:      END SUBROUTINE cal_fem3d_rhs
480:!#################################################################
481:
482:
483:      ! Calculate fem3d (Dirichlet boundary condition)
484:!#################################################################
485:      SUBROUTINE cal_fem3d_dirichletbc(fem3d)
486:!#################################################################
487:
488:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
489:
490:!-----------------------------------------------------------------
491:
492:      INTEGER :: ns3d_n
493:      INTEGER, ALLOCATABLE :: ns3d_bc(:)
494:      INTEGER :: idof, jdof
495:      INTEGER :: idof_g
496:      INTEGER :: ndofs_given
497:      INTEGER, ALLOCATABLE :: idof_given(:)
498:
499:      REAL(8), ALLOCATABLE :: ns3d_u(:)
```

```
500:      REAL(8) :: u_given
501:
502:!-----------------------------------------------------------------
503:
504:      CALL get_nodes3d_n(fem3d%ns3d, ns3d_n)
505:      ALLOCATE( ns3d_u(fem3d%ndofs) )
506:      CALL get_nodes3d_u(fem3d%ns3d, ns3d_u)
507:      ALLOCATE( ns3d_bc(fem3d%ndofs) )
508:      CALL get_nodes3d_bc(fem3d%ns3d, ns3d_bc)
509:
510:!-----------------------------------------------------------------
511:
512:      idof_g = 0
513:
514:      DO idof = 1, fem3d%ndofs
515:
516:       IF( ns3d_bc(idof) .EQ. 1 ) THEN
517:
518:        idof_g = idof_g+1
519:
520:       END IF
521:
522:      END DO
523:
524:      ndofs_given = idof_g
525:
526:      !---------------------------------------------------------------
527:
528:      ALLOCATE( idof_given(ndofs_given) )
529:
530:      !---------------------------------------------------------------
531:
532:      idof_g = 0
533:
534:      DO idof = 1, fem3d%ndofs
535:
```

```
536:       IF( ns3d_bc(idof) .EQ. 1 ) THEN
537:
538:        idof_g = idof_g+1
539:
540:        idof_given(idof_g) = idof
541:
542:       END IF
543:
544:      END DO
545:
546:!-----------------------------------------------------------------------
547:
548:      ! Dirichlet boundary conditions
549:
550:      DO idof_g = 1, ndofs_given
551:
552:       jdof = idof_given(idof_g)
553:
554:       DO idof = 1, fem3d%ndofs
555:
556:        fem3d%f(idof) = fem3d%f(idof)-fem3d%k(idof, jdof)*ns3d_u(jdof)
557:
558:       END DO
559:
560:      END DO
561:
562:      DO idof_g = 1, ndofs_given
563:
564:       idof = idof_given(idof_g)
565:
566:       fem3d%f(idof) = ns3d_u(idof)
567:
568:      END DO
569:
570:      DO idof_g = 1, ndofs_given
571:
```

```
572:       jdof = idof_given(idof_g)
573:
574:       DO idof = 1, fem3d%ndofs
575:
576:        fem3d%k(idof, jdof) = 0.0D0
577:        fem3d%k(jdof, idof) = 0.0D0
578:
579:       END DO
580:
581:       fem3d%k(jdof, jdof) = 1.0D0
582:
583:      END DO
584:
585:!-----------------------------------------------------------------
586:
587:      DEALLOCATE( ns3d_u )
588:      DEALLOCATE( ns3d_bc )
589:
590:      DEALLOCATE( idof_given )
591:
592:!-----------------------------------------------------------------
593:
594:      RETURN
595:
596:!#################################################################
597:      END SUBROUTINE cal_fem3d_dirichletbc
598:!#################################################################
599:
600:
601:      ! Calculate fem3d (linear solver)
602:!#################################################################
603:      SUBROUTINE cal_fem3d_linearsolver(fem3d)
604:!#################################################################
605:
606:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
607:
```

```
608:!-------------------------------------------------------------------
609:
610:      INTEGER :: ipiv(fem3d%ndofs)
611:      INTEGER :: info
612:
613:!-------------------------------------------------------------------
614:
615:      ! Linear solver
616:
617:      ipiv = 0
618:
619:      CALL DGESV                                          &
620:           (fem3d%ndofs, 1, fem3d%k, fem3d%ndofs, ipiv, &
621:            fem3d%f, fem3d%ndofs, info)
622:
623:      CALL set_nodes3d_u(fem3d%ns3d, fem3d%f)
624:
625:!-------------------------------------------------------------------
626:
627:      RETURN
628:
629:!#####################################################################
630:      END SUBROUTINE cal_fem3d_linearsolver
631:!#####################################################################
632:
633:
634:      ! Delete fem3d
635:!#####################################################################
636:      SUBROUTINE del_fem3d(fem3d)
637:!#####################################################################
638:
639:      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d
640:
641:!-------------------------------------------------------------------
642:
643:      IF( fem3d%ndofs .EQ. 0 ) THEN
```

