

第 5 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 5 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_elemexforcevec3d の修正

1. 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:     ! nelemboundaries
34:     ! The total number of element boundaries
35:     !
36:     ! table_ie(:)
37:     ! Table of element boundary no. and element no.
38:     !
39:     ! table_ma(:)
40:     ! Table of element boundary no. and
41:     ! boundary no. in a local element
42:     !
43:     ! t(:, :)
44:     ! Traction vector
45:     ! tx, ty, tz
46:     !
47:     !-----
48:
49:     ! rho(:)
50:     ! Density
51:     !
52:     ! g
53:     ! Gravitational acceleration
54:     !
55:     !-----
56:
57:     INTEGER :: nelemboundaries
58:     INTEGER, ALLOCATABLE :: table_ie(:)

```

```

59:      INTEGER, ALLOCATABLE :: table_ma(:)
60:
61:      REAL(8), ALLOCATABLE :: f(:, :)
62:      REAL(8), ALLOCATABLE :: t(:, :)
63:      REAL(8), ALLOCATABLE :: rho(:)
64:      REAL(8) :: g
65:
66:      !-----
67:
68:      END TYPE struct_elemexforcevec3d
69:
70: !-----
71:
72:      CONTAINS
73:
74:
75:      ! Get element external force vector
76: !#####
77:      SUBROUTINE get_elemexforcevec3d_f(efv3d, f)
78: !#####
79:
80:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
81:
82:      REAL(8), INTENT(OUT) :: f(:, :)
83:
84: !-----
85:
86:      f = efv3d%f
87:
88: !-----
89:
90:      RETURN
91:
92: !#####
93:      END SUBROUTINE get_elemexforcevec3d_f
94: !#####

```

```

95:
96:
97:      ! Get the total number of element boundaries
98: !#####
99:      SUBROUTINE get_elemexforcevec3d_nelemboundaries &
100:          (efv3d, nelemboundaries)
101: !#####
102:
103:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
104:
105:      INTEGER, INTENT(OUT) :: nelemboundaries
106:
107: !-----
108:
109:      nelemboundaries = efv3d%nelemboundaries
110:
111: !-----
112:
113:      RETURN
114:
115: !#####
116:      END SUBROUTINE get_elemexforcevec3d_nelemboundaries
117: !#####
118:
119:
120:      ! Set traction vector
121: !#####
122:      SUBROUTINE set_elemexforcevec3d_t          &
123:          (efv3d, table_ie, table_ma, t)
124: !#####
125:
126:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
127:
128:      INTEGER, INTENT(IN) :: table_ie(:)
129:      INTEGER, INTENT(IN) :: table_ma(:)
130:      REAL(8), INTENT(IN) :: t(:, :)

```

```

131:
132: !-----
133:
134:     efv3d%table_ie = table_ie
135:     efv3d%table_ma = table_ma
136:     efv3d%t = t
137:
138: !-----
139:
140:     RETURN
141:
142: !#####
143:     END SUBROUTINE set_elemexforcevec3d_t
144: !#####
145:
146:
147:     ! Get traction vector
148: !#####
149:     SUBROUTINE get_elemexforcevec3d_t      &
150:         (efv3d, table_ie, table_ma, t)
151: !#####
152:
153:     TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
154:
155:     INTEGER, INTENT(OUT) :: table_ie(:)
156:     INTEGER, INTENT(OUT) :: table_ma(:)
157:     REAL(8), INTENT(OUT) :: t(:, :)
158:
159: !-----
160:
161:     table_ie = efv3d%table_ie
162:     table_ma = efv3d%table_ma
163:     t        = efv3d%t
164:
165: !-----
166:

```

```

167:      RETURN
168:
169: !#####
170:      END SUBROUTINE get_elemexforcevec3d_t
171: !#####
172:
173:
174:      ! Set density
175: !#####
176:      SUBROUTINE set_elemexforcevec3d_rho(efv3d, rho)
177: !#####
178:
179:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
180:
181:      REAL(8), INTENT(IN) :: rho(:)
182:
183: !-----
184:
185:      efv3d%rho = rho
186:
187: !-----
188:
189:      RETURN
190:
191: !#####
192:      END SUBROUTINE set_elemexforcevec3d_rho
193: !#####
194:
195:
196:      ! Get density
197: !#####
198:      SUBROUTINE get_elemexforcevec3d_rho(efv3d, rho)
199: !#####
200:
201:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
202:

```

```

203:      REAL (8), INTENT (OUT) :: rho(:)
204:
205: !-----
206:
207:      rho = efv3d%rho
208:
209: !-----
210:
211:      RETURN
212:
213: !#####
214:      END SUBROUTINE get_elemexforcevec3d_rho
215: !#####
216:
217:
218:      ! Set gravitational acceleration
219: !#####
220:      SUBROUTINE set_elemexforcevec3d_g(efv3d, g)
221: !#####
222:
223:      TYPE(struct_elemexforcevec3d), INTENT (INOUT) :: efv3d
224:
225:      REAL (8), INTENT (IN) :: g
226:
227: !-----
228:
229:      efv3d%g = g
230:
231: !-----
232:
233:      RETURN
234:
235: !#####
236:      END SUBROUTINE set_elemexforcevec3d_g
237: !#####
238:

```

```

239:
240:      ! Get gravitational acceleration
241: !#####
242:      SUBROUTINE get_elemexforcevec3d_g(efv3d, g)
243: !#####
244:
245:      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d
246:
247:      REAL(8), INTENT(OUT) :: g
248:
249: !-----
250:
251:      g = efv3d%g
252:
253: !-----
254:
255:      RETURN
256:
257: !#####
258:      END SUBROUTINE get_elemexforcevec3d_g
259: !#####
260:
261:
262: !#####
263:      SUBROUTINE init_elemexforcevec3d                                &
264:          (efv3d, ns3d, le3d, es3d, nelemboundaries)
265: !#####
266:
267:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
268:
269:      TYPE(struct_nodes3d), TARGET, INTENT(IN)      :: ns3d
270:      TYPE(struct_localelement3d), TARGET, INTENT(IN) :: le3d
271:      TYPE(struct_elements3d), TARGET, INTENT(IN)    :: es3d
272:
273:      INTEGER, INTENT(IN) :: nelemboundaries
274:

```



```

275: !-----
276:
277:     INTEGER :: le3d_nnodes
278:     INTEGER :: es3d_n
279:
280: !-----
281:
282:     efv3d%ns3d => ns3d
283:     efv3d%le3d => le3d
284:     efv3d%es3d => es3d
285:
286: !-----
287:
288:     CALL get_localelement3d_nnodes(efv3d%le3d, le3d_nnodes)
289:
290:     CALL get_elements3d_n(efv3d%es3d, es3d_n)
291:
292: !-----
293:
294:     ALLOCATE( efv3d%f(3*le3d_nnodes, es3d_n) )
295:
296:     efv3d%f = 0.0D0
297:
298:     !-----
299:
300:     efv3d%nelemboundaries = nelemboundaries
301:
302:     ALLOCATE( efv3d%table_ie(nelemboundaries) )
303:
304:     efv3d%table_ie = 0
305:
306:     ALLOCATE( efv3d%table_ma(nelemboundaries) )
307:
308:     efv3d%table_ma = 0
309:
310:     ALLOCATE( efv3d%t(3, nelemboundaries) )

```

```

311:
312:     efv3d%t = 0.0D0
313:
314:     !-----
315:
316:     ALLOCATE( efv3d%rho(es3d_n) )
317:
318:     efv3d%rho = 0.0D0
319:
320:     efv3d%g = 0.0D0
321:
322: !-----
323:
324:     RETURN
325:
326: !#####
327:     END SUBROUTINE init_elemexforcevec3d
328: !#####
329:
330:
331: !#####
332:     SUBROUTINE cal_elemexforcevec3d(efv3d)
333: !#####
334:
335:     TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
336:
337: !-----
338:
339:     INTEGER :: ns3d_n
340:     INTEGER :: le3d_nboundaries
341:     INTEGER :: le3d_nnodes
342:     INTEGER :: le3d_nnodes_boundary
343:     INTEGER, ALLOCATABLE :: le3d_table_na(:, :)
344:     INTEGER :: le3d_nqps
345:     INTEGER :: es3d_n
346:     INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)

```

```

347:      INTEGER :: nqps_tot
348:      INTEGER :: i, k
349:      INTEGER :: id
350:      INTEGER :: ma
351:      INTEGER :: na, nb
352:      INTEGER :: naa
353:      INTEGER :: ie
354:      INTEGER :: ib
355:      INTEGER :: isize
356:      INTEGER :: jsize1, jsize2, jsize3
357:      INTEGER :: ijk
358:
359:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
360:      REAL(8), ALLOCATABLE :: le3d_xi_qp(:, :)
361:      REAL(8), ALLOCATABLE :: le3d_w_qp(:, :)
362:      REAL(8), ALLOCATABLE :: le3d_n_qp(:, :)
363:      REAL(8), ALLOCATABLE :: le3d_dndxi_qp(:, :, :)
364:      REAL(8), ALLOCATABLE :: x_local(:, :)
365:      REAL(8) :: w_xi, w_eta, w_zeta
366:      REAL(8) :: g1(3), g2(3), g3(3)
367:      REAL(8) :: x31(3), x42(3)
368:      REAL(8) :: area, area_inv
369:      REAL(8) :: area_nx, area_ny, area_nz
370:      REAL(8) :: det_j
371:      REAL(8), ALLOCATABLE :: n(:)
372:      REAL(8) :: w_w_w_det_j
373:      REAL(8), ALLOCATABLE :: nmat(:, :)
374:      REAL(8) :: bvec(3)
375:
376: !-----
377:
378:      CALL get_nodes3d_n(efv3d%ns3d, ns3d_n)
379:      ALLOCATE( ns3d_x(3, ns3d_n) )
380:      CALL get_nodes3d_x(efv3d%ns3d, ns3d_x)
381:
382:      CALL get_localelement3d_nboundaries(efv3d%le3d, le3d_nboundaries)

```

```

383:      CALL get_localelement3d_nnodes(efv3d%le3d, le3d_nnodes)
384:      CALL get_localelement3d_nnodes_boundary &
385:           (efv3d%le3d, le3d_nnodes_boundary)
386:      ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
387:      CALL get_localelement3d_table_na(efv3d%le3d, le3d_table_na)
388:      CALL get_localelement3d_nqps(efv3d%le3d, le3d_nqps)
389:      nqps_tot = le3d_nqps*le3d_nqps*le3d_nqps
390:      ALLOCATE( le3d_xi_qp(3, nqps_tot) )
391:      ALLOCATE( le3d_w_qp(3, nqps_tot) )
392:      CALL get_localelement3d_xi_w_qp(efv3d%le3d, le3d_xi_qp, le3d_w_qp)
393:      ALLOCATE( le3d_n_qp(le3d_nnodes, nqps_tot) )
394:      ALLOCATE( le3d_dndxi_qp(3, le3d_nnodes, nqps_tot) )
395:      CALL get_localelement3d_n_qp(efv3d%le3d, le3d_n_qp, le3d_dndxi_qp)
396:
397:      CALL get_elements3d_n(efv3d%es3d, es3d_n)
398:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
399:      CALL get_elements3d_connectivity(efv3d%es3d, es3d_connectivity)
400:
401:      ALLOCATE( x_local(3, le3d_nnodes) )
402:      ALLOCATE( n(le3d_nnodes_boundary) )
403:      ALLOCATE( nmat(3, 3*le3d_nnodes) )
404:
405: !-----
406:
407:      efv3d%f = 0.0D0
408:
409:      DO ib = 1, efv3d%nelemboundaries
410:
411: !-----
412:
413:      ie = efv3d%table_ie(ib)
414:      ma = efv3d%table_ma(ib)
415:
416:      DO naa = 1, le3d_nnodes_boundary
417:
418:      na = le3d_table_na(naa, ma)

```

