7.3 動的解析入力ファイル

ここで説明するファイルは、動的ソルバーで必要となる入力ファイルについてであり、特に SPACE で設定したパラメータ用ファイルを除いたものである。このパラメータ用ファイルは既に前節で説明した。

動的解析に必要となる入力用データファイルは以下のようであり、また、その右側に書かれている名前は、そのファイルを読み込むための専用サブルーチンである。

１．構造用データファイル	Get_structure( )
２．地震波加速度ファイル	Get_earth_load( )
３．節点荷重データファイル	Get_point_load( )
４．初期不整データファイル	Get_imperfection( )
５．ファイバーデータファイル	Fiber_input( )
６．質量データファイル	Get_mass( )
７．レーリー減衰データファイル	Get_damp( )

後節では、これらのファイル仕様と専用サブルーチンについて詳細に説明する。

7.3.1 構造用データファイル

本節では、構造用データファイルの仕様について述べる。構造用データファイルの詳細な仕様はリファレンスマニュアルを参照されたい。ここでは、サブルーチン Get_structure()の説明をすることでファイルの仕様を見ていこう。

まず、構造用データファイルの構造を示す。構造用データファイルは、次の５つのデータ群から構成されている。

１．　タイトル
２．　制御パラメータ
３．　節点に関するデータ
　　3.1　節点座標
　　3.2　節点局所座標
　　3.3　節点拘束条件
４．　要素データ
５．　部材に関するデータ
　　5.1　部材モデルデータ
　　5.2　部材主軸回転データ

このサブルーチンは、単にデータ入力するだけでなく、後に必要となる情報をこの入力データから作り出すコードを含む。以下にサブルーチン Get_structure()を示す。プログラムの右端についている番号、例えば、No.1 などは、上記のデータ群に対応している。なお、構造体や配列、その他の変数については、付録のインクルードファイル submain.h や数値

部材情報を入力する際、要素データと部材データとに分けて設定するようになっている。これは、解析モデルが大きくなると部材全てにヤング係数、他の材料定数などを設定する労力を省くためである。
構造解析システムでは、要素に関する情報は、構造体 Elemnt にまた、部材に関する情報は構造体 Member に設定される。一般に構造体 Element は、解析途中で情報が変化しないデータを保持し、Member は解析が進むに従って変化するデータか、もしくは各部材で異なった情報を保持する必要があるデータを集めて作られている。

計算用主サブルーチン submain_dynamic_a()を参照されたい。また、このファイルの仕様はリファレンスマニュアルを参照されたい。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_structure
C       ―――――――――――――――――――――――――――――――
C     ●   構造データを入力し、データをダンプファイルに出力する。
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_structure(Point,Member,Element,Parameter_C,
     *           Model_type,ierr)
c――――――――――――――――――――――――――――――――――★計算結果のダンプ出力ファイル番号
      parameter(damp_out = 76)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s  / Parameter_C
      record / member_s     / Member
      record / point_s      / Point
      record / element_s    / Element
      record / n_model_s    / Model_type
      dimension Member(*),Point(*),Element(*)
      character title*80
      dimension ire(6)                          ! ワークエリアとして使用
      integer RO_MODEL_NUMBER,TRI_MODEL_NUMBER
      data RO_MODEL_NUMBER/12/
      data TRI_MODEL_NUMBER/11/
c――――――――――――――――――――――――――――――――――★タイトル入力          No,1
      ierr=0
      read(5,*,err=9911,end=9918) n                                    ! 1
      if(n.ne.0) then
      write(damp_out,103)
  103 format(///5X,'----------- Title of dynamic analysis -----------')
      do j=1,n
      read(5,'(a80)',err=9911,end=9918) title                          !2
      write(damp_out,'(10x,a80)') title
      end do
      end if
c――――――――――――――――――――――――――――――――――★制御データ入力        No.2
      read(5,*,err=9911,end=9918) node,nelem,memb,nrbound,locod,njiku    !3
      write(damp_out,1001)'   節点数 = ',node,
     *                    '   要素数 = ',nelem,
     *                    '   部材数 = ',memb,          ! 部材数
     *                    '拘束節点数= ',nrbound,       ! 境界条件を与える拘束節点数
     *                    '局所座標数= ',locod,         ! 局所座標を与える節点数
     *                    '回転部材数= ',njiku          ! ｘ軸回りの回転を与える部材数
 1001 format(6(/2x,a,i8))
c――――――――――――――――――――――――――――――――――★節点データ入力        No.3
      write(damp_out,1002)
 1002 format(///1h ,'     節点座標'/)
      do i=1,node
      read(5,*,err=9912,end=9918) ii,x,y,z,idm                        !No.3.1  4
     write(damp_out,'(i4,3f12.3,i4)')ii,x,y,z,idm  ! ii:節点番号 idm:ダミー（０をセット）
      Point(ii).coord(1)  = x                                          !5
      Point(ii).coord(2)  = y
```

```
      Point(ii).coord(3)  = z
      end do
c―――――――――――――――――――――――――――――――★節点局所座標入力      No.3.2
c――――――――――――――――――――――★初期ゼロ設定
      do i=1,node                                                    ! 6
      Point(i).local_coord      = 0                                  ! 7
      do j=1,3
      Point(i).coord_local(j)   = 0.0                                ! 8
      Point(i).disp_initial(j)  = 0.0                                ! 9
      end do
      end do
c――――――――――――――――――――――★節点局所座標入力
      if(locod.ne.0) then                                            ! 10
      write(damp_out,1003)
 1003 format(///1h ,'     局所座標'/)
      it = 0
      do i=1,locod                                                   ! 11
      read(5,*,err=9912,end=9918) j,tlx,tly,tlz                      ! 12
      write(damp_out,'(i4,3f12.3)') j,tlx,tly,tlz
      it = it+ 1
      Point(j).local_coord      = it      ! この番号は、局所座標系への変換行列の番号となる。  13
      Point(j).coord_local(1)   = tlx                                ! 14
      Point(j).coord_local(2)   = tly
      Point(j).coord_local(3)   = tlz
      end do
      end if
c―――――――――――――――――――――――――――――――★境界拘束条件入力      No.3.3
c――――――――――――――――――――――★初期ゼロ設定
      do i=1,node                                                    ! 15
      do j=1,6
      Point(i).irest(j)  = 0                                         ! 16
      end do
      end do
c――――――――――――――――――――――★境界拘束条件入力
      if(nrbound.ne.0) then                                          ! 17
      write(damp_out,1004)
 1004 format(///1h ,'     境界条件'/)
      do i=1,nrbound                                                 ! 18
      read(5,*,err=9912,end=9918) j,(ire(k),k=1,6)                   ! 19
      write(damp_out,'(7i8)') j,(ire(k),k=1,6)
      do k=1,6
      Point(j).irest(k) = ire(k)                                     ! 20
      end do
      end do
      end if
c―――――――――――――――――――――――――――――――★線形要素モデルデータ入力  No.4
      write(damp_out,1005)
 1005 format(///1h ,'     要素モデルデータ'/)
      do  i=1,nelem                                                  ! 21
      rd1=0                                                          ! 22
      rd2=0
      sg1=0
      sg2=0
      read(5,*,err=9913,end=9918) m_type,e,g,a,rix,riy,riz,asy,asz,  ! 23
```

```
*          am1,am2,anp,ampy,ampz,nm_type
 write(damp_out,'(2i4,13e12.4,i4)') i,m_type,e,g,a,rix,riy,riz,
*                         asy,asz,am1,am2,anp,ampy,ampz,nm_type
 Element(i).n_section(1) = 0                ! 断面におけるファイバー数をゼロセット
c――――――――――――――――――――★要素番号 13,33 に対してデータ入力
      if(m_type .eq. 13.or.m_type .eq. 33) then                          ! 24
      Element(i).n_section(1) =  nm_type
C     if(riy.eq.0.) riy=riz
C     if(riy.gt.riz) riy=riz            ! 弱軸の断面二次モーメントを riy にセット
      endif
c――――――――――――――――――――★要素番号 11，15，21 に対してデータ入力
      if(m_type .eq. 11.or.m_type .eq. 15
     * .or.m_type .eq. 21) then                                           ! 25
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,
     *   (Element(i).n_section(j),j=1,2)
      write(damp_out,'(4f12.4,4i4)') rd1,rd2,sg1,sg2,
     *     (Element(i).n_section(j),j=1,2)
      endif
c――――――――――――――――――――★要素番号１2，22 に対してデータ入力
      if(m_type .eq. 12.or.m_type .eq. 22) then                          ! 26
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,
     *     (Element(i).n_section(j),j=1,3)
      write(damp_out,'(4f12.4,4i4)') rd1,rd2,sgi,sg2,
     *     (Element(i).n_section(j),j=1,3)
                                        endif
c――――――――――――――――――――★要素番号 31 に対してデータ入力
      if(m_type .eq. 31) then                                            ! 27
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,
     *   (Element(i).n_section(j),j=1,2)
      write(damp_out,'(4f12.4,4i4)') rd1,rd2,sg1,sg2,
     *   (Element(i).n_section(j),j=1,2)
      endif
c――――――――――――――――――――★要素番号 32 に対してデータ入力
      if(m_type .eq. 32) then                                            ! 28
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,
     *     (Element(i).n_section(j),j=1,3)
      write(damp_out,'(4f12.4,4i4)') rd1,rd2,sg1,sg2,
     *     (Element(i).n_section(j),j=1,3)
                                        endif
c――――――――――――――――――――★せん断タイプで修正 RO モデルの数をかぞえる
      if(m_type .eq. 2) then
      if(nm_type.eq.RO_MODEL_NUMBER.or.
     *   nm_type.eq.TRI_MODEL_NUMBER) then                               ! 29
      Model_type.n_m_ro_model= Model_type.n_m_ro_model + 1
      Element(i).n_section(1) = Model_type.n_m_ro_model
      endif
      endif
c――――――――――――――――――――★要素データを構造体にセット
      Element(i).element_type  =  m_type     ! 要素タイプ番号            ! 30
      Element(i).E             =  e          ! ヤング係数
      Element(i).G             =  g          ! せん断力係数
      Element(i).A             =  a          ! 断面積
      Element(i).RIx           =  rix        ! ねじり剛性
      Element(i).RIy           =  riy        ! y 軸回りの断面二次モーメント
```

```
      Element(i).RIz          =  riz       ! z 軸回りの断面二次モーメント
      Element(i).ASy          =  asy       ! y 軸に関するせん断変形に関係する断面積
      Element(i).ASz          =  asz       ! z 軸に関するせん断変形に関係する断面積
      Element(i).nm_damp      =  0         ! 部材減衰有無 （システムが自動でセットする）
      Element(i).ANP          =  anp       ! 軸方向耐力
      Element(i).AMPY         =  ampy      ! y 軸塑性モーメント
      Element(i).AMPZ         =  ampz      ! z 軸塑性モーメント
      Element(i).nm_type      =  nm_type   ! 履歴タイプ番号
      Element(i).i_rigid_length   =  rd1   ! i 端剛域長さ
      Element(i).j_rigid_length   =  rd2   ! j 端剛域長さ
      Element(i).i_shear_G        =  sg1   ! i 端せん断剛性
      Element(i).j_shear_G        =  sg2   ! j 端せん断剛性
c――――――――――――――――――――――★部材要素は自重を構造体にセット
      if(m_type.eq.1.or.m_type.gt.10) then                              ! 31
      Element(i).AM(1)        =  am1/980.    ! 要素単位長さ当たり(cm*2)質量
      Element(i).AM(2)        =  am2/980.    ! 要素単位長さ当たり(cm*2)質量
      else
c――――――――――――――――――――――★その他はそのままの値を構造体にセット
      Element(i).AM(1)        =  am1                                    ! 32
      Element(i).AM(2)        =  am2
      endif
      end do
c――――――――――――――――――――――――――★モデルタイプ別の要素数の計算（セロセット）
      do i=1,Model_type.n_e_models                                      ! 33
      Model_type.n_e_model(i)=0
      Model_type.n_m_model(i)=0
      end do
      Parameter_C.n_element_dll=0
c――――――――――――――――――――――★各タイプ別の要素数を数える。
      DO i=1,nelem                                                      ! 34
      m_type = Element(i).element_type
      do j=1,Model_type.n_e_models
      if(m_type .eq. Model_type.no_e_model(j)) then
      Model_type.n_e_model(j) = Model_type.n_e_model(j)+1
      Element(i).nm_damp = Model_type.n_damp(j)
      Element(i).element_type = j       ! モデル番号からモデルの記憶領域番号に変換
      goto 19
      end if
      end do
      ierr=14
      write(damp_out,'(a,i4,a)') ' 要素：',i,
     *            'のモデルタイプが存在しない。'
   19 continue
c――――――――――――――――――――――★dll を使用した要素数を数える。
      if(m_type .gt.1000)                                               ! 35
     *   Parameter_C.n_element_dll = Parameter_C.n_element_dll+1
      end do
      if(ierr.ne.0) goto 9914
c――――――――――――――――――――――――――★モデル別連続番号のセット（ゼロセット）
      do i=1,nelem                                                      ! 36
      Element(i).n_element=0
      enddo
c――――――――――――――――――――――★
      do i=1,nelem                                                      ! 37
```