```
644:
645:       RETURN
646:
647:      END IF
648:
649:!-----------------------------------------------------------------------
650:
651:      NULLIFY( fem3d%ns3d )
652:      NULLIFY( fem3d%le3d )
653:      NULLIFY( fem3d%es3d )
654:      NULLIFY( fem3d%esm3d )
655:      NULLIFY( fem3d%efv3d )
656:
657:!-----------------------------------------------------------------------
658:
659:      fem3d%ndofs = 0
660:
661:      !-------------------------------------------------------------
662:
663:      DEALLOCATE( fem3d%k )
664:      DEALLOCATE( fem3d%f )
665:
666:!-----------------------------------------------------------------------
667:
668:      IF( fem3d%nnodes_loaded .EQ. 0 ) THEN
669:
670:       RETURN
671:
672:      END IF
673:
674:!-----------------------------------------------------------------------
675:
676:      fem3d%nnodes_loaded = 0
677:
678:      !-------------------------------------------------------------
679:
```

```
680:      DEALLOCATE( fem3d%id_loaded )
681:      DEALLOCATE( fem3d%f_loaded )
682:
683:!-----------------------------------------------------------------
684:
685:      RETURN
686:
687:!#################################################################
688:      END SUBROUTINE del_fem3d
689:!#################################################################
690:
691:
692:!#################################################################
693:      END MODULE mod_fem3d
```

第 5 節のプログラム：メッシング用のアプリケーションモジュール mod_appli の修正

４．アプリケーションモジュール mod_appli.f90

```
 1:      MODULE mod_appli
 2:!#################################################################
 3:
 4:      USE mod_nodes3d
 5:      USE mod_localelement3d
 6:      USE mod_elements3d
 7:      USE mod_elemstiffmat3d
 8:      USE mod_elemexforcevec3d
 9:      USE mod_fem3d
10:      USE mod_rectmesher3d
11:
12:!-----------------------------------------------------------------
13:
14:      IMPLICIT NONE
15:
```

```
16:!-------------------------------------------------------------------
17:
18:      TYPE(struct_nodes3d), POINTER          :: ns3d
19:      TYPE(struct_localelement3d), POINTER   :: le3d
20:      TYPE(struct_elements3d), POINTER       :: es3d
21:      TYPE(struct_elemstiffmat3d), POINTER   :: esm3d
22:      TYPE(struct_elemexforcevec3d), POINTER :: efv3d
23:      TYPE(struct_fem3d), POINTER            :: fem3d
24:      TYPE(struct_rectmesher3d), POINTER     :: rm3d
25:
26:      !-------------------------------------------------------------
27:      !
28:      ! Problem number
29:      ! prob
30:      !
31:      !-------------------------------------------------------------
32:
33:      INTEGER :: prob
34:
35:!-------------------------------------------------------------------
36:
37:      CONTAINS
38:
39:
40:      ! Start appli
41:!####################################################################
42:      SUBROUTINE start_appli()
43:!####################################################################
44:
45:      INTEGER :: ns3d_n
46:      INTEGER :: le3d_nboundaries
47:      INTEGER :: le3d_nnodes
48:      INTEGER :: le3d_nqps
49:      INTEGER :: es3d_n
50:      INTEGER :: rm3d_n_x(3)
51:
```

```
52:       REAL(8) :: rm3d_x_start(3)
53:       REAL(8) :: rm3d_x_end(3)
54:       REAL(8) :: e
55:       REAL(8) :: nu
56:       REAL(8) :: rho
57:       REAL(8) :: g
58:
59:       CHARACTER(1) :: dataname
60:
61:!-----------------------------------------------------------------
62:
63:       ALLOCATE( ns3d )
64:       ALLOCATE( le3d )
65:       ALLOCATE( es3d )
66:       ALLOCATE( esm3d )
67:       ALLOCATE( efv3d )
68:       ALLOCATE( fem3d )
69:       ALLOCATE( rm3d )
70:
71:!-----------------------------------------------------------------
72:
73:       OPEN(13, FILE = 'param_meshing.dat')
74:
75:       READ(13, *) dataname
76:       READ(13, *) rm3d_n_x(1), rm3d_n_x(2), rm3d_n_x(3)
77:       READ(13, *) dataname
78:       READ(13, *) rm3d_x_start(1), rm3d_x_start(2), rm3d_x_start(3)
79:       READ(13, *) dataname
80:       READ(13, *) rm3d_x_end(1), rm3d_x_end(2), rm3d_x_end(3)
81:       READ(13, *) dataname
82:       READ(13, *) prob
83:       READ(13, *) dataname
84:       READ(13, *) e
85:       READ(13, *) dataname
86:       READ(13, *) nu
87:       READ(13, *) dataname
```