```
419:      id = es3d_connectivity(na, ie)
420:
421:      DO i = 1, 3
422:
423:          x_local(i, naa) = ns3d_x(i, id)
424:
425:      END DO
426:
427:  END DO
428:
429:  IF( le3d_nboundaries .EQ. 6 ) THEN
430:
431:      x31(1) = x_local(1, 3)-x_local(1, 1)
432:      x31(2) = x_local(2, 3)-x_local(2, 1)
433:      x31(3) = x_local(3, 3)-x_local(3, 1)
434:
435:      x42(1) = x_local(1, 4)-x_local(1, 2)
436:      x42(2) = x_local(2, 4)-x_local(2, 2)
437:      x42(3) = x_local(3, 4)-x_local(3, 2)
438:
439:      area_nx = 0.5D0*( x31(2)*x42(3)-x42(2)*x31(3) )
440:      area_ny = 0.5D0*( x31(3)*x42(1)-x42(3)*x31(1) )
441:      area_nz = 0.5D0*( x31(1)*x42(2)-x42(1)*x31(2) )
442:
443:      area = DSQRT( area_nx*area_nx   &
444:                   +area_ny*area_ny   &
445:                   +area_nz*area_nz )
446:
447:      det_j = 0.25D0*area
448:
449:  IF( le3d_nnodes .EQ. 8 ) THEN
450:
451:      n(1) = 1.0D0
452:      n(2) = 1.0D0
453:      n(3) = 1.0D0
454:      n(4) = 1.0D0
```

```

455:
456:     END IF
457:
458:     END IF
459:
460:     DO naa = 1, le3d_nnodes_boundary
461:
462:         na = le3d_table_na(naa, ma)
463:
464:         isize = 3*(na-1)+1
465:         efv3d%f(isize, ie)                                &
466:         = efv3d%f(isize, ie)+n(naa)*efv3d%t(1, ib)*det_j
467:
468:         isize = 3*(na-1)+2
469:         efv3d%f(isize, ie)                                &
470:         = efv3d%f(isize, ie)+n(naa)*efv3d%t(2, ib)*det_j
471:
472:         isize = 3*(na-1)+3
473:         efv3d%f(isize, ie)                                &
474:         = efv3d%f(isize, ie)+n(naa)*efv3d%t(3, ib)*det_j
475:
476:     END DO
477:
478: END DO
479:
480: !-----
481:
482: DO ie = 1, es3d_n
483:
484:     !-----
485:
486:     DO na = 1, le3d_nnodes
487:
488:         id = es3d_connectivity(na, ie)
489:
490:         DO i = 1, 3

```

```

491:
492:     x_local(i, na) = ns3d_x(i, id)
493:
494:     END DO
495:
496: END DO
497:
498: !-----
499:
500: DO ijk = 1, nqps_tot
501:
502:     !-----
503:
504:     ! Covariant basis vector
505:     DO i = 1, 3
506:
507:         g1(i) = 0.0D0
508:         g2(i) = 0.0D0
509:         g3(i) = 0.0D0
510:
511:         DO na = 1, le3d_nnodes
512:
513:             g1(i) = g1(i)+le3d_dndxi_qp(1, na, ijk)*x_local(i, na)
514:             g2(i) = g2(i)+le3d_dndxi_qp(2, na, ijk)*x_local(i, na)
515:             g3(i) = g3(i)+le3d_dndxi_qp(3, na, ijk)*x_local(i, na)
516:
517:         END DO
518:
519:     END DO
520:
521:     !-----
522:
523:     ! Jacobian
524:     det_j = g1(1)*( g2(2)*g3(3)-g2(3)*g3(2) ) &
525:             +g1(2)*( g2(3)*g3(1)-g2(1)*g3(3) ) &
526:             +g1(3)*( g2(1)*g3(2)-g2(2)*g3(1) )

```

```

527:
528:         w_w_w_det_j                                &
529:         = le3d_w_qp(1, ijk)*le3d_w_qp(2, ijk)*le3d_w_qp(3, ijk) &
530:         *det_j
531:
532:         !-----
533:
534:         ! N matrix
535:
536:         nmat = 0.0D0
537:
538:         DO nb = 1, le3d_nnodes
539:
540:             jsize1 = 3*(nb-1)+1
541:             jsize2 = 3*(nb-1)+2
542:             jsize3 = 3*(nb-1)+3
543:
544:             nmat(1, jsize1) = le3d_n_qp(nb, ijk)
545:             nmat(2, jsize2) = le3d_n_qp(nb, ijk)
546:             nmat(3, jsize3) = le3d_n_qp(nb, ijk)
547:
548:         END DO
549:
550:         !-----
551:
552:         bvec(1) = 0.0D0
553:         bvec(2) = 0.0D0
554:         bvec(3) = efv3d%g
555:
556:         !-----
557:
558:         DO isize = 1, 3*le3d_nnodes
559:
560:             DO k = 1, 3
561:
562:                 efv3d%f(isize, ie)                                &

```



```

563:         = efv3d%f( isize, ie)           &
564:         +w_w_w_det_j           &
565:         *efv3d%rho(ie)*nmat(k, isize)*bvec(k)
566:
567:     END DO
568:
569: END DO
570:
571: !-----
572:
573: END DO
574:
575: !-----
576:
577: END DO
578:
579: !-----
580:
581: DEALLOCATE( ns3d_x )
582:
583: DEALLOCATE( le3d_table_na )
584: DEALLOCATE( le3d_xi_qp )
585: DEALLOCATE( le3d_w_qp )
586: DEALLOCATE( le3d_n_qp )
587: DEALLOCATE( le3d_dndxi_qp )
588:
589: DEALLOCATE( es3d_connectivity )
590:
591: DEALLOCATE( x_local )
592: DEALLOCATE( n )
593: DEALLOCATE( nmat )
594:
595: !-----
596:
597: RETURN
598:

```

```

599: !#####
600:      END SUBROUTINE cal_elemexforcevec3d
601: !#####
602:
603:
604: !#####
605:      SUBROUTINE del_elemexforcevec3d(efv3d)
606: !#####
607:
608:      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d
609:
610: !-----
611:
612:      NULLIFY( efv3d%ns3d )
613:      NULLIFY( efv3d%le3d )
614:      NULLIFY( efv3d%es3d )
615:
616: !-----
617:
618:      DEALLOCATE( efv3d%f )
619:
620:      !-----
621:
622:      DEALLOCATE( efv3d%table_ie )
623:      DEALLOCATE( efv3d%table_ma )
624:      DEALLOCATE( efv3d%t )
625:
626:      !-----
627:
628:      DEALLOCATE( efv3d%rho )
629:
630:      efv3d%g = 0.0D0
631:
632: !-----
633:
634:      RETURN

```

```

635:
636: !#####
637:      END SUBROUTINE del_elemexforcevec3d
638: !#####
639:
640:
641: !#####
642:      END MODULE mod_elemexforcevec3d

```

第 4 節のプログラム：単純引張変形解析

2. アプリケーションモジュール mod_appli.f90 (直方体メッシング用)

```

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 :: efv3d_nelemboundaries
160:      INTEGER, ALLOCATABLE :: efv3d_table_ie(:)
161:      INTEGER, ALLOCATABLE :: efv3d_table_ma(:)
162:      INTEGER :: fem3d_ndofs
163:      INTEGER :: fem3d_nnodes_loaded
164:      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
165:      INTEGER :: i
166:      INTEGER :: id

```

```

167:      INTEGER :: id_l
168:      INTEGER :: ma
169:      INTEGER :: na
170:      INTEGER :: ie
171:      INTEGER :: idof
172:      INTEGER :: ib
173:
174:      REAL (8), ALLOCATABLE :: ns3d_x(:, :)
175:      REAL (8), ALLOCATABLE :: ns3d_u(:)
176:      REAL (8), ALLOCATABLE :: es3d_volume(:)
177:      REAL (8) :: es3d_max_volume
178:      REAL (8) :: es3d_min_volume
179:      REAL (8) :: es3d_sum_volume
180:      REAL (8), ALLOCATABLE :: efv3d_t(:, :)
181:      REAL (8), ALLOCATABLE :: fem3d_f_loaded(:, :)
182:      REAL (8) :: rm3d_x_start(3)
183:      REAL (8) :: rm3d_x_end(3)
184:      REAL (8) :: rm3d_x_center(3)
185:      REAL (8) :: rm3d_length_x(3)
186:      REAL (8), ALLOCATABLE :: x_local(:, :)
187:
188: !-----
189:
190:      CALL cal_rectmesher3d(rm3d)
191:
192:      CALL cal_elements3d(es3d)
193:
194: !-----
195:
196:      CALL get_nodes3d_n(ns3d, ns3d_n)
197:
198:      ALLOCATE( ns3d_x(3, ns3d_n) )
199:      CALL get_nodes3d_x(ns3d, ns3d_x)
200:      ALLOCATE( ns3d_u(3*ns3d_n) )
201:      ns3d_u = 0.0D0
202:      ALLOCATE( ns3d_bc(3*ns3d_n) )

```



```

203:      ns3d_bc = 0
204:
205:      CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
206:      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
207:      CALL get_localelement3d_nnodes_boundary(le3d, le3d_nnodes_boundary)
208:      ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
209:      CALL get_localelement3d_table_na(le3d, le3d_table_na)
210:
211:      CALL get_elements3d_n(es3d, es3d_n)
212:      ALLOCATE( es3d_volume(es3d_n) )
213:      CALL get_elements3d_volume              &
214:      (es3d, es3d_volume,                    &
215:      es3d_max_volume, es3d_ie_max_volume, &
216:      es3d_min_volume, es3d_ie_min_volume, &
217:      es3d_sum_volume)
218:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
219:      CALL get_elements3d_connectivity(es3d, es3d_connectivity)
220:      CALL get_rectmesher3d_x_start_x_end    &
221:      (rm3d, rm3d_x_start, rm3d_x_end, &
222:      rm3d_x_center, rm3d_length_x)
223:
224:      ALLOCATE( x_local(3, le3d_nnodes_boundary) )
225:
226: !-----
227:
228:      ! Tensile deformation
229:      IF( prob .EQ. 1 ) THEN
230:
231:      !-----
232:
233:      id_l = 0
234:      ib = 0
235:
236:      DO id = 1, ns3d_n
237:
238:      IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN

```

```

239:
240:     id_l = id_l+1
241:
242:     END IF
243:
244:     END DO
245:
246:     fem3d_nnodes_loaded = id_l
247:     efv3d_nelemboundaries = ib
248:
249:     !-----
250:
251:     ! Tensile deformation
252:     ELSE IF( prob .EQ. 2 ) THEN
253:
254:     !-----
255:
256:     id_l = 0
257:     ib = 0
258:
259:     DO ie = 1, es3d_n
260:
261:         DO ma = 1, le3d_nboundaries
262:
263:             na = le3d_table_na(1, ma)
264:             id = es3d_connectivity(na, ie)
265:             x_local(1, 1) = ns3d_x(1, id)
266:             x_local(2, 1) = ns3d_x(2, id)
267:             x_local(3, 1) = ns3d_x(3, id)
268:
269:             na = le3d_table_na(2, ma)
270:             id = es3d_connectivity(na, ie)
271:             x_local(1, 2) = ns3d_x(1, id)
272:             x_local(2, 2) = ns3d_x(2, id)
273:             x_local(3, 2) = ns3d_x(3, id)
274:

```

```

275:      na = le3d_table_na(3, ma)
276:      id = es3d_connectivity(na, ie)
277:      x_local(1, 3) = ns3d_x(1, id)
278:      x_local(2, 3) = ns3d_x(2, id)
279:      x_local(3, 3) = ns3d_x(3, id)
280:
281:      na = le3d_table_na(4, ma)
282:      id = es3d_connectivity(na, ie)
283:      x_local(1, 4) = ns3d_x(1, id)
284:      x_local(2, 4) = ns3d_x(2, id)
285:      x_local(3, 4) = ns3d_x(3, id)
286:
287:      IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
288:          ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
289:          ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
290:          ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
291:
292:          ib = ib+1
293:
294:      END IF
295:
296:      END DO
297:
298:      END DO
299:
300:      fem3d_nnodes_loaded = id_l
301:      efv3d_nelemboundaries = ib
302:
303:      !-----
304:
305:      END IF
306:
307:      !-----
308:
309:      CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
310:      CALL init_elemexforcevec3d                                     &

```

