第 4 回演習プログラム

新領域創成科学研究科　人間環境学専攻
橋本　学

第 4 回演習資料のプログラムを以下に示します.

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

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

```
      MODULE mod_elemstiffmat3d
!####################################################################

      USE mod_nodes3d
      USE mod_localelement3d
      USE mod_elements3d

!--------------------------------------------------------------------

      IMPLICIT NONE

!--------------------------------------------------------------------

      TYPE :: struct_elemstiffmat3d

       !--------------------------------------------------------

       PRIVATE

       !--------------------------------------------------------
```

```

  TYPE(struct_nodes3d), POINTER        :: ns3d => NULL()
  TYPE(struct_localelement3d), POINTER :: le3d => NULL()
  TYPE(struct_elements3d), POINTER     :: es3d => NULL()

  !--------------------------------------------------------
  !
  ! k(:, :, :)
  ! Element stiffness matrix
  !
  !--------------------------------------------------------
  !
  ! e(:)
  ! Young's modulus
  !
  ! nu(:)
  ! Poisson's ratio
  !
  !--------------------------------------------------------
  !
  ! evec(:, :)
  ! Infinitesimal strain
  !
  ! svec(:, :)
  ! Stress
  !
  ! s_mises(:)
  ! Mises stress
  !
  !--------------------------------------------------------

  REAL(8), ALLOCATABLE :: k(:, :, :)
  REAL(8), ALLOCATABLE :: e(:)
  REAL(8), ALLOCATABLE :: nu(:)
  REAL(8), ALLOCATABLE :: evec(:, :)
  REAL(8), ALLOCATABLE :: svec(:, :)
```

```
       REAL(8), ALLOCATABLE :: s_mises(:)

       !--------------------------------------------------------

      END TYPE

!--------------------------------------------------------------------

      CONTAINS


      ! Get element stiffness matrix
!####################################################################
      SUBROUTINE get_elemstiffmat3d_k(esm3d, k)
!####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d

      REAL(8), INTENT(OUT) :: k(:, :, :)

!--------------------------------------------------------------------

      k = esm3d%k

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE get_elemstiffmat3d_k
!####################################################################


      ! Set Young's modulus and Poisson's ratio
!####################################################################
      SUBROUTINE set_elemstiffmat3d_e_nu(esm3d, e, nu)
```

```
!#####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d

      REAL(8), INTENT(IN) :: e(:)
      REAL(8), INTENT(IN) :: nu(:)

!---------------------------------------------------------------------

      esm3d%e  = e
      esm3d%nu = nu

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE set_elemstiffmat3d_e_nu
!#####################################################################


      ! Get Young's modulus and Poisson's ratio
!#####################################################################
      SUBROUTINE get_elemstiffmat3d_e(esm3d, e, nu)
!#####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d

      REAL(8), INTENT(OUT) :: e(:)
      REAL(8), INTENT(OUT) :: nu(:)

!---------------------------------------------------------------------

      e  = esm3d%e
      nu = esm3d%nu

```

```
!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE get_elemstiffmat3d_e
!#####################################################################


      ! Get infinitesimal strain and stress
!#####################################################################
      SUBROUTINE get_elemstiffmat3d_evec_svec &
                 (esm3d, evec, svec, s_mises)
!#####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(IN) :: esm3d

      REAL(8), INTENT(OUT) :: evec(:, :)
      REAL(8), INTENT(OUT) :: svec(:, :)
      REAL(8), INTENT(OUT) :: s_mises(:)

!---------------------------------------------------------------------

      evec = esm3d%evec
      svec = esm3d%svec

      s_mises = esm3d%s_mises

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE get_elemstiffmat3d_evec_svec
!#####################################################################

```

```

!####################################################################
      SUBROUTINE init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
!####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d

      TYPE(struct_nodes3d), TARGET, INTENT(IN)        :: ns3d
      TYPE(struct_localelement3d), TARGET, INTENT(IN) :: le3d
      TYPE(struct_elements3d), TARGET, INTENT(IN)     :: es3d

!--------------------------------------------------------------------

      INTEGER :: le3d_nnodes
      INTEGER :: es3d_n

!--------------------------------------------------------------------

      esm3d%ns3d => ns3d
      esm3d%le3d => le3d
      esm3d%es3d => es3d

!--------------------------------------------------------------------

      CALL get_localelement3d_nnodes(esm3d%le3d, le3d_nnodes)

      CALL get_elements3d_n(esm3d%es3d, es3d_n)

!--------------------------------------------------------------------

      ALLOCATE( esm3d%k(3*le3d_nnodes, 3*le3d_nnodes, es3d_n) )

      esm3d%k = 0.0D0

      !--------------------------------------------------------------

      ALLOCATE( esm3d%e(es3d_n)  )
      ALLOCATE( esm3d%nu(es3d_n) )

      esm3d%e  = 0.0D0
      esm3d%nu = 0.0D0

      !---------------------------------------------------------------

      ALLOCATE( esm3d%evec(6, es3d_n) )

      esm3d%evec = 0.0D0

      ALLOCATE( esm3d%svec(6, es3d_n) )

      esm3d%svec = 0.0D0

      ALLOCATE( esm3d%s_mises(es3d_n) )

      esm3d%s_mises = 0.0D0

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE init_elemstiffmat3d
!#####################################################################


!#####################################################################
      SUBROUTINE cal_elemstiffmat3d(esm3d)
!#####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d

!---------------------------------------------------------------------
```

```

    INTEGER :: ns3d_n
    INTEGER :: le3d_nnodes
    INTEGER :: le3d_nqps
    INTEGER :: es3d_n
    INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
    INTEGER :: nqps_tot
    INTEGER :: i, j, k
    INTEGER :: id
    INTEGER :: na, nb
    INTEGER :: ie
    INTEGER :: idof
    INTEGER :: isize, jsize
    INTEGER :: jsize1, jsize2, jsize3
    INTEGER :: ijk

    REAL(8), ALLOCATABLE :: ns3d_x(:, :)
    REAL(8), ALLOCATABLE :: ns3d_u(:)
    REAL(8), ALLOCATABLE :: le3d_xi_qp(:, :)
    REAL(8), ALLOCATABLE :: le3d_w_qp(:, :)
    REAL(8), ALLOCATABLE :: le3d_n_qp(:, :)
    REAL(8), ALLOCATABLE :: le3d_dndxi_qp(:, :, :)
    REAL(8), ALLOCATABLE :: x_local(:, :)
    REAL(8), ALLOCATABLE :: u_local(:)
    REAL(8) :: nqps_tot_inv
    REAL(8) :: g1(3), g2(3), g3(3)
    REAL(8) :: det_j, det_j_inv
    REAL(8) :: w_w_w_det_j
    REAL(8) :: cg1(3), cg2(3), cg3(3)
    REAL(8), ALLOCATABLE :: dndx(:, :)
    REAL(8), ALLOCATABLE :: bmat(:, :)
    REAL(8) :: evec(6)
    REAL(8) :: lambda, mu
    REAL(8) :: dmat(6, 6)
    REAL(8), ALLOCATABLE :: cmat(:, :)
    REAL(8) :: svec(6)
```

```

!------------------------------------------------------------------

      CALL get_nodes3d_n(esm3d%ns3d, ns3d_n)
      ALLOCATE( ns3d_x(3, ns3d_n) )
      CALL get_nodes3d_x(esm3d%ns3d, ns3d_x)
      ALLOCATE( ns3d_u(3*ns3d_n) )
      CALL get_nodes3d_u(esm3d%ns3d, ns3d_u)

      CALL get_localelement3d_nnodes(esm3d%le3d, le3d_nnodes)
      CALL get_localelement3d_nqps(esm3d%le3d, le3d_nqps)
      nqps_tot = le3d_nqps*le3d_nqps*le3d_nqps
      nqps_tot_inv = 1.0D0/DFLOAT( nqps_tot )
      ALLOCATE( le3d_xi_qp(3, nqps_tot) )
      ALLOCATE( le3d_w_qp(3, nqps_tot) )
      CALL get_localelement3d_xi_w_qp         &
           (esm3d%le3d, le3d_xi_qp, le3d_w_qp)
      ALLOCATE( le3d_n_qp(le3d_nnodes, nqps_tot) )
      ALLOCATE( le3d_dndxi_qp(3, le3d_nnodes, nqps_tot) )
      CALL get_localelement3d_n_qp               &
           (esm3d%le3d, le3d_n_qp, le3d_dndxi_qp)

      CALL get_elements3d_n(esm3d%es3d, es3d_n)
      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
      CALL get_elements3d_connectivity(esm3d%es3d, es3d_connectivity)

      ALLOCATE( x_local(3, le3d_nnodes) )
      ALLOCATE( u_local(3*le3d_nnodes) )
      ALLOCATE( dndx(3, le3d_nnodes) )
      ALLOCATE( bmat(6, 3*le3d_nnodes) )
      ALLOCATE( cmat(6, 3*le3d_nnodes) )

!------------------------------------------------------------------

      esm3d%k = 0.0D0