```
 88:      READ(13, *) rho
 89:      READ(13, *) dataname
 90:      READ(13, *) g
 91:      READ(13, *) dataname
 92:
 93:      CLOSE(13)
 94:
 95:!-----------------------------------------------------------------
 96:
 97:      OPEN(13, FILE = 'param_fea.dat')
 98:
 99:      WRITE(13, '(A)') '!ANALYSIS_TYPE'
100:      WRITE(13, '(A)') 'STATIC_ANALYSIS'
101:      WRITE(13, '(A)') "!YOUNG'S_MODULUS"
102:      WRITE(13, '(E17.8)') e
103:      WRITE(13, '(A)') "!POISSON'S_RATIO"
104:      WRITE(13, '(E17.8)') nu
105:      WRITE(13,  '(A)') '!DENSITY'
106:      WRITE(13, '(E17.8)') rho
107:      WRITE(13,  '(A)') '!GRAVITATIONAL_ACCELERATION'
108:      WRITE(13, '(E17.8)') g
109:
110:!-----------------------------------------------------------------
111:
112:      CALL init_rectmesher3d                    &
113:           (rm3d, ns3d, le3d, es3d,             &
114:            rm3d_n_x, rm3d_x_start, rm3d_x_end)
115:
116:!-----------------------------------------------------------------
117:
118:      CALL get_nodes3d_n(ns3d, ns3d_n)
119:
120:      CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
121:      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
122:      CALL get_localelement3d_nqps(le3d, le3d_nqps)
123:
```

```
124:      CALL get_elements3d_n(es3d, es3d_n)
125:
126:      !-------------------------------------------------------------
127:
128:      CALL init_nodes3d(ns3d, ns3d_n)
129:
130:      CALL init_localelement3d                              &
131:           (le3d, le3d_nboundaries, le3d_nnodes, le3d_nqps)
132:
133:      CALL init_elements3d(es3d, ns3d, le3d, es3d_n)
134:
135:!-----------------------------------------------------------------------
136:
137:      RETURN
138:
139:!#######################################################################
140:      END SUBROUTINE start_appli
141:!#######################################################################
142:
143:
144:      ! Run appli
145:!#######################################################################
146:      SUBROUTINE run_appli()
147:!#######################################################################
148:
149:      INTEGER :: ns3d_n
150:      INTEGER, ALLOCATABLE :: ns3d_bc(:)
151:      INTEGER :: le3d_nboundaries
152:      INTEGER :: le3d_nnodes
153:      INTEGER :: le3d_nnodes_boundary
154:      INTEGER, ALLOCATABLE :: le3d_table_na(:, :)
155:      INTEGER :: es3d_n
156:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
157:      INTEGER :: es3d_ie_max_volume
158:      INTEGER :: es3d_ie_min_volume
159:      INTEGER :: fem3d_ndofs
```

```
160:      INTEGER :: fem3d_nnodes_loaded
161:      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
162:      INTEGER :: i
163:      INTEGER :: id
164:      INTEGER :: id_l
165:      INTEGER :: na
166:      INTEGER :: ie
167:      INTEGER :: idof
168:
169:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
170:      REAL(8), ALLOCATABLE :: ns3d_u(:)
171:      REAL(8), ALLOCATABLE :: es3d_volume(:)
172:      REAL(8) :: es3d_max_volume
173:      REAL(8) :: es3d_min_volume
174:      REAL(8) :: es3d_sum_volume
175:      REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
176:      REAL(8) :: rm3d_x_start(3)
177:      REAL(8) :: rm3d_x_end(3)
178:      REAL(8) :: rm3d_x_center(3)
179:      REAL(8) :: rm3d_length_x(3)
180:      REAL(8), ALLOCATABLE :: x_local(:, :)
181:
182:!-----------------------------------------------------------------------
183:
184:      CALL cal_rectmesher3d(rm3d)
185:
186:      CALL cal_elements3d(es3d)
187:
188:!-----------------------------------------------------------------------
189:
190:      CALL get_nodes3d_n(ns3d, ns3d_n)
191:
192:      ALLOCATE( ns3d_x(3, ns3d_n) )
193:      CALL get_nodes3d_x(ns3d, ns3d_x)
194:      ALLOCATE( ns3d_u(3*ns3d_n) )
195:      ns3d_u = 0.0D0
```

```
196:      ALLOCATE( ns3d_bc(3*ns3d_n) )
197:      ns3d_bc = 0
198:
199:      CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
200:      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
201:      CALL get_localelement3d_nnodes_boundary(le3d, le3d_nnodes_boundary)
202:      ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
203:      CALL get_localelement3d_table_na(le3d, le3d_table_na)
204:
205:      CALL get_elements3d_n(es3d, es3d_n)
206:      ALLOCATE( es3d_volume(es3d_n) )
207:      CALL get_elements3d_volume                &
208:           (es3d, es3d_volume,                  &
209:            es3d_max_volume, es3d_ie_max_volume, &
210:            es3d_min_volume, es3d_ie_min_volume, &
211:            es3d_sum_volume)
212:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
213:      CALL get_elements3d_connectivity(es3d, es3d_connectivity)
214:      CALL get_rectmesher3d_x_start_x_end   &
215:           (rm3d, rm3d_x_start, rm3d_x_end, &
216:            rm3d_x_center, rm3d_length_x)
217:
218:      ALLOCATE( x_local(3, le3d_nnodes_boundary) )
219:
220:!-----------------------------------------------------------------------
221:
222:      ! Tensile deformation
223:      IF( prob .EQ. 1 ) THEN
224:
225:       !-------------------------------------------------------
226:
227:       id_l = 0
228:
229:       DO id = 1, ns3d_n
230:
231:        IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
```