```
      iel=Element(i).element_type
      if(Element(i).n_element.eq.0) then
      ii=0
      do j=i,nelem                                                    ! 38
      if(iel.eq.Element(j).element_type)then
      ii=ii+1
      Element(j).n_element=ii
      endif
      enddo
      endif
      enddo
c―――――――――――――――――――――★モデル別連続番号の出力
      write(damp_out,*) ' モデル別番号'                               ! 39
      do i=1,nelem
      write(damp_out,'(a,6i4)') ' element:',i,Element(i).n_element
      enddo
c――――――――――――――――――――――――――――――★部材データ入力      No.5
      write(damp_out,1006)                                            ! 40
 1006 format(///1h ,'     部材データ'/)
      ii1=1
      ii2=1
      ian=1
      do i=1,memb                                                     ! 41
      read(5,*,err=9915,end=9918) ii,i1,i2,ie,ian,ig,iso,ii1,ii2,     ! No.5.1
     *          rigid_i,rigid_j,shear_i,shear_j
      ii1=1            ! 現在  端部ピンは扱わない
      ii2=1            ! 現在  端部ピンは扱わない
      Member(ii).nm_element      = ie    ! 要素番号
      Member(ii).nm_point(1)     = i1    ! i 節点番号
      Member(ii).nm_point(2)     = i2    ! j 節点番号
      Member(ii).nm_analysis     = ian   ! 0:弾性 1:弾塑性
      Member(ii).nm_group        = ig    ! 部材グループ （解析に関係なし）
      Member(ii).ijp(1)          = ii1   ! i 節点結合状態 1:剛接合   0:ピン接合（現在はダミー）
      Member(ii).ijp(2)          = ii2   ! i 節点結合状態 1:剛接合   0:ピン接合（現在はダミー）
      Member(ii).nm_dll_element  = 0     ! dll 部材かどうかを示す（0:通常 1:dll 部材）
      Member(ii).rot_x           = 0.0   ! 主軸回転（度） 必要ならば後からデータ入力
      Member(ii).nm_so           = iso   ! 部材層番号 （解析に関係なし）
      Member(ii).nm_damp         = 0     !
      if(rigid_i.lt.0.)rigid_i=Element(ie).i_rigid_length             ! 42
      Member(ii).i_rigid_length=rigid_i     !  i 端剛域長さ
      if(rigid_j.lt.0.)rigid_j=Element(ie).j_rigid_length
      Member(ii).j_rigid_length=rigid_j     !  j 端剛域長さ
      if(shear_i.lt.0.)shear_i=Element(ie).i_shear_G
      Member(ii).i_shear_G=shear_i          !  i 端せん断剛性
      if(shear_j.lt.0.)shear_j=Element(ie).j_shear_G
      Member(ii).j_shear_G=shear_j          !  j 端せん断剛性
      if(ie.le. nelem ) then
      Member(ii).element_type = Element(ie).element_type
      if(Member(ii).element_type .gt. 50 )Member(ii).nm_dll_element=1
      write(damp_out,'(6i8,i12,2i8,4f10.2)')
     *          ii,i1,i2,ie,ian,ig,iso,ii1,ii2,
     *          rigid_i,rigid_j,shear_i,shear_j
      else
      write(damp_out,'(a,6i8)') ' data err:',ii,i1,i2,ie
```

```
      endif
c―――――――――――――――――――――――――★部材の弾塑性状態を全部材弾性にセット
      do j=1,3
      Member(ii).d_stat(j)=0                                                ! 43
      enddo
      end do
c―――――――――――――――――――――――――――――――――――★要素タイプ別個数チェック
      do 31 i=1,nelem                                                       ! 44
      j=0
      do ii=1,memb
      if(i.eq.Member(ii).nm_element) then
      j=j+1
      Member(ii).n_element_type=j
      endif
      enddo
   31 continue
c―――――――――――――――――――――――――――――――――――★モデルタイプ別部材数
      Parameter_C.n_member_dll=0
      DO 30 ii=1,memb                                                       ! 45
      i = Member(ii).nm_element
      if(i .le. nelem ) then
      m_type = Element(i).element_type
      do j=1,Model_type.n_e_models
      if(m_type .eq. j) then
      Model_type.n_m_model(j) = Model_type.n_m_model(j)+1
      Member(ii).n_model= j
      goto 29
      end if
      end do
      ierr=16
      write(76,'(a,i4,a)') ' 部材：',i,
     *       '  の要素データはモデルタイプに適合しない。'
   29 continue
      if(m_type .gt.1000)                                                   ! 46
     *   Parameter_C.n_member_dll = Parameter_C.n_member_dll+1
       else
      ierr = 16
      write(76,'(a,i4,a)') ' 部材：',i,
     *       '  は要素データに適合しない。'
      endif
   30 continue
      if(ierr.ne.0) goto 9916
c―――――――――――――――――――――――――――――――――――★モデル別通し番号計算
      do 32 i=1,Model_type.n_e_models                                       ! 47
      if(Model_type.no_e_model(i).ne.0) then
      j=0
      do ii=1,memb
      if(i.eq.Member(ii).n_model) then
      j=j+1
      Member(ii).n_model_type = j
      endif
      enddo
      endif
   32 continue
```

```
c——————————————————————————————★部材主軸回転入力      No.5.2
      if(njiku .ne. 0 ) then                                       ! 48
      write(damp_out,1007)
 1007 format(///1h ,'     部材主軸回転データ'/)
      do i=1,njiku
      read(5,*,err=9917,end=9918) ii,ax
      write(damp_out,'(i6,f12.2)') ii,ax
      Member(ii).rot_x          = ax    ! 主軸回転（度）
      end do
      end if
c——————————————————————————————★部材の両端局所座標のチェック
      do i=1,memb                                                  ! 49
      do j=1,2
      ie=Member(i).nm_point(j)
      Member(i).nm_local_coord(j) = Point(ie).local_coord
      end do
      end do
c——————————————————————————————★部材減衰の個数チェック
      Model_type.n_m_damp=0                                        ! 50
      write(damp_out,*) ' 部材データ'
      write(damp_out,*) 'ii, nm_damp, nm_element, element_type',
     *      'n_model, n_model_type, n_element_type'
      do ii=1,memb
      ie= Member(ii).nm_element
      if(Element(ie).nm_damp.ne.0) then
      Model_type.n_m_damp = Model_type.n_m_damp + 1
      Member(ii).nm_damp = Model_type.n_m_damp
      endif
      write(damp_out,'(i4,8i5)') ii,Member(ii).nm_damp,
     *      Member(ii).nm_element,Member(ii).element_type,
     *      Member(ii).n_model,Member(ii).n_model_type,
     *      Member(ii).n_element_type
      end do
c—————————————————————————★部材減衰の個数をセット
      Parameter_C.nc_member = Model_type.n_m_damp
      write(damp_out,'(a,i4,i4)') ' Total no. damp :',
     *             Parameter_C.nc_member
      return
 9911 continue
      ierr=11                                                      ! 51
      write(damp_out,'(a,a)') ' タイトル及び制御データ',
     *    'にエラー、矛盾がある。'
      return
 9912 continue
      ierr=12
      write(damp_out,'(a,a)') ' 節点データ、局所座標データ、',
     *      '境界条件にエラーがある。'
      return
 9913 continue
      ierr=13
      write(damp_out,'(a,a)') ' 要素データにエラーがある。'
      return
 9914 continue
      ierr=14
```

```
      write(damp_out,'(a)') ' 要素データのモデルタイプにエラーがある。'
      return
 9915 continue
      ierr=15
      write(damp_out,'(a,a)') ' 部材データにエラーがある。'
      return
 9916 continue
      ierr=16
      write(damp_out,'(a,a)') ' 部材データに矛盾がある。'
      return
 9917 continue
      ierr=17
      write(damp_out,'(a,a)') ' 主軸回転データにエラーがある。'
      return
 9918 continue
      ierr=18
      write(damp_out,'(a,a)') ' 構造データファイルが不足している。'
      return
       end
```

ここでは、構造データ入力のプログラムの内容について説明しておこう。データの細部仕様についてはリファレンスマニュアルを併せて参照されたい。以下の番号は上記プログラムに付された番号に対応する。

１．タイトルに使用する行数を読み込む。
２．タイトルを読み込み、次の行で DOUTPUT ファイルに書き出す。ここで指定している damp_out ファイルのユニット番号は 76 番である。
３．構造用データを読み込むための制御データを読み込む。その仕様は次の行の write 文で理解できる。
４．節点データを読み込む。ここでは、節点番号と全体座標系で表した節点の 3 次元座標である。次の行で DOUTPUT ファイルに書き出す。
５．節点に関する構造体に、入力した 3 次元座標をセットする。
６．節点に関する構造体で、座標以外の構造体成分のゼロクリアを行う。
７．その節点に局所座標の適用有無のパラメータをゼロセットする。なお、0 は適用しないを意味する。
８．3 方向の局所座標をゼロクリアする。
９．3 方向の初期変位をゼロクリアする。
10. 制御データで、局所座標の入力個数がない場合は、局所座標入力処理はスキップする。
11. 制御データで指定した局所座標の入力個数分入力する。
12. 節点番号とその節点の局所座標を入力する。局所座標は、x 軸、y 軸、z 軸について、全体座標系から何度回転しているかで設定する。
13. 構造体 Point(j).local_coord は、0 がその節点で局所座標を使用せ

ず、また、その他は局所座標を使用する節点の通し番号となっており、この番号が局所座標への変換行列のリンク情報として使用する。
14. 構造体成分 Point(j).coord_local に 3 方向の局所座標をセットする。
15. 全節点に対し、次の処理を行う。
16. 構造体成分 Point(i).irest（節点拘束条件）をゼロクリアする。１節点当たり 6 自由度に対し、全てゼロクリアする。
17. 境界条件の制御データ nrbound がゼロでない場合、境界条件を入力する。もし nrbound がゼロの場合、入力処理をスキップする。
18. 境界条件の制御データ nrbound の個数分、境界条件を入力する。
19. 境界条件として、節点番号の次に 6 自由度分入力する。
20. 境界条件を節点の構造体成分 Point(j).irest にセットする。
21. ここから、要素データ入力処理を行う。以下の処理は、制御データである要素数 nelem 分、データを入力する。要素データは部材モデルによって入力仕様が異なり、かなり煩雑となっている。
22. 入力がない場合に備えて、部材両端の剛域長さ rd1,rd2 とせん断剛性 sg1,sg2 をゼロにセットする。
23. 要素に関する第 1 レコードを入力する。
24. 第 2 レコードは要素によって異なる。ここは、部材モデル番号 13,33 について設定する。ただし、第 2 レコードはないが、部材中央に取り付くファイバー断面などの特殊断面の番号を設定する。
25. 部材モデル番号 11,15,21 に関するデータを入力する。部材両端の剛域長さ rd1,rd2 とせん断剛性 sg1,sg2、及び、両端に取り付くファイバー断面などの特殊断面の番号を入力する。
26. 部材モデル番号 12,22 に関するデータを入力する。部材両端の剛域長さ rd1,rd2 とせん断剛性 sg1,sg2、及び、両端及び中央に取り付くファイバー断面などの特殊断面の番号を入力する。
27. 部材モデル番号 31 に関するデータを入力する。部材両端の剛域長さ rd1,rd2 とせん断剛性 sg1,sg2、及び、部材両端に取り付くアナロジー断面の番号を入力する。
28. 部材モデル番号 32 についてデータを入力する。部材両端の剛域長さ rd1,rd2 とせん断剛性 sg1,sg2、及び、部材両端及び中央に取り付くアナロジー断面の番号を入力する。
29. 部材モデル番号 2 でしかも、その要素が修正 RO モデルか修正トリリニアモデルである場合はその数を数え、リンク情報を設定する。
30. 要素データを構造体 Element に代入する。
31. 入力した部材モデル番号が１か、もしくは 11 以上は単位長さ当たり

の重量データを質量に変換する。
32. 上記以外は、そのままデータを設定する。
33. モデルタイプ別の要素数を数える。最初に全モデルに対し、モデル別要素数を入れる構造体成分をゼロセットする。
34. 各部材モデル別に要素数を数える。要素構造体に保存されている減衰部材データを、要素構造体にセット、最後にモデル番号からモデルの記憶領域番号（連続した番号）を振り直す。
35. ダイナミックリンク用部材モデルの要素数を数える。（現在は使用していない）
36. 部材モデル別に分類した要素に、リンク情報として要素モデル別に連続番号を割り付ける。最初に連続番号を入れる構造体成分 Element(i).n_element をゼロクリアする。
38. 次に、連続番号を割り付ける処理を行う。
39. モデル別連続番号を出力する。
40. ここから、部材データ入力に関する処理を行う。
41. 部材数分、部材データを読み込み、構造体 Member にセットする。その入力仕様は、構造体にセットするコードにコメントとして書かれているので参照されたい。
42. 部材両端の剛域長さとせん断剛性をセットする。剛域長さとせん断剛性の設定仕様は、該当するデータに-1 がセットされていれば、要素で定義した剛域長さ及びせん断剛性がセットされ、0 以上の値が入力されていれば、その値がセットされる。
43. 部材の弾塑性状態を示す構造体成分 Member(ii).d_stat(j)を、全部材について 3 箇所（部材両端と中央）ゼロセット（弾性）する。
44.部材が指定している要素ごとに、部材の最初から連続番号を、部材の構造体 Member(ii).nm_element_type にセットする。
45. 全部材についてモデルタイプを設定する。部材で設定している要素がない場合は、エラー表示を行う。
46. dll 要素（動的リンク要素：現在使用不可）の数を数える。
47. 全部材について、部材の構造体成分 Member(ii).n_model_type に、モデル別に連続番号をセットする。つまり、部材データから要素を特定できるリンク情報をセットする。
48. 主軸回転を行う部材数がゼロでない場合、部材番号と主軸回転角度を入力する。ゼロの場合、入力処理をスキップする。主軸回転角度を構造体成分 Member(ii).rot_x にセットする。
49. 局所座標を表す節点構造体成分 Point(ie).local_coord を部材両端

の局所座標を表す構造体成分 Mmember(i).nm_local_coord にセットする。

50. 部材減衰を有している要素を部材が使用しているどうかチェックし、その数を数える。部材減衰を使用していれば部材減衰の構造体成分 Member(ii).nm_damp に連続番号をセットする。

51. 構造データ入力時、もしくはこのサブルーチン処理中にエラーが生じた場合、これ以降のエラー処理が実行される。ここでは、各エラーに合わせてエラー出力が damp_out ファイルに出力され、エラーコードを ierr にセットした後、このサブルーチンより戻る。

7.3.2 地震波加速度ファイル

本節で説明する地震波加速度ファイルは、簡単な構造となっており、また、他のサブシステムでも使用されている。地震加速度ファイルは、一般に、SPACE の管理者が用意するもので、システムを利用するユーザーはこのファイルの設定方法を知る必要はない。地震波ファイルは地震波専用のフォルダーを作り、まとめて管理することをお勧めする。

ここで扱える加速度ファイルは、以下のような仕様となっているので注意されたい。これ以外の仕様ファイルは、他のソフトあるいはマニュアルで本仕様に変換した後、使用することになる。このファイルのキーワードは、

X方向入力地震波「xeathf」
Y方向入力地震波「yeathf」
Z方向入力地震波「zeathf」

である。このファイルは、次のように

1.　ヘッダー部
2.　加速度データ

に分かれている。

この加速度ファイルを使用して解析する場合、SPACE の動的解析ソルバーの解析制限によって、同時に入力する加速度のサンプリング間隔は同じでなければならない。また、加速度ファイルで設定されているサンプリング間隔より短い増分時間で解析を行う場合、SPACE では、自動的に線形補間した値を用いる。

同時入力に使用する加速度のサンプリング間隔は同じでなくてはならない。サンプリング間隔より短い、増分時間では、線形補間する。

ヘッダー部は２行からなり、第１行目はタイトル、第２行目は加速度データの個数などである。まず、ヘッダーの第１行目は加速度ファイル

のタイトルであり、プレゼンターなどで表示される。例えば以下のようである。