```

```
 DO ie = 1, es3d_n

  !--------------------------------------------------------

  DO na = 1, le3d_nnodes

   id = es3d_connectivity(na, ie)

   DO i = 1, 3

    idof  = 3*(id-1)+i
    isize = 3*(na-1)+i

    x_local(i, na) = ns3d_x(i, id)
    u_local(isize) = ns3d_u(idof)

   END DO

  END DO

  !--------------------------------------------------------

  DO ijk = 1, nqps_tot

   !--------------------------------------------------

   ! Covariant basis vector
   DO i = 1, 3

    g1(i) = 0.0D0
    g2(i) = 0.0D0
    g3(i) = 0.0D0

    DO na = 1, le3d_nnodes

     g1(i) = g1(i)+le3d_dndxi_qp(1, na, ijk)*x_local(i, na)
     g2(i) = g2(i)+le3d_dndxi_qp(2, na, ijk)*x_local(i, na)
     g3(i) = g3(i)+le3d_dndxi_qp(3, na, ijk)*x_local(i, na)

    END DO

   END DO

   !--------------------------------------------------

   ! Jacobian
   det_j = g1(1)*( g2(2)*g3(3)-g2(3)*g3(2) ) &
          +g1(2)*( g2(3)*g3(1)-g2(1)*g3(3) ) &
          +g1(3)*( g2(1)*g3(2)-g2(2)*g3(1) )

   det_j_inv = 1.0D0/det_j

   w_w_w_det_j                                             &
   = le3d_w_qp(1, ijk)*le3d_w_qp(2, ijk)*le3d_w_qp(3, ijk) &
     *det_j

   !--------------------------------------------------

   ! Contravariant basis vector
   cg1(1) = det_j_inv                     &
            *( g2(2)*g3(3)-g2(3)*g3(2) )
   cg1(2) = det_j_inv                     &
            *( g2(3)*g3(1)-g2(1)*g3(3) )
   cg1(3) = det_j_inv                     &
            *( g2(1)*g3(2)-g2(2)*g3(1) )
   cg2(1) = det_j_inv                     &
            *( g3(2)*g1(3)-g3(3)*g1(2) )
   cg2(2) = det_j_inv                     &
            *( g3(3)*g1(1)-g3(1)*g1(3) )
   cg2(3) = det_j_inv                     &
            *( g3(1)*g1(2)-g3(2)*g1(1) )
   cg3(1) = det_j_inv                     &
```

```
               *( g1(2)*g2(3)-g1(3)*g2(2) )
      cg3(2) = det_j_inv                    &
               *( g1(3)*g2(1)-g1(1)*g2(3) )
      cg3(3) = det_j_inv                    &
               *( g1(1)*g2(2)-g1(2)*g2(1) )

      !--------------------------------------------------

      DO na = 1, le3d_nnodes

       dndx(1, na)                        &
       = cg1(1)*le3d_dndxi_qp(1, na, ijk) &
        +cg2(1)*le3d_dndxi_qp(2, na, ijk) &
        +cg3(1)*le3d_dndxi_qp(3, na, ijk)
       dndx(2, na)                        &
       = cg1(2)*le3d_dndxi_qp(1, na, ijk) &
        +cg2(2)*le3d_dndxi_qp(2, na, ijk) &
        +cg3(2)*le3d_dndxi_qp(3, na, ijk)
       dndx(3, na)                        &
       = cg1(3)*le3d_dndxi_qp(1, na, ijk) &
        +cg2(3)*le3d_dndxi_qp(2, na, ijk) &
        +cg3(3)*le3d_dndxi_qp(3, na, ijk)

      END DO

      !--------------------------------------------------

      ! B matrix

      bmat = 0.0D0

      DO nb = 1, le3d_nnodes

       jsize1 = 3*(nb-1)+1
       jsize2 = 3*(nb-1)+2
       jsize3 = 3*(nb-1)+3
```

```

         bmat(1, jsize1) = dndx(1, nb)
         bmat(4, jsize1) = dndx(2, nb)
         bmat(6, jsize1) = dndx(3, nb)
         bmat(2, jsize2) = dndx(2, nb)
         bmat(4, jsize2) = dndx(1, nb)
         bmat(5, jsize2) = dndx(3, nb)
         bmat(3, jsize3) = dndx(3, nb)
         bmat(5, jsize3) = dndx(2, nb)
         bmat(6, jsize3) = dndx(1, nb)

        END DO

        !--------------------------------------------------

        evec = 0.0D0

        DO i = 1, 6

         DO jsize = 1, 3*le3d_nnodes

          evec(i) = evec(i)+bmat(i, jsize)*u_local(jsize)

         END DO

        END DO

        !------------------------------------------------

        lambda                                                      &
        = ( esm3d%e(ie)*esm3d%nu(ie) )                              &
          /( ( 1.0D0+esm3d%nu(ie) )*( 1.0D0-2.0D0*esm3d%nu(ie) ) )
        mu = esm3d%e(ie)/( 2.0D0*( 1.0D0+esm3d%nu(ie) ) )

        !------------------------------------------------

```

```
! D matrix

dmat = 0.0D0

dmat(1, 1) = lambda+2.0D0*mu
dmat(2, 2) = lambda+2.0D0*mu
dmat(3, 3) = lambda+2.0D0*mu
dmat(1, 2) = lambda
dmat(1, 3) = lambda
dmat(2, 1) = lambda
dmat(2, 3) = lambda
dmat(3, 1) = lambda
dmat(3, 2) = lambda
dmat(4, 4) = mu
dmat(5, 5) = mu
dmat(6, 6) = mu

!--------------------------------------------------

DO i = 1, 6

 DO jsize = 1, 3*le3d_nnodes

  cmat(i, jsize) = 0.0D0

  DO k = 1, 6

   cmat(i, jsize) = cmat(i, jsize)+dmat(i, k)*bmat(k, jsize)

  END DO

 END DO

END DO

DO isize = 1, 3*le3d_nnodes
```

```

 DO jsize = 1, 3*le3d_nnodes

  DO k = 1, 6

   esm3d%k(isize, jsize, ie)                   &
   = esm3d%k(isize, jsize, ie)                 &
    +w_w_w_det_j*bmat(k, isize)*cmat(k, jsize)

  END DO

 END DO

END DO

!-------------------------------------------------

svec = 0.0D0

DO i = 1, 6

 DO j = 1, 6

  svec(i) = svec(i)+dmat(i, j)*evec(j)

 END DO

END DO

!-------------------------------------------------

DO i = 1, 6

 esm3d%evec(i, ie) = esm3d%evec(i, ie)+evec(i)
 esm3d%svec(i, ie) = esm3d%svec(i, ie)+svec(i)