```
232:
233:         id_l = id_l+1
234:
235:        END IF
236:
237:       END DO
238:
239:       fem3d_nnodes_loaded = id_l
240:
241:       !-------------------------------------------------------
242:
243:      END IF
244:
245:      !-------------------------------------------------------------
246:
247:      CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
248:      CALL init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
249:      CALL init_fem3d                              &
250:           (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
251:            fem3d_nnodes_loaded)
252:
253:      CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
254:      ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
255:      fem3d_id_loaded = 0
256:      ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
257:      fem3d_f_loaded = 0.0D0
258:
259:      !-------------------------------------------------------------
260:
261:      ! Tensile deformation
262:      IF( prob .EQ. 1 ) THEN
263:
264:       !-------------------------------------------------------
265:
266:       DO id = 1, ns3d_n
267:
```

```
268:        IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN
269:
270:         idof = 3*(id-1)+1
271:
272:         ns3d_u(idof) = 0.0D0
273:         ns3d_bc(idof) = 1
274:
275:         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
276:
277:          idof = 3*(id-1)+2
278:
279:          ns3d_u(idof) = 0.0D0
280:          ns3d_bc(idof) = 1
281:
282:         END IF
283:
284:         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
285:
286:          idof = 3*(id-1)+3
287:
288:          ns3d_u(idof) = 0.0D0
289:          ns3d_bc(idof) = 1
290:
291:         END IF
292:
293:        END IF
294:
295:       END DO
296:
297:       !-------------------------------------------------------
298:
299:      END IF
300:
301:      !-------------------------------------------------------------
302:
303:      id_l = 0
```

```
304:
305:      ! Tensile deformation
306:      IF( prob .EQ. 1 ) THEN
307:
308:       !-----------------------------------------------------
309:
310:       DO id = 1, ns3d_n
311:
312:        IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
313:
314:         id_l = id_l+1
315:
316:         fem3d_id_loaded(id_l) = id
317:         fem3d_f_loaded(1, id_l) = 2.5D6
318:
319:         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
320:
321:          fem3d_f_loaded(1, id_l) = 5.0D6
322:
323:         END IF
324:
325:         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
326:
327:          fem3d_f_loaded(1, id_l) = 5.0D6
328:
329:         END IF
330:
331:         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
332:
333:          IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
334:
335:           fem3d_f_loaded(1, id_l) = 1.0D7
336:
337:          END IF
338:
339:         END IF
```

```
340:
341:        END IF
342:
343:       END DO
344:
345:       !-------------------------------------------------------
346:
347:      END IF
348:
349:      CALL set_fem3d_f_loaded                          &
350:            (fem3d, fem3d_id_loaded, fem3d_f_loaded)
351:
352:!---------------------------------------------------------------------
353:
354:      OPEN(10, FILE = 'mesh.dat')
355:
356:      WRITE(10, '(A)') '!NODE'
357:
358:      DO id = 1, ns3d_n
359:
360:       WRITE( 10,'( I8, 3(A, E17.8) )' )            &
361:              id, ( ',', ns3d_x(i, id), i = 1, 3  )
362:
363:      END DO
364:
365:      WRITE( 10, '(A, 3(A, I3) )' )                 &
366:             '!ELEMENT', ', ', le3d_nboundaries,    &
367:                         ', ', le3d_nnodes, ', ', 2
368:
369:
370:      DO ie = 1, es3d_n
371:
372:       WRITE( 10, '( I8, 27(A, I8) )' )             &
373:              ie, ( ',', es3d_connectivity(na, ie), &
374:                    na = 1, le3d_nnodes )
375:
```

```
376:      END DO
377:
378:      WRITE(10,'(A)') '!END'
379:
380:      CLOSE(10)
381:
382:!-----------------------------------------------------------------
383:
384:      OPEN(11, FILE = 'ic.dat')
385:
386:      WRITE(11, '(A)') '!DISPLACEMENT'
387:
388:      DO id = 1, ns3d_n
389:
390:       WRITE( 11, '(I8, 3(A, E17.8) )' )                    &
391:              id, ( ', ', ns3d_u( 3*(id-1)+i ), i = 1, 3 )
392:
393:      END DO
394:
395:      WRITE(11, '(A)') '!END'
396:
397:      CLOSE(11)
398:
399:!-----------------------------------------------------------------
400:
401:      OPEN(12, FILE = 'bc.dat')
402:
403:      WRITE(12, '(A)') '!DISPLACEMENT'
404:
405:      DO id = 1, ns3d_n
406:
407:       WRITE( 12, '(I8, 3(A, I8) )' )                       &
408:              id, ( ', ', ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
409:
410:      END DO
411:
```