```
EL CENTRO NS   1940  dt=0.02  Amax= 341.70
```

このデータは文字データで、先頭から、15バイト分が入力地震波名、6バイト分が年代、7バイト分が加速度の増分時間（秒）、14バイト分が加速度の最大値を示す。ただし、この部分のデータは動的ソルバーではコメント扱いであり、解析そのものには関係しない。

第２行目は、加速度の個数と加速度のサンプリング間隔（秒）である。

```
 2674        0.02
```

加速度の個数は整数、加速度のサンプリング間隔は実数で、形式は自由形式である。加速度データ部は、ヘッダー部で定義した加速度の個数分だけ、以下の仕様で繰り返す。加速度の単位は、Gal(cm/sec^2)である。

```
 0.3000000E+00  0.1900000E+01  0.6800000E+01  0.2900000E+01  0.2900000E+01
 0.5400000E+01  0.8300000E+01  0.5300000E+01  0.1400000E+01  0.5300000E+01
 0.1340000E+02  0.1240000E+02  0.6600000E+01  0.6100000E+01  0.1330000E+02
```

加速度データの仕様は、実数15桁で、１行に５つの加速度データを持つ形式である。ただし、現在では、加速度波形データは自由形式仕様となっており、加速度の個数分、ブランクやカンマで区切り、データを設定すれば良い。

地震加速度ファイルを入力するサブプログラムGet_earth_load()を以下に示す。地震加速度をセットするために、このプログラムは、２度呼ばれることになる。最初はヘッダー部を読み、加速度データの動的記憶領域をいくつに設定すべきかを決定する。記憶領域を動的に確保した後、再度このプログラムを呼び、加速度データを読み込む。ここでは、3 方向の加速度について、同様な方法でファイルを読むことになる。また、動的解析では、入力最大加速度を設定する場合があり、ここでは指定した値が最大加速度となるように処理する。

```
C     ------------------------------------------------------
C     ●   SUBROUTINE /Get_earth_load
C     ------------------------------------------------------
C     ●   地震加速度データを入力する(ok)
C     ------------------------------------------------------
      subroutine Get_earth_load(nx,acc_earth,Dynamic_load,ierrx)
```

```
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s    / Parameter_C
      record / dynamic_load_s / Dynamic_load
      dimension acc_earth(3,*)
      character aa*1
C     ------------------------------------------------------
c     nx             :1: パラメータセット 2:データ入力
c     Dynamic_load   :structure
c     acc_earth      :real*8
c     acc_earth(3,*) :real*8  地震加速度
c      structure / dynamic_load_s/
c      integer    load_point(3)         !  節点荷重ありか
c      integer    load_dynamic(3)       !  地震荷重ありか
c      real*8     amp_load_dynamic(3)   !  地震荷重の大きさ
c      real*8     dt_load_dynamic(3)    !＊地震荷重の増分時間(ここで設定)
c      integer    n_load_dynamic        !＊荷重ファイルの最大個数(ここで設定)
c      integer    load_mass             !  整合質量ありか
c      end structure
c      record / dynamic_load_s / Dynamic_load
C     ------------------------------------------------------
      ierrx=0
      if(nx.eq.1) then                                          ! 1
C     ------------------------------------------------------
c     ★        地震データを予備入力し、構造体の値を設定する
C     ------------------------------------------------------
      Dynamic_load.n_load_dynamic = 0
      do i=1,3
      write(76,*) Dynamic_load.load_dynamic(i),'=load_dynamic'
      enddo
C     ---------------------------------  x方向
      if (Dynamic_load.load_dynamic(1) .ne. 0) then             ! 2
      nfix=5
      nfi=57
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then                                        ! 3
      read(5,'(a)')  aa
      read(5,*) it,dt
      close(nfix)
      Dynamic_load.dt_load_dynamic(1)= dt
      if(it .gt. Dynamic_load.n_load_dynamic)
     *          Dynamic_load.n_load_dynamic = it
      else
      Dynamic_load.load_dynamic(1) = 0
      ierrx=1
      endif
      endif
C     ---------------------------------  y方向
      if (Dynamic_load.load_dynamic(2) .ne. 0) then             ! 4
      nfix=5
      nfi=58
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa
```

```
      read(5,*) it,dt
      close(nfix)
      Dynamic_load.dt_load_dynamic(2)= dt
      if(it .gt. Dynamic_load.n_load_dynamic)
     *                     Dynamic_load.n_load_dynamic = it
      else
      Dynamic_load.load_dynamic(2) = 0
      ierrx=1
      endif
      endif
C     ------------------------------  z方向
      if (Dynamic_load.load_dynamic(3) .ne. 0) then                    ! 5
      nfix=5
      nfi=59
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa
      read(5,*) it,dt
      close(nfix)
      Dynamic_load.dt_load_dynamic(3)= dt
      if(it .gt. Dynamic_load.n_load_dynamic)
     *                     Dynamic_load.n_load_dynamic = it
      else
      Dynamic_load.load_dynamic(3) = 0
      ierrx=1
      endif
      endif

      write(76,'(//a)') ' 構造体：Dynamic_load'
      write(76,*) Dynamic_load.n_load_dynamic,'=n_load_dynamic'
      do i=1,3
      write(76,*) Dynamic_load.dt_load_dynamic(i),'=dt_load_dynamic'
      write(76,*) Dynamic_load.load_dynamic(i),'=load_dynamic'
      enddo
C     ------------------------------------------------
c     ★        地震加速度を入力する
C     ------------------------------------------------
      else                                                             ! 6
      nfix=5
      do i=1,3                                                         ! 7
      if(Dynamic_load.load_dynamic(i).ne.0) then
      nfi=56 + i
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then                                               ! 8
      read(5,'(a)')  aa                                                ! 9
      read(5,*) it,dt
      read(5,*)(acc_earth(i,k),k=1,it)
      close(nfix)
      end if
      ss=0.
      do k=1,it
      if(ss.le.dabs(acc_earth(i,k))) ss = dabs(acc_earth(i,k))         ! 10
      end do
      if(it .lt. Dynamic_load.n_load_dynamic)then                      ! 11
```

```
do k=it+1,Dynamic_load.n_load_dynamic
acc_earth(i,k) = 0.0
enddo
endif
if(ss.ne.0.) then
ss = Dynamic_load.amp_load_dynamic(i)/ss                    ! 12
endif
do k=1,it
acc_earth(i,k) =acc_earth(i,k)*ss                           ! 13
end do
end if
end do
end if
return
 end
```

ここで、サブルーチン Get_earth_load()について説明する。

1．　このサブルーチンコールが 1 回目か 2 回目かを判定し、1 回目であればこの行以降を実行する。また、2 回目であれば、６へ飛ぶ。
2．　X 方向加速度波形のファイルを仮読みする。まず、ファイルをオープンするための標準的サブルーチン infile()をコールする。ここで、nfix=5 の 5 は read 文のユニット番号を表し、nfi=57 はファイル番号である。
3．もしファイルがない場合や、読み込みが許可されていない場合はエラーとして戻る。ファイルがオープンされると、ヘッダー部の 2 行を読み込み、加速度のサンプリング間隔を構造体にセットする。また、個数の最大値を構造体にセットする。
4．上記と同様に、Y 方向の加速度波形を仮読みする。
5．同じく、Z 方向の加速度波形を仮読みする。３方向の加速度波形の仮読みが終了すると、一旦サブルーチンを抜けることになる。その後で、加速度波形個数の最大値を用いて、動的領域を確保する。
6．第 2 回目のサブルーチンコールによる処理がこれ以降で行われる。
7．ここで、3 方向の加速度波形を読む。
8．ファイルをオープンし、エラーがある場合は、処理を中止する。
9．　第 2 レコードのヘッダー部分を読んだ後､加速度波形を読む｡ファイルの仕様は、自由形式でよい。
10．加速度の最大値を求める。
11．個数が最大値より小さい場合は、残りの部分はゼロセットする。
12．解析用加速度として指定した値を構造体成分 Dynamic_load.amp_load_dynamic(i)にセットする。
13．加速度波形の最大値を、指定した値に変換する。

7.3.3 節点荷重データファイル

本節では、節点荷重データファイルの仕様について述べる。節点荷重データファイルの仕様は以下のようである。第 1 行目は節点荷重が加わる節点数を表す。第 2 行目以降は、節点番号に続いて、6 自由度に対する荷重である。荷重の座標は、全体座標系に従う。したがって局所座標系を含む場合は、その局所座標系に自動的に変換する。以下に節点荷重データファイルの一例を示す。