```

```
        END DO

        !--------------------------------------------------

       END DO

       !--------------------------------------------------------

       DO I = 1, 6

        esm3d%evec(i, ie) = nqps_tot_inv*esm3d%evec(i, ie)
        esm3d%svec(i, ie) = nqps_tot_inv*esm3d%svec(i, ie)

       END DO

       esm3d%s_mises(ie)                                           &
       = DSQRT( 0.5D0*( ( esm3d%svec(1, ie)-esm3d%svec(2, ie) )**2 &
                       +( esm3d%svec(2, ie)-esm3d%svec(3, ie) )**2 &
                       +( esm3d%svec(3, ie)-esm3d%svec(1, ie) )**2 &
                       +6.0D0                                      &
                        *( esm3d%svec(4, ie)**2                    &
                          +esm3d%svec(5, ie)**2                    &
                          +esm3d%svec(6, ie)**2 ) ) )

       !--------------------------------------------------------

      END DO

!--------------------------------------------------------------------

      DEALLOCATE( ns3d_x )
      DEALLOCATE( ns3d_u )

      DEALLOCATE( le3d_xi_qp )
      DEALLOCATE( le3d_w_qp )
      DEALLOCATE( le3d_n_qp )
      DEALLOCATE( le3d_dndxi_qp )

      DEALLOCATE( es3d_connectivity )

      DEALLOCATE( x_local )
      DEALLOCATE( u_local )
      DEALLOCATE( dndx )
      DEALLOCATE( bmat )
      DEALLOCATE( cmat )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE cal_elemstiffmat3d
!####################################################################


!####################################################################
      SUBROUTINE del_elemstiffmat3d(esm3d)
!####################################################################

      TYPE(struct_elemstiffmat3d), INTENT(INOUT) :: esm3d

!--------------------------------------------------------------------

      NULLIFY( esm3d%ns3d )
      NULLIFY( esm3d%le3d )
      NULLIFY( esm3d%es3d )

!--------------------------------------------------------------------

      DEALLOCATE( esm3d%k )

      !--------------------------------------------------------------

      DEALLOCATE( esm3d%e  )
      DEALLOCATE( esm3d%nu )

      !-------------------------------------------------------------

      DEALLOCATE( esm3d%evec )
      DEALLOCATE( esm3d%svec )
      DEALLOCATE( esm3d%s_mises )

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE del_elemstiffmat3d
!#####################################################################


!#####################################################################
      END MODULE mod_elemstiffmat3d
```

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

```
      MODULE mod_elemexforcevec3d
!#####################################################################

      USE mod_nodes3d
      USE mod_localelement3d
      USE mod_elements3d

!---------------------------------------------------------------------

      IMPLICIT NONE

!---------------------------------------------------------------------
```

```

  TYPE :: struct_elemexforcevec3d

   !--------------------------------------------------------

   PRIVATE

   !--------------------------------------------------------

   TYPE(struct_nodes3d), POINTER        :: ns3d => NULL()
   TYPE(struct_localelement3d), POINTER :: le3d => NULL()
   TYPE(struct_elements3d), POINTER     :: es3d => NULL()

   !--------------------------------------------------------
   !
   ! f(:, :)
   ! Element external force vector
   !
   !--------------------------------------------------------

   ! rho(:)
   ! Density
   !
   ! g
   ! Gravitational acceleration
   !
   !--------------------------------------------------------

   REAL(8), ALLOCATABLE :: f(:, :)
   REAL(8), ALLOCATABLE :: rho(:)
   REAL(8) :: g

   !--------------------------------------------------------

  END TYPE struct_elemexforcevec3d

```

```
!---------------------------------------------------------------------

      CONTAINS


      ! Get element external force vector
!#####################################################################
      SUBROUTINE get_elemexforcevec3d_f(efv3d, f)
!#####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d

      REAL(8), INTENT(OUT) :: f(:, :)

!---------------------------------------------------------------------

      f = efv3d%f

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE get_elemexforcevec3d_f
!#####################################################################


      ! Set density
!#####################################################################
      SUBROUTINE set_elemexforcevec3d_rho(efv3d, rho)
!#####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d

      REAL(8), INTENT(IN) :: rho(:)

```

```
!--------------------------------------------------------------------

      efv3d%rho = rho

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE set_elemexforcevec3d_rho
!####################################################################


      ! Get density
!####################################################################
      SUBROUTINE get_elemexforcevec3d_rho(efv3d, rho)
!####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d

      REAL(8), INTENT(OUT) :: rho(:)

!--------------------------------------------------------------------

      rho = efv3d%rho

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE get_elemexforcevec3d_rho
!####################################################################


      ! Set gravitational acceleration
```

```
!#####################################################################
      SUBROUTINE set_elemexforcevec3d_g(efv3d, g)
!#####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d

      REAL(8), INTENT(IN) :: g

!---------------------------------------------------------------------

      efv3d%g = g

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE set_elemexforcevec3d_g
!#####################################################################


      ! Get gravitational acceleration
!#####################################################################
      SUBROUTINE get_elemexforcevec3d_g(efv3d, g)
!#####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(IN) :: efv3d

      REAL(8), INTENT(OUT) :: g

!---------------------------------------------------------------------

      g = efv3d%g

!---------------------------------------------------------------------

```

```
      RETURN

!######################################################################
      END SUBROUTINE get_elemexforcevec3d_g
!######################################################################


!######################################################################
      SUBROUTINE init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
!######################################################################

      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d

      TYPE(struct_nodes3d), TARGET, INTENT(IN)        :: ns3d
      TYPE(struct_localelement3d), TARGET, INTENT(IN) :: le3d
      TYPE(struct_elements3d), TARGET, INTENT(IN)     :: es3d

!----------------------------------------------------------------------

      INTEGER :: le3d_nnodes
      INTEGER :: es3d_n

!----------------------------------------------------------------------

      efv3d%ns3d => ns3d
      efv3d%le3d => le3d
      efv3d%es3d => es3d

!----------------------------------------------------------------------

      CALL get_localelement3d_nnodes(efv3d%le3d, le3d_nnodes)

      CALL get_elements3d_n(efv3d%es3d, es3d_n)

!----------------------------------------------------------------------

```

```
      ALLOCATE( efv3d%f(3*le3d_nnodes, es3d_n) )

      efv3d%f = 0.0D0

      !--------------------------------------------------------------

      ALLOCATE( efv3d%rho(es3d_n) )

      efv3d%rho = 0.0D0

      efv3d%g = 0.0D0

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE init_elemexforcevec3d
!####################################################################


!####################################################################
      SUBROUTINE cal_elemexforcevec3d(efv3d)
!####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d

!--------------------------------------------------------------------

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE cal_elemexforcevec3d
!####################################################################
```

```


!####################################################################
      SUBROUTINE del_elemexforcevec3d(efv3d)
!####################################################################

      TYPE(struct_elemexforcevec3d), INTENT(INOUT) :: efv3d

!--------------------------------------------------------------------

      NULLIFY( efv3d%ns3d )
      NULLIFY( efv3d%le3d )
      NULLIFY( efv3d%es3d )

!--------------------------------------------------------------------

      DEALLOCATE( efv3d%f )

      !--------------------------------------------------------------

      DEALLOCATE( efv3d%rho )

      efv3d%g = 0.0D0

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE del_elemexforcevec3d
!####################################################################


!####################################################################
      END MODULE mod_elemexforcevec3d
```

第 4 節のプログラム：モジュール mod_fem3d の作成

３．有限要素法モジュール mod_fem3d.f90

```fortran
      MODULE mod_fem3d
!####################################################################

      USE mod_nodes3d
      USE mod_localelement3d
      USE mod_elements3d
      USE mod_elemstiffmat3d
      USE mod_elemexforcevec3d

!--------------------------------------------------------------------

      IMPLICIT NONE

!--------------------------------------------------------------------

      TYPE :: struct_fem3d

       !--------------------------------------------------------

       PRIVATE

       !--------------------------------------------------------

       TYPE(struct_nodes3d), POINTER          :: ns3d  => NULL()
       TYPE(struct_localelement3d), POINTER   :: le3d  => NULL()
       TYPE(struct_elements3d), POINTER       :: es3d  => NULL()
       TYPE(struct_elemstiffmat3d), POINTER   :: esm3d => NULL()
       TYPE(struct_elemexforcevec3d), POINTER :: efv3d => NULL()

       !--------------------------------------------------------
       !
       ! ndofs
       ! The total number of DOFs
       !
       ! k(:, :)
       ! Stiffness matrix
       !
       ! f(:)
       ! External force vector
       !
       !--------------------------------------------------------
       !
       ! nnodes_loaded
       ! The total number of nodes with an equivalent nodal force
       !
       ! id_loaded(:)
       ! Node no. with an equivalent nodal force
       !
       ! f_loaded(:, :)
       ! Equivalent nodal force
       !
       !--------------------------------------------------------

       INTEGER :: ndofs
       INTEGER :: nnodes_loaded
       INTEGER, ALLOCATABLE :: id_loaded(:)

       REAL(8), ALLOCATABLE :: k(:, :)
       REAL(8), ALLOCATABLE :: f(:)
       REAL(8), ALLOCATABLE :: f_loaded(:, :)

       !--------------------------------------------------------

      END TYPE struct_fem3d

  !--------------------------------------------------------------------