```
412:      WRITE(12, '(A)') '!END'
413:
414:      CLOSE(12)
415:
416:!-----------------------------------------------------------------
417:
418:      WRITE(13, '(A)') '!F_LOADED'
419:
420:      DO id_l = 1, fem3d_nnodes_loaded
421:
422:       WRITE( 13, '(I8, 3(A, E17.8) )' )                  &
423:              fem3d_id_loaded(id_l),                      &
424:              ( ', ', fem3d_f_loaded(i, id_l), i = 1, 3 )
425:
426:      END DO
427:
428:      WRITE(13, '(A)') '!END'
429:
430:      CLOSE(13)
431:
432:!-----------------------------------------------------------------
433:
434:      OPEN(14, FILE = 'mesh.inp')
435:
436:      WRITE(14, '( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
437:
438:      DO id = 1, ns3d_n
439:
440:       WRITE( 14, '( (I8, 1X), 3(E17.8, 1X) )' ) &
441:              id, ( ns3d_x(i, id), i = 1, 3 )
442:
443:      END DO
444:
445:      DO ie = 1, es3d_n
446:
447:       WRITE( 14, '( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )' )        &
```

```
448:              ie, 1, '  hex',                                    &
449:              ( es3d_connectivity(na, ie), na = 1, le3d_nnodes )
450:
451:      END DO
452:
453:      WRITE(14, '( 4(I8, 1X) )') 1, 3
454:      WRITE(14, '(A)') 'DISPLACEMENT, m'
455:
456:      DO id = 1, ns3d_n
457:
458:       WRITE( 14, '( (I8, 1X), 3(E17.8, 1X) )' )     &
459:              id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
460:
461:      END DO
462:
463:      WRITE(14, '( 14I8 )') 1, 1
464:      WRITE(14, '( (A, 1X) )') 'VOLUME, m3'
465:
466:      DO ie = 1, es3d_n
467:
468:       WRITE( 14, '( (I8, 1X), (E17.8, 1X) )' ) &
469:              ie, es3d_volume(ie)
470:
471:      END DO
472:
473:      CLOSE(14)
474:
475:!-----------------------------------------------------------------
476:
477:      DEALLOCATE( ns3d_x )
478:      DEALLOCATE( ns3d_u )
479:
480:      DEALLOCATE( le3d_table_na )
481:
482:      DEALLOCATE( es3d_volume )
483:      DEALLOCATE( es3d_connectivity )
```

```
484:
485:      DEALLOCATE( fem3d_id_loaded )
486:      DEALLOCATE( fem3d_f_loaded )
487:
488:      DEALLOCATE( x_local )
489:
490:!-------------------------------------------------------------------
491:
492:      RETURN
493:
494:!#####################################################################
495:      END SUBROUTINE run_appli
496:!#####################################################################
497:
498:
499:!#####################################################################
500:      SUBROUTINE finish_appli()
501:!#####################################################################
502:
503:      CALL del_nodes3d(ns3d)
504:      CALL del_localelement3d(le3d)
505:      CALL del_elements3d(es3d)
506:      CALL del_elemstiffmat3d(esm3d)
507:      CALL del_elemexforcevec3d(efv3d)
508:      CALL del_fem3d(fem3d)
509:      CALL del_rectmesher3d(rm3d)
510:
511:!-------------------------------------------------------------------
512:
513:      DEALLOCATE( ns3d )
514:      DEALLOCATE( le3d )
515:      DEALLOCATE( es3d )
516:      DEALLOCATE( esm3d )
517:      DEALLOCATE( efv3d )
518:      DEALLOCATE( fem3d )
519:      DEALLOCATE( rm3d )
```

```
520:
521:!------------------------------------------------------------------
522:
523:      RETURN
524:
525:!######################################################################
526:      END SUBROUTINE finish_appli
527:!######################################################################
528:
529:
530:!######################################################################
531:      END MODULE mod_appli
```

第 6 節のプログラム:有限要素解析用のアプリケーションモジュール mod_appli の作成

５．アプリケーションモジュール mod_appli.f90

```
 1:      MODULE mod_appli
 2:!######################################################################
 3:
 4:      USE mod_nodes3d
 5:      USE mod_localelement3d
 6:      USE mod_elements3d
 7:      USE mod_elemstiffmat3d
 8:      USE mod_elemexforcevec3d
 9:      USE mod_fem3d
10:
11:!------------------------------------------------------------------
12:
13:      IMPLICIT NONE
14:
15:!------------------------------------------------------------------
16:
17:      TYPE(struct_nodes3d), POINTER          :: ns3d
```