```
      271
       8       0.      0.    -50.00       0.      0.      0.
      11       0.      0.    -50.00       0.      0.      0.
      12       0.      0.    -50.00       0.      0.      0.
      13       0.      0.    -50.00       0.      0.      0.
      14       0.      0.    -50.00       0.      0.      0.
      17       0.      0.    -50.00       0.      0.      0.
```

ここでは、サブルーチン Get_point_loadf()について述べる。このサブルーチンは、上記仕様にしたがって書かれたファイルを読み込む。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_point_loadf
C       ―――――――――――――――――――――――――――――――
C     ●   節点分布荷重を入力する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_point_loadf(fll_static_point,Parameter_C,
     *        Dynamic_load,ierr)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record /parameter_s/ Parameter_C
      record / Dynamic_load_s / Dynamic_load
      dimension fll_static_point(3,6,*),app(6)
C     ―――――――――――――――――――――――――――――――――
c           fll_static_point    : real*8   節点荷重
c           Parameter_C         : structure
c           Control             : structure
c           ierr                : integer エラーコード
C     ―――――――――――――――――――――――――――――――――
      ierr = 0
      n_point = Parameter_C.n_point
      do i= 1, 3                                                    ! 1
      do j=1,n_point                                                ! 2
      do k=1,6
      fll_static_point(i,k,j) = 0                                   ! 3
      enddo
      enddo
      if(Dynamic_load.load_point(i) .ne.0 ) then                    ! 4
      nfix=5
      nfi=4 + i
      call infile(nfi,nfix,ierr)                                    ! 5
      write(76,*) ierr
```

```
if(ierr.eq. 0 ) then                                    ! 6
write(76,*) ierr
read(5,*) ipp                                           ! 7
write(76,'(a,i4)') ' 節点荷重数：',ipp
do j=1,ipp
read(5,*) ip,(app(k),k=1,6)                             ! 8
write(76,'(i4,6f12.4)') ip,(app(k),k=1,6)
do k=1,6
fll_static_point(i,k,ip) = app(k)                       ! 9
end do
end do
close(nfix)
else                                                    ! 10
Dynamic_load.load_point(i)=0
endif
end if
end do
return
end
```

右端に付した番号にしたがって、プログラムの説明を行う。

1. 動的解析では、3 つの荷重ファイルを用いている。そこで、この 3 つの節点荷重ファイルについてデータ入力処理を行う。
2. まず、全節点、6 自由度について以降の処理を行う。
3. 荷重配列 fll_static_point をゼロにセットする。これで、最初に全節点、全自由度に対しゼロセットしたことになる。
4. 動的解析で、1 番目の荷重ファイルを使用するか否かについてチェックする。使用する場合はファイルをオープンし、データを読み込むことになる。
5. 指定した番号（nif=4+i）ファイルをオープンする。
6. ファイルオープンにエラーがある場合は、10 番に飛び、エラー処理を行う。
7. 荷重が加わっている節点数 ipp を読み込み、以降その節点数分処理を行う。
8. 節点番号に続いて、その節点の 6 自由度に対する荷重を読み込む。
9. 荷重をその節点の荷重配列 fll_static_point にセットする。データ入力終了後、ファイルを閉じる。
10. ファイルオープンにエラーがある場合、荷重構造体成分にゼロセットする。

7.3.4 節点荷重時刻歴データ

前節では、節点に分布する荷重ファイルを読み込むプログラムの説明を行った。後は、この荷重分布が時間的にどのように変化していくかについて決める事になる。本節では、この時間的変化を設定するサブルーチン Get_point_load()について説明する。この時間的変化に対し SPACE では、以下に示す 2 種類の荷重状態を想定している。

１．擬似的な静的荷重
２．風荷重（ただし、現バージョンでは使用不可となっている）

このサブルーチンは２回コールすることで、必要となる動的領域の確保を行っている。１回目のサブルーチンコールでは、風荷重の時刻歴変化を表すデータを読み込むための動的記憶領域の大きさを設定する。しかし、現在では、この風荷重は使用不可となっている。そのため、ここでは、擬似的な静的荷重を設定するための前処理のみ行っている。

この風荷重の時間的変化は、仕様が決まっていないため現在は使用できない。ただし、このサブルーチンでは、時間的変化を記述するファイルを読み込むように作られている。

このサブルーチンを２度目にコールすると擬似的な静的荷重の時刻歴を設定する。静的荷重に時刻歴とは何か違和感があるが、この動的解析では、あくまでも解析事態は動的解析手法を用いて数値解析を行うため、ゆっくりと荷重を増加させるという時間的変化を設定する必要が生じる。ここでは、第 7.2.3 節で説明した配列 fs、fl、ifp を用いて、荷重の履歴 fdd_point にデータ設定を行う。ただし、配列 fs はステップを表す時間を、fl は荷重係数を、ifp は荷重の形式を表す。以下に、サブルーチン Get_point_load()を示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_point_load
C       ―――――――――――――――――――――――――――――――――
C     ●   節点分布荷重を入力する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_point_load(nx,fdd_point,
     *         Dynamic_load,ierrx,Newmark_P,fs_st,fl_st,ifp_st)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / Dynamic_load_s / Dynamic_load
      record / newmark_s / Newmark_P
      dimension fdd_point(3,*)
      dimension fs_st(10,3),fl_st(10,3),ifp_st(10,3)
C     ―――――――――――――――――――――――――――――――――
c           nx                : nx=1 予備、２：実際の設定およびデータ入力
c           fdd_point         : real*8   節点荷重履歴
c           Dynamic_load      : structure
c           Newmark_P         : structure
c           ierrx             :integer エラーコード
C     ―――――――――――――――――――――――――――――――――
```

```
      if(nx.eq.1) then                                                  ! 1
      Dynamic_load.n_load_point = 0
      ierrx=0
      do i=1,3
      write(76,*) Dynamic_load.load_point(i),'=load_point'
      enddo
      if (Dynamic_load.load_point(1) .eq. 2) then                       ! 2
c―――――――――――――――――――――――――――――――――★節点荷重履歴をファイルより入力
      nfix=5
      nfi=61
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa
      read(5,*) it,dt                                                   ! 3
      close(nfix)
      Dynamic_load.dt_load_point(1)= dt
      if(it .gt. Dynamic_load.n_load_point)
     *                    Dynamic_load.n_load_point = it
      else
      Dynamic_load.load_point(1) = 0
      ierrx=1
      endif
c―――――――――――――――――――――――――――――――――★静的節点荷重をセットする
      elseif (Dynamic_load.load_point(1) .eq. 1) then                   ! 4
      Dynamic_load.dt_load_point(1)=Newmark_P.dt
      it=Newmark_P.n2_step
      if(it .gt. Dynamic_load.n_load_point)
     *                     Dynamic_load.n_load_point = it
      endif
      if (Dynamic_load.load_point(2) .eq. 2) then                       ! 5
c―――――――――――――――――――――――――――――――――★節点荷重履歴をファイルより入力
      nfix=5
      nfi=62
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa
      read(5,*) it,dt
      close(nfix)
      Dynamic_load.dt_load_point(2)= dt
      if(it .gt. Dynamic_load.n_load_point)
     *                     Dynamic_load.n_load_point = it
      else
      Dynamic_load.load_point(2) = 0
      ierrx=1
      endif
c―――――――――――――――――――――――――――――――――★静的節点荷重をセットする
      elseif (Dynamic_load.load_point(2) .eq. 1) then
      Dynamic_load.dt_load_point(2)=Newmark_P.dt
      it=Newmark_P.n2_step
      if(it .gt. Dynamic_load.n_load_point)
     *                     Dynamic_load.n_load_point = it
      endif
      if (Dynamic_load.load_point(3) .eq. 2) then                       ! 6
c―――――――――――――――――――――――――――――――――★節点荷重履歴をファイルより入力
```

```
      nfix=5
      nfi=63
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa
      read(5,*) it,dt
      close(nfix)
      Dynamic_load.dt_load_point(3)= dt
      if(it .gt. Dynamic_load.n_load_point)
     *                    Dynamic_load.n_load_point = it
      else
      Dynamic_load.load_point(3) = 0
      ierrx=1
      endif
c―――――――――――――――――――――――――――――――――★静的節点荷重をセットする
      elseif (Dynamic_load.load_point(3) .eq. 1) then
      Dynamic_load.dt_load_point(3)=Newmark_P.dt
      it=Newmark_P.n2_step
      if(it .gt. Dynamic_load.n_load_point)
     *                    Dynamic_load.n_load_point = it
      endif
      write(76,'(//a)') ' 構造体：Dynamic_load'                              ! 7
      write(76,*) Dynamic_load.n_load_point,'=n_load_point'
      do i=1,3
      write(76,*) Dynamic_load.dt_load_point(i),'=dt_load_point'
      write(76,*) Dynamic_load.load_point(i),'=load_point'
      enddo
      else                                                                  ! 8
c―――――――――――――――――――――――――――――――――★静的節点荷重を実際に入力、セットする
      ierr = 0
      nfix=5
      do i=1,3                                                              ! 9
      it=0
      if(Dynamic_load.load_point(i).eq.2) then                              ! 10
c―――――――――――――――――――――――――――――――――★節点荷重履歴をファイルより入力
      nfi=60 + i
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(5,'(a)')  aa                                                     ! 11
      read(5,*) it,dt
      read(5,*)(fdd_point(i,k),k=1,it)
      close(nfix)
      endif
      if(it .lt. Dynamic_load.n_load_point)then
      do k=it+1,Dynamic_load.n_load_point
      fdd_point(i,k) = 0.0
      enddo
      endif
      elseif(Dynamic_load.load_point(i).eq.1) then                          ! 12
c―――――――――――――――――――――――――――――――――★静的節点荷重をセットする
      dt = Dynamic_load.dt_load_point(i)
      f1sec=Newmark_P.f1_T
      nstl=Dynamic_load.n_load_point-1
      call static_load_set(dt,nstl,f1sec,fs_st,fl_st,ifp_st,                ! 13
```

