第７章　ファイルの仕様と管理

7.1 はじめに

本章では、動的解析システム内に組み込まれているファイル管理技法とファイルの仕様について解説する。ここでは、動的ソルバーが必要とする入力データファイルと、同じく動的ソルバーが出力する全ファイルについて、その仕様とプログラムコードを示し、詳細に解説する。

7.2 制御ファイル

7.2.1 spacesys.xxx ファイル

SPACE システムから動的解析システムを立ち上げるとき、動的解析システムに情報の伝達を行う必要がある。ここでは、その手法について説明する。SPACE から動的解析システムへの情報は、WINDOWS ファイルシステムの中にある TEMP フォルダー内に伝達情報を出力し、それらを解析システムが入力することで伝達される。ファイル名は SPACE システムが自動的に決定し、制御ファイル spacesys.xxx としている。ファイルの内容は２行で構成され、１行目は解析種別を、２行目はコントロールファイルの絶対パス名である。動的解析システムは、立ち上がると直ぐに、このファイルを読み込み、解析種別とコントロールファイルの名前を設定する。この制御ファイルが存在しない場合は、動的ソルバーは直ちに処理を終了し、制御は SPACE に戻ることになる。それでは、制御ファイルの一例を以下に示そう。

```
10
D:¥space-frame¥DISK2¥動的解析編¥アーチの動的安定¥両端ピン支持アーチ¥Arch_弾性分岐動座屈.ctl
```

ファイル第１行目の解析種別番号は、静的ソルバーもしくは動的ソルバーにおける解析手法番号と対応し、これによって解析方法が選択される。解析種別番号の仕様を以下に示す。

静的解析

1: 線形解析
2: 幾何学的非線形解析
3: 弾塑性解析
4: 幾何学的非線形性を考慮した弾塑性解析
5: 線形座屈解析

動的解析

6: 線形値固有解析（固有振動数、固有振動モード）
7: 線形解析
8: 幾何学的非線形解析
9: 弾塑性解析
10: 幾何学的非線形性を考慮した弾塑性解析

7.2.2 コントロールファイル

SPACE システムで使用される全ファイルは、ここで解説するコントロールファイルに記述されている。そのため、静的解析・動的解析など全てのサブシステムは、使用するファイルをこのコントロールファイルの内容を参照して該当するファイル名を決定する。このようにコントロールファイルは重要なファイルであり、そのため SPACE ではこのファイルを自動的に管理し、手動でファイルの内容を変更することのないように設計されている。

コントロールファイルの内容は、キィワードを含み、以下のようである。そのファイルの仕様は、１行でひとつのファイルを管理し、その１行は５つのデータで構成され、最後の２つの領域を除いて、記号||で４つの領域に分割されている。第１番目は、１桁目で、読み込み許可か不許可を表す。記号＊がある場合は不許可を意味し、ブランクの場合は許可を意味する。また、記号 T はタイトル行を表す。タイトル行は第１行目にのみ現れる。第２番目のデータは、５桁のキィワードであり、キィワードは、各サブシステムがこのキィワードを使用して該当するファイル名を特定する。第３番目は、実際のファイル名を記述する場所であり、コントロールファイルと同一のフォルダーに存在する場合は、単にファイル名を記述すれば良いが、他のフォルダーに存在する場合は、コントロールファイルが存在するフォルダーからの相対パス名か、もしくは絶対パス名を記述する。最後の領域は２つの情報を表し、第１桁目は、このファイルへの書き込みの許可か不許可を表す。許可はブランクであり、不許可は記号＊で示す。第２桁目以降は、このファイルに書き込みを行った最終日時がシステムによって自動的に記入される。

ファイル名はキィワードによって検索される。

コントロールファイルの例を以下に示す。第１行目は第２行目以下の有効レコード数を表し、そのレコード数以降は SPACE システムでは無視される。第２行目以降は、先に示したとおり、ファイルの管理情報が記述されている。

```
      60
||T||      ||Fiber element66   second rigidity0.01          ||
|| ||struct||struct.dat                                     || ------------------
|| ||mass  ||mass.dat                                       || ------------------
||*||sload1||sload1.dat                                     ||*------------------
||*||sload2||sload2.dat                                     ||*------------------
```