```

311:      (efv3d, ns3d, le3d, es3d, efv3d_nelemboundaries)
312:      CALL init_fem3d                                &
313:      (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
314:      fem3d_nnodes_loaded)
315:
316:      ALLOCATE( efv3d_table_ie(efv3d_nelemboundaries) )
317:      efv3d_table_ie = 0
318:      ALLOCATE( efv3d_table_ma(efv3d_nelemboundaries) )
319:      efv3d_table_ma = 0
320:      ALLOCATE( efv3d_t(3, efv3d_nelemboundaries) )
321:      efv3d_t = 0.0D0
322:
323:      CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
324:      ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
325:      fem3d_id_loaded = 0
326:      ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
327:      fem3d_f_loaded = 0.0D0
328:
329:      !-----
330:
331:      ! Tensile deformation
332:      IF( ( prob .EQ. 1 ) .OR. ( prob .EQ. 2 ) ) THEN
333:
334:      !-----
335:
336:      DO id = 1, ns3d_n
337:
338:      IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN
339:
340:      idof = 3*(id-1)+1
341:
342:      ns3d_u(idof) = 0.0D0
343:      ns3d_bc(idof) = 1
344:
345:      IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
346:

```

```

347:         idof = 3*(id-1)+2
348:
349:         ns3d_u(idof) = 0.0D0
350:         ns3d_bc(idof) = 1
351:
352:     END IF
353:
354:     IF ( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
355:
356:         idof = 3*(id-1)+3
357:
358:         ns3d_u(idof) = 0.0D0
359:         ns3d_bc(idof) = 1
360:
361:     END IF
362:
363: END IF
364:
365: END DO
366:
367: !-----
368:
369: END IF
370:
371: !-----
372:
373: ! Tensile deformation
374: IF( prob .EQ. 1 ) THEN
375:
376: !-----
377:
378:     id_l = 0
379:
380:     DO id = 1, ns3d_n
381:
382:         IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN

```

```

383:
384:     id_l = id_l+1
385:
386:     fem3d_id_loaded(id_l) = id
387:     fem3d_f_loaded(1, id_l) = 2.5D6
388:
389:     IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
390:
391:         fem3d_f_loaded(1, id_l) = 5.0D6
392:
393:     END IF
394:
395:     IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
396:
397:         fem3d_f_loaded(1, id_l) = 5.0D6
398:
399:     END IF
400:
401:     IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
402:
403:         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
404:
405:             fem3d_f_loaded(1, id_l) = 1.0D7
406:
407:         END IF
408:
409:     END IF
410:
411: END IF
412:
413: END DO
414:
415: !-----
416:
417:     ! Tensile deformation
418: ELSE IF( prob .EQ. 2 ) THEN

```

```

419:
420:     !-----
421:
422:     ib = 0
423:
424:     D0 ie = 1, es3d_n
425:
426:     D0 ma = 1, le3d_nboundaries
427:
428:     na = le3d_table_na(1, ma)
429:     id = es3d_connectivity(na, ie)
430:     x_local(1, 1) = ns3d_x(1, id)
431:     x_local(2, 1) = ns3d_x(2, id)
432:     x_local(3, 1) = ns3d_x(3, id)
433:
434:     na = le3d_table_na(2, ma)
435:     id = es3d_connectivity(na, ie)
436:     x_local(1, 2) = ns3d_x(1, id)
437:     x_local(2, 2) = ns3d_x(2, id)
438:     x_local(3, 2) = ns3d_x(3, id)
439:
440:     na = le3d_table_na(3, ma)
441:     id = es3d_connectivity(na, ie)
442:     x_local(1, 3) = ns3d_x(1, id)
443:     x_local(2, 3) = ns3d_x(2, id)
444:     x_local(3, 3) = ns3d_x(3, id)
445:
446:     na = le3d_table_na(4, ma)
447:     id = es3d_connectivity(na, ie)
448:     x_local(1, 4) = ns3d_x(1, id)
449:     x_local(2, 4) = ns3d_x(2, id)
450:     x_local(3, 4) = ns3d_x(3, id)
451:
452:     IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
453:          ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
454:          ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &

```

```

455:          ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
456:
457:          ib = ib+1
458:
459:          efv3d_table_ie(ib) = ie
460:          efv3d_table_ma(ib) = ma
461:          efv3d_t(1, ib) = -4.0D7
462:          efv3d_t(2, ib) = 0.0D0
463:          efv3d_t(3, ib) = 0.0D0
464:
465:          END IF
466:
467:          END DO
468:
469:          END DO
470:
471:          !-----
472:
473:          END IF
474:
475:          CALL set_fem3d_f_loaded          &
476:          (fem3d, fem3d_id_loaded, fem3d_f_loaded)
477:          CALL set_elemexforcevec3d_t          &
478:          (efv3d, efv3d_table_ie, efv3d_table_ma, efv3d_t)
479:
480:          !-----
481:
482:          OPEN(10, FILE = 'mesh.dat')
483:
484:          WRITE(10, ' (A)') ' !NODE'
485:
486:          DO id = 1, ns3d_n
487:
488:          WRITE( 10, ' ( I8, 3(A, E17.8) )' )          &
489:          id, ( ', ', ns3d_x(i, id), i = 1, 3 )
490:

```



```

491:      END DO
492:
493:      WRITE( 10, ' (A, 3(A, I3) )' )          &
494:          ' !ELEMENT', ' ', ' ', le3d_nboundaries, &
495:          ' ', ' ', le3d_nnodes, ' ', ' ', 2
496:
497:
498:      DO ie = 1, es3d_n
499:
500:          WRITE( 10, ' ( I8, 27(A, I8) )' )      &
501:              ie, ( ' ', ' ', es3d_connectivity(na, ie), &
502:                  na = 1, le3d_nnodes )
503:
504:      END DO
505:
506:      WRITE(10, ' (A)' ) ' !END'
507:
508:      CLOSE(10)
509:
510: !-----
511:
512:      OPEN(11, FILE = 'ic.dat')
513:
514:      WRITE(11, ' (A)' ) ' !DISPLACEMENT'
515:
516:      DO id = 1, ns3d_n
517:
518:          WRITE( 11, ' (I8, 3(A, E17.8) )' )      &
519:              id, ( ' ', ' ', ns3d_u( 3*(id-1)+i ), i = 1, 3 )
520:
521:      END DO
522:
523:      WRITE(11, ' (A)' ) ' !END'
524:
525:      CLOSE(11)
526:

```

```

527: !-----
528:
529:     OPEN(12, FILE = 'bc.dat')
530:
531:     WRITE(12, '(A)') '!DISPLACEMENT'
532:
533:     DO id = 1, ns3d_n
534:
535:         WRITE( 12, '(I8, 3(A, I8) )' )                                &
536:             id, ( ', ', ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
537:
538:     END DO
539:
540:     WRITE(12, '(A)') '!END'
541:
542:     CLOSE(12)
543:
544: !-----
545:
546:     WRITE(13, '(A)') '!F_LOADED'
547:
548:     DO id_l = 1, fem3d_nnodes_loaded
549:
550:         WRITE( 13, '(I8, 3(A, E17.8) )' )                                &
551:             fem3d_id_loaded(id_l),                                &
552:             ( ', ', fem3d_f_loaded(i, id_l), i = 1, 3 )
553:
554:     END DO
555:
556:     WRITE(13, '(A)') '!TRACTION'
557:
558:     DO ib = 1, efv3d_nelemboundaries
559:
560:         WRITE( 13, '(I8, 2(A, I8), 3(A, E17.8) )' ) &
561:             ib, ', ', efv3d_table_ie(ib),            &
562:             ', ', efv3d_table_ma(ib),                &

```

```

563:          ( ', ', efv3d_t(i, ib), i = 1, 3 )
564:
565:      END DO
566:
567:      WRITE(13, ' (A)') '!END'
568:
569:      CLOSE(13)
570:
571: !-----
572:
573:      OPEN(14, FILE = 'mesh.inp')
574:
575:      WRITE(14, ' ( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
576:
577:      DO id = 1, ns3d_n
578:
579:          WRITE( 14, ' ( (I8, 1X), 3(E17.8, 1X) )' ) &
580:              id, ( ns3d_x(i, id), i = 1, 3 )
581:
582:      END DO
583:
584:      DO ie = 1, es3d_n
585:
586:          WRITE( 14, ' ( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )' )      &
587:              ie, 1, ' hex',                                       &
588:              ( es3d_connectivity(na, ie), na = 1, le3d_nnodes )
589:
590:      END DO
591:
592:      WRITE(14, ' ( 4(I8, 1X) )') 1, 3
593:      WRITE(14, ' (A)') 'DISPLACEMENT, m'
594:
595:      DO id = 1, ns3d_n
596:
597:          WRITE( 14, ' ( (I8, 1X), 3(E17.8, 1X) )' )      &
598:              id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )

```

```

599:
600:     END DO
601:
602:     WRITE(14, '( 14I8 )') 1, 1
603:     WRITE(14, '( (A, 1X) )') 'VOLUME, m3'
604:
605:     DO ie = 1, es3d_n
606:
607:         WRITE( 14, '( (I8, 1X), (E17.8, 1X) )' ) &
608:             ie, es3d_volume(ie)
609:
610:     END DO
611:
612:     CLOSE(14)
613:
614: !-----
615:
616:     DEALLOCATE( ns3d_x )
617:     DEALLOCATE( ns3d_u )
618:
619:     DEALLOCATE( le3d_table_na )
620:
621:     DEALLOCATE( es3d_volume )
622:     DEALLOCATE( es3d_connectivity )
623:
624:     DEALLOCATE( efv3d_table_ie )
625:     DEALLOCATE( efv3d_table_ma )
626:     DEALLOCATE( efv3d_t )
627:
628:     DEALLOCATE( fem3d_id_loaded )
629:     DEALLOCATE( fem3d_f_loaded )
630:
631:     DEALLOCATE( x_local )
632:
633: !-----
634:

```

```

635:      RETURN
636:
637: !#####
638:      END SUBROUTINE run_appli
639: !#####
640:
641:
642: !#####
643:      SUBROUTINE finish_appli()
644: !#####
645:
646:      CALL del_nodes3d(ns3d)
647:      CALL del_localelement3d(le3d)
648:      CALL del_elements3d(es3d)
649:      CALL del_elemstiffmat3d(esm3d)
650:      CALL del_elemexforcevec3d(efv3d)
651:      CALL del_fem3d(fem3d)
652:      CALL del_rectmesher3d(rm3d)
653:
654: !-----
655:
656:      DEALLOCATE( ns3d )
657:      DEALLOCATE( le3d )
658:      DEALLOCATE( es3d )
659:      DEALLOCATE( esm3d )
660:      DEALLOCATE( efv3d )
661:      DEALLOCATE( fem3d )
662:      DEALLOCATE( rm3d )
663:
664: !-----
665:
666:      RETURN
667:
668: !#####
669:      END SUBROUTINE finish_appli
670: !#####

```

```
671:
672:
673: !#####
674:      END MODULE mod_appli
```

3. アプリケーションモジュール mod_appli.f90 (有限要素解析用)

```
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 :: efv3d_nelemboundaries
44:     INTEGER, ALLOCATABLE :: efv3d_table_ie(:)
45:     INTEGER, ALLOCATABLE :: efv3d_table_ma(:)
46:     INTEGER :: fem3d_ndofs
47:     INTEGER :: i
48:     INTEGER :: id
49:     INTEGER :: id_l
50:     INTEGER :: na
51:     INTEGER :: ie
52:     INTEGER :: ib
53:     INTEGER :: number
54:
55:     REAL(8), ALLOCATABLE :: ns3d_x(:, :)
56:     REAL(8), ALLOCATABLE :: ns3d_u(:)
57:     REAL(8), ALLOCATABLE :: esm3d_e(:)
58:     REAL(8), ALLOCATABLE :: esm3d_nu(:)
59:     REAL(8), ALLOCATABLE :: efv3d_rho(:)
60:     REAL(8), ALLOCATABLE :: efv3d_t(:, :)
61:     REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
62:     REAL(8) :: e
63:     REAL(8) :: nu
64:     REAL(8) :: rho
65:     REAL(8) :: g

```