```
*                   fdd_point,Dynamic_load.n_load_point,i)
 endif
 enddo

 endif
 return
 end
```

上記プログラムの説明を、コード右側に付した番号に従って行う。

1. このサブルーチンコールが１度目か２度目かを判定し、１度目の場合は、以下の処理を行う。２度目の場合は、処理は８．へ移動する。
2. 節点荷重分布のファイル１に対する時刻歴設定を行う。最初に、風荷重か擬似的静的荷重かを判定する。Dynamic_load.load_point(1)が２の場合は風荷重を表し、以下の処理を行う。
3. 対象となるファイルをオープンし、ファイルのヘッダー部分を読み込む。ファイル番号は 61 番に割り付けられている。このファイルの仕様は加速度波形と同一である。個数とサンプル間隔を構造体にセットする。ここでは、ヘッダー部を仮読みして、動的領域確保に使用する。
4. 静的荷重の場合は、擬似的な静的荷重が加わる時間に関するデータとして、構造体に増分時間と個数を設定する。
5. 節点荷重分布のファイル２に対する時刻歴設定を行う。ファイル番号は 62 番である。処理は、2、3、4 と同様である。
6. 上記の 5 と同様に、節点荷重分布のファイル３に対する時刻歴設定を行う。ファイル番号は、63 番である。処理は、2、3、4 と同様である。
7. 設定した荷重履歴に関する構造体成分の内容を出力する。その後、サブルーチンから抜け出し、必要な動的領域を確保した後、再度このサブルーチンをコールしてファイルの内容をシステム内に取り込むことになる。
8. ２度目のサブルーチンコールでは、以降の処理を行う。
9. 節点荷重分布を表す各ファイルに対し、時刻歴設定処理を行う。
10. 最初に、風荷重か擬似的静的荷重かを判定する。構造体成分である Dynamic_load.load_point(1)が２の場合は風荷重を表し、以下の処理を行う。また、１の場合は擬似的静的荷重を表す。
11. 風荷重用の時刻歴データファイルを読み込む。このファイルの仕様は、地震波形と同じである。
12. 擬似的静的解析の場合は以下の処理を行う。ここで、f1sec は第１段階（擬似的静的解析）の解析時間を表し、nstl は、動的解析の全ステップ数である。

13. 使用者が設定した静的荷重に関するパラメータから、擬似的な静的荷重履歴を作り出すサブルーチンをコールする。サブルーチン名は、static_load_set()である。

次に、サブルーチン static_load_set()について説明する。このサブルーチンは、使用者がダイアログで設定した静的荷重の時刻歴パラメータを用いて、実際の増分時間毎の時刻歴データに変換するものである。この時刻歴データは擬似的な静的荷重なので、動的解析の第１段階に対応する。以下に、このサブルーチンとそれに関連するサブルーチンpsetpt()と pset()を示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /ctlstx
C       ―――――――――――――――――――――――――――――――――
C     ●   節点静的荷重を定義する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine  static_load_set(delt,nstl,f1sec,fs,fl,ifp,
     *                            fdd_point,mstep,i_st)
      IMPLICIT REAL*8(A-H,O-Z)
      dimension fs(10,3),fl(10,3),ifp(10,3)
      dimension fdd_point(3,*)
      do j=1,mstep                                                  ! 1
      fdd_point(i_st,j)=0.
      enddo
      if(nstl.le.1.or.delt.le.0.) return                            ! 2
      i=i_st
      stt=0.
      m=1
      fls=0.
      do 22 k=1,10                                                  ! 3
      ett=fs(k,i )
      if(stt.gt.ett) goto 20
      if(stt.eq.ett) then
      call qsetpt(i ,m,0,fdd_point,fls,fl(k,i ),                    ! 4
     *            ifp(k,i ),mstep)
      else
      if(ett.gt.f1sec) ett=f1sec                                    ! 5
      ifs=(ett-stt)/delt
      call qsetpt(i ,m,ifs,fdd_point,fls,fl(k,i ),                  ! 6
     *           ifp(k,i ),mstep)
      endif
      if(fs(k,i).gt.f1sec) goto 20                                  ! 7
      if(fs(k,i).eq.0.and.k.gt.2) goto 20                           ! 8
      stt=ett                                                       ! 9
      fls=fl(k,i)
 22   continue
 20   continue
      if(m.eq.mstep) return                                         ! 10
      do j=ｓm+1,mstep                                              ! 11
```

```
      fdd_point(i,j)=fdd_point(i,m)
      enddo
      return
      end
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /qsetpt
C       ――――――――――――――――――――――――――――――――――
C     ●   静的荷重（時刻歴）を計算(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine qsetpt(i,m,ifs,sp,fls,fl,
     *           ifp,mstep)
      IMPLICIT REAL*8(A-H,O-Z)
      dimension sp(3,mstep)
      if(ifs.le.0) then                                          ! 12
      sp(i,m)=fls
      sp(i,m+1)=fl
      m=m+1
      else
      call qset(ifs,m,fls,fl,sp,ifp,mstep,i)                     ! 13
      endif
      return
      end
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /qset
C       ――――――――――――――――――――――――――――――――――
C     ●   静的荷重（時刻歴）を計算その２(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine qset(ifs,m,fls,fl,p,ifp,mstep,i)
      IMPLICIT REAL*8(A-H,O-Z)
      dimension p(3,mstep)
      data pai/3.1415926/
      if(ifp.eq.1) then                                          ! 14
      dt=(fl-fls)/ifs
      do 10 k=1,ifs
      p(i,m)=fls+dt*k
      p(i,m+1)=p(i,m)
      m=m+1
 10   continue
      else                                                       ! 15
      dt=pai/ifs
      dtt=(fl-fls)*0.5
      pai2=pai/2.
      flsx=fls+dtt
      do 15 k=1,ifs
      p(i,m)=flsx+dtt*sin(dt*k-pai2)
      p(i,m+1)=p(i,m)
      m=m+1
 15   continue
      endif
      return
       end
```

上記プログラムの説明を、コード右側に付した番号に従って行う。

1. 荷重履歴を入れる領域をゼロクリアする。ここで、mstep は解析で使用する全ステップ数であり、i_st は荷重ファイルの番号を表し、順次1、2、3 の順にサブルーチンが３度コールされる。
2. 解析ステップ数と増分時間が所定の値以下である場合は、このサブルーチンから戻ることになる。
3. 使用者が設定可能な静的荷重の個数は 10 までであり、ここでは、10回にわたって、データ有無のチェックとその間の荷重履歴を作り出す。ここで、stt は前ステップの時刻を示し、ett は、現ステップの時刻を示す。ett が stt より値が大きい場合は、データ終了とみなして設定を中止する。例えば、途中で 0 が入っているとその前ステップでデータ設定が終わっているとみなす。
4. 前ステップと現ステップの時刻が同一の場合の処理を行う。ここでは、解析ステップである ifs をゼロにセットして、サブルーチン qsetpt() をコールする。
5. 現ステップの時刻が第１段階の解析時間を越えているかどうかチェックする。超えている場合は、現ステップの時刻を第１段階の解析時間に変更する。また、前ステップの時刻と現ステップの時刻の間で、解析ステップ ifs がいくつ必要となるか、増分時間 delt を用いて計算する。
6. 解析ステップである ifs をセットして、サブルーチン qsetpt()をコールする。
7. 現ステップの時刻が第１段階の解析時間を越えている場合は、処理を終了する。
8. 現ステップの時刻が 0 で、しかも、そのステップ番号が 2 以上であれば終了する。
9. 現ステップのために、時刻と荷重係数 **fl** を現ステップ用から前ステップ用に設定し直す。
10. 以降のコードは、静的荷重の第１段階動的解析における時刻歴を設定した後の処理である。静的荷重の時刻歴の全個数 m が解析ステップ数を超えている場合は、ここでサブルーチンから戻る。
11. 上記で設定した時刻歴の全個数 m から、解析ステップ数までの残りの時刻歴データを設定する。残りの時刻歴データは、上記で設定した最後の値をそのまま使用する。つまり、第２段階の動的解析では、この最後の値がそのまま使用され、静的荷重が加わった状態となる。

12．このサブルーチン qsetpt()では、前ステップの時刻と現ステップの時刻が一致する場合と、そうでない場合とに分けて処理する。一致する場合は、構造体に引数で受け渡された値を設定する。ここで値が設定されると、同時刻でステップ荷重のように荷重の値が上下する。
13．一致しない場合は、その間の時刻歴データを作ることになる。この時刻歴データを作成するサブルーチン qset()をコールする。
14．このサブルーチン qset()では、荷重形式 ifp の違いによって、２つの処理が行われる。荷重形式が直線の場合（ifp=1）は、以降の処理を行い、荷重履歴を作成する。
15．荷重形式が SIN 型の場合（ifp=2）は、以降の処理を行う。

7.3.5 初期不整データファイル

ここでは、初期不整データ、特に初期変位データファイルの仕様とその入力プログラムについて説明する。ファイルの仕様は、非常に単純であり、以下のようである。

```
      2
     3     0.    0.   0.1
     4     0.    0.   0.1
```

第１行目は、初期変位が存在する節点の個数（整数）を表す。第２行目以降は、節点の初期変位を表し、全て同じ仕様である。左から、節点番号（整数)、次の３つは、全体座標系における x、y、z 方向に対応する各節点のずれ（初期変位：単位 cm）を表す。

サブルーチン Get_imperfection()は、初期不整データを読み込むもので、内容を以下に示す。

```
C     ------------------------------------------
C     ●   SUBROUTINE /Get_imperfection
C       ------------------------------------------
C     ●   初期不整データを入力し、節点座標値に加える(ok)
C     ------------------------------------------
      subroutine Get_imperfection(amp_imperfection,Point,Parameter_C)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s  / Parameter_C
      record / point_s      / Point
      dimension Point(*)
C     ------------------------------------------
c     amp_imperfection :real*8 初期不整の大きさ
c     Parameter_C      :structure
c     Point            :structure
```

```
C     ―――――――――――――――――――――――――――――――――
      read(5,*) npoint,                                                  ! 1
      do i=1,npoint
      read(5,*) i1,am1,am2,am3,                                          ! 2
      Point(i1).disp_initial(1) =  am1 * amp_imperfection  ,             ! 3
      Point(i1).disp_initial(2) =  am2 * amp_imperfection
      Point(i1).disp_initial(3) =  am3 * amp_imperfection
      end do
      do i=1,Parameter_C.n_point
      do j=1,3
      Point(i).coord(j) = Point(i).coord(j)+Point(i).disp_initial(j) ,   ! 4
      end do
      end do
      return
      end
```

上記プログラムの説明を、コード右側に付した番号に従って行う。

1. 初期不整が存在する節点の個数を入力する。
2. その個数分、初期変位を入力する。
3. 入力した３方向の初期変位を構造体成分 Point(i1).disp_initial に設定する。ダイアログで設定した初期不整 amp_imperfection を、この構造体の初期変位に掛け算し、実際の初期変位とする。また、この構造体は、既にゼロ初期設定を行っている。
4. 全節点に対し、先に入力した初期変位を節点の座標に加えて、節点座標を移動させる。

7.3.6 ファイバーデータファイル

本節では、ファイバーデータの入力ファイルの仕様とそのファイルの入力プログラムについて解説する。ファイバーデータファイルは、特殊断面用としても用いられており、他にアナロジーモデルの断面データ、マルチスプリング用の断面データのファイルとして使用される。ファイバー用データファイルの一例を以下に示し、内容を説明する。第 1 行目は断面数を表し、この値の数だけ断面データを読み込むことになる。第 2 行目は断面形状に関するパラメータを表す。その行の 1 桁目はこの断面データに付けられたコード番号であり、第 2 のパラメータはこの断面に含まれるファイバー数を表す。第 3 行目以降は、このファイバー数分のデータを示す。このファイバーデータは履歴特性を表し、1 行もしくは 2 行となる。

```
  1
 1  66  3.00  1.00  16.00  32.00  20.00  40.00  0.80  1.30  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00        0.00        0.00        0.00        0.00
   1   1  1.0400   0.0000   19.3500   2100.0000  21.0000  2.4000  810.0000   0.0000  0.0000
   2   1  1.0400   0.0000  -19.3500   2100.0000  21.0000  2.4000  810.0000   0.0000  0.0000
   3   1  1.5600  -9.4000   19.3500   2100.0000  21.0000  2.4000  810.0000   0.0000  0.0000
```

それでは、このファイルを読み込むプログラムを見てみよう。このプログラムは、データを読み込むと同時に、後のデータ処理のために、多くの処理を行っている。ここでは、それらについて解説する。ただし、各部材モデルに対し、同様の処理がほとんどなので、同じような処理は省くことにする。プログラム全体を見たい場合は、付録を参照されたい。

このサブプログラムは、2 回コールされる。一回目は、データ格納のための動的領域の大きさを決める処理と、モデル別の個数を調査する処理が行われる。このサブルーチンを抜けた後、動的領域を確保し、2 回目のサブルーチンコールが行われ、実際のデータを入力する。以下にこのサブルーチンを示す。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_input
C     ―――――――――――――――――――――――――――――――
C     ●   ファイバー要素の入力(ok)
C     ―――――――――――――――――――――――――――――――
c      ファイバー履歴タイプ
c      nm_type: 1      バイリニア型
c               2      トリリニア型
c               3      直線コンクリート型
c               4      曲線コンクリート型
c               5      等方硬化＋移動硬化バイリニア型
c               6      等方硬化＋移動硬化トリリニア型
c               7      非対称バイリニア型
c               8      非対称トリリニア型
c
c        アナロジーモデル履歴タイプ
c              11      完全弾塑性型
c              12      等方硬化弾塑性型（開発中）
c              13      移動硬化弾塑性型（開発中）
c              14      等方硬化＋移動硬化弾塑性型（開発中）
c
c        マルチスプリング履歴タイプ
c              21      武田モデル（開発中）
c              22      等方硬化＋移動硬化トリリニア型（開発中）
c
C     ―――――――――――――――――――――――――――――――
      subroutine Fiber_input(it,ierr,n_member,n_element,Member,
     *      Element,Model_type,E_model_fiber,M_model_fiber,
     *      E_model11,M_model11,E_model12,M_model12,
     *      E_model13,M_model13,E_model15,M_model15,
     *      E_model21,M_model21,E_model22,M_model22,
```