```
18:      TYPE(struct_localelement3d), POINTER   :: le3d
19:      TYPE(struct_elements3d), POINTER       :: es3d
20:      TYPE(struct_elemstiffmat3d), POINTER   :: esm3d
21:      TYPE(struct_elemexforcevec3d), POINTER :: efv3d
22:      TYPE(struct_fem3d), POINTER            :: fem3d
23:
24:!-----------------------------------------------------------------
25:
26:      CONTAINS
27:
28:
29:      ! Start appli
30:!######################################################################
31:      SUBROUTINE start_appli()
32:!######################################################################
33:
34:      INTEGER :: ns3d_n
35:      INTEGER, ALLOCATABLE :: ns3d_bc(:)
36:      INTEGER :: le3d_nboundaries
37:      INTEGER :: le3d_nnodes
38:      INTEGER :: le3d_nqps
39:      INTEGER :: es3d_n
40:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
41:      INTEGER :: fem3d_nnodes_loaded
42:      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
43:      INTEGER :: fem3d_ndofs
44:      INTEGER :: i
45:      INTEGER :: id
46:      INTEGER :: na
47:      INTEGER :: ie
48:      INTEGER :: ib
49:      INTEGER :: idof_g
50:      INTEGER :: idof_l
51:      INTEGER :: number
52:
53:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
```

```
54:      REAL(8), ALLOCATABLE :: ns3d_u(:)
55:      REAL(8), ALLOCATABLE :: esm3d_e(:)
56:      REAL(8), ALLOCATABLE :: esm3d_nu(:)
57:      REAL(8), ALLOCATABLE :: efv3d_rho(:)
58:      REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
59:      REAL(8) :: e
60:      REAL(8) :: nu
61:      REAL(8) :: rho
62:      REAL(8) :: g
63:
64:      CHARACTER(1) :: dataname
65:
66:!-----------------------------------------------------------------------
67:
68:      ALLOCATE( ns3d )
69:      ALLOCATE( le3d )
70:      ALLOCATE( es3d )
71:      ALLOCATE( esm3d )
72:      ALLOCATE( efv3d )
73:      ALLOCATE( fem3d )
74:
75:!-----------------------------------------------------------------------
76:
77:      OPEN(10, FILE = 'mesh.dat')
78:
79:      READ(10, *) dataname
80:
81:      ns3d_n = 0
82:
83:      DO
84:
85:       READ(10, *) dataname
86:
87:       ns3d_n = ns3d_n+1
88:
89:       IF( dataname .EQ. '!' ) THEN
```

```
 90:
 91:        EXIT
 92:
 93:       END IF
 94:
 95:      END DO
 96:
 97:      ns3d_n = ns3d_n-1
 98:
 99:      es3d_n = 0
100:
101:      DO
102:
103:       READ(10, *) dataname
104:
105:       es3d_n = es3d_n+1
106:
107:       IF( dataname .EQ. '!' ) THEN
108:
109:        EXIT
110:
111:       END IF
112:
113:      END DO
114:
115:      es3d_n = es3d_n-1
116:
117:      CLOSE(10)
118:
119:      !-------------------------------------------------------------
120:
121:      OPEN(13, FILE = 'param_fea.dat')
122:
123:      READ(13, *) dataname
124:      READ(13, *) dataname
125:      READ(13, *) dataname
```

```
126:      READ(13, *) dataname
127:      READ(13, *) dataname
128:      READ(13, *) dataname
129:      READ(13, *) dataname
130:      READ(13, *) dataname
131:      READ(13, *) dataname
132:      READ(13, *) dataname
133:
134:      READ(13, *) dataname
135:
136:      fem3d_nnodes_loaded = 0
137:
138:      DO
139:
140:       READ(13, *) dataname
141:
142:       fem3d_nnodes_loaded = fem3d_nnodes_loaded+1
143:
144:       IF( dataname .EQ. '!' ) THEN
145:
146:        EXIT
147:
148:       END IF
149:
150:      END DO
151:
152:      fem3d_nnodes_loaded = fem3d_nnodes_loaded-1
153:
154:      CLOSE(13)
155:
156:!-----------------------------------------------------------------------
157:
158:      OPEN(10, FILE = 'mesh.dat')
159:
160:      READ(10, *) dataname
161:
```

```
162:      CALL init_nodes3d(ns3d, ns3d_n)
163:
164:      ALLOCATE( ns3d_x(3, ns3d_n) )
165:
166:      DO id = 1, ns3d_n
167:
168:       READ(10, *) number, ( ns3d_x(i, id), i = 1, 3 )
169:
170:      END DO
171:
172:      CALL set_nodes3d_x(ns3d, ns3d_x)
173:
174:      READ(10, *) dataname, le3d_nboundaries, le3d_nnodes, le3d_nqps
175:
176:      CALL init_localelement3d                              &
177:           (le3d, le3d_nboundaries, le3d_nnodes, le3d_nqps)
178:      CALL init_elements3d(es3d, ns3d, le3d, es3d_n)
179:
180:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
181:
182:      DO ie = 1, es3d_n
183:
184:       READ(10, *) number, ( es3d_connectivity(na, ie), &
185:                             na = 1, le3d_nnodes )
186:
187:      END DO
188:
189:      CALL set_elements3d_connectivity(es3d, es3d_connectivity)
190:
191:      CLOSE(10)
192:
193:      !-------------------------------------------------------------
194:
195:      CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
196:      CALL init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
197:      CALL init_fem3d                                   &
```