```
66:
67:     CHARACTER(1) :: dataname
68:
69: !-----
70:
71:     ALLOCATE( ns3d )
72:     ALLOCATE( le3d )
73:     ALLOCATE( es3d )
74:     ALLOCATE( esm3d )
75:     ALLOCATE( efv3d )
76:     ALLOCATE( fem3d )
77:
78: !-----
79:
80:     OPEN(10, FILE = 'mesh.dat')
81:
82:     READ(10, *) dataname
83:
84:     ns3d_n = 0
85:
86:     DO
87:
88:         READ(10, *) dataname
89:
90:         ns3d_n = ns3d_n+1
91:
92:         IF( dataname .EQ. '!' ) THEN
93:
94:             EXIT
95:
96:         END IF
97:
98:     END DO
99:
100:    ns3d_n = ns3d_n-1
101:
```



```
102:      es3d_n = 0
103:
104:      DO
105:
106:          READ(10, *) dataname
107:
108:          es3d_n = es3d_n+1
109:
110:          IF( dataname .EQ. '!' ) THEN
111:
112:              EXIT
113:
114:          END IF
115:
116:      END DO
117:
118:      es3d_n = es3d_n-1
119:
120:      CLOSE(10)
121:
122:      !-----
123:
124:      OPEN(13, FILE = 'param_fea.dat')
125:
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:      READ(13, *) dataname
134:      READ(13, *) dataname
135:      READ(13, *) dataname
136:
137:      READ(13, *) dataname
```

```
138:
139:     fem3d_nnodes_loaded = 0
140:
141:     DO
142:
143:         READ(13, *) dataname
144:
145:         fem3d_nnodes_loaded = fem3d_nnodes_loaded+1
146:
147:         IF( dataname .EQ. '!' ) THEN
148:
149:             EXIT
150:
151:         END IF
152:
153:     END DO
154:
155:     fem3d_nnodes_loaded = fem3d_nnodes_loaded-1
156:
157:     efv3d_nelemboundaries = 0
158:
159:     DO
160:
161:         READ(13, *) dataname
162:
163:         efv3d_nelemboundaries = efv3d_nelemboundaries+1
164:
165:         IF( dataname .EQ. '!' ) THEN
166:
167:             EXIT
168:
169:         END IF
170:
171:     END DO
172:
173:     efv3d_nelemboundaries = efv3d_nelemboundaries-1
```

```

174:
175:     CLOSE(13)
176:
177: !-----
178:
179:     OPEN(10, FILE = 'mesh.dat')
180:
181:     READ(10, *) dataname
182:
183:     CALL init_nodes3d(ns3d, ns3d_n)
184:
185:     ALLOCATE( ns3d_x(3, ns3d_n) )
186:
187:     DO id = 1, ns3d_n
188:
189:         READ(10, *) number, ( ns3d_x(i, id), i = 1, 3 )
190:
191:     END DO
192:
193:     CALL set_nodes3d_x(ns3d, ns3d_x)
194:
195:     READ(10, *) dataname, le3d_nboundaries, le3d_nnodes, le3d_nqps
196:
197:     CALL init_localelement3d                                &
198:         (le3d, le3d_nboundaries, le3d_nnodes, le3d_nqps)
199:     CALL init_elements3d(es3d, ns3d, le3d, es3d_n)
200:
201:     ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
202:
203:     DO ie = 1, es3d_n
204:
205:         READ(10, *) number, ( es3d_connectivity(na, ie), &
206:                               na = 1, le3d_nnodes )
207:
208:     END DO
209:

```

```

210:      CALL set_elements3d_connectivity(es3d, es3d_connectivity)
211:
212:      CLOSE(10)
213:
214:      !-----
215:
216:      CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
217:      CALL init_elemexforcevec3d                                &
218:          (efv3d, ns3d, le3d, es3d, efv3d_nelemboundaries)
219:      CALL init_fem3d                                            &
220:          (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
221:          fem3d_nnodes_loaded)
222:
223:      CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
224:
225:      !-----
226:
227:      OPEN(11, FILE = 'ic.dat')
228:
229:      READ(11, *) dataname
230:
231:      ALLOCATE( ns3d_u(3*ns3d_n) )
232:
233:      DO id = 1, ns3d_n
234:
235:          READ(11, *) number, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
236:
237:      END DO
238:
239:      CALL set_nodes3d_u(ns3d, ns3d_u)
240:
241:      READ(11, *) dataname
242:
243:      CLOSE(11)
244:
245:      !-----

```

```

246:
247:     OPEN(12, FILE = 'bc.dat')
248:
249:     READ(12, *) dataname
250:
251:     ALLOCATE( ns3d_bc(3*ns3d_n) )
252:
253:     DO id = 1, ns3d_n
254:
255:         READ(12, *) number, ( ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
256:
257:     END DO
258:
259:     CALL set_nodes3d_bc(ns3d, ns3d_bc)
260:
261:     READ(12, *) dataname
262:
263:     CLOSE(12)
264:
265: !-----
266:
267:     OPEN(13, FILE = 'param_fea.dat')
268:
269:     READ(13, *) dataname
270:     READ(13, *) dataname
271:     READ(13, *) dataname
272:     READ(13, *) e
273:     READ(13, *) dataname
274:     READ(13, *) nu
275:     READ(13, *) dataname
276:     READ(13, *) rho
277:     READ(13, *) dataname
278:     READ(13, *) g
279:
280: !-----
281:

```

```

282:    ALLOCATE( esm3d_e(es3d_n) )
283:    ALLOCATE( esm3d_nu(es3d_n) )
284:
285:    ALLOCATE( efv3d_rho(es3d_n) )
286:
287:    DO ie = 1, es3d_n
288:
289:        esm3d_e(ie) = e
290:        esm3d_nu(ie) = nu
291:
292:        efv3d_rho(ie) = rho
293:
294:    END DO
295:
296:    CALL set_elemstiffmat3d_e_nu(esm3d, esm3d_e, esm3d_nu)
297:
298:    CALL set_elemexforcevec3d_rho(efv3d, efv3d_rho)
299:    CALL set_elemexforcevec3d_g(efv3d, g)
300:
301:    !-----
302:
303:    READ(13, *) dataname
304:
305:    ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
306:    ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
307:
308:    DO id_l = 1, fem3d_nnodes_loaded
309:
310:        READ(13, *) fem3d_id_loaded(id_l),          &
311:                ( fem3d_f_loaded(i, id_l), i = 1, 3 )
312:
313:    END DO
314:
315:    CALL set_fem3d_f_loaded          &
316:        (fem3d, fem3d_id_loaded, fem3d_f_loaded)
317:

```

```

318:      READ(13, *) dataname
319:
320:      ALLOCATE( efv3d_table_ie(efv3d_nelemboundaries) )
321:      ALLOCATE( efv3d_table_ma(efv3d_nelemboundaries) )
322:      ALLOCATE( efv3d_t(3, efv3d_nelemboundaries) )
323:
324:      DO ib = 1, efv3d_nelemboundaries
325:
326:          READ(13, *) number,                                &
327:              efv3d_table_ie(ib), efv3d_table_ma(ib), &
328:              ( efv3d_t(i, ib), i = 1, 3 )
329:
330:      END DO
331:
332:      CALL set_elemexforcevec3d_t                                &
333:          (efv3d, efv3d_table_ie, efv3d_table_ma, efv3d_t)
334:
335:      READ(13, *) dataname
336:
337:      CLOSE(13)
338:
339: !-----
340:
341:      DEALLOCATE( ns3d_x )
342:      DEALLOCATE( ns3d_u )
343:      DEALLOCATE( ns3d_bc )
344:
345:      DEALLOCATE( es3d_connectivity )
346:
347:      DEALLOCATE( esm3d_e )
348:      DEALLOCATE( esm3d_nu )
349:
350:      DEALLOCATE( efv3d_rho )
351:      DEALLOCATE( efv3d_table_ie )
352:      DEALLOCATE( efv3d_table_ma )
353:      DEALLOCATE( efv3d_t )

```

```

354:
355:     DEALLOCATE( fem3d_id_loaded )
356:     DEALLOCATE( fem3d_f_loaded )
357:
358: !-----
359:
360:     RETURN
361:
362: !#####
363:     END SUBROUTINE start_appli
364: !#####
365:
366:
367:     ! Run appli
368: !#####
369:     SUBROUTINE run_appli()
370: !#####
371:
372:     INTEGER :: ns3d_n
373:     INTEGER :: le3d_nnodes
374:     INTEGER :: es3d_n
375:     INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
376:     INTEGER :: i
377:     INTEGER :: id
378:     INTEGER :: na
379:     INTEGER :: ie
380:
381:     REAL(8), ALLOCATABLE :: ns3d_x(:, :)
382:     REAL(8), ALLOCATABLE :: ns3d_u(:)
383:     REAL(8), ALLOCATABLE :: esm3d_evec(:, :)
384:     REAL(8), ALLOCATABLE :: esm3d_svec(:, :), esm3d_s_mises(:)
385:
386: !-----
387:
388:     CALL cal_elements3d(es3d)
389:

```



```

390:      CALL cal_fem3d(fem3d)
391:
392: !-----
393:
394:      CALL get_nodes3d_n(ns3d, ns3d_n)
395:      ALLOCATE( ns3d_x(3, ns3d_n) )
396:      CALL get_nodes3d_x(ns3d, ns3d_x)
397:      ALLOCATE( ns3d_u(3*ns3d_n) )
398:      CALL get_nodes3d_u(ns3d, ns3d_u)
399:
400:      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
401:
402:      CALL get_elements3d_n(es3d, es3d_n)
403:      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
404:      CALL get_elements3d_connectivity(es3d, es3d_connectivity)
405:
406:      ALLOCATE( esm3d_evec(6, es3d_n) )
407:      ALLOCATE( esm3d_svec(6, es3d_n) )
408:      ALLOCATE( esm3d_s_mises(es3d_n) )
409:      CALL get_elemstiffmat3d_evec_svec      &
410:          (esm3d, esm3d_evec, esm3d_svec, esm3d_s_mises)
411:
412: !-----
413:
414:      OPEN(14, FILE = 'result.inp')
415:
416:      WRITE(14, '( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
417:
418:      DO id = 1, ns3d_n
419:
420:          WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )') &
421:              id, ( ns3d_x(i, id), i = 1, 3 )
422:
423:      END DO
424:
425:      DO ie = 1, es3d_n

```

```

426:
427:     WRITE(14, '( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )')      &
428:         ie, 1, ' hex',                                     &
429:         ( es3d_connectivity(na, ie), na = 1, le3d_nnodes)
430:
431: END DO
432:
433: WRITE(14, '( 4(I8, 1X) )') 1, 3
434: WRITE(14, '(A)') 'DISPLACEMENT, m'
435:
436: DO id = 1, ns3d_n
437:
438:     WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )')      &
439:         id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
440:
441: END DO
442:
443: WRITE(14, '( 14I8 )') 13, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
444: WRITE(14, '( (A, 1X) )') 'STRAIN_11, unit_unknown'
445: WRITE(14, '( (A, 1X) )') 'STRAIN_22, unit_unknown'
446: WRITE(14, '( (A, 1X) )') 'STRAIN_33, unit_unknown'
447: WRITE(14, '( (A, 1X) )') 'STRAIN_12, unit_unknown'
448: WRITE(14, '( (A, 1X) )') 'STRAIN_23, unit_unknown'
449: WRITE(14, '( (A, 1X) )') 'STRAIN_31, unit_unknown'
450: WRITE(14, '( (A, 1X) )') 'STRESS_11, Pa'
451: WRITE(14, '( (A, 1X) )') 'STRESS_22, Pa'
452: WRITE(14, '( (A, 1X) )') 'STRESS_33, Pa'
453: WRITE(14, '( (A, 1X) )') 'STRESS_12, Pa'
454: WRITE(14, '( (A, 1X) )') 'STRESS_23, Pa'
455: WRITE(14, '( (A, 1X) )') 'STRESS_31, Pa'
456: WRITE(14, '( (A, 1X) )') 'STRESS_MISES, Pa'
457:
458: DO ie = 1, es3d_n
459:
460:     WRITE(14, '( (I8, 1X), 13(E17.8, 1X) )')      &
461:         ie, ( esm3d_evec(i, ie), i = 1, 6 ),      &

```