```

```
      PRIVATE :: cal_fem3d_stiffmat
      PRIVATE :: cal_fem3d_rhs
      PRIVATE :: cal_fem3d_dirichletbc
      PRIVATE :: cal_fem3d_linearsolver

!--------------------------------------------------------------------

      CONTAINS


      ! Get the total number of DOFs
!####################################################################
      SUBROUTINE get_fem3d_ndofs(fem3d, ndofs)
!####################################################################

      TYPE(struct_fem3d), INTENT(IN) :: fem3d

      INTEGER, INTENT(OUT) :: ndofs

!--------------------------------------------------------------------

      ndofs = fem3d%ndofs

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE get_fem3d_ndofs
!####################################################################


      ! Get stiffness matrix
!####################################################################
      SUBROUTINE get_fem3d_k(fem3d, k)
!####################################################################
```

```

      TYPE(struct_fem3d), INTENT(IN) :: fem3d

      INTEGER, INTENT(OUT) :: k(:, :)

!--------------------------------------------------------------------

      k = fem3d%k

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE get_fem3d_k
!####################################################################


      ! Get external force vector
!####################################################################
      SUBROUTINE get_fem3d_f(fem3d, f)
!####################################################################

      TYPE(struct_fem3d), INTENT(IN) :: fem3d

      INTEGER, INTENT(OUT) :: f(:)

!--------------------------------------------------------------------

      f = fem3d%f

!--------------------------------------------------------------------

      RETURN

!####################################################################
```

```
      END SUBROUTINE get_fem3d_f
!####################################################################


      ! Set equivalent nodal force
!####################################################################
      SUBROUTINE set_fem3d_f_loaded(fem3d, id_loaded, f_loaded)
!####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

      INTEGER, INTENT(IN) :: id_loaded(:)
      REAL(8), INTENT(IN) :: f_loaded(:, :)

!--------------------------------------------------------------------

      fem3d%id_loaded = id_loaded
      fem3d%f_loaded  = f_loaded

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE set_fem3d_f_loaded
!####################################################################


      ! Get equivalent nodal force
!####################################################################
      SUBROUTINE get_fem3d_f_loaded(fem3d, id_loaded, f_loaded)
!####################################################################

      TYPE(struct_fem3d), INTENT(IN) :: fem3d

      INTEGER, INTENT(OUT) :: id_loaded(:)
      REAL(8), INTENT(OUT) :: f_loaded(:, :)

!---------------------------------------------------------------------

      id_loaded = fem3d%id_loaded
      f_loaded  = fem3d%f_loaded

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE get_fem3d_f_loaded
!#####################################################################


      ! Initialize fem3d
!#####################################################################
      SUBROUTINE init_fem3d                              &
                 (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
                  nnodes_loaded)
!#####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

      TYPE(struct_nodes3d), TARGET, INTENT(IN)           :: ns3d
      TYPE(struct_localelement3d), TARGET, INTENT(IN)   :: le3d
      TYPE(struct_elements3d), TARGET, INTENT(IN)       :: es3d
      TYPE(struct_elemstiffmat3d), TARGET, INTENT(IN)   :: esm3d
      TYPE(struct_elemexforcevec3d), TARGET, INTENT(IN) :: efv3d

      INTEGER, INTENT(IN) :: nnodes_loaded

!---------------------------------------------------------------------

      INTEGER :: ns3d_n
```

```

!--------------------------------------------------------------------

      fem3d%ns3d => ns3d
      fem3d%le3d => le3d
      fem3d%es3d => es3d
      fem3d%esm3d => esm3d
      fem3d%efv3d => efv3d

!--------------------------------------------------------------------

      CALL get_nodes3d_n(fem3d%ns3d, ns3d_n)

!--------------------------------------------------------------------

      fem3d%ndofs = 3*ns3d_n

      !--------------------------------------------------------------

      ALLOCATE( fem3d%k(fem3d%ndofs, fem3d%ndofs) )

      fem3d%k = 0.0D0

      ALLOCATE( fem3d%f(fem3d%ndofs) )

      fem3d%f = 0.0D0

      !--------------------------------------------------------------

      fem3d%nnodes_loaded = nnodes_loaded

      ALLOCATE( fem3d%id_loaded(nnodes_loaded) )

      fem3d%id_loaded = 0

      ALLOCATE( fem3d%f_loaded(3, nnodes_loaded) )

      fem3d%f_loaded = 0.0D0

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE init_fem3d
!####################################################################


      ! Calculate fem3d
!####################################################################
      SUBROUTINE cal_fem3d(fem3d)
!####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

!--------------------------------------------------------------------

      ! Stiffness matrix
      CALL cal_fem3d_stiffmat(fem3d)

!--------------------------------------------------------------------

      ! RHS vector
      CALL cal_fem3d_rhs(fem3d)

!--------------------------------------------------------------------

      ! Dirichlet boundary condition
      CALL cal_fem3d_dirichletbc(fem3d)

!--------------------------------------------------------------------

```

```
      ! Linear solver
      CALL cal_fem3d_linearsolver(fem3d)

!--------------------------------------------------------------------

      ! Strain and stress
      CALL cal_elemstiffmat3d(fem3d%esm3d)

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE cal_fem3d
!####################################################################


      ! Calculate fem3d (stiffness matrix)
!####################################################################
      SUBROUTINE cal_fem3d_stiffmat(fem3d)
!####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

!--------------------------------------------------------------------

      INTEGER :: le3d_nnodes
      INTEGER :: es3d_n
      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
      INTEGER :: i, j
      INTEGER :: id, jd
      INTEGER :: na, nb
      INTEGER :: ie
      INTEGER :: idof, jdof
      INTEGER :: isize, jsize

      REAL(8), ALLOCATABLE :: esm3d_k(:, :, :)
      REAL(8), ALLOCATABLE :: efv3d_f(:, :)

!--------------------------------------------------------------------

      ! Element stiffness matrix
      CALL cal_elemstiffmat3d(fem3d%esm3d)

!--------------------------------------------------------------------

      CALL get_localelement3d_nnodes(fem3d%le3d, le3d_nnodes)

      CALL get_elements3d_n(fem3d%es3d, es3d_n)
      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
      CALL get_elements3d_connectivity(fem3d%es3d, es3d_connectivity)

      ALLOCATE( esm3d_k(3*le3d_nnodes, 3*le3d_nnodes, es3d_n) )
      CALL get_elemstiffmat3d_k(fem3d%esm3d, esm3d_k)

      ! Stiffness matrix

      fem3d%k = 0.0D0

      DO ie = 1, es3d_n

       DO na = 1, le3d_nnodes

        id = es3d_connectivity(na, ie)

        DO i = 1, 3

         idof  = 3*(id-1)+i
         isize = 3*(na-1)+i

         DO nb = 1, le3d_nnodes

```

```
          jd = es3d_connectivity(nb, ie)

          DO j = 1, 3

           jdof  = 3*(jd-1)+j
           jsize = 3*(nb-1)+j

           fem3d%k(idof, jdof)                              &
           = fem3d%k(idof, jdof)+esm3d_k(isize, jsize, ie)

          END DO

         END DO

        END DO

       END DO

      END DO

!---------------------------------------------------------------------

      DEALLOCATE( es3d_connectivity )

      DEALLOCATE( esm3d_k )

!---------------------------------------------------------------------

      RETURN

!#####################################################################
      END SUBROUTINE cal_fem3d_stiffmat
!#####################################################################


      ! Calculate fem3d (RHS vector)
```

```
!######################################################################
      SUBROUTINE cal_fem3d_rhs(fem3d)
!######################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

!----------------------------------------------------------------------

      INTEGER :: le3d_nnodes
      INTEGER :: es3d_n
      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
      INTEGER :: i
      INTEGER :: id
      INTEGER :: id_l
      INTEGER :: na
      INTEGER :: ie
      INTEGER :: idof
      INTEGER :: isize

      REAL(8), ALLOCATABLE :: efv3d_f(:, :)

!----------------------------------------------------------------------

      ! Element external force vector
      CALL cal_elemexforcevec3d(fem3d%efv3d)

!----------------------------------------------------------------------

      CALL get_localelement3d_nnodes(fem3d%le3d, le3d_nnodes)

      CALL get_elements3d_n(fem3d%es3d, es3d_n)
      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
      CALL get_elements3d_connectivity(fem3d%es3d, es3d_connectivity)

      ALLOCATE( efv3d_f(3*le3d_nnodes, es3d_n) )
      CALL get_elemexforcevec3d_f(fem3d%efv3d, efv3d_f)
```

```

!--------------------------------------------------------------

! External force vector

fem3d%f = 0.0D0

DO id_l = 1, fem3d%nnodes_loaded

 id = fem3d%id_loaded(id_l)

 DO i = 1, 3

  idof = 3*(id-1)+i

  fem3d%f(idof) = fem3d%f_loaded(i, id_l)

 END DO

END DO

DO ie = 1, es3d_n

 DO na = 1, le3d_nnodes

  id = es3d_connectivity(na, ie)

  DO i = 1, 3

   idof  = 3*(id-1)+i
   isize = 3*(na-1)+i

   fem3d%f(idof) = fem3d%f(idof)+efv3d_f(isize, ie)

  END DO