以下に、キィワードとファイル名を示す。ここで（　)内はそのキィワードが表すファイルの内容を、＃は現在（Ver.2.10）では使用されていないファイルを示す。例に示すファイル名は、SPACE システムが自動的につける既定値であり、コントロールファイルを作成した当初は、読み込み・書き込み共に全ファイル不許可となっている。

```
||T||      ||このコントロールファイルのタイトルを表す          ||
||*||struct||struct.dat（構造データファイル ）                ||*------------------
||*||mass  ||mass.dat  （質量データファイル）                  ||*------------------
||*||sload1||sload1.dat（静的荷重ファイルその１ ）            ||*------------------
||*||sload2||sload2.dat（静的荷重ファイルその２ ）            ||*------------------
||*||dload1||dload1.dat（動的荷重ファイルその１ ）            ||*------------------
||*||dload2||dload2.dat（動的荷重ファイルその２ ）            ||*------------------
||*||dload3||dload3.dat（動的荷重ファイルその３ ）            ||*------------------
||*||scontl||scontl.dat（静的解析コントロールファイル）       ||*------------------
||*||seigen||seigen.dat（静的座屈解析コントロールファイル）   ||*------------------
||*||soutcl||soutcl.dat（静的解析結果出力コントロールファイル）||*------------------
||*||dcontl||dcontl.dat（動的解析コントロールファイルその１）  ||*------------------
||*||dcalcl||dcalcl.dat（動的解析コントロールファイルその２）  ||*------------------
||*||damp  ||damp.dat  （動的解析減衰ファイル）               ||*------------------
||*||deigen||deigen.dat（動的固有問題解析コントロールファイル）||*------------------
||*||doutcl||doutcl.dat（動的解析結果出力コントロールファイル）||*------------------
||*||inidis||inidis.dat（初期変位ファイル）                   ||*------------------
||*||perscl||perscl.dat（透視図用コントロールファイル）       ||*------------------
||*||windcl||outfle.dat（ウインドウコントロールファイル）#    ||*------------------
||*||info  ||info.dat  （情報用コントロールファイル）         ||*------------------
||*||nofc_s||nofc_s.dat（静的節点荷重、反力ファイル）         ||*------------------
||*||disp_s||unbf_s.dat（静的節点不釣合力ファイル）           ||*------------------
||*||disp_l||disp_s.dat（静的節点変位ファイル）               ||*------------------
||*||dibm_s||dibm_s.dat  #                                   ||*------------------
||*||stsp_s||stsp_s.dat  （静的解析結果部材材端応力ファイル） ||*------------------
||*||stbm_s||stbm_s.dat  （静的解析結果断面内応力ファイル）   ||*------------------
||*||mode_s||mode_s.dat  （座屈モードファイル）               ||*------------------
||*||engy_s||engy_s.dat  #                                   ||*------------------
||*||shar_s||shar_s.dat  #                                   ||*------------------
||*||linr_s||linr_s.dat  #                                   ||*------------------
||*||pd_s  ||pd_s.dat  （荷重・変位データファイル）           ||*------------------
||*||sry_s ||sry_s.dat   #                                   ||*------------------
||*||maxd_s||maxd_s.dat  #                                   ||*------------------
||*||axis_s||axis_s.dat  #                                   ||*------------------
||*||moment||moment.dat  #                                   ||*------------------
```

```
||*||nofc_d||nofc_d.dat  (動的節点荷重、反力ファイル)          ||*------------------
||*||hing_d||unbf_d.dat  (動的節点不釣合力ファイル)            ||*------------------
||*||disp_d||disp_d.dat  (動的節点変位ファイル)                ||*------------------
||*||dibm_d||dibm_d.dat  #                                    ||*------------------
||*||stsp_d||stsp_d.dat  (動的解析結果部材材端応力ファイル)    ||*------------------
||*||stbm_d||stbm_d.dat  (動的解析結果断面内応力ファイル)      ||*------------------
||*||mode_d||mode_d.dat  (振動モードファイル)                  ||*------------------
||*||omeg_d||omeg_d.dat  (固有振動数、刺激係数などファイル)    ||*------------------
||*||shar_d||shar_d.dat  (動的層せん断力ファイル) #            ||*------------------
||*||acc_d ||acc_d.dat   (節点応答相対加速度ファイル)          ||*------------------
||*||vel_d ||vel_d.dat   (節点応答絶対速度ファイル)            ||*------------------
||*||max_d ||max_d.dat   (最大変位、速度、加速度ファイル)      ||*------------------
||*||absa_d||absa_d.dat  (節点応答絶対加速度ファイル)          ||*------------------
||*||engy_d||engy_d.dat  #                                    ||*------------------
||*||ave_d ||ave_d.dat   #                                    ||*------------------
||*||beta_d||max_sp.dat  (最大応力ファイル)                    ||*------------------
||*||inistr||inistr.dat  (初期応力データファイル)              ||*------------------
||*||romodl||romodl.dat  (RO データファイル)                   ||*------------------
||*||fiberm||fiberm.dat  (ファイバーデータファイル)            ||*------------------
||*||sdout1||sdout1.dat   #                                   ||*------------------
||*||sdout2||sdout2.dat   #                                   ||*------------------
||*||xeathf||xeathf.dat  (x方向地震波ファイル)                ||*------------------
||*||yeathf||yeathf.dat  (y方向地震波ファイル)                ||*------------------
||*||zeathf||zeathf.dat  (UD方向地震波ファイル)               ||*------------------
||*||sectio||sectio.dat  (断面データファイル)                  ||*------------------
```