```

462:          ( esm3d_svec(i, ie), i = 1, 6 ), esm3d_s_mises(ie)
463:
464:      END DO
465:
466:      CLOSE(14)
467:
468: !-----
469:
470:      DEALLOCATE( ns3d_x )
471:      DEALLOCATE( ns3d_u )
472:
473:      DEALLOCATE( es3d_connectivity )
474:
475:      DEALLOCATE( esm3d_evec )
476:      DEALLOCATE( esm3d_svec )
477:      DEALLOCATE( esm3d_s_mises )
478:
479: !-----
480:
481:      RETURN
482:
483: !#####
484:      END SUBROUTINE run_appli
485: !#####
486:
487:
488:      ! Finish appli
489: !#####
490:      SUBROUTINE finish_appli()
491: !#####
492:
493:      CALL del_nodes3d(ns3d)
494:      CALL del_localelement3d(le3d)
495:      CALL del_elements3d(es3d)
496:      CALL del_elemstiffmat3d(esm3d)
497:      CALL del_elemexforcevec3d(efv3d)

```

```

498:      CALL del_fem3d(fem3d)
499:
500: !-----
501:
502:      DEALLOCATE( ns3d )
503:      DEALLOCATE( le3d )
504:      DEALLOCATE( es3d )
505:      DEALLOCATE( esm3d )
506:      DEALLOCATE( efv3d )
507:      DEALLOCATE( fem3d )
508:
509: !-----
510:
511:      RETURN
512:
513: !#####
514:      END SUBROUTINE finish_appli
515: !#####
516:
517:
518: !#####
519:      END MODULE mod_appli

```

第 5 節のプログラム：単純せん断変形解析

4. アプリケーションモジュール mod_appli.f90 (直方体メッシング用)

```

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 :: efv3d_nelemboundaries
160:    INTEGER, ALLOCATABLE :: efv3d_table_ie(:)
161:    INTEGER, ALLOCATABLE :: efv3d_table_ma(:)
162:    INTEGER :: fem3d_ndofs
163:    INTEGER :: fem3d_nnodes_loaded
164:    INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
165:    INTEGER :: i
166:    INTEGER :: id
167:    INTEGER :: id_l
168:    INTEGER :: ma
169:    INTEGER :: na
170:    INTEGER :: ie
171:    INTEGER :: idof
172:    INTEGER :: ib
173:
174:    REAL (8), ALLOCATABLE :: ns3d_x(:, :)
175:    REAL (8), ALLOCATABLE :: ns3d_u(:)
176:    REAL (8), ALLOCATABLE :: es3d_volume(:)
177:    REAL (8) :: es3d_max_volume
178:    REAL (8) :: es3d_min_volume
179:    REAL (8) :: es3d_sum_volume
180:    REAL (8), ALLOCATABLE :: efv3d_t(:, :)
181:    REAL (8), ALLOCATABLE :: fem3d_f_loaded(:, :)
182:    REAL (8) :: rm3d_x_start(3)
183:    REAL (8) :: rm3d_x_end(3)
184:    REAL (8) :: rm3d_x_center(3)
185:    REAL (8) :: rm3d_length_x(3)
186:    REAL (8), ALLOCATABLE :: x_local(:, :)
187:
188: !-----

```

```

189:
190:     CALL cal_rectmesher3d(rm3d)
191:
192:     CALL cal_elements3d(es3d)
193:
194: !-----
195:
196:     CALL get_nodes3d_n(ns3d, ns3d_n)
197:
198:     ALLOCATE( ns3d_x(3, ns3d_n) )
199:     CALL get_nodes3d_x(ns3d, ns3d_x)
200:     ALLOCATE( ns3d_u(3*ns3d_n) )
201:     ns3d_u = 0.0D0
202:     ALLOCATE( ns3d_bc(3*ns3d_n) )
203:     ns3d_bc = 0
204:
205:     CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
206:     CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
207:     CALL get_localelement3d_nnodes_boundary(le3d, le3d_nnodes_boundary)
208:     ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
209:     CALL get_localelement3d_table_na(le3d, le3d_table_na)
210:
211:     CALL get_elements3d_n(es3d, es3d_n)
212:     ALLOCATE( es3d_volume(es3d_n) )
213:     CALL get_elements3d_volume           &
214:         (es3d, es3d_volume,             &
215:         es3d_max_volume, es3d_ie_max_volume, &
216:         es3d_min_volume, es3d_ie_min_volume, &
217:         es3d_sum_volume)
218:     ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
219:     CALL get_elements3d_connectivity(es3d, es3d_connectivity)
220:     CALL get_rectmesher3d_x_start_x_end  &
221:         (rm3d, rm3d_x_start, rm3d_x_end, &
222:         rm3d_x_center, rm3d_length_x)
223:
224:     ALLOCATE( x_local(3, le3d_nnodes_boundary) )

```

```

225:
226: !-----
227:
228:     ! Tensile deformation
229:     IF( prob .EQ. 1 ) THEN
230:
231:         !-----
232:
233:         id_l = 0
234:         ib = 0
235:
236:         DO id = 1, ns3d_n
237:
238:             IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
239:
240:                 id_l = id_l+1
241:
242:             END IF
243:
244:         END DO
245:
246:         fem3d_nnodes_loaded = id_l
247:         efv3d_nelemboundaries = ib
248:
249:         !-----
250:
251:         ! Tensile deformation
252:         ELSE IF( prob .EQ. 2 ) THEN
253:
254:             !-----
255:
256:             id_l = 0
257:             ib = 0
258:
259:             DO ie = 1, es3d_n
260:

```

```

261:      DO ma = 1, le3d_nboundaries
262:
263:          na = le3d_table_na(1, ma)
264:          id = es3d_connectivity(na, ie)
265:          x_local(1, 1) = ns3d_x(1, id)
266:          x_local(2, 1) = ns3d_x(2, id)
267:          x_local(3, 1) = ns3d_x(3, id)
268:
269:          na = le3d_table_na(2, ma)
270:          id = es3d_connectivity(na, ie)
271:          x_local(1, 2) = ns3d_x(1, id)
272:          x_local(2, 2) = ns3d_x(2, id)
273:          x_local(3, 2) = ns3d_x(3, id)
274:
275:          na = le3d_table_na(3, ma)
276:          id = es3d_connectivity(na, ie)
277:          x_local(1, 3) = ns3d_x(1, id)
278:          x_local(2, 3) = ns3d_x(2, id)
279:          x_local(3, 3) = ns3d_x(3, id)
280:
281:          na = le3d_table_na(4, ma)
282:          id = es3d_connectivity(na, ie)
283:          x_local(1, 4) = ns3d_x(1, id)
284:          x_local(2, 4) = ns3d_x(2, id)
285:          x_local(3, 4) = ns3d_x(3, id)
286:
287:          IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
288:              ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
289:              ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
290:              ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
291:
292:              ib = ib+1
293:
294:          END IF
295:
296:      END DO

```

```

297:
298:     END DO
299:
300:     fem3d_nnodes_loaded = id_l
301:     efv3d_nelemboundaries = ib
302:
303:     !-----
304:
305:     ! Shear deformation
306:     ELSE IF( prob.EQ. 3 ) THEN
307:
308:     !-----
309:
310:     id_l = 0
311:     ib = 0
312:
313:     DO ie = 1, es3d_n
314:
315:         DO ma = 1, le3d_nboundaries
316:
317:             na = le3d_table_na(1, ma)
318:             id = es3d_connectivity(na, ie)
319:             x_local(1, 1) = ns3d_x(1, id)
320:             x_local(2, 1) = ns3d_x(2, id)
321:             x_local(3, 1) = ns3d_x(3, id)
322:
323:             na = le3d_table_na(2, ma)
324:             id = es3d_connectivity(na, ie)
325:             x_local(1, 2) = ns3d_x(1, id)
326:             x_local(2, 2) = ns3d_x(2, id)
327:             x_local(3, 2) = ns3d_x(3, id)
328:
329:             na = le3d_table_na(3, ma)
330:             id = es3d_connectivity(na, ie)
331:             x_local(1, 3) = ns3d_x(1, id)
332:             x_local(2, 3) = ns3d_x(2, id)

```

```

333:      x_local(3, 3) = ns3d_x(3, id)
334:
335:      na = le3d_table_na(4, ma)
336:      id = es3d_connectivity(na, ie)
337:      x_local(1, 4) = ns3d_x(1, id)
338:      x_local(2, 4) = ns3d_x(2, id)
339:      x_local(3, 4) = ns3d_x(3, id)
340:
341:      IF( ( DABS( x_local(1, 1)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
342:          ( DABS( x_local(1, 2)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
343:          ( DABS( x_local(1, 3)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
344:          ( DABS( x_local(1, 4)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) ) THEN
345:
346:          ib = ib+1
347:
348:      ELSE IF( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
349:              ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
350:              ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
351:              ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
352:
353:          ib = ib+1
354:
355:      END IF
356:
357:      IF( ( DABS( x_local(3, 1)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
358:          ( DABS( x_local(3, 2)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
359:          ( DABS( x_local(3, 3)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
360:          ( DABS( x_local(3, 4)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) ) THEN
361:
362:          ib = ib+1
363:
364:      END IF
365:
366:      END DO
367:
368:      END DO

```

```

369:
370:     fem3d_nnodes_loaded = id_l
371:     efv3d_nelemboundaries = ib
372:
373:     !-----
374:
375: END IF
376:
377: !-----
378:
379: CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
380: CALL init_elemexforcevec3d                                &
381:     (efv3d, ns3d, le3d, es3d, efv3d_nelemboundaries)
382: CALL init_fem3d                                            &
383:     (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
384:     fem3d_nnodes_loaded)
385:
386: ALLOCATE( efv3d_table_ie(efv3d_nelemboundaries) )
387: efv3d_table_ie = 0
388: ALLOCATE( efv3d_table_ma(efv3d_nelemboundaries) )
389: efv3d_table_ma = 0
390: ALLOCATE( efv3d_t(3, efv3d_nelemboundaries) )
391: efv3d_t = 0.0D0
392:
393: CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
394: ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
395: fem3d_id_loaded = 0
396: ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
397: fem3d_f_loaded = 0.0D0
398:
399: !-----
400:
401: ! Tensile deformation
402: IF( ( prob .EQ. 1 ) .OR. ( prob .EQ. 2 ) ) THEN
403:
404:     !-----

```

```

405:
406:     DO id = 1, ns3d_n
407:
408:         IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN
409:
410:             idof = 3*(id-1)+1
411:
412:             ns3d_u(idof) = 0.0D0
413:             ns3d_bc(idof) = 1
414:
415:             IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
416:
417:                 idof = 3*(id-1)+2
418:
419:                 ns3d_u(idof) = 0.0D0
420:                 ns3d_bc(idof) = 1
421:
422:             END IF
423:
424:             IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
425:
426:                 idof = 3*(id-1)+3
427:
428:                 ns3d_u(idof) = 0.0D0
429:                 ns3d_bc(idof) = 1
430:
431:             END IF
432:
433:         END IF
434:
435:     END DO
436:
437:     !-----
438:
439:     ! Shear deformation
440:     ELSE IF( prob .EQ. 3 ) THEN

```



```

441:
442:     !-----
443:
444:     DO id = 1, ns3d_n
445:
446:         IF( DABS( ns3d_x(3, id)-rm3d_x_start(3) ) .LT. EPSILON(1.0D0) ) THEN
447:
448:             DO i = 1, 3
449:
450:                 idof = 3*(id-1)+i
451:
452:                 ns3d_u(idof) = 0.0D0
453:                 ns3d_bc(idof) = 1
454:
455:             END DO
456:
457:         END IF
458:
459:     END DO
460:
461:     !-----
462:
463: END IF
464:
465: !-----
466:
467: ! Tensile deformation
468: IF( prob .EQ. 1 ) THEN
469:
470:     !-----
471:
472:     id_l = 0
473:
474:     DO id = 1, ns3d_n
475:
476:         IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN

```

```

477:
478:     id_l = id_l+1
479:
480:     fem3d_id_loaded(id_l) = id
481:     fem3d_f_loaded(1, id_l) = 2.5D6
482:
483:     IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
484:
485:         fem3d_f_loaded(1, id_l) = 5.0D6
486:
487:     END IF
488:
489:     IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
490:
491:         fem3d_f_loaded(1, id_l) = 5.0D6
492:
493:     END IF
494:
495:     IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
496:
497:         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
498:
499:             fem3d_f_loaded(1, id_l) = 1.0D7
500:
501:         END IF
502:
503:     END IF
504:
505: END IF
506:
507: END DO
508:
509: !-----
510:
511: ! Tensile deformation
512: ELSE IF( prob .EQ. 2 ) THEN

```