```

```
         END DO

       END DO

!---------------------------------------------------------------------

       DEALLOCATE( es3d_connectivity )

       DEALLOCATE( efv3d_f )

!---------------------------------------------------------------------

       RETURN

!#####################################################################
       END SUBROUTINE cal_fem3d_rhs
!#####################################################################


       ! Calculate fem3d (Dirichlet boundary condition)
!#####################################################################
       SUBROUTINE cal_fem3d_dirichletbc(fem3d)
!#####################################################################

       TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

!---------------------------------------------------------------------

       INTEGER :: ns3d_n
       INTEGER, ALLOCATABLE :: ns3d_bc(:)
       INTEGER :: idof, jdof
       INTEGER :: idof_g
       INTEGER :: ndofs_given
       INTEGER, ALLOCATABLE :: idof_given(:)

       REAL(8), ALLOCATABLE :: ns3d_u(:)
```

```
     REAL(8) :: u_given

!------------------------------------------------------------------

     CALL get_nodes3d_n(fem3d%ns3d, ns3d_n)
     ALLOCATE( ns3d_u(fem3d%ndofs) )
     CALL get_nodes3d_u(fem3d%ns3d, ns3d_u)
     ALLOCATE( ns3d_bc(fem3d%ndofs) )
     CALL get_nodes3d_bc(fem3d%ns3d, ns3d_bc)

!------------------------------------------------------------------

     idof_g = 0

     DO idof = 1, fem3d%ndofs

      IF( ns3d_bc(idof) .EQ. 1 ) THEN

       idof_g = idof_g+1

      END IF

     END DO

     ndofs_given = idof_g

     !-------------------------------------------------------------

     ALLOCATE( idof_given(ndofs_given) )

     !-------------------------------------------------------------

     idof_g = 0

     DO idof = 1, fem3d%ndofs

```

```
       IF( ns3d_bc(idof) .EQ. 1 ) THEN

        idof_g = idof_g+1

        idof_given(idof_g) = idof

       END IF

      END DO

!---------------------------------------------------------------------

      ! Dirichlet boundary conditions

      DO idof_g = 1, ndofs_given

       jdof = idof_given(idof_g)

       DO idof = 1, fem3d%ndofs

        fem3d%f(idof) = fem3d%f(idof)-fem3d%k(idof, jdof)*ns3d_u(jdof)

       END DO

      END DO

      DO idof_g = 1, ndofs_given

       idof = idof_given(idof_g)

       fem3d%f(idof) = ns3d_u(idof)

      END DO

      DO idof_g = 1, ndofs_given

```

```
       jdof = idof_given(idof_g)

       DO idof = 1, fem3d%ndofs

        fem3d%k(idof, jdof) = 0.0D0
        fem3d%k(jdof, idof) = 0.0D0

       END DO

       fem3d%k(jdof, jdof) = 1.0D0

      END DO

!--------------------------------------------------------------------

      DEALLOCATE( ns3d_u )
      DEALLOCATE( ns3d_bc )

      DEALLOCATE( idof_given )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE cal_fem3d_dirichletbc
!####################################################################


      ! Calculate fem3d (linear solver)
!####################################################################
      SUBROUTINE cal_fem3d_linearsolver(fem3d)
!####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

```

```
!--------------------------------------------------------------------

      INTEGER :: ipiv(fem3d%ndofs)
      INTEGER :: info

!--------------------------------------------------------------------

      ! Linear solver

      ipiv = 0

      CALL DGESV                                        &
           (fem3d%ndofs, 1, fem3d%k, fem3d%ndofs, ipiv, &
            fem3d%f, fem3d%ndofs, info)

      CALL set_nodes3d_u(fem3d%ns3d, fem3d%f)

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE cal_fem3d_linearsolver
!####################################################################


      ! Delete fem3d
!####################################################################
      SUBROUTINE del_fem3d(fem3d)
!####################################################################

      TYPE(struct_fem3d), INTENT(INOUT) :: fem3d

!--------------------------------------------------------------------

      IF( fem3d%ndofs .EQ. 0 ) THEN
```

```

       RETURN

      END IF

!---------------------------------------------------------------------

      NULLIFY( fem3d%ns3d )
      NULLIFY( fem3d%le3d )
      NULLIFY( fem3d%es3d )
      NULLIFY( fem3d%esm3d )
      NULLIFY( fem3d%efv3d )

!---------------------------------------------------------------------

      fem3d%ndofs = 0

      !---------------------------------------------------------------

      DEALLOCATE( fem3d%k )
      DEALLOCATE( fem3d%f )

!---------------------------------------------------------------------

      IF( fem3d%nnodes_loaded .EQ. 0 ) THEN

       RETURN

      END IF

!---------------------------------------------------------------------

      fem3d%nnodes_loaded = 0

      !---------------------------------------------------------------

      DEALLOCATE( fem3d%id_loaded )
      DEALLOCATE( fem3d%f_loaded )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE del_fem3d
!####################################################################


!####################################################################
      END MODULE mod_fem3d
```

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

４．アプリケーションモジュール mod_appli.f90

```
      MODULE mod_appli
!####################################################################

      USE mod_nodes3d
      USE mod_localelement3d
      USE mod_elements3d
      USE mod_elemstiffmat3d
      USE mod_elemexforcevec3d
      USE mod_fem3d
      USE mod_rectmesher3d

!--------------------------------------------------------------------

      IMPLICIT NONE