7.2.3 動的制御パラメータその１ファイル

動的解析に必要な制御情報は、SPACE 中のダイアログで設定し、ファイルとして保存される。この保存された動的制御ファイルその１は、キィワード dcontl で示されるファイルであり、ファイルの仕様を以下に示す。このファイルのデータは、下図のダイアログ中で設定したデータを整理して保存したものである。最初に、ファイルの一例を示す。このファイルの仕様は、SPACE の制御用データファイルの標準仕様となっている。

```
    32         63
     1         2         1         0         0         0         0         0         0         0
     0         0         0         0         0         0         0         0         0         0
     0         0         0         0         0         0         0         0         0         0
     0         0
 0.100000E+01  0.100000E+00  0.100000E-01  0.100000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.550000E+01  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
```

```
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00
```

上記の第１行の 32 と 63 は、このファイルで使用されている整数と実数のパラメータ数を表す。つまり、32 は、第 2 行目以降の整数の数を、63 は、第 6 行目以降の実数の数を意味する。

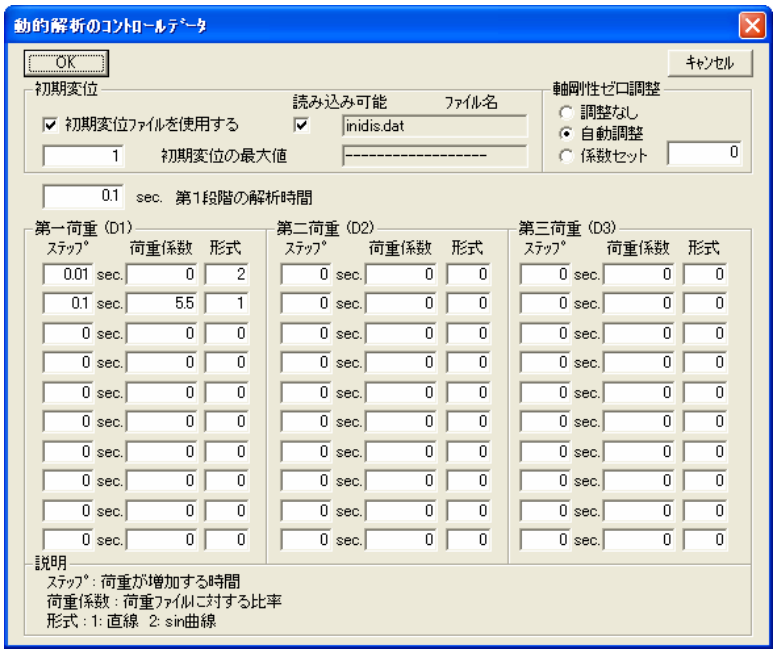

図 7-1 動的解析のコントロールデータダイアログ

最初に、整数から説明する。図 7-1 のダイアログ中で、初期不整の読み込み許可とファイル名はコントロールファイルより情報を得ており、従って、このファイルで管理しているデータは初期不整の最大値以降である。整数データの第 1 番目は、初期不整を使用するか否かを 1 と 0 で表している。最後の 32 番目の整数データは、軸剛性ゼロ調整に関するデータであり、0 は調整なし、1 は自動調整、2 は係数セットを意味する。整数データの第 2 から第 31 番目までの 30 個のデータは、荷重の形式を表す。最初の 10 個のデータが第 1 荷重に対応し、続いて 10 個づつ第 2 荷重、第 3 荷重に対応する。

実数データは、第 1 番目は初期変位の大きさを、第 2 番目は第 1 段階の解析時間を表す。第 3 から第 62 番目までの 60 個のデータは荷重に関するデータで、ステップ（秒）と荷重係数の順に第 1 荷重が 20 個、続いて、第 2 荷重が 20 個、第 3 荷重が 20 個となっている。最後に、第 63 番目は、軸剛性に関する係数の値である。

ファイルの入力とデータの設定は、サブルーチン dyctl1()で行っている。このサブルーチンを以下に示す。この中で datset()というサブルーチンをコールしているが、ここでは、キィワードに従って、該当するファイルをオープンし、標準的な仕様となっているデータを読み込み、整数配列 id と実数配列 fd にデータをセットして戻している。オープンするファイル名は引数としてこのサブルーチンに渡す。この中にデータ数がいくつセットされているかは、各配列の最初の番地に収められている。