```

513:
514:     !-----
515:
516:     ib = 0
517:
518:     DO ie = 1, es3d_n
519:
520:         DO ma = 1, le3d_nboundaries
521:
522:             na = le3d_table_na(1, ma)
523:             id = es3d_connectivity(na, ie)
524:             x_local(1, 1) = ns3d_x(1, id)
525:             x_local(2, 1) = ns3d_x(2, id)
526:             x_local(3, 1) = ns3d_x(3, id)
527:
528:             na = le3d_table_na(2, ma)
529:             id = es3d_connectivity(na, ie)
530:             x_local(1, 2) = ns3d_x(1, id)
531:             x_local(2, 2) = ns3d_x(2, id)
532:             x_local(3, 2) = ns3d_x(3, id)
533:
534:             na = le3d_table_na(3, ma)
535:             id = es3d_connectivity(na, ie)
536:             x_local(1, 3) = ns3d_x(1, id)
537:             x_local(2, 3) = ns3d_x(2, id)
538:             x_local(3, 3) = ns3d_x(3, id)
539:
540:             na = le3d_table_na(4, ma)
541:             id = es3d_connectivity(na, ie)
542:             x_local(1, 4) = ns3d_x(1, id)
543:             x_local(2, 4) = ns3d_x(2, id)
544:             x_local(3, 4) = ns3d_x(3, id)
545:
546:             IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
547:                 ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
548:                 ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &

```

```

549:      ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
550:
551:      ib = ib+1
552:
553:      efv3d_table_ie(ib) = ie
554:      efv3d_table_ma(ib) = ma
555:      efv3d_t(1, ib) = -4.0D7
556:      efv3d_t(2, ib) = 0.0D0
557:      efv3d_t(3, ib) = 0.0D0
558:
559:      END IF
560:
561:      END DO
562:
563:      END DO
564:
565:      !-----
566:
567:      ! Shear deformation
568:      ELSE IF( prob .EQ. 3 ) THEN
569:
570:      !-----
571:
572:      ib = 0
573:
574:      DO ie = 1, es3d_n
575:
576:      DO ma = 1, le3d_nboundaries
577:
578:      na = le3d_table_na(1, ma)
579:      id = es3d_connectivity(na, ie)
580:      x_local(1, 1) = ns3d_x(1, id)
581:      x_local(2, 1) = ns3d_x(2, id)
582:      x_local(3, 1) = ns3d_x(3, id)
583:
584:      na = le3d_table_na(2, ma)

```

```

585:      id = es3d_connectivity(na, ie)
586:      x_local(1, 2) = ns3d_x(1, id)
587:      x_local(2, 2) = ns3d_x(2, id)
588:      x_local(3, 2) = ns3d_x(3, id)
589:
590:      na = le3d_table_na(3, ma)
591:      id = es3d_connectivity(na, ie)
592:      x_local(1, 3) = ns3d_x(1, id)
593:      x_local(2, 3) = ns3d_x(2, id)
594:      x_local(3, 3) = ns3d_x(3, id)
595:
596:      na = le3d_table_na(4, ma)
597:      id = es3d_connectivity(na, ie)
598:      x_local(1, 4) = ns3d_x(1, id)
599:      x_local(2, 4) = ns3d_x(2, id)
600:      x_local(3, 4) = ns3d_x(3, id)
601:
602:      IF( ( DABS( x_local(1, 1)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
603:          ( DABS( x_local(1, 2)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
604:          ( DABS( x_local(1, 3)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
605:          ( DABS( x_local(1, 4)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) ) THEN
606:
607:          ib = ib+1
608:
609:          efv3d_table_ie(ib) = ie
610:          efv3d_table_ma(ib) = ma
611:          efv3d_t(1, ib) = 0.0D0
612:          efv3d_t(2, ib) = 0.0D0
613:          efv3d_t(3, ib) = -1.0D8
614:
615:      ELSE IF( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
616:              ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
617:              ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
618:              ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
619:
620:          ib = ib+1

```

```

621:
622:     efv3d_table_ie(ib) = ie
623:     efv3d_table_ma(ib) = ma
624:     efv3d_t(1, ib) = 0.0D0
625:     efv3d_t(2, ib) = 0.0D0
626:     efv3d_t(3, ib) = 1.0D8
627:
628:     END IF
629:
630:     IF ( ( DABS( x_local(3, 1)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
631:         ( DABS( x_local(3, 2)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
632:         ( DABS( x_local(3, 3)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
633:         ( DABS( x_local(3, 4)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) ) THEN
634:
635:         ib = ib+1
636:
637:         efv3d_table_ie(ib) = ie
638:         efv3d_table_ma(ib) = ma
639:         efv3d_t(1, ib) = 1.0D8
640:         efv3d_t(2, ib) = 0.0D0
641:         efv3d_t(3, ib) = 0.0D0
642:
643:     END IF
644:
645: END DO
646:
647: END DO
648:
649: !-----
650:
651: END IF
652:
653: CALL set_fem3d_f_loaded                                &
654:     (fem3d, fem3d_id_loaded, fem3d_f_loaded)
655: CALL set_elemexforcevec3d_t                            &
656:     (efv3d, efv3d_table_ie, efv3d_table_ma, efv3d_t)

```

```

657:
658: !-----
659:
660:     OPEN(10, FILE = 'mesh.dat')
661:
662:     WRITE(10, ' (A)') '!NODE'
663:
664:     DO id = 1, ns3d_n
665:
666:         WRITE( 10, ' ( I8, 3(A, E17.8) )' )           &
667:             id, ( ', ', ns3d_x(i, id), i = 1, 3 )
668:
669:     END DO
670:
671:     WRITE( 10, ' (A, 3(A, I3) )' )           &
672:         '!ELEMENT', ', ', ', ', le3d_nboundaries, &
673:         ', ', ', ', le3d_nnodes, ', ', ', 2
674:
675:
676:     DO ie = 1, es3d_n
677:
678:         WRITE( 10, ' ( I8, 27(A, I8) )' )           &
679:             ie, ( ', ', es3d_connectivity(na, ie), &
680:                 na = 1, le3d_nnodes )
681:
682:     END DO
683:
684:     WRITE(10, ' (A)') '!END'
685:
686:     CLOSE(10)
687:
688: !-----
689:
690:     OPEN(11, FILE = 'ic.dat')
691:
692:     WRITE(11, ' (A)') '!DISPLACEMENT'

```

```

693:
694:      DO id = 1, ns3d_n
695:
696:          WRITE( 11, ' (I8, 3(A, E17.8) )' )          &
697:              id, ( ' ', ' ', ns3d_u( 3*(id-1)+i ), i = 1, 3 )
698:
699:      END DO
700:
701:      WRITE(11, ' (A)' ) ' !END'
702:
703:      CLOSE(11)
704:
705: !-----
706:
707:      OPEN(12, FILE = 'bc.dat')
708:
709:      WRITE(12, ' (A)' ) ' !DISPLACEMENT'
710:
711:      DO id = 1, ns3d_n
712:
713:          WRITE( 12, ' (I8, 3(A, I8) )' )          &
714:              id, ( ' ', ' ', ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
715:
716:      END DO
717:
718:      WRITE(12, ' (A)' ) ' !END'
719:
720:      CLOSE(12)
721:
722: !-----
723:
724:      WRITE(13, ' (A)' ) ' !F_LOADED'
725:
726:      DO id_l = 1, fem3d_nnodes_loaded
727:
728:          WRITE( 13, ' (I8, 3(A, E17.8) )' )          &

```



```

729:          fem3d_id_loaded(id_l),          &
730:          ( ', ', fem3d_f_loaded(i, id_l), i = 1, 3 )
731:
732:      END DO
733:
734:      WRITE(13, ' (A)') '!TRACTION'
735:
736:      DO ib = 1, efv3d_nelemboundaries
737:
738:          WRITE( 13, ' (I8, 2(A, I8), 3(A, E17.8) )' ) &
739:              ib, ', ', efv3d_table_ie(ib),          &
740:              ', ', efv3d_table_ma(ib),              &
741:              ( ', ', efv3d_t(i, ib), i = 1, 3 )
742:
743:      END DO
744:
745:      WRITE(13, ' (A)') '!END'
746:
747:      CLOSE(13)
748:
749: !-----
750:
751:      OPEN(14, FILE = 'mesh. inp')
752:
753:      WRITE(14, ' ( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
754:
755:      DO id = 1, ns3d_n
756:
757:          WRITE( 14, ' ( (I8, 1X), 3(E17.8, 1X) )' ) &
758:              id, ( ns3d_x(i, id), i = 1, 3 )
759:
760:      END DO
761:
762:      DO ie = 1, es3d_n
763:
764:          WRITE( 14, ' ( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )' )          &

```

```

765:          ie, 1, ' hex',                                &
766:          ( es3d_connectivity(na, ie), na = 1, le3d_nnodes )
767:
768:      END DO
769:
770:      WRITE(14, '( 4(I8, 1X) )') 1, 3
771:      WRITE(14, '(A)') 'DISPLACEMENT, m'
772:
773:      DO id = 1, ns3d_n
774:
775:          WRITE( 14, '( (I8, 1X), 3(E17.8, 1X) )' )      &
776:              id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
777:
778:      END DO
779:
780:      WRITE(14, '( 14I8 )') 1, 1
781:      WRITE(14, '( (A, 1X) )') 'VOLUME, m3'
782:
783:      DO ie = 1, es3d_n
784:
785:          WRITE( 14, '( (I8, 1X), (E17.8, 1X) )' ) &
786:              ie, es3d_volume(ie)
787:
788:      END DO
789:
790:      CLOSE(14)
791:
792: !-----
793:
794:      DEALLOCATE( ns3d_x )
795:      DEALLOCATE( ns3d_u )
796:
797:      DEALLOCATE( le3d_table_na )
798:
799:      DEALLOCATE( es3d_volume )
800:      DEALLOCATE( es3d_connectivity )

```

```

801:
802:     DEALLOCATE( efv3d_table_ie )
803:     DEALLOCATE( efv3d_table_ma )
804:     DEALLOCATE( efv3d_t )
805:
806:     DEALLOCATE( fem3d_id_loaded )
807:     DEALLOCATE( fem3d_f_loaded )
808:
809:     DEALLOCATE( x_local )
810:
811: !-----
812:
813:     RETURN
814:
815: !#####
816:     END SUBROUTINE run_appli
817: !#####
818:
819:
820: !#####
821:     SUBROUTINE finish_appli()
822: !#####
823:
824:     CALL del_nodes3d(ns3d)
825:     CALL del_localelement3d(le3d)
826:     CALL del_elements3d(es3d)
827:     CALL del_elemstiffmat3d(esm3d)
828:     CALL del_elemexforcevec3d(efv3d)
829:     CALL del_fem3d(fem3d)
830:     CALL del_rectmesher3d(rm3d)
831:
832: !-----
833:
834:     DEALLOCATE( ns3d )
835:     DEALLOCATE( le3d )
836:     DEALLOCATE( es3d )

```

```

837:      DEALLOCATE( esm3d )
838:      DEALLOCATE( efv3d )
839:      DEALLOCATE( fem3d )
840:      DEALLOCATE( rm3d )
841:
842: !-----
843:
844:      RETURN
845:
846: !#####
847:      END SUBROUTINE finish_appli
848: !#####
849:
850:
851: !#####
852:      END MODULE mod_appli

```

第 6 節のプログラム：単純支持梁の曲げ変形解析

5. アプリケーションモジュール mod_appli.f90 (直方体メッシング用)