```

```
!--------------------------------------------------------------------

      TYPE(struct_nodes3d), POINTER          :: ns3d
      TYPE(struct_localelement3d), POINTER   :: le3d
      TYPE(struct_elements3d), POINTER       :: es3d
      TYPE(struct_elemstiffmat3d), POINTER   :: esm3d
      TYPE(struct_elemexforcevec3d), POINTER :: efv3d
      TYPE(struct_fem3d), POINTER            :: fem3d
      TYPE(struct_rectmesher3d), POINTER     :: rm3d

      !--------------------------------------------------------------
      !
      ! Problem number
      ! prob
      !
      !--------------------------------------------------------------

      INTEGER :: prob

!--------------------------------------------------------------------

      CONTAINS


      ! Start appli
!####################################################################
      SUBROUTINE start_appli()
!####################################################################

      INTEGER :: ns3d_n
      INTEGER :: le3d_nboundaries
      INTEGER :: le3d_nnodes
      INTEGER :: le3d_nqps
      INTEGER :: es3d_n
      INTEGER :: rm3d_n_x(3)

      REAL(8) :: rm3d_x_start(3)
      REAL(8) :: rm3d_x_end(3)
      REAL(8) :: e
      REAL(8) :: nu
      REAL(8) :: rho
      REAL(8) :: g

      CHARACTER(1) :: dataname

!--------------------------------------------------------------------

      ALLOCATE( ns3d )
      ALLOCATE( le3d )
      ALLOCATE( es3d )
      ALLOCATE( esm3d )
      ALLOCATE( efv3d )
      ALLOCATE( fem3d )
      ALLOCATE( rm3d )

!--------------------------------------------------------------------

      OPEN(13, FILE = 'param_meshing.dat')

      READ(13, *) dataname
      READ(13, *) rm3d_n_x(1), rm3d_n_x(2), rm3d_n_x(3)
      READ(13, *) dataname
      READ(13, *) rm3d_x_start(1), rm3d_x_start(2), rm3d_x_start(3)
      READ(13, *) dataname
      READ(13, *) rm3d_x_end(1), rm3d_x_end(2), rm3d_x_end(3)
      READ(13, *) dataname
      READ(13, *) prob
      READ(13, *) dataname
      READ(13, *) e
      READ(13, *) dataname
      READ(13, *) nu
      READ(13, *) dataname
      READ(13, *) rho
      READ(13, *) dataname
      READ(13, *) g
      READ(13, *) dataname

      CLOSE(13)

!---------------------------------------------------------------------

      OPEN(13, FILE = 'param_fea.dat')

      WRITE(13, '(A)') '!ANALYSIS_TYPE'
      WRITE(13, '(A)') 'STATIC_ANALYSIS'
      WRITE(13, '(A)') "!YOUNG'S_MODULUS"
      WRITE(13, '(E17.8)') e
      WRITE(13, '(A)') "!POISSON'S_RATIO"
      WRITE(13, '(E17.8)') nu
      WRITE(13,  '(A)') '!DENSITY'
      WRITE(13, '(E17.8)') rho
      WRITE(13,  '(A)') '!GRAVITATIONAL_ACCELERATION'
      WRITE(13, '(E17.8)') g

!---------------------------------------------------------------------

      CALL init_rectmesher3d                          &
           (rm3d, ns3d, le3d, es3d,                   &
            rm3d_n_x, rm3d_x_start, rm3d_x_end)

!---------------------------------------------------------------------

      CALL get_nodes3d_n(ns3d, ns3d_n)

      CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
      CALL get_localelement3d_nqps(le3d, le3d_nqps)

      CALL get_elements3d_n(es3d, es3d_n)

      !----------------------------------------------------------

      CALL init_nodes3d(ns3d, ns3d_n)

      CALL init_localelement3d                           &
          (le3d, le3d_nboundaries, le3d_nnodes, le3d_nqps)

      CALL init_elements3d(es3d, ns3d, le3d, es3d_n)

!----------------------------------------------------------------

      RETURN

!################################################################
      END SUBROUTINE start_appli
!################################################################


      ! Run appli
!################################################################
      SUBROUTINE run_appli()
!################################################################

      INTEGER :: ns3d_n
      INTEGER, ALLOCATABLE :: ns3d_bc(:)
      INTEGER :: le3d_nboundaries
      INTEGER :: le3d_nnodes
      INTEGER :: le3d_nnodes_boundary
      INTEGER, ALLOCATABLE :: le3d_table_na(:, :)
      INTEGER :: es3d_n
      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
      INTEGER :: es3d_ie_max_volume
      INTEGER :: es3d_ie_min_volume
      INTEGER :: fem3d_ndofs
      INTEGER :: fem3d_nnodes_loaded
      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
      INTEGER :: i
      INTEGER :: id
      INTEGER :: id_l
      INTEGER :: na
      INTEGER :: ie
      INTEGER :: idof

      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
      REAL(8), ALLOCATABLE :: ns3d_u(:)
      REAL(8), ALLOCATABLE :: es3d_volume(:)
      REAL(8) :: es3d_max_volume
      REAL(8) :: es3d_min_volume
      REAL(8) :: es3d_sum_volume
      REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
      REAL(8) :: rm3d_x_start(3)
      REAL(8) :: rm3d_x_end(3)
      REAL(8) :: rm3d_x_center(3)
      REAL(8) :: rm3d_length_x(3)
      REAL(8), ALLOCATABLE :: x_local(:, :)

!--------------------------------------------------------------------

      CALL cal_rectmesher3d(rm3d)

      CALL cal_elements3d(es3d)

!--------------------------------------------------------------------

      CALL get_nodes3d_n(ns3d, ns3d_n)

      ALLOCATE( ns3d_x(3, ns3d_n) )
      CALL get_nodes3d_x(ns3d, ns3d_x)
      ALLOCATE( ns3d_u(3*ns3d_n) )
      ns3d_u = 0.0D0
      ALLOCATE( ns3d_bc(3*ns3d_n) )
      ns3d_bc = 0

      CALL get_localelement3d_nboundaries(le3d, le3d_nboundaries)
      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)
      CALL get_localelement3d_nnodes_boundary(le3d, le3d_nnodes_boundary)
      ALLOCATE( le3d_table_na(le3d_nnodes_boundary, le3d_nboundaries) )
      CALL get_localelement3d_table_na(le3d, le3d_table_na)

      CALL get_elements3d_n(es3d, es3d_n)
      ALLOCATE( es3d_volume(es3d_n) )
      CALL get_elements3d_volume                 &
           (es3d, es3d_volume,                   &
            es3d_max_volume, es3d_ie_max_volume, &
            es3d_min_volume, es3d_ie_min_volume, &
            es3d_sum_volume)
      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
      CALL get_elements3d_connectivity(es3d, es3d_connectivity)
      CALL get_rectmesher3d_x_start_x_end   &
           (rm3d, rm3d_x_start, rm3d_x_end, &
            rm3d_x_center, rm3d_length_x)

      ALLOCATE( x_local(3, le3d_nnodes_boundary) )

!--------------------------------------------------------------------

      ! Tensile deformation
      IF( prob .EQ. 1 ) THEN

       !--------------------------------------------------------

       id_l = 0

       DO id = 1, ns3d_n

        IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN
```

```

     id_l = id_l+1

    END IF

   END DO

   fem3d_nnodes_loaded = id_l

   !--------------------------------------------------------

  END IF

  !--------------------------------------------------------------

  CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
  CALL init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
  CALL init_fem3d                              &
       (fem3d, ns3d, le3d, es3d, esm3d, efv3d, &
        fem3d_nnodes_loaded)

  CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)
  ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
  fem3d_id_loaded = 0
  ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )
  fem3d_f_loaded = 0.0D0

  !--------------------------------------------------------------

  ! Tensile deformation
  IF( prob .EQ. 1 ) THEN

   !--------------------------------------------------------

   DO id = 1, ns3d_n

    IF( DABS( ns3d_x(1, id)-rm3d_x_start(1) ) .LT. EPSILON(1.0D0) ) THEN

     idof = 3*(id-1)+1

     ns3d_u(idof) = 0.0D0
     ns3d_bc(idof) = 1

     IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN

      idof = 3*(id-1)+2

      ns3d_u(idof) = 0.0D0
      ns3d_bc(idof) = 1

     END IF

     IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN

      idof = 3*(id-1)+3

      ns3d_u(idof) = 0.0D0
      ns3d_bc(idof) = 1

     END IF

    END IF

   END DO

   !--------------------------------------------------------

  END IF

  !--------------------------------------------------------------

  id_l = 0
```

```

      ! Tensile deformation
      IF( prob .EQ. 1 ) THEN

       !-------------------------------------------------------

       DO id = 1, ns3d_n

        IF( DABS( ns3d_x(1, id)-rm3d_x_end(1) ) .LT. EPSILON(1.0D0) ) THEN

         id_l = id_l+1

         fem3d_id_loaded(id_l) = id
         fem3d_f_loaded(1, id_l) = 2.5D6

         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN

          fem3d_f_loaded(1, id_l) = 5.0D6

         END IF

         IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN

          fem3d_f_loaded(1, id_l) = 5.0D6

         END IF

         IF( DABS( ns3d_x(2, id)-rm3d_x_center(2) ) .LT. EPSILON(1.0D0) ) THEN

          IF( DABS( ns3d_x(3, id)-rm3d_x_center(3) ) .LT. EPSILON(1.0D0) ) THEN

           fem3d_f_loaded(1, id_l) = 1.0D7

          END IF

         END IF

        END IF

       END DO

       !--------------------------------------------------------

      END IF

      CALL set_fem3d_f_loaded                              &
            (fem3d, fem3d_id_loaded, fem3d_f_loaded)

!--------------------------------------------------------------------

      OPEN(10, FILE = 'mesh.dat')

      WRITE(10, '(A)') '!NODE'

      DO id = 1, ns3d_n

       WRITE( 10,'( I8, 3(A, E17.8) )' )                  &
              id, ( ',', ns3d_x(i, id), i = 1, 3  )

      END DO

      WRITE( 10, '(A, 3(A, I3) )' )                       &
             '!ELEMENT', ', ', le3d_nboundaries,          &
                         ', ', le3d_nnodes, ', ', 2


      DO ie = 1, es3d_n

       WRITE( 10, '( I8, 27(A, I8) )' )                   &
              ie, ( ',', es3d_connectivity(na, ie),       &
                    na = 1, le3d_nnodes )

      END DO

      WRITE(10,'(A)') '!END'

      CLOSE(10)

!---------------------------------------------------------------------

      OPEN(11, FILE = 'ic.dat')

      WRITE(11, '(A)') '!DISPLACEMENT'

      DO id = 1, ns3d_n

       WRITE( 11, '(I8, 3(A, E17.8) )' )                  &
              id, ( ', ', ns3d_u( 3*(id-1)+i ), i = 1, 3 )

      END DO

      WRITE(11, '(A)') '!END'

      CLOSE(11)

!---------------------------------------------------------------------

      OPEN(12, FILE = 'bc.dat')

      WRITE(12, '(A)') '!DISPLACEMENT'

      DO id = 1, ns3d_n

       WRITE( 12, '(I8, 3(A, I8) )' )                       &
              id, ( ', ', ns3d_bc( 3*(id-1)+i ), i = 1, 3 )

      END DO

      WRITE(12, '(A)') '!END'

      CLOSE(12)

!---------------------------------------------------------------------

      WRITE(13, '(A)') '!F_LOADED'

      DO id_l = 1, fem3d_nnodes_loaded

       WRITE( 13, '(I8, 3(A, E17.8) )' )                  &
              fem3d_id_loaded(id_l),                      &
              ( ', ', fem3d_f_loaded(i, id_l), i = 1, 3 )

      END DO

      WRITE(13, '(A)') '!END'

      CLOSE(13)

!---------------------------------------------------------------------

      OPEN(14, FILE = 'mesh.inp')

      WRITE(14, '( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0

      DO id = 1, ns3d_n

       WRITE( 14, '( (I8, 1X), 3(E17.8, 1X) )' ) &
              id, ( ns3d_x(i, id), i = 1, 3 )

      END DO

      DO ie = 1, es3d_n

       WRITE( 14, '( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )' )       &
```

```
              ie, 1, '  hex',                                    &
              ( es3d_connectivity(na, ie), na = 1, le3d_nnodes )

      END DO

      WRITE(14, '( 4(I8, 1X) )') 1, 3
      WRITE(14, '(A)') 'DISPLACEMENT, m'

      DO id = 1, ns3d_n

       WRITE( 14, '( (I8, 1X), 3(E17.8, 1X) )' )     &
              id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )

      END DO

      WRITE(14, '( 14I8 )') 1, 1
      WRITE(14, '( (A, 1X) )') 'VOLUME, m3'

      DO ie = 1, es3d_n

       WRITE( 14, '( (I8, 1X), (E17.8, 1X) )' ) &
              ie, es3d_volume(ie)

      END DO

      CLOSE(14)

!-------------------------------------------------------------------

      DEALLOCATE( ns3d_x )
      DEALLOCATE( ns3d_u )

      DEALLOCATE( le3d_table_na )

      DEALLOCATE( es3d_volume )
      DEALLOCATE( es3d_connectivity )

      DEALLOCATE( fem3d_id_loaded )
      DEALLOCATE( fem3d_f_loaded )

      DEALLOCATE( x_local )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE run_appli
!####################################################################


!####################################################################
      SUBROUTINE finish_appli()
!####################################################################

      CALL del_nodes3d(ns3d)
      CALL del_localelement3d(le3d)
      CALL del_elements3d(es3d)
      CALL del_elemstiffmat3d(esm3d)
      CALL del_elemexforcevec3d(efv3d)
      CALL del_fem3d(fem3d)
      CALL del_rectmesher3d(rm3d)

!--------------------------------------------------------------------

      DEALLOCATE( ns3d )
      DEALLOCATE( le3d )
      DEALLOCATE( es3d )
      DEALLOCATE( esm3d )
      DEALLOCATE( efv3d )
      DEALLOCATE( fem3d )
      DEALLOCATE( rm3d )
```

```

!----------------------------------------------------------------------

      RETURN

!######################################################################
      END SUBROUTINE finish_appli
!######################################################################


!######################################################################
      END MODULE mod_appli
```