```
*      E_model31,M_model31,E_model32,M_model32,E_model33,M_model33)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s          / Member
      record / element_s         / Element
      record / E_model11_s       / E_model11
      record / E_model12_s       / E_model12
      record / E_model13_s       / E_model13
      record / E_model15_s       / E_model15
      record / E_model21_s       / E_model21
      record / E_model22_s       / E_model22
      record / E_model31_s       / E_model31
      record / E_model32_s       / E_model32
      record / E_model33_s       / E_model33
      record / M_model11_s       / M_model11
      record / M_model12_s       / M_model12
      record / M_model13_s       / M_model13
      record / M_model15_s       / M_model15
      record / M_model21_s       / M_model21
      record / M_model22_s       / M_model22
      record / M_model31_s       / M_model31
      record / M_model32_s       / M_model32
      record / M_model33_s       / M_model33
      record / E_model_fiber_s   / E_model_fiber
      record / M_model_fiber_s   / M_model_fiber
      record / n_model_s         / Model_type
C     ――――――――――――――――――――――――――――――――――
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension E_model11(*),M_model11(*),E_model12(*),M_model12(*)
      dimension E_model13(*),M_model13(*),E_model15(*),M_model15(*)
      dimension E_model21(*),M_model21(*),E_model22(*),M_model22(*)
      dimension E_model31(*),M_model31(*),E_model32(*),M_model32(*)
      dimension E_model33(*),M_model33(*)
      dimension Member(*),Element(*)
      dimension ddm(20)
      ierr=0
      if(it.eq.0) then                                                    ! 1
c――――――――――――――――――――――――――――――――★ファイバーデータの予備入力
      read(5,*,err=999) nm          ! 最大断面数                          ! 2
      write(76,'(a,i4)') ' ファイバー断面総数：',nm
      ii=0
      do i=1,nm                                                           ! 3
      read(5,*,err=999) n_m,nmm,(ddm(j),j=1,20)     ! 断面番号、エレメント数
      write(76,'(a,i4,a,i4)') '断面番号：',n_m,'  ファイバー数：',nmm
c――――――――――――――――――――――――――――――――★断面ファイバー数のセット
      kk1  = 0
      kk2  = 0
      kk3  = 0
      kk5  = 0
      kk21 = 0
      kk22 = 0
      kk31 = 0
      kk32 = 0
```

```
      kk33 = 0
      do i1=1,n_element                                          ! 4
      itype_m = Model_type.no_e_model(Element(i1).element_type)
      if(itype_m.eq.11) then                                     ! 5
c―――――――――――――――――――――――★モデル１１
      kk1 = kk1 + 1
      do k=1,2
      if(Element(i1).n_section(k).eq.n_m) then
      Element(i1).nm_section(k)   = nmm
      if(k.eq.1) then
      E_model11(kk1).n_section_1  = nmm
      E_model11(kk1).nm_section_1 = ii + 1
      endif
      if(k.eq.2) then
      E_model11(kk1).n_section_2  = nmm
      E_model11(kk1).nm_section_2 = ii + 1
      endif
      endif
      enddo
      endif
c―――――――――――――――――――――――★モデル１２
      if(itype_m.eq.12) then                                     ! 6
      endif
c―――――――――――――――――――――――★モデル１３
      if(itype_m.eq.13) then
      endif
c―――――――――――――――――――――――★モデル１５
      if(itype_m.eq.15) then
      endif
c―――――――――――――――――――――――★モデル２１
      if(itype_m.eq.21) then
      endif
c―――――――――――――――――――――――★モデル２２
      if(itype_m.eq.22) then
      endif
c―――――――――――――――――――――――★モデル３１
      if(itype_m.eq.31) then
      endif
c―――――――――――――――――――――――★モデル３２
      if(itype_m.eq.32) then
      endif
c―――――――――――――――――――――――★モデル３３
      if(itype_m.eq.33) then
      endif
      enddo
c―――――――――――――――――――――――★
      do j=1,nmm                                                 ! 7
      ii = ii + 1
      read(5,*,err=999) n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az
      if(nm_type.le.10) then
      goto(901,902,903,904,905,906,907,908,909,910),nm_type      ! 8
 901  continue
      goto 900
 902  continue                                                   ! 9
```

```
      read(5,*,err=999) E_3,Q_2,beta,beta_2                ! 対称トリリニア型 （スチール用）
      goto 900
 903  continue
      read(5,*,err=999) AK_3,AK_4,Q_2,Q_3,Q_4              ! 直線コンクリート型
      goto 900
 904  continue
      read(5,*,err=999) AK_4,Q_3,STR_3,STR_7               ! 曲線コンクリート型
      goto 900
 905  continue
      read(5,*,err=999) beta                               ! 等方硬化＋移動硬化バイリニア型
      goto 900
 906  continue
      read(5,*,err=999) E_3,Q_2,beta,beta_2                ! 等方硬化＋移動硬化トリリニア型
      goto 900
 907  continue
      read(5,*,err=999) Ec_1,Ec_2,Qc_1,beta                ! 非対称バイリニア型
      goto 900
 908  continue
      read(5,*,err=999) E_3,Q_2,Ec_1,Ec_2,Ec_3,Qc_1,Qc_2,beta,beta_2 ! 非対称トリリニア型
      goto 900
 909  continue
      goto 900
 910  continue
      goto 900
      elseif(nm_type.le.20) then
      endif
 900  continue
      enddo
      enddo
c―――――――――――――――――――――――★要素ファイバー数セット
      Model_type.nm_div_felement= ii                                          ! 10
      write(76,'(a,i5)') ' ファイバー数：',ii
c―――――――――――――――――――――――――――★部材断面ファイバー数のセット
      jj = 0
      do i=1,n_member                                                         ! 11
      i1   = Member(i).nm_element
      immm = Member(i).n_model_type               ! モデルタイプ別番号
      itype_m = Model_type.no_e_model(Element(i1).element_type)
c      write(76,'(a,4i4)') ' fiber ',i,i1,immm,itype_m
c―――――――――――――――――――――――★モデル１１
      if(itype_m.eq.11) then                                                  ! 12
      k = 1
      M_model11(immm).n_section_1   = Element(i1).nm_section(k)
      M_model11(immm).nm_section_1  = jj + 1
      jj = jj + Element(i1).nm_section(k)
      k  = 2
      M_model11(immm).n_section_2   = Element(i1).nm_section(k)
      M_model11(immm).nm_section_2  = jj + 1
      jj = jj + Element(i1).nm_section(k)
      endif
c―――――――――――――――――――――――★モデル１２
      if(itype_m.eq.12) then                                                  ! 13
      endif
c―――――――――――――――――――――――★モデル１３
```

```
      if(itype_m.eq.13) then
      endif
c―――――――――――――――――――――★モデル１５
      if(itype_m.eq.15) then
      endif
c―――――――――――――――――――――★モデル２１
      if(itype_m.eq.21) then
      endif
c―――――――――――――――――――――★モデル２２
      if(itype_m.eq.22) then
      endif
c―――――――――――――――――――――★モデル３１
      if(itype_m.eq.31) then
      endif
c―――――――――――――――――――――★モデル３２
      if(itype_m.eq.32) then
      endif
c―――――――――――――――――――――★モデル３３
      if(itype_m.eq.33) then
      endif
      enddo
      Model_type.nm_div_fmodel =  jj       ! ファイバー要素の最大数
C     ―――――――――――――――――――――――――――――――――――
C        ★     ファイバーデータの入力その２
C     ―――――――――――――――――――――――――――――――――――
      else                                                          ! 14

      read(5,*,err=999) nm                                          ! 15
      write(76,'(a,i4)') ' Number of sections:',nm
      ii = 0
      do i=1,nm
      read(5,*,err=999) n_m,nmm,(ddm(j),j=1,20)                     ! 16
      write(76,'(a,2i4,20f10.3)') ' mem:',n_m,nmm,(ddm(j),j=1,20)
      do j=1,nmm                                                    ! 17
      read(5,*,err=999) n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az       ! 標準データ
c―――――――――――――――――――――★部材断面ファイバー数セット
      ii = ii + 1                                                   ! 18
      E_model_fiber(ii).nm_type = nm_type             !  履歴特性番号
      E_model_fiber(ii).E_1     = E_1                 !  ファイバーの第一剛性 E1
      E_model_fiber(ii).E_2     = E_2                 !  ファイバーの第二剛性 E2
      E_model_fiber(ii).Q_1     = Q_1                 !  ファイバーの第一折れ点
      E_model_fiber(ii).G       = G                   !  ファイバー断面積 G
      E_model_fiber(ii).A       = A                   !  ファイバー断面積
      E_model_fiber(ii).Ay      = Ay                  !  ファイバーｙ軸せん断用断面積
      E_model_fiber(ii).Az      = Az                  !  ファイバーｚ軸せん断用断面積
      E_model_fiber(ii).ry      = ry                  !  中立軸から断面中心までのｙ方向距離
      E_model_fiber(ii).rz      = rz                  !  中立軸から断面中心までのｚ方向距離
      if(nm_type.le.10) then
      goto(801,802,803,804,805,806,807,808,809,810),nm_type         ! 19
 801  continue
c―――――――――――――――――――――★バイリニア型
      write(76,'(2i4,9e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az
      E_model_fiber(ii).E_3     = 0.                  !  ファイバーの第三剛性 E3
      E_model_fiber(ii).Q_2     = 0.                  !  ファイバーの第二折れ点
```

```
      E_model_fiber(ii).Ec_1    = E_1                 !  ファイバーの圧縮側第一剛性 E1
      E_model_fiber(ii).Ec_2    = 0.                  !  ファイバーの圧縮側第二剛性 E2
      E_model_fiber(ii).Ec_3    = 0.                  !  ファイバーの圧縮側第三剛性 E3
      E_model_fiber(ii).Qc_1    = 0.                  !  ファイバーの圧縮側第一折れ点
      E_model_fiber(ii).Qc_2    = 0.                  !  ファイバーの圧縮側第二折れ点
      goto 800
 802  continue
c――――――――――――――――――――――――★トリリニア型
      goto 800
 803  continue
c――――――――――――――――――――――――★直線コンクリート型                          ! 20
      read(5,*,err=999) AK_3,AK_4,Q_2,Q_3,Q_4                ! 直線コンクリート型
      write(76,'(2i4,18e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az,
     *     AK_3,AK_4,Q_2,Q_3,Q_4
      E_model_fiber(ii).E_3     = AK_3                !  圧縮第三勾配
      E_model_fiber(ii).Q_2     = Q_2                 !  圧縮側第一折れ点の応力
      E_model_fiber(ii).Ec_1    = Q_3                 !  圧縮強度
      E_model_fiber(ii).Ec_2    = Q_4                 !  圧縮流れ点
      E_model_fiber(ii).Ec_3    = AK_4                !  引張第二勾配
      E_model_fiber(ii).Qc_1    = 0.                  !  ダミー
      E_model_fiber(ii).Qc_2    = 0.                  !  ダミー
      goto 800
 804  continue
c――――――――――――――――――――――――★曲線コンクリート型
      goto 800
 805  continue
c――――――――――――――――――――――――★等方硬化＋移動硬化バイリニア型
      goto 800
 806  continue
c――――――――――――――――――――――――★等方硬化＋移動硬化トリリニア型
      goto 800
 807  continue
c――――――――――――――――――――――――★非対称バイリニア型
      goto 800
    808  continue
c――――――――――――――――――――――――★非対称トリリニア型
      goto 800
 809  continue
      goto 800
 810  continue
      goto 800

      elseif(nm_type.le.20) then
      goto(811,812,813,814,815,816,817,818,819,820),nm_type-10
 811  continue
      goto 800
 812  continue
      goto 800
 813  continue
      goto 800
      enddo
c――――――――――――――――――――――――★ファイバーモデル剛性出力
      call Fiber_output(E_model_fiber(ii-nmm+1),nmm)                          ! 21
      enddo
```

```
c―――――――――――――――――――――――★ファイバー履歴特性セット
      n_m_bilinear    = 0                                              ! 22
      n_m_trilinear   = 0
      n_m_concrete    = 0
      n_m_analogy     = 0
      do i=1,n_member                                                  ! 23
      ie  = Member(i).nm_element
      imm = Element(ie).n_element
      im  = Member(i).n_model_type
c―――――――――――――――――――――――★モデル１１                  ! 24
      itype_m = Model_type.no_e_model(Element(ie).element_type)
      if(itype_m.eq.11) then
      ii  = E_model11(imm).n_section_1
      nmm = E_model11(imm).nm_section_1 - 1
      nnmm=M_model11(im).nm_section_1 - 1
      do j=1,ii                                                        ! 25
      nmm = nmm  + 1
      nnmm= nnmm + 1
      if(E_model_fiber(nmm).nm_type.eq.1.or.                           ! 26
     *          E_model_fiber(nmm).nm_type.eq.5.or.
     *          E_model_fiber(nmm).nm_type.eq.7) then
      n_m_bilinear  = n_m_bilinear + 1
      M_model_fiber(nnmm).n_type = n_m_bilinear
      elseif(E_model_fiber(nmm).nm_type.eq.2.or.                       ! 27
     *          E_model_fiber(nmm).nm_type.eq.6.or.
     *          E_model_fiber(nmm).nm_type.eq.8) then
      n_m_trilinear = n_m_trilinear + 1
      M_model_fiber(nnmm).n_type = n_m_trilinear
      elseif(E_model_fiber(nmm).nm_type.eq.3.or.                       ! 28
     *          E_model_fiber(nmm).nm_type.eq.4) then
      n_m_concrete  = n_m_concrete + 1
      M_model_fiber(nnmm).n_type = n_m_concrete
      endif
      enddo
      ii  = E_model11(imm).n_section_2                                 ! 29
      nmm = E_model11(imm).nm_section_2 - 1
      nnmm= M_model11(im).nm_section_2 - 1
      do j=1,ii
      nmm  = nmm  + 1
      nnmm = nnmm + 1
      if(E_model_fiber(nmm).nm_type.eq.1.or.
     *          E_model_fiber(nmm).nm_type.eq.5.or.
     *          E_model_fiber(nmm).nm_type.eq.7) then
      n_m_bilinear = n_m_bilinear + 1
      M_model_fiber(nnmm).n_type = n_m_bilinear
      elseif(E_model_fiber(nmm).nm_type.eq.2.or.
     *          E_model_fiber(nmm).nm_type.eq.6.or.
     *          E_model_fiber(nmm).nm_type.eq.8) then
      n_m_trilinear = n_m_trilinear + 1
      M_model_fiber(nnmm).n_type = n_m_trilinear
      elseif(E_model_fiber(nmm).nm_type.eq.3.or.
     *          E_model_fiber(nmm).nm_type.eq.4) then
      n_m_concrete  = n_m_concrete + 1
                M_model_fiber(nnmm).n_type = n_m_concrete
```