```

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 :: efv3d_nelemboundaries
160:      INTEGER, ALLOCATABLE :: efv3d_table_ie(:)
161:      INTEGER, ALLOCATABLE :: efv3d_table_ma(:)
162:      INTEGER :: fem3d_ndofs
163:      INTEGER :: fem3d_nnodes_loaded
164:      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
165:      INTEGER :: i
166:      INTEGER :: id
167:      INTEGER :: id_l
168:      INTEGER :: ma
169:      INTEGER :: na
170:      INTEGER :: ie
171:      INTEGER :: idof
172:      INTEGER :: ib
173:
174:      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
175:      REAL(8), ALLOCATABLE :: ns3d_u(:)
176:      REAL(8), ALLOCATABLE :: es3d_volume(:)
177:      REAL(8) :: es3d_max_volume
178:      REAL(8) :: es3d_min_volume
179:      REAL(8) :: es3d_sum_volume
180:      REAL(8), ALLOCATABLE :: efv3d_t(:, :)
181:      REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
182:      REAL(8) :: rm3d_x_start(3)
183:      REAL(8) :: rm3d_x_end(3)
184:      REAL(8) :: rm3d_x_center(3)
185:      REAL(8) :: rm3d_length_x(3)
186:      REAL(8), ALLOCATABLE :: x_local(:, :)
187:
188: !-----
189:
190:      CALL cal_rectmesher3d(rm3d)
191:
192:      CALL cal_elements3d(es3d)
193:
194: !-----

```

```

195:
196:     CALL get_nodes3d_n(ns3d, ns3d_n)
197:
198:     ALLOCATE( ns3d_x(3, ns3d_n) )
199:     CALL get_nodes3d_x(ns3d, ns3d_x)
200:     ALLOCATE( ns3d_u(3*ns3d_n) )
201:     ns3d_u = 0.0D0
202:     ALLOCATE( ns3d_bc(3*ns3d_n) )
203:     ns3d_bc = 0
204:
205:     CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
206:     CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
207:     CALL get_localelement3d_nnodes_boundary(le3d, le3d_nnodes_boundary)
208:     ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
209:     CALL get_localelement3d_table_na(le3d, le3d_table_na)
210:
211:     CALL get_elements3d_n(es3d, es3d_n)
212:     ALLOCATE( es3d_volume(es3d_n) )
213:     CALL get_elements3d_volume           &
214:         (es3d, es3d_volume,             &
215:         es3d_max_volume, es3d_ie_max_volume, &
216:         es3d_min_volume, es3d_ie_min_volume, &
217:         es3d_sum_volume)
218:     ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
219:     CALL get_elements3d_connectivity(es3d, es3d_connectivity)
220:     CALL get_rectmesher3d_x_start_x_end  &
221:         (rm3d, rm3d_x_start, rm3d_x_end, &
222:         rm3d_x_center, rm3d_length_x)
223:
224:     ALLOCATE( x_local(3, le3d_nnodes_boundary) )
225:
226: !-----
227:
228:     ! Tensile deformation
229:     IF( prob .EQ. 1 ) THEN
230:

```

```

231:      !-----
232:
233:      id_l = 0
234:      ib = 0
235:
236:      DO id = 1, ns3d_n
237:
238:      IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
239:
240:      id_l = id_l+1
241:
242:      END IF
243:
244:      END DO
245:
246:      fem3d_nnodes_loaded = id_l
247:      efv3d_nelemboundaries = ib
248:
249:      !-----
250:
251:      ! Tensile deformation
252:      ELSE IF( prob .EQ. 2 ) THEN
253:
254:      !-----
255:
256:      id_l = 0
257:      ib = 0
258:
259:      DO ie = 1, es3d_n
260:
261:      DO ma = 1, le3d_nboundaries
262:
263:      na = le3d_table_na(1, ma)
264:      id = es3d_connectivity(na, ie)
265:      x_local(1, 1) = ns3d_x(1, id)
266:      x_local(2, 1) = ns3d_x(2, id)

```

```

267:      x_local(3, 1) = ns3d_x(3, id)
268:
269:      na = le3d_table_na(2, ma)
270:      id = es3d_connectivity(na, ie)
271:      x_local(1, 2) = ns3d_x(1, id)
272:      x_local(2, 2) = ns3d_x(2, id)
273:      x_local(3, 2) = ns3d_x(3, id)
274:
275:      na = le3d_table_na(3, ma)
276:      id = es3d_connectivity(na, ie)
277:      x_local(1, 3) = ns3d_x(1, id)
278:      x_local(2, 3) = ns3d_x(2, id)
279:      x_local(3, 3) = ns3d_x(3, id)
280:
281:      na = le3d_table_na(4, ma)
282:      id = es3d_connectivity(na, ie)
283:      x_local(1, 4) = ns3d_x(1, id)
284:      x_local(2, 4) = ns3d_x(2, id)
285:      x_local(3, 4) = ns3d_x(3, id)
286:
287:      IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
288:          ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
289:          ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
290:          ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
291:
292:          ib = ib+1
293:
294:      END IF
295:
296:  END DO
297:
298:  END DO
299:
300:  fem3d_nnodes_loaded = id_l
301:  efv3d_nelemboundaries = ib
302:

```

```

303:      !-----
304:
305:      ! Shear deformation
306:      ELSE IF( prob .EQ. 3 ) THEN
307:
308:      !-----
309:
310:      id_l = 0
311:      ib = 0
312:
313:      DO ie = 1, es3d_n
314:
315:      DO ma = 1, le3d_nboundaries
316:
317:      na = le3d_table_na(1, ma)
318:      id = es3d_connectivity(na, ie)
319:      x_local(1, 1) = ns3d_x(1, id)
320:      x_local(2, 1) = ns3d_x(2, id)
321:      x_local(3, 1) = ns3d_x(3, id)
322:
323:      na = le3d_table_na(2, ma)
324:      id = es3d_connectivity(na, ie)
325:      x_local(1, 2) = ns3d_x(1, id)
326:      x_local(2, 2) = ns3d_x(2, id)
327:      x_local(3, 2) = ns3d_x(3, id)
328:
329:      na = le3d_table_na(3, ma)
330:      id = es3d_connectivity(na, ie)
331:      x_local(1, 3) = ns3d_x(1, id)
332:      x_local(2, 3) = ns3d_x(2, id)
333:      x_local(3, 3) = ns3d_x(3, id)
334:
335:      na = le3d_table_na(4, ma)
336:      id = es3d_connectivity(na, ie)
337:      x_local(1, 4) = ns3d_x(1, id)
338:      x_local(2, 4) = ns3d_x(2, id)

```

```

339:      x_local(3, 4) = ns3d_x(3, id)
340:
341:      IF ( ( DABS( x_local(1, 1)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
342:          ( DABS( x_local(1, 2)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
343:          ( DABS( x_local(1, 3)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
344:          ( DABS( x_local(1, 4)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) ) THEN
345:
346:          ib = ib+1
347:
348:      ELSE IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
349:              ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
350:              ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
351:              ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
352:
353:          ib = ib+1
354:
355:      END IF
356:
357:      IF ( ( DABS( x_local(3, 1)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
358:          ( DABS( x_local(3, 2)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
359:          ( DABS( x_local(3, 3)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
360:          ( DABS( x_local(3, 4)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) ) THEN
361:
362:          ib = ib+1
363:
364:      END IF
365:
366:  END DO
367:
368:  END DO
369:
370:  fem3d_nnodes_loaded = id_l
371:  efv3d_nelemboundaries = ib
372:
373:  !-----
374:

```

```

375:      ! Bending deformation
376:      ELSE IF( prob .EQ. 4 ) THEN
377:
378:      !-----
379:
380:      id_l = 0
381:      ib = 0
382:
383:      DO id = 1, ns3d_n
384:
385:      IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
386:
387:      id_l = id_l+1
388:
389:      END IF
390:
391:      END DO
392:
393:      fem3d_nnodes_loaded = id_l
394:      efv3d_nelemboundaries = ib
395:
396:      !-----
397:
398:      END IF
399:
400:      !-----
401:
402:      CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
403:      CALL init_elemexforcevec3d                                     &
404:      (efv3d, ns3d, le3d, es3d, efv3d_nelemboundaries)
405:      CALL init_fem3d                                               &
406:      (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
407:      fem3d_nnodes_loaded)
408:
409:      ALLOCATE( efv3d_table_ie(efv3d_nelemboundaries) )
410:      efv3d_table_ie = 0

```

```

411:    ALLOCATE( efv3d_table_ma(efv3d_nelemboundaries) )
412:    efv3d_table_ma = 0
413:    ALLOCATE( efv3d_t(3, efv3d_nelemboundaries) )
414:    efv3d_t = 0.0D0
415:
416:    CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
417:    ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
418:    fem3d_id_loaded = 0
419:    ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
420:    fem3d_f_loaded = 0.0D0
421:
422:    !-----
423:
424:    ! Tensile deformation
425:    IF( ( prob .EQ. 1 ) .OR. ( prob .EQ. 2 ) ) THEN
426:
427:    !-----
428:
429:    DO id = 1, ns3d_n
430:
431:    IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN
432:
433:        idof = 3*(id-1)+1
434:
435:        ns3d_u(idof) = 0.0D0
436:        ns3d_bc(idof) = 1
437:
438:    IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
439:
440:        idof = 3*(id-1)+2
441:
442:        ns3d_u(idof) = 0.0D0
443:        ns3d_bc(idof) = 1
444:
445:    END IF
446:

```



```

447:      IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
448:
449:          idof = 3*(id-1)+3
450:
451:          ns3d_u(idof) = 0.0D0
452:          ns3d_bc(idof) = 1
453:
454:      END IF
455:
456:  END IF
457:
458:  END DO
459:
460:  !-----
461:
462:  ! Shear deformation
463:  ELSE IF( prob .EQ. 3 ) THEN
464:
465:  !-----
466:
467:  DO id = 1, ns3d_n
468:
469:      IF( DABS( ns3d_x(3, id)-rm3d_x_start(3) ) .LT. EPSILON(1.0D0) ) THEN
470:
471:          DO i = 1, 3
472:
473:              idof = 3*(id-1)+i
474:
475:              ns3d_u(idof) = 0.0D0
476:              ns3d_bc(idof) = 1
477:
478:          END DO
479:
480:      END IF
481:
482:  END DO

```

```

483:
484:     !-----
485:
486:     ! Bending deformation
487:     ELSE IF( prob .EQ. 4 ) THEN
488:
489:     !-----
490:
491:     DO id = 1, ns3d_n
492:
493:     IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN
494:
495:         idof = 3*(id-1)+1
496:
497:         ns3d_u(idof) = 0.0D0
498:         ns3d_bc(idof) = 1
499:
500:         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
501:
502:             idof = 3*(id-1)+2
503:
504:             ns3d_u(idof) = 0.0D0
505:             ns3d_bc(idof) = 1
506:
507:         END IF
508:
509:         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
510:
511:             idof = 3*(id-1)+3
512:
513:             ns3d_u(idof) = 0.0D0
514:             ns3d_bc(idof) = 1
515:
516:         END IF
517:
518:     END IF

```

```

519:
520:     END DO
521:
522:     !-----
523:
524: END IF
525:
526: !-----
527:
528: ! Tensile deformation
529: IF( prob .EQ. 1 ) THEN
530:
531:     !-----
532:
533:     id_l = 0
534:
535:     DO id = 1, ns3d_n
536:
537:         IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
538:
539:             id_l = id_l+1
540:
541:             fem3d_id_loaded(id_l) = id
542:             fem3d_f_loaded(1, id_l) = 2.5D6
543:
544:             IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
545:
546:                 fem3d_f_loaded(1, id_l) = 5.0D6
547:
548:             END IF
549:
550:             IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
551:
552:                 fem3d_f_loaded(1, id_l) = 5.0D6
553:
554:             END IF

```

```

555:
556:     IF ( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
557:
558:     IF ( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN
559:
560:         fem3d_f_loaded(1, id_l) = 1.0D7
561:
562:     END IF
563:
564: END IF
565:
566: END IF
567:
568: END DO
569:
570: !-----
571:
572: ! Tensile deformation
573: ELSE IF( prob .EQ. 2 ) THEN
574:
575: !-----
576:
577:     ib = 0
578:
579:     DO ie = 1, es3d_n
580:
581:         DO ma = 1, le3d_nboundaries
582:
583:             na = le3d_table_na(1, ma)
584:             id = es3d_connectivity(na, ie)
585:             x_local(1, 1) = ns3d_x(1, id)
586:             x_local(2, 1) = ns3d_x(2, id)
587:             x_local(3, 1) = ns3d_x(3, id)
588:
589:             na = le3d_table_na(2, ma)
590:             id = es3d_connectivity(na, ie)

```