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

５．アプリケーションモジュール mod_appli.f90

```
      MODULE mod_appli
!######################################################################

      USE mod_nodes3d
      USE mod_localelement3d
      USE mod_elements3d
      USE mod_elemstiffmat3d
      USE mod_elemexforcevec3d
      USE mod_fem3d

!----------------------------------------------------------------------

      IMPLICIT NONE

!----------------------------------------------------------------------

      TYPE(struct_nodes3d), POINTER          :: ns3d
      TYPE(struct_localelement3d), POINTER   :: le3d
      TYPE(struct_elements3d), POINTER       :: es3d
      TYPE(struct_elemstiffmat3d), POINTER   :: esm3d
      TYPE(struct_elemexforcevec3d), POINTER :: efv3d
      TYPE(struct_fem3d), POINTER            :: fem3d

!--------------------------------------------------------------------

      CONTAINS


      ! Start appli
!####################################################################
      SUBROUTINE start_appli()
!####################################################################

      INTEGER :: ns3d_n
      INTEGER, ALLOCATABLE :: ns3d_bc(:)
      INTEGER :: le3d_nboundaries
      INTEGER :: le3d_nnodes
      INTEGER :: le3d_nqps
      INTEGER :: es3d_n
      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
      INTEGER :: fem3d_nnodes_loaded
      INTEGER, ALLOCATABLE :: fem3d_id_loaded(:)
      INTEGER :: fem3d_ndofs
      INTEGER :: i
      INTEGER :: id
      INTEGER :: na
      INTEGER :: ie
      INTEGER :: ib
      INTEGER :: idof_g
      INTEGER :: idof_l
      INTEGER :: number

      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
      REAL(8), ALLOCATABLE :: ns3d_u(:)
      REAL(8), ALLOCATABLE :: esm3d_e(:)
      REAL(8), ALLOCATABLE :: esm3d_nu(:)
      REAL(8), ALLOCATABLE :: efv3d_rho(:)
      REAL(8), ALLOCATABLE :: fem3d_f_loaded(:, :)
      REAL(8) :: e
      REAL(8) :: nu
      REAL(8) :: rho
      REAL(8) :: g

      CHARACTER(1) :: dataname

!--------------------------------------------------------------------

      ALLOCATE( ns3d )
      ALLOCATE( le3d )
      ALLOCATE( es3d )
      ALLOCATE( esm3d )
      ALLOCATE( efv3d )
      ALLOCATE( fem3d )

!--------------------------------------------------------------------

      OPEN(10, FILE = 'mesh.dat')

      READ(10, *) dataname

      ns3d_n = 0

      DO

       READ(10, *) dataname

       ns3d_n = ns3d_n+1

       IF( dataname .EQ. '!' ) THEN
```

```

  EXIT

 END IF

END DO

ns3d_n = ns3d_n-1

es3d_n = 0

DO

 READ(10, *) dataname

 es3d_n = es3d_n+1

 IF( dataname .EQ. '!' ) THEN

  EXIT

 END IF

END DO

es3d_n = es3d_n-1

CLOSE(10)

!--------------------------------------------------------------

OPEN(13, FILE = 'param_fea.dat')

READ(13, *) dataname
READ(13, *) dataname
READ(13, *) dataname
```

```
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname

      READ(13, *) dataname

      fem3d_nnodes_loaded = 0

      DO

       READ(13, *) dataname

       fem3d_nnodes_loaded = fem3d_nnodes_loaded+1

       IF( dataname .EQ. '!' ) THEN

        EXIT

       END IF

      END DO

      fem3d_nnodes_loaded = fem3d_nnodes_loaded-1

      CLOSE(13)

!--------------------------------------------------------------------

      OPEN(10, FILE = 'mesh.dat')

      READ(10, *) dataname