```
      endif
      enddo
      endif
c―――――――――――――――――――――★モデル１２
      if(itype_m.eq.12) then                                          ! 30
      endif
c―――――――――――――――――――――★モデル１３
      if(itype_m.eq.13) then
      endif
c―――――――――――――――――――――★モデル１５
      if(itype_m.eq.15) then
      endif
c―――――――――――――――――――――★モデル２１
      if(itype_m.eq.21) then
      endif
c―――――――――――――――――――――★モデル２２
      if(itype_m.eq.22) then
      endif
c―――――――――――――――――――――★モデル３１
      if(itype_m.eq.31) then
      endif
c―――――――――――――――――――――★モデル３２
      if(itype_m.eq.32) then
      endif
c―――――――――――――――――――――★モデル３３
      if(itype_m.eq.33) then
      endif
c―――――――――――――――――――――★
      enddo
      Model_type.n_m_bilinear   = n_m_bilinear                        ! 31
      Model_type.n_m_trilinear  = n_m_trilinear
      Model_type.n_m_concrete   = n_m_concrete
      Model_type.n_m_analogy    = n_m_analogy
      write(76,'(a,i8)') ' 履歴 NO.1:',n_m_bilinear
      write(76,'(a,i8)') ' 履歴 NO.2:',n_m_trilinear
      write(76,'(a,i8)') ' 履歴 NO.3:',n_m_concrete
      write(76,'(a,i8)') ' アナロジーモデル:',n_m_analogy
      endif
      return
 999  continue
      ierr=1
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_output
C       ―――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の剛性出力(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Fiber_output(E_model_fiber,nm_div)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*)
```

```
      aa =0.                                                          ! 32
      raz=0
      ray=0
      raz2=0
      ray2=0
      aNp=0.
      aMyp=0.
      aMzp=0.
      do nn=1,nm_div                                                  ! 33
      if(E_model_fiber(nn).nm_type.gt.10) return  ! アナロジーモデルは計算しない
      y=E_model_fiber(nn).ry
      z=E_model_fiber(nn).rz
      a=E_model_fiber(nn).A
      ray      = ray  + z*a                                           ! 34
      raz      = raz  + y*a                                           ! 35
      ray2     = ray2 +  z*z*a                                        ! 36
      raz2     = raz2 +  y*y*a                                        ! 37
      aa       = aa   + a                                             ! 38
      asigy    = a*E_model_fiber(nn).Q_1                              ! 39
      aNp=aNp + asigy                                                 ! 40
      aMyp=aMyp + dabs(z*asigy)                                       ! 41
      aMzp=aMzp + dabs(y*asigy)                                       ! 42
      enddo
      write(76,'(//4a/9e18.8//)')'          E                A       ',
     *                          '          Sz               Sy      ',
     *                         '          Iz               Iy      ',
     *         '          Np               Mzp              Myp',
     * E_model_fiber(1).E_1,aa,raz,ray,raz2,ray2,aNp,aMzp,aMyP
      return
      end
```

上記サブルーチンの説明を、コード右側に付した番号に従って行う。

1. 第１回目のサブルーチンコールか第２回目かを判定し、１回目のコールの場合は、以降の処理を行う。２回目の場合は 11. に処理が移動する。
2. このファイルで定義されている断面の数を読み込む。後は、この数分、同じ処理を行う。
3. 特殊断面の番号、断面に含まれるファイバー数、及び断面の形状に関するパラメータを入力する。この断面に関するパラメータは、リファレンスマニュアルのファイル仕様を参照されたい。
   ここで、断面番号は n_m、また断面に含まれるファイバー数は nmm である。
4. 入力したファイバー断面がどの要素に結合しているかを調査する。先に入力した要素数分、次の処理を行う。
5. そのファイバー断面がモデル番号 11 である場合、部材の両端に付い

ている断面が入力した断面番号 n_m であるかどうかチェックする。ファイバー断面番号が一致すると構造体 Element(i1).section と E_model11(kk1)にその断面に含まれるファイバー数と、ファイバー連続番号の最初の番号 ii をセットする。この値を用いてファイバーの各データへの書き込みや読み込みを行う。

6. 各部材モデルについて、上記と同様の処理を行う。この処理を行うことによって、各要素が有するファイバー総数が計算される。
7. 断面に含まれるファイバー数分、２レコードのデータを読み込む。この内容についてはコード内に書かれている。また、nm_type はそのファイバーの履歴特性番号を表す。
8. ファイバー履歴特性番号によって、第２レコードのデータ個数が異なるので、特性番号にしたがって各自のデータ入力プログラムが記述されている。
9. ファイバー履歴特性番号が１の対称トリリニアのデータを読み込む。これ以降は履歴特性番号にしたがって、データ入力処理が行われる。
10. ここで、第１回目のデータ入力を終わり、構造体にファイバー総数をセットする。
11. ここでは、全部材に対するファイバー数を数える。該当する部材の要素番号 i1、モデルタイプコード連続番号 imm、モデル番号 itype_m をセットする。
12. 部材モデルが 11 の場合、次の処理を行う。最初に i 端の断面について、ファイバー数を構造体成分 M_model11(immm).n_section_1 に、ファイバー連続番号の最初の番号を M_model11(immm).nm_section_1 にセットする。続いて、j 端についても同様の処理を行う。
13. ここ以降では、各部材モデル別に上記の処理を行い、部材内にあるファイバー数をセットする。
14. ここまでが、第１回目の処理であり、これ以降のコードが第２回目のデータ入力となる。
15. このファイルで定義されている断面の数を読み込む。後は、この断面数分、同じ処理を行う。
16. 特殊断面の番号、断面に含まれるファイバー数、及び断面の形状に関するパラメータを入力する。
17. 断面内ファイバー数分、各ファイバーの履歴特性値を読む。このデータは、通常２レコードで構成されているが、履歴特性番号によっては、２レコード目がないものや、また、データ数が異なっているものもある。その内容については、コード内のコメントや、リファレンスマニ

ュアルを参照されたい。

18. 入力した各ファイバーの履歴特性値を構造体 E_model_fiber にセットする。これらの値を参照する場合は、ファイバー連続番号の最初の番号を示す M_model11(immm).nm_section_1 を用いる。
19. ファイバー履歴コード nm_type にしたがって処理を変える。次のバイリニア型では、第２レコードはなく、構造体にデータをセットするのみである。以降のコードは各ファイバー履歴モデルに対するプログラムである。
20. ここでは直線コンクリート型に対する処理を行う。以降のコードにおける他のモデルに対しては、付録を参照されたい
21. ファイバー断面の剛性を評価するサブルーチン Fiber_output()をコールする。
22. 以降の処理は、ファイバーの履歴特性を構造体にセットする。また、バイリニアモデル、トリリニアモデル、コンクリートモデル、アナロジーモデルの数を数えるために、各モデルの数を数える変数をゼロクリアする。
23. 以降の処理を全部材について行う。
24. 部材モデルが 11 の場合、以降の処理を行う。最初に、部材 i 端の断面について調査する。
25. 部材 i 端の断面におけるファイバー数分、以降の処理を行う。ここで、nmm は構造体 E_model_fiber へのアクセス番号であり、nnmm は構造体 M_model_fiber へのアクセス番号である。
26. このファイバーの履歴タイプが 1、5、7 番の場合、バイリニアとして設定し、バイリニアのファイバーを数える変数 n_m_bilinear に１を加える。その値を構造体にセットする。この構造体の値が、バイリニアモデルに対するデータのアクセス番号となる。
27. このファイバーの履歴タイプが 2、6、8 番の場合、トリリニアとして設定し、トリリニアのファイバーを数える変数 n_m_trilinear に１を加える。その値を構造体にセットする。
28. このファイバーの履歴タイプが 3、4 番の場合、コンクリートとして設定し、コンクリートのファイバーを数える変数 n_m_concrete に１を加える。その値を構造体にセットする。
29. 次に部材 j 端について、上記と同様な処理を行う。
30. これ以降では、他の部材モデルについて同様な処理を行う。
31. 上記の処理によって、３種類のファイバー履歴モデルとアナロジーモデル１種類の数を求めた。この値を構造体にセットする。これで、２

ファイバー履歴タイプ
nm_type:
1 ：バイリニア型
2 ：トリリニア型
3 ：直線コンクリート型
4 ：曲線コンクリート型
5 ：等方硬化＋移動硬化バイリニア型
6 ：等方硬化＋移動硬化トリリニア型
7 ：非対称バイリニア型
8 ：非対称トリリニア型

回目のサブルーチンの処理を終了する。
32. ファイバーの剛性を計算するサブルーチンで、和を取る変数を全てゼロクリアする。
33. 断面内のファイバー数分、以下の処理を行う。
34. ファイバーのｙ軸に関する断面一次モーメントを計算し、和を取る。
35. ファイバーのｚ軸に関する断面一次モーメントを計算し、和を取る。
36. ファイバーのｙ軸に関する断面二次モーメントを計算し、和を取る。
37. ファイバーのｚ軸に関する断面二次モーメントを計算し、和を取る。
38. ファイバーの断面積を計算し、和を取る。
39. ファイバーのせん断剛性を計算し、和を取る。
40. ファイバーの軸方向耐力を計算し、和を取る。
41. ファイバーのｙ軸に関する塑性モーメントを計算し、和を取る。
42. ファイバーのｚ軸に関する塑性モーメントを計算し、和を取る。

7.3.7 質量データファイル

本節では、質量データファイルの仕様とそのファイルの入力プログラムについて解説する。質量データファイルの内部仕様は単純で、容易に理解できる。質量データファイルの一例を以下に示し、内容を説明する。第 1 行目は、質量が存在する節点数を表し、この値の数だけ質量データを読み込むことになる。第 2 行目以降は、節点番号とその節点における質量を設定する。SPACE での動的解析は 2 段階で行われており、それに対応して２つのデータを設定し、2 つの段階で質量を変化させることができる。一つ目が第１段階で、二つ目が第２段階で使用される。ただし、ここで設定するデータは、重量（tf）であり、システム内部で質量に変換する。また、SPACE における集中質量系の動的解析では、回転慣性項に関連する質量項は無視するため、設定しないことになる。

```
12
 2       1299.9     1299.9
 3       1218.0     1218.0
 4       1198.3     1198.3
 5       1194.5     1194.5
 6       1186.3     1186.3
 7       1219.8     1219.8
 8       1402.0     1402.0
 9       1511.9     1511.9
10       1264.0     1264.0
11       1302.2     1302.2
12        208.5      208.5
13         80.2       80.2
```

次に、上記のファイルを読み込むサブプログラム Get_mass()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_mass
C       ―――――――――――――――――――――――――――――――――
C     ●   節点集中質量入力(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_mass(ierr,Point,Parameter_C)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s  / Parameter_C
      record / point_s      / Point
      dimension Point(*)
C     ―――――――――――――――――――――――――――――――――
c     Parameter_C    :structure
c     Point          :structure
C     ―――――――――――――――――――――――――――――――――
      ierr=0
      do i=1,Parameter_C.n_point                                    ! 1
      Point(i).mass_1 =0.
      Point(i).mass_2 =0.
      end do
      read(5,*,err=999) npoint                                      ! 2
      write(76,*) ' 質量節点数： ',npoint
      do i=1,npoint                                                 ! 3
      read(5,*,err=999) i1,am1,am2                                  ! 4
      Point(i1).mass_1 = am1/980.     ! 第１段階質量               ! 5
      Point(i1).mass_2 = am2/980.     ! 第２段階質量
      write(76,'(i4,2f12.4)') i1,Point(i1).mass_1,Point(i1).mass_2
      end do
      return
 999  continue
      ierr=1
      end
```