```
198:           (fem3d, ns3d, le3d, es3d, esm3d, efv3d, 9)
199:
200:      CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
201:
202:      !-------------------------------------------------------------
203:
204:      OPEN(11, FILE = 'ic.dat')
205:
206:      READ(11, *) dataname
207:
208:      ALLOCATE( ns3d_u(3*ns3d_n) )
209:
210:      DO id = 1, ns3d_n
211:
212:       READ(11, *) number, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
213:
214:      END DO
215:
216:      CALL set_nodes3d_u(ns3d, ns3d_u)
217:
218:      READ(11, *) dataname
219:
220:      CLOSE(11)
221:
222:      !-------------------------------------------------------------
223:
224:      OPEN(12, FILE = 'bc.dat')
225:
226:      READ(12, *) dataname
227:
228:      ALLOCATE( ns3d_bc(3*ns3d_n) )
229:
230:      DO id = 1, ns3d_n
231:
232:       READ(12, *) number, ( ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
233:
```

```
234:      END DO
235:
236:      CALL set_nodes3d_bc(ns3d, ns3d_bc)
237:
238:      READ(12, *) dataname
239:
240:      CLOSE(12)
241:
242:!-----------------------------------------------------------------
243:
244:      OPEN(13, FILE = 'param_fea.dat')
245:
246:      READ(13, *) dataname
247:      READ(13, *) dataname
248:      READ(13, *) dataname
249:      READ(13, *) e
250:      READ(13, *) dataname
251:      READ(13, *) nu
252:      READ(13, *) dataname
253:      READ(13, *) rho
254:      READ(13, *) dataname
255:      READ(13, *) g
256:
257:      !-------------------------------------------------------------
258:
259:      ALLOCATE( esm3d_e(es3d_n) )
260:      ALLOCATE( esm3d_nu(es3d_n) )
261:
262:      ALLOCATE( efv3d_rho(es3d_n) )
263:
264:      DO ie = 1, es3d_n
265:
266:       esm3d_e(ie)  = e
267:       esm3d_nu(ie) = nu
268:
269:       efv3d_rho(ie) = rho
```

```
270:
271:      END DO
272:
273:      CALL set_elemstiffmat3d_e_nu(esm3d, esm3d_e, esm3d_nu)
274:
275:      CALL set_elemexforcevec3d_rho(efv3d, efv3d_rho)
276:      CALL set_elemexforcevec3d_g(efv3d, g)
277:
278:      !-------------------------------------------------------------
279:
280:      READ(13, *) dataname
281:
282:      ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
283:      ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
284:
285:      DO idof_l = 1, fem3d_nnodes_loaded
286:
287:       READ(13, *) fem3d_id_loaded(idof_l),                &
288:                   ( fem3d_f_loaded(i, idof_l), i = 1, 3 )
289:
290:      END DO
291:
292:      CALL set_fem3d_f_loaded                         &
293:           (fem3d, fem3d_id_loaded, fem3d_f_loaded)
294:
295:      READ(13, *) dataname
296:
297:      CLOSE(13)
298:
299:!-----------------------------------------------------------------
300:
301:      DEALLOCATE( ns3d_x )
302:      DEALLOCATE( ns3d_u )
303:      DEALLOCATE( ns3d_bc )
304:
305:      DEALLOCATE( es3d_connectivity )
```

```
306:
307:      DEALLOCATE( esm3d_e  )
308:      DEALLOCATE( esm3d_nu )
309:
310:      DEALLOCATE( efv3d_rho )
311:
312:      DEALLOCATE( fem3d_id_loaded )
313:      DEALLOCATE( fem3d_f_loaded )
314:
315:!-----------------------------------------------------------------
316:
317:      RETURN
318:
319:!#######################################################################
320:      END SUBROUTINE start_appli
321:!#######################################################################
322:
323:
324:      ! Run appli
325:!#######################################################################
326:      SUBROUTINE run_appli()
327:!#######################################################################
328:
329:      INTEGER :: ns3d_n
330:      INTEGER :: le3d_nnodes
331:      INTEGER :: es3d_n
332:      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
333:      INTEGER :: i
334:      INTEGER :: id
335:      INTEGER :: na
336:      INTEGER :: ie
337:
338:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
339:      REAL(8), ALLOCATABLE :: ns3d_u(:)
340:      REAL(8), ALLOCATABLE :: esm3d_evec(:, :)
341:      REAL(8), ALLOCATABLE :: esm3d_svec(:, :), esm3d_s_mises(:)
```