```

```
CALL init_nodes3d(ns3d, ns3d_n)

ALLOCATE( ns3d_x(3, ns3d_n) )

DO id = 1, ns3d_n

 READ(10, *) number, ( ns3d_x(i, id), i = 1, 3 )

END DO

CALL set_nodes3d_x(ns3d, ns3d_x)

READ(10, *) dataname, le3d_nboundaries, le3d_nnodes, le3d_nqps

CALL init_localelement3d                               &
     (le3d, le3d_nboundaries, le3d_nnodes, le3d_nqps)
CALL init_elements3d(es3d, ns3d, le3d, es3d_n)

ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )

DO ie = 1, es3d_n

 READ(10, *) number, ( es3d_connectivity(na, ie), &
                       na = 1, le3d_nnodes )

END DO

CALL set_elements3d_connectivity(es3d, es3d_connectivity)

CLOSE(10)

!---------------------------------------------------------------

CALL init_elemstiffmat3d(esm3d, ns3d, le3d, es3d)
CALL init_elemexforcevec3d(efv3d, ns3d, le3d, es3d)
CALL init_fem3d                                  &
```

```
       (fem3d, ns3d, le3d, es3d, esm3d, efv3d, 9)

  CALL get_fem3d_ndofs(fem3d, fem3d_ndofs)

  !--------------------------------------------------------------

  OPEN(11, FILE = 'ic.dat')

  READ(11, *) dataname

  ALLOCATE( ns3d_u(3*ns3d_n) )

  DO id = 1, ns3d_n

   READ(11, *) number, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )

  END DO

  CALL set_nodes3d_u(ns3d, ns3d_u)

  READ(11, *) dataname

  CLOSE(11)

  !--------------------------------------------------------------

  OPEN(12, FILE = 'bc.dat')

  READ(12, *) dataname

  ALLOCATE( ns3d_bc(3*ns3d_n) )

  DO id = 1, ns3d_n

   READ(12, *) number, ( ns3d_bc( 3*(id-1)+i ), i = 1, 3 )

```

```
      END DO

      CALL set_nodes3d_bc(ns3d, ns3d_bc)

      READ(12, *) dataname

      CLOSE(12)

!-------------------------------------------------------------------

      OPEN(13, FILE = 'param_fea.dat')

      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) dataname
      READ(13, *) e
      READ(13, *) dataname
      READ(13, *) nu
      READ(13, *) dataname
      READ(13, *) rho
      READ(13, *) dataname
      READ(13, *) g

      !-------------------------------------------------------------

      ALLOCATE( esm3d_e(es3d_n) )
      ALLOCATE( esm3d_nu(es3d_n) )

      ALLOCATE( efv3d_rho(es3d_n) )

      DO ie = 1, es3d_n

       esm3d_e(ie)  = e
       esm3d_nu(ie) = nu

       efv3d_rho(ie) = rho

      END DO

      CALL set_elemstiffmat3d_e_nu(esm3d, esm3d_e, esm3d_nu)

      CALL set_elemexforcevec3d_rho(efv3d, efv3d_rho)
      CALL set_elemexforcevec3d_g(efv3d, g)

      !---------------------------------------------------------------

      READ(13, *) dataname

      ALLOCATE( fem3d_id_loaded(fem3d_nnodes_loaded) )
      ALLOCATE( fem3d_f_loaded(3, fem3d_nnodes_loaded) )

      DO idof_l = 1, fem3d_nnodes_loaded

       READ(13, *) fem3d_id_loaded(idof_l),                 &
                   ( fem3d_f_loaded(i, idof_l), i = 1, 3 )

      END DO

      CALL set_fem3d_f_loaded                        &
           (fem3d, fem3d_id_loaded, fem3d_f_loaded)

      READ(13, *) dataname

      CLOSE(13)

!---------------------------------------------------------------------

      DEALLOCATE( ns3d_x )
      DEALLOCATE( ns3d_u )
      DEALLOCATE( ns3d_bc )

      DEALLOCATE( es3d_connectivity )
      DEALLOCATE( esm3d_e  )
      DEALLOCATE( esm3d_nu )

      DEALLOCATE( efv3d_rho )

      DEALLOCATE( fem3d_id_loaded )
      DEALLOCATE( fem3d_f_loaded )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE start_appli
!####################################################################


      ! Run appli
!####################################################################
      SUBROUTINE run_appli()
!####################################################################

      INTEGER :: ns3d_n
      INTEGER :: le3d_nnodes
      INTEGER :: es3d_n
      INTEGER, ALLOCATABLE :: es3d_connectivity(:, :)
      INTEGER :: i
      INTEGER :: id
      INTEGER :: na
      INTEGER :: ie

      REAL(8), ALLOCATABLE :: ns3d_x(:, :)
      REAL(8), ALLOCATABLE :: ns3d_u(:)
      REAL(8), ALLOCATABLE :: esm3d_evec(:, :)
      REAL(8), ALLOCATABLE :: esm3d_svec(:, :), esm3d_s_mises(:)
```

```

!---------------------------------------------------------------------

      CALL cal_elements3d(es3d)

      CALL cal_fem3d(fem3d)

!---------------------------------------------------------------------

      CALL get_nodes3d_n(ns3d, ns3d_n)
      ALLOCATE( ns3d_x(3, ns3d_n) )
      CALL get_nodes3d_x(ns3d, ns3d_x)
      ALLOCATE( ns3d_u(3*ns3d_n) )
      CALL get_nodes3d_u(ns3d, ns3d_u)

      CALL get_localelement3d_nnodes(le3d, le3d_nnodes)

      CALL get_elements3d_n(es3d, es3d_n)
      ALLOCATE( es3d_connectivity(le3d_nnodes, es3d_n) )
      CALL get_elements3d_connectivity(es3d, es3d_connectivity)

      ALLOCATE( esm3d_evec(6, es3d_n) )
      ALLOCATE( esm3d_svec(6, es3d_n) )
      ALLOCATE( esm3d_s_mises(es3d_n) )
      CALL get_elemstiffmat3d_evec_svec                   &
           (esm3d, esm3d_evec, esm3d_svec, esm3d_s_mises)

!---------------------------------------------------------------------

      OPEN(14, FILE = 'result.inp')

      WRITE(14, '( 5(I8, 1X) )') ns3d_n, es3d_n, 3, 13, 0

      DO id = 1, ns3d_n

       WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )') &
```

```
           id, ( ns3d_x(i, id), i = 1, 3 )

    END DO

    DO ie = 1, es3d_n

     WRITE(14, '( 2(I8, 1X), (A5, 1X), 27(I8, 1X) )')         &
           ie, 1, '  hex',                                    &
           ( es3d_connectivity(na, ie),  na = 1, le3d_nnodes)

    END DO

    WRITE(14, '( 4(I8, 1X) )') 1, 3
    WRITE(14, '(A)') 'DISPLACEMENT, m'

    DO id = 1, ns3d_n

     WRITE(14, '( (I8, 1X), 3(E17.8, 1X) )')     &
           id, ( ns3d_u( 3*(id-1)+i ), i = 1, 3 )

    END DO

    WRITE(14, '( 14I8 )') 13, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    WRITE(14, '( (A, 1X) )') 'STRAIN_11, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRAIN_22, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRAIN_33, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRAIN_12, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRAIN_23, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRAIN_31, unit_unknown'
    WRITE(14, '( (A, 1X) )') 'STRESS_11, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_22, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_33, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_12, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_23, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_31, Pa'
    WRITE(14, '( (A, 1X) )') 'STRESS_MISES, Pa'
```

```

      DO ie = 1, es3d_n

       WRITE(14, '( (I8, 1X), 13(E17.8, 1X) )')                    &
             ie, ( esm3d_evec(i, ie), i = 1, 6 ),                  &
                 ( esm3d_svec(i, ie), i = 1, 6 ), esm3d_s_mises(ie)

      END DO

      CLOSE(14)

!-------------------------------------------------------------------

      DEALLOCATE( ns3d_x )
      DEALLOCATE( ns3d_u )

      DEALLOCATE( es3d_connectivity )

      DEALLOCATE( esm3d_evec )
      DEALLOCATE( esm3d_svec )
      DEALLOCATE( esm3d_s_mises )

!-------------------------------------------------------------------

      RETURN

!###################################################################
      END SUBROUTINE run_appli
!###################################################################


      ! Finish appli
!###################################################################
      SUBROUTINE finish_appli()
!###################################################################

```

```
      CALL del_nodes3d(ns3d)
      CALL del_localelement3d(le3d)
      CALL del_elements3d(es3d)
      CALL del_elemstiffmat3d(esm3d)
      CALL del_elemexforcevec3d(efv3d)
      CALL del_fem3d(fem3d)

!--------------------------------------------------------------------

      DEALLOCATE( ns3d )
      DEALLOCATE( le3d )
      DEALLOCATE( es3d )
      DEALLOCATE( esm3d )
      DEALLOCATE( efv3d )
      DEALLOCATE( fem3d )

!--------------------------------------------------------------------

      RETURN

!####################################################################
      END SUBROUTINE finish_appli
!####################################################################


!####################################################################
      END MODULE mod_appli
```