上記プログラムの説明を、コード右側に付した番号に従って行う。

１．全節点に対し、構造体成分の質量項 Point(i).mass をゼロクリアする。
２．質量が存在する節点数 npoint を読み込む。
３．節点数 npoint 分、以下の処理を繰り返す。
４．節点番号、第１段階質量（データは重量(tf)）、第２段階質量（データは重量(tf)）を読み込む。
５．重量を質量に変換し、構造体の質量項にセットする。

7.3.8 レーリー減衰データファイル

本節では、減衰データファイルの仕様とそのファイルの入力プログラムについて解説する。このファイルは、固有値解析における結果を出力したもので、ASKII ファイルである。減衰データファイルの一例を以下に示し、内容を説明する。第 1 行目はモード数を表し、この値の数だけ固有振動、周期などのデータを読み込むことになる。第 2 行目は、3 行目以下のデータに関するヘッダーであり、左よりモード番号、固有振動数、固有周期、減衰定数、x 方向の刺激係数、同じく y 方向、z 方向の刺激係数である。第 3 行目以降は、この順番にモード数分データがセットされている。このファイルには、第１段階の固有値解析の結果と第２段階の結果が続いてセットされており、上記のデータ群がその後に続いて出力されている。

```
    5
 MODE       FREQ.        PERIOD          DAMPING         BETA(x)         BETA(y)         BETA(z)
   1  0.62979813E+01  0.99765068E+00  0.10000000E+01  0.14413898E+01  0.00000000E+00  0.00000000E+00
   2  0.15798292E+02  0.39771294E+00  0.20000000E+01 -0.74104711E+00  0.00000000E+00  0.00000000E+00
   3  0.24765401E+02  0.25370820E+00  0.30458805E+01 -0.52723585E+00  0.00000000E+00  0.00000000E+00
   4  0.32639791E+02  0.19250078E+00  0.39800737E+01 -0.36410378E+00  0.00000000E+00  0.00000000E+00
   5  0.40741401E+02  0.15422114E+00  0.49471888E+01 -0.23392539E+00  0.00000000E+00  0.00000000E+00
    5
 MODE       FREQ.        PERIOD          DAMPING         BETA(x)         BETA(y)         BETA(z)
   1  0.62979813E+01  0.99765068E+00  0.29999999E-01  0.14413898E+01  0.00000000E+00  0.00000000E+00
   2  0.15798292E+02  0.39771294E+00  0.29999999E-01 -0.74104711E+00  0.00000000E+00  0.00000000E+00
   3  0.24765401E+02  0.25370820E+00  0.39078529E-01 -0.52723585E+00  0.00000000E+00  0.00000000E+00
   4  0.32639791E+02  0.19250078E+00  0.48453603E-01 -0.36410378E+00  0.00000000E+00  0.00000000E+00
   5  0.40741401E+02  0.15422114E+00  0.58630114E-01 -0.23392539E+00  0.00000000E+00  0.00000000E+00
```

上記のファイルを読み込むサブルーチン Get_damp()と Get_omega()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_damp
C       ―――――――――――――――――――――――――――――――――
C     ●    減衰データをセットする(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_damp(Newmark_P,ierr)
c――――――――――――――――――――――――――――――――――――――★計算結果のダンプ出力ファイル番号
      parameter(damp_out = 76)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s  / Newmark_P
C     ―――――――――――――――――――――――――――――――――
      itype = Newmark_P.n_damp_type
      write(76,*) ' レーリー減衰',itype
c――――――――――――――――――――――――――――――――――――――★構造物の固有振動数を入力する
      ierr=0
```

```
      call Get_omega(itype,Newmark_P.n_damp_1,Newmark_P.n_damp_2,          ! 1
     *          OMG1_1,OMG1_2,OMG2_1,OMG2_2,ierr)
      if(ierr.ne.0) return
      write(76,*) 'Newmark_P.alf1_1:',Newmark_P.alf1_1                     ! 2
      write(76,*) 'Newmark_P.alf1_2:',Newmark_P.alf1_2
      write(76,*) 'Newmark_P.alf2_1:',Newmark_P.alf2_1
      write(76,*) 'Newmark_P.alf2_2:',Newmark_P.alf2_2
      write(76,*) 'Newmark_P.n_damp_1:',Newmark_P.n_damp_1
      write(76,*) 'Newmark_P.n_damp_2:',Newmark_P.n_damp_2
      write(76,*) 'OMG1_1:',OMG1_1
      write(76,*) 'OMG1_2:',OMG1_2
      write(76,*) 'OMG2_1:',OMG2_1
      write(76,*) 'OMG2_2:',OMG2_2
c―――――――――――――――――――――――――――――――★質量比例型
      if ( itype.EQ.1 ) then                                               ! 3
      Newmark_P.alf1_1 = 2.D0*Newmark_P.alf1_1*OMG1_1
      Newmark_P.alf2_1 = 2.D0*Newmark_P.alf2_1*OMG2_1
      Newmark_P.alf1_2 = 0.0
      Newmark_P.alf2_2 = 0.0
c―――――――――――――――――――――――――――――――★剛性比例型
      elseif ( itype.EQ.2 ) then                                           ! 4
      if (OMG1_1.NE.0.0D0 ) then
      Newmark_P.alf1_2 = 2.D0*Newmark_P.alf1_1/OMG1_1
      else
      ierr =1
      return
      end if
      if (OMG2_1.NE.0.0D0 ) then                                           ! 5
      Newmark_P.alf2_2 = 2.D0*Newmark_P.alf2_1/OMG2_1
      else
      ierr =1
      return
      end if
      Newmark_P.alf1_1 = 0.0
      Newmark_P.alf2_1 = 0.0
c―――――――――――――――――――――――――――――――★レーリー減衰型
      elseif ( itype.EQ.3 ) then                                           ! 6
      a = OMG1_2*OMG1_2-OMG1_1*OMG1_1
      if (a.ne.0.0D0 ) then
      a0 = 2.D0*OMG1_1*OMG1_2*(Newmark_P.alf1_1*OMG1_2
     *                     -Newmark_P.alf1_2*OMG1_1)/a
      a1 = 2.D0*(Newmark_P.alf1_2*OMG1_2-Newmark_P.alf1_1*OMG1_1)/a
      Newmark_P.alf1_1 = a0
      Newmark_P.alf1_2 = a1
      else
      ierr = 1
      return
      end if
      a = OMG2_2*OMG2_2-OMG2_1*OMG2_1
      if (a.ne.0.0D0 ) then
      a0 = 2.D0*OMG2_1*OMG2_2*(Newmark_P.alf2_1*OMG2_2
     *                     -Newmark_P.alf2_2*OMG2_1)/a
      a1 = 2.D0*(Newmark_P.alf2_2*OMG2_2-Newmark_P.alf2_1*OMG2_1)/a
      Newmark_P.alf2_1 = a0
```

```
      Newmark_P.alf2_2 = a1
      else
      ierr = 1
      return
      end if
      endif
      write(76,*) 'Newmark_P.alf1_1:',Newmark_P.alf1_1
      write(76,*) 'Newmark_P.alf1_2:',Newmark_P.alf1_2
      write(76,*) 'Newmark_P.alf2_1:',Newmark_P.alf2_1
      write(76,*) 'Newmark_P.alf2_2:',Newmark_P.alf2_2
c―――――――――――――――――――――――――――――――――★一般化レーリー減衰型
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_omega
C     ―――――――――――――――――――――――――――――――――――
C     ●   構造物の固有振動数を入力する(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Get_omega(itype,n_damp_1,n_damp_2,
     *          OMG1_1,OMG1_2,OMG2_1,OMG2_2,ierr)
      implicit real*8(A-H,O-Z)
      character dummy*1
      ierr=0
      OMG1_1=0.                                                    ! 7
      OMG1_2=0.
      OMG2_1=0.
      OMG2_2=0.
      if(itype .eq. 3 ) then                                       ! 8
      read(5,'(I6)',err=999) NNMOD                                 ! 9
      read(5,'(A)',err=999) DUMMY                                  ! 10
      do JMODE=1,NNMOD                                             ! 11
      read(5,*,err=999) imode,OMG,dm1,dm2,dm3,dm4,dm5              ! 12
      if(n_damp_1 .eq. imode ) OMG1_1 = OMG                        ! 13
      if(n_damp_2 .eq. imode ) OMG1_2 = OMG                        ! 14
      end do
      read(5,'(I6)',err=999) NNMOD
      read(5,'(A)',err=999) DUMMY
      do JMODE=1,NNMOD
      read(5,*,err=999) imode,OMG,dm1,dm2,dm3,dm4,dm5
      if(n_damp_1 .eq. imode ) OMG2_1 = OMG                        ! 15
      if(n_damp_2 .eq. imode ) OMG2_2 = OMG                        ! 16
      end do
      else

      read(5,'(I6)',err=999) NNMOD                                 ! 17
      read(5,'(A)',err=999) DUMMY
      do JMODE=1,NNMOD
      read(5,*,err=999) imode,OMG,dm1,dm2,dm3,dm4,dm5
      if(n_damp_1 .eq. imode ) OMG1_1 = OMG
      end do
      read(5,'(I6)',err=999) NNMOD
      read(5,'(A)',err=999) DUMMY
      do JMODE=1,NNMOD
      read(5,*,err=999) imode,OMG,dm1,dm2,dm3,dm4,dm5
```

```
      if(n_damp_1 .eq. imode ) OMG2_1 = OMG
      end do
      endif
      return
  537 FORMAT(I5,6F8.4)
  999 continue
      ierr=1
      end
```

上記プログラムの説明を、コード右側に付した番号に従って行う。

１．減衰データファイルを読み込むサブルーチン Get_omega()をコールし、レーリー減衰に関連するデータを設定する。
２．セットされたデータをダンプファイルに出力する。
３．減衰の型別に減衰パラメータを構造体成分 Newmark_P.alf に設定する。これ以降は質量比例型（itype=1）となっている。
４．減衰型が剛性比例型（itype=2）であり、第１段階の振動数がゼロでないとき、値を構造体にセットする。ゼロの場合は、計算不可となるためエラーとして処理を中止する。
５．同じく、剛性比例型（itype=2）であり、第２段階の振動数がゼロでないとき、値を構造体にセットする。
６．減衰型がレーリー型（itype=3）であり、第１段階と第２段階に対するパラメータを計算し、構造体にセットする。選択したモード番号の振動数が同じとなると分母がゼロとなるため、計算不可となり、システムではエラーとして処理を中止する。

OMEG1_1: 第１段階で、第１選択モード番号に対応する振動数
OMEG1_2: 第１段階で、第２選択モード番号に対応する振動数
OMEG2_1: 第２段階で、第１選択モード番号に対応する振動数
OMEG2_2: 第２段階で、第２選択モード番号に対応する振動数
計算前は、
Newmark_P.alf1_1：第１段階で、第１選択モード番号に対応する減衰定数
Newmark_P.alf1_2：第１段階で、第２選択モード番号に対応する減衰定数
Newmark_P.alf2_1：第２段階で、第１選択モード番号に対応する減衰定数
Newmark_P.alf2_2：第２段階で、第２選択モード番号に対応する減衰定数
であり、計算後は、第１段階と第２段階における減衰に関するパラメータ a0 と a1 である（理論マニュアルの 3.8 節を参照）。

７．第１選択、第２選択モードに対する振動数をゼロクリアする。
８．減衰型がレーリー減衰の場合は、以降の処理を行う。
９．モード数を読み込む。
10．　ヘッドの部分を読み捨てる。
11．モード数分、以下の処理を行う。

12．１行分データを読み、モード番号と振動数をセットする。
13. 第１選択モード n_damp_1 が入力したモード番号に一致した場合は、振動数を第一段階振動数にセットする。
14. 第２選択モード n_damp_2 が入力したモード番号に一致した場合は、振動数を第１段階の振動数にセットする。
15. 第１選択モード n_damp_1 が入力したモード番号に一致した場合は、振動数を第２段階振動数にセットする。
16. 第２選択モード n_damp_2 が入力したモード番号に一致した場合は、振動数を第２段階振動数セットする。
17．上記と同様の処理を減衰型が質量比例型と剛性比例型に対し行う。