```

591:      x_local(1, 2) = ns3d_x(1, id)
592:      x_local(2, 2) = ns3d_x(2, id)
593:      x_local(3, 2) = ns3d_x(3, id)
594:
595:      na = le3d_table_na(3, ma)
596:      id = es3d_connectivity(na, ie)
597:      x_local(1, 3) = ns3d_x(1, id)
598:      x_local(2, 3) = ns3d_x(2, id)
599:      x_local(3, 3) = ns3d_x(3, id)
600:
601:      na = le3d_table_na(4, ma)
602:      id = es3d_connectivity(na, ie)
603:      x_local(1, 4) = ns3d_x(1, id)
604:      x_local(2, 4) = ns3d_x(2, id)
605:      x_local(3, 4) = ns3d_x(3, id)
606:
607:      IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
608:          ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
609:          ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
610:          ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
611:
612:          ib = ib+1
613:
614:          efv3d_table_ie(ib) = ie
615:          efv3d_table_ma(ib) = ma
616:          efv3d_t(1, ib) = -4.0D7
617:          efv3d_t(2, ib) = 0.0D0
618:          efv3d_t(3, ib) = 0.0D0
619:
620:      END IF
621:
622:      END DO
623:
624:      END DO
625:
626:      !-----

```

```

627:
628:      ! Shear deformation
629:      ELSE IF( prob .EQ. 3 ) THEN
630:
631:      !-----
632:
633:      ib = 0
634:
635:      DO ie = 1, es3d_n
636:
637:      DO ma = 1, le3d_nboundaries
638:
639:          na = le3d_table_na(1, ma)
640:          id = es3d_connectivity(na, ie)
641:          x_local(1, 1) = ns3d_x(1, id)
642:          x_local(2, 1) = ns3d_x(2, id)
643:          x_local(3, 1) = ns3d_x(3, id)
644:
645:          na = le3d_table_na(2, ma)
646:          id = es3d_connectivity(na, ie)
647:          x_local(1, 2) = ns3d_x(1, id)
648:          x_local(2, 2) = ns3d_x(2, id)
649:          x_local(3, 2) = ns3d_x(3, id)
650:
651:          na = le3d_table_na(3, ma)
652:          id = es3d_connectivity(na, ie)
653:          x_local(1, 3) = ns3d_x(1, id)
654:          x_local(2, 3) = ns3d_x(2, id)
655:          x_local(3, 3) = ns3d_x(3, id)
656:
657:          na = le3d_table_na(4, ma)
658:          id = es3d_connectivity(na, ie)
659:          x_local(1, 4) = ns3d_x(1, id)
660:          x_local(2, 4) = ns3d_x(2, id)
661:          x_local(3, 4) = ns3d_x(3, id)
662:

```

```

663:      IF ( ( DABS( x_local(1, 1)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
664:          ( DABS( x_local(1, 2)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
665:          ( DABS( x_local(1, 3)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) .AND. &
666:          ( DABS( x_local(1, 4)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) ) THEN
667:
668:          ib = ib+1
669:
670:          efv3d_table_ie(ib) = ie
671:          efv3d_table_ma(ib) = ma
672:          efv3d_t(1, ib) = 0.0D0
673:          efv3d_t(2, ib) = 0.0D0
674:          efv3d_t(3, ib) = -1.0D8
675:
676:      ELSE IF ( ( DABS( x_local(1, 1)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
677:          ( DABS( x_local(1, 2)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
678:          ( DABS( x_local(1, 3)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) .AND. &
679:          ( DABS( x_local(1, 4)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) ) THEN
680:
681:          ib = ib+1
682:
683:          efv3d_table_ie(ib) = ie
684:          efv3d_table_ma(ib) = ma
685:          efv3d_t(1, ib) = 0.0D0
686:          efv3d_t(2, ib) = 0.0D0
687:          efv3d_t(3, ib) = 1.0D8
688:
689:      END IF
690:
691:      IF ( ( DABS( x_local(3, 1)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
692:          ( DABS( x_local(3, 2)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
693:          ( DABS( x_local(3, 3)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) .AND. &
694:          ( DABS( x_local(3, 4)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) ) THEN
695:
696:          ib = ib+1
697:
698:          efv3d_table_ie(ib) = ie

```

```

699:      efv3d_table_ma(ib) = ma
700:      efv3d_t(1, ib) = 1.0D8
701:      efv3d_t(2, ib) = 0.0D0
702:      efv3d_t(3, ib) = 0.0D0
703:
704:      END IF
705:
706:      END DO
707:
708:      END DO
709:
710:      !-----
711:
712:      ! Bending deformation
713:      ELSE IF( prob .EQ. 4 ) THEN
714:
715:      !-----
716:
717:      id_l = 0
718:
719:      DO id = 1, ns3d_n
720:
721:      IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
722:
723:      id_l = id_l+1
724:
725:      fem3d_id_loaded(id_l) = id
726:      fem3d_f_loaded(1, id_l) = 0.0D0
727:
728:      IF( DABS( ns3d_x(3, id)-rm3d_x_start(3) ) .LT. EPSILON(1.0D0) ) THEN
729:
730:      fem3d_f_loaded(1, id_l) = -250.0D0
731:
732:      IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
733:
734:      fem3d_f_loaded(1, id_l) = -500.0D0

```



```

735:
736:         END IF
737:
738:         END IF
739:
740:         IF ( DABS( ns3d_x(3, id)-rm3d_x_end(3) ) .LT. EPSILON(1.0D0) ) THEN
741:
742:             fem3d_f_loaded(1, id_l) = 250.0D0
743:
744:             IF ( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN
745:
746:                 fem3d_f_loaded(1, id_l) = 500.0D0
747:
748:             END IF
749:
750:         END IF
751:
752:     END IF
753:
754: END DO
755:
756: !-----
757:
758: END IF
759:
760: CALL set_fem3d_f_loaded                &
761:     (fem3d, fem3d_id_loaded, fem3d_f_loaded)
762: CALL set_elemexforcevec3d_t            &
763:     (efv3d, efv3d_table_ie, efv3d_table_ma, efv3d_t)
764:
765: !-----
766:
767: OPEN(10, FILE = 'mesh.dat')
768:
769: WRITE(10, ' (A)') '!NODE'
770:

```

```

771:      DO id = 1, ns3d_n
772:
773:        WRITE( 10, ' ( I8, 3(A, E17.8) )' )          &
774:          id, ( ', ', ns3d_x(i, id), i = 1, 3 )
775:
776:      END DO
777:
778:      WRITE( 10, ' (A, 3(A, I3) )' )                  &
779:        '!ELEMENT', ', ', le3d_nboundaries,          &
780:        ', ', le3d_nnodes, ', ', 2
781:
782:
783:      DO ie = 1, es3d_n
784:
785:        WRITE( 10, ' ( I8, 27(A, I8) )' )              &
786:          ie, ( ', ', es3d_connectivity(na, ie), &
787:            na = 1, le3d_nnodes )
788:
789:      END DO
790:
791:      WRITE(10, ' (A)' ) '!END'
792:
793:      CLOSE(10)
794:
795: !-----
796:
797:      OPEN(11, FILE = 'ic.dat')
798:
799:      WRITE(11, ' (A)' ) '!DISPLACEMENT'
800:
801:      DO id = 1, ns3d_n
802:
803:        WRITE( 11, ' (I8, 3(A, E17.8) )' )              &
804:          id, ( ', ', ns3d_u( 3*(id-1)+i ), i = 1, 3 )
805:
806:      END DO

```

```

807:
808:     WRITE(11, '(A)') '!END'
809:
810:     CLOSE(11)
811:
812: !-----
813:
814:     OPEN(12, FILE = 'bc.dat')
815:
816:     WRITE(12, '(A)') '!DISPLACEMENT'
817:
818:     DO id = 1, ns3d_n
819:
820:         WRITE( 12, '(I8, 3(A, I8) )' )           &
821:             id, ( ', ', ns3d_bc( 3*(id-1)+i ), i = 1, 3 )
822:
823:     END DO
824:
825:     WRITE(12, '(A)') '!END'
826:
827:     CLOSE(12)
828:
829: !-----
830:
831:     WRITE(13, '(A)') '!F_LOADED'
832:
833:     DO id_l = 1, fem3d_nnodes_loaded
834:
835:         WRITE( 13, '(I8, 3(A, E17.8) )' )           &
836:             fem3d_id_loaded(id_l),                 &
837:             ( ', ', fem3d_f_loaded(i, id_l), i = 1, 3 )
838:
839:     END DO
840:
841:     WRITE(13, '(A)') '!TRACTION'
842:

```

```

843:      DO ib = 1, efv3d_nelemboundaries
844:
845:         WRITE( 13, ' (I8, 2(A, I8), 3(A, E17.8) )' ) &
846:            ib, ', ', efv3d_table_ie(ib),           &
847:            ', ', efv3d_table_ma(ib),               &
848:            ( ', ', efv3d_t(i, ib), i = 1, 3 )
849:
850:      END DO
851:
852:      WRITE(13, ' (A)') '!END'
853:
854:      CLOSE(13)
855:
856: !-----
857:
858:      OPEN(14, FILE = 'mesh.inp')
859:
860:      WRITE(14, ' ( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0
861:
862:      DO id = 1, ns3d_n
863:
864:         WRITE( 14, ' ( (I8, 1X), 3(E17.8, 1X) )' ) &
865:            id, ( ns3d_x(i, id), i = 1, 3 )
866:
867:      END DO
868:
869:      DO ie = 1, es3d_n
870:
871:         WRITE( 14, ' ( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )' ) &
872:            ie, 1, ' hex',                                     &
873:            ( es3d_connectivity(na, ie), na = 1, le3d_nnodes )
874:
875:      END DO
876:
877:      WRITE(14, ' ( 4(I8, 1X) )') 1, 3
878:      WRITE(14, ' (A)') 'DISPLACEMENT, m'

```

```

879:
880:     DO id = 1, ns3d_n
881:
882:         WRITE( 14, ' ( (I8, 1X), 3(E17.8, 1X) )' )      &
883:             id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )
884:
885:     END DO
886:
887:     WRITE(14, ' ( 14I8 )') 1, 1
888:     WRITE(14, ' ( (A, 1X) )') 'VOLUME, m3'
889:
890:     DO ie = 1, es3d_n
891:
892:         WRITE( 14, ' ( (I8, 1X), (E17.8, 1X) )' ) &
893:             ie, es3d_volume(ie)
894:
895:     END DO
896:
897:     CLOSE(14)
898:
899: !-----
900:
901:     DEALLOCATE( ns3d_x )
902:     DEALLOCATE( ns3d_u )
903:
904:     DEALLOCATE( le3d_table_na )
905:
906:     DEALLOCATE( es3d_volume )
907:     DEALLOCATE( es3d_connectivity )
908:
909:     DEALLOCATE( efv3d_table_ie )
910:     DEALLOCATE( efv3d_table_ma )
911:     DEALLOCATE( efv3d_t )
912:
913:     DEALLOCATE( fem3d_id_loaded )
914:     DEALLOCATE( fem3d_f_loaded )

```

```

915:
916:    DEALLOCATE( x_local )
917:
918: !-----
919:
920:    RETURN
921:
922: !#####
923:    END SUBROUTINE run_appli
924: !#####
925:
926:
927: !#####
928:    SUBROUTINE finish_appli()
929: !#####
930:
931:    CALL del_nodes3d(ns3d)
932:    CALL del_localelement3d(le3d)
933:    CALL del_elements3d(es3d)
934:    CALL del_elemstiffmat3d(esm3d)
935:    CALL del_elmemxforcevec3d(efv3d)
936:    CALL del_fem3d(fem3d)
937:    CALL del_rectmesher3d(rm3d)
938:
939: !-----
940:
941:    DEALLOCATE( ns3d )
942:    DEALLOCATE( le3d )
943:    DEALLOCATE( es3d )
944:    DEALLOCATE( esm3d )
945:    DEALLOCATE( efv3d )
946:    DEALLOCATE( fem3d )
947:    DEALLOCATE( rm3d )
948:
949: !-----
950:

```

```
951:      RETURN
952:
953: !#####
954:      END SUBROUTINE finish_appli
955: !#####
956:
957:
958: !#####
959:      END MODULE mod_appli
```