```
342:
343:!-----------------------------------------------------------------
344:
345:      CALL cal_elements3d(es3d)
346:
347:      CALL cal_fem3d(fem3d)
348:
349:!-----------------------------------------------------------------
350:
351:      CALL get_nodes3d_n(ns3d, ns3d_n)
352:      ALLOCATE( ns3d_x(3, ns3d_n) )
353:      CALL get_nodes3d_x(ns3d, ns3d_x)
354:      ALLOCATE( ns3d_u(3*ns3d_n) )
355:      CALL get_nodes3d_u(ns3d, ns3d_u)
356:
357:      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
358:
359:      CALL get_elements3d_n(es3d, es3d_n)
360:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
361:      CALL get_elements3d_connectivity(es3d, es3d_connectivity)
362:
363:      ALLOCATE( esm3d_evec(6, es3d_n) )
364:      ALLOCATE( esm3d_svec(6, es3d_n) )
365:      ALLOCATE( esm3d_s_mises(es3d_n) )
366:      CALL get_elemstiffmat3d_evec_svec                    &
367:           (esm3d, esm3d_evec, esm3d_svec, esm3d_s_mises)
368:
369:!-----------------------------------------------------------------
370:
371:      OPEN(14, FILE = 'result.inp')
372:
373:      WRITE(14, '( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
374:
375:      DO id = 1, ns3d_n
376:
377:       WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )') &
```

```
378:             id, ( ns3d_x(i, id), i = 1, 3 )
379:
380:       END DO
381:
382:       DO ie = 1, es3d_n
383:
384:        WRITE(14, '( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )')         &
385:             ie, 1, '  hex',                                      &
386:             ( es3d_connectivity(na, ie),  na = 1, le3d_nnodes)
387:
388:       END DO
389:
390:       WRITE(14, '( 4(I8, 1X) )') 1, 3
391:       WRITE(14, '(A)') 'DISPLACEMENT, m'
392:
393:       DO id = 1, ns3d_n
394:
395:        WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )')      &
396:             id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
397:
398:       END DO
399:
400:       WRITE(14, '( 14I8 )') 13, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
401:       WRITE(14, '( (A, 1X) )') 'STRAIN_11, unit_unknown'
402:       WRITE(14, '( (A, 1X) )') 'STRAIN_22, unit_unknown'
403:       WRITE(14, '( (A, 1X) )') 'STRAIN_33, unit_unknown'
404:       WRITE(14, '( (A, 1X) )') 'STRAIN_12, unit_unknown'
405:       WRITE(14, '( (A, 1X) )') 'STRAIN_23, unit_unknown'
406:       WRITE(14, '( (A, 1X) )') 'STRAIN_31, unit_unknown'
407:       WRITE(14, '( (A, 1X) )') 'STRESS_11, Pa'
408:       WRITE(14, '( (A, 1X) )') 'STRESS_22, Pa'
409:       WRITE(14, '( (A, 1X) )') 'STRESS_33, Pa'
410:       WRITE(14, '( (A, 1X) )') 'STRESS_12, Pa'
411:       WRITE(14, '( (A, 1X) )') 'STRESS_23, Pa'
412:       WRITE(14, '( (A, 1X) )') 'STRESS_31, Pa'
413:       WRITE(14, '( (A, 1X) )') 'STRESS_MISES, Pa'
```

```
414:
415:      DO ie = 1, es3d_n
416:
417:       WRITE(14, '( (I8, 1X), 13(E17.8, 1X) )')                    &
418:             ie, ( esm3d_evec(i, ie), i = 1, 6 ),                  &
419:                 ( esm3d_svec(i, ie), i = 1, 6 ), esm3d_s_mises(ie)
420:
421:      END DO
422:
423:      CLOSE(14)
424:
425:!-----------------------------------------------------------------------
426:
427:      DEALLOCATE( ns3d_x )
428:      DEALLOCATE( ns3d_u )
429:
430:      DEALLOCATE( es3d_connectivity )
431:
432:      DEALLOCATE( esm3d_evec )
433:      DEALLOCATE( esm3d_svec )
434:      DEALLOCATE( esm3d_s_mises )
435:
436:!-----------------------------------------------------------------------
437:
438:      RETURN
439:
440:!#######################################################################
441:      END SUBROUTINE run_appli
442:!#######################################################################
443:
444:
445:      ! Finish appli
446:!#######################################################################
447:      SUBROUTINE finish_appli()
448:!#######################################################################
449:
```

```
450:      CALL del_nodes3d(ns3d)
451:      CALL del_localelement3d(le3d)
452:      CALL del_elements3d(es3d)
453:      CALL del_elemstiffmat3d(esm3d)
454:      CALL del_elemexforcevec3d(efv3d)
455:      CALL del_fem3d(fem3d)
456:
457:!-----------------------------------------------------------------------
458:
459:      DEALLOCATE( ns3d )
460:      DEALLOCATE( le3d )
461:      DEALLOCATE( es3d )
462:      DEALLOCATE( esm3d )
463:      DEALLOCATE( efv3d )
464:      DEALLOCATE( fem3d )
465:
466:!-----------------------------------------------------------------------
467:
468:      RETURN
469:
470:!#######################################################################
471:      END SUBROUTINE finish_appli
472:!#######################################################################
473:
474:
475:!#######################################################################
476:      END MODULE mod_appli
```