このプログラムは、非常に単純なので理解し易い。ここで、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 dyctl1()と datset()を示す。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /dyctl1
C     ------------------------------------------------------------------
C     ●    動的制御ファイル No.1 を入力(ok)
C     ------------------------------------------------------------------
      subroutine dyctl1(ierr,nindis,gindis,f1sec,fs,fl,ifp,ist,
     *                  JIKUZERO,G_JIKUZERO_ALPH)
      implicit real*8(A-H,O-Z)
      character fcode*6
      dimension id(100)
      real*4 fd(100)
      dimension fs(10,3),fl(10,3),ifp(10,3),ist(3)
      ierr=0
      ihan=0
      numb=1
      fcode='dcontl'
      call datset(ihan,fcode,id ,fd )
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      JIKUZERO=id(33)
      G_JIKUZERO_ALPH=fd(64)
      nindis=id(2)
      gindis=fd(2)
      f1sec=fd(3)
      do i=1,10
      fs(i,1)=fd(i+3)
      fs(i,2)=fd(i+23)
      fs(i,3)=fd(i+43)
      fl(i,1)=fd(i+13)
      fl(i,2)=fd(i+33)
      fl(i,3)=fd(i+53)
      ifp(i,1)=id(i+2)
      ifp(i,2)=id(i+12)
      ifp(i,3)=id(i+22)
      end do
      do i=1,3
      ist(i)=0
      if(f1sec.ne.0.) then
      do j=1,10
      if(ifp(j,i) .ne. 0 ) ist(i)=1
      end do
      endif
      end do
      return
      end
C     ------------------------------------------------------------------
```

```
C     ●   SUBROUTINE /datset
C       ―――――――――――――――――――――――――――――――――――――――
C     ●   ファイルからデータを入力する(ok)
C       ―――――――――――――――――――――――――――――――――――――――
      SUBROUTINE datset(ihan ,fcode,idd ,fdd)
       common /comctl/ctl,ctlf
       common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100),fcode*6
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character CTL(100)*6,ctlf(100)*10
      integer*4 idd(100)
       real*4    fdd(100)
      character fnxfl(100)*10
      character clockt*9,timexx*20
      if(ihan.ne.0) goto 2000                              ! 入力か出力を判定する。
C                                        標準仕様のデータを入力する。
      do ilen=1,100
       if(fcode .eq.CTL(ilen) ) goto 102                   ! キィーワードでファイル名を特定する。
      end do
      ihan=-1
       write(76,'(//a,a,a)') ' エラー：ファイル[ ',fcode, ' ]をオープンできません。'
       return
  102 continue
      if(jdfile(ilen).eq.-1.or.fnfile(ilen).eq.' ') then  ! ファイル名が設定されているかチェック
       ihan =-2
       write(76,'(//a,a,a)') ' エラー：ファイル[ ',fcode, ' ]が存在しません。'
       return
      endif
      if(jdfile(ilen).eq.0) then
       ihan=-3
       write(76,'(//a,a,a)') ' エラー：ファイル[ ',fcode,  ! ファイルが入力許可されているかチェック
     *                       ' ]が入力可能状態になっていません。'
       return
       endif
      open(7,FILE=FNFILE(ILEN),STATUS='OLD',ERR=814)     ! ファイルをオープンする。
       goto 812
  814 continue
      ihan = -1
       write(76,'(//a,a,a)') ' エラー：ファイル[ ',fcode, ' ]をオープンできません。'
       return
  812 continue
      read(7,'(2i10)') i1,i2                               ! 整数、実数データ数を読み込む
       IDD(1)=I1
       FDD(1)=I2+0.5
       if(I1.EQ.0) goto 902
       II1=I1/10
      if(II1.NE.0)then
       do i=1,II1
       read(7,'(10I8)')(IDD((i-1)*10+j+1),j=1,10)      ! 整数データを読み込む
      end do
      endif
       III1=I1-II1*10
       IF(III1.NE.0)then
       read(7,'(10I8)')(IDD(II1*10+j+1),j=1,III1)
```

```
      endif
  902 continue
      if(I2.EQ.0) goto 903
      II1=I2/5
      if(II1.NE.0)then
      do i=1,II1
      read(7,'(5E16.6)')(FDD((i-1)*5+j+1),j=1,5)          ! 実数データを読み込む
      end do
      endif
      III1=I2-II1*5
      if(III1.NE.0)then
      read(7,'(5E16.6)')(FDD(II1*5+j+1),j=1,III1)
      endif
  903 continue
      close(7)
      return
 2000 continue
      if(kDFILE(Ihan).ne.1) then
      ihan=-1
      return
      endif
      call dstamp(timex(Ihan))
      close(7)
      return
      end
```

7.2.4 動的制御パラメータその２ファイル

本節では、制御パラメータに関するファイルその２について解説する。このファイルは、SPACE 中のダイアログ(図 7-2)で入力したパラメータを保存する。このファイルに関するキィワードは、dcalcl であり、その例を以下に示す。

```
      7         11
      0         0         0         2        10         0         1
 0.120000E+02    0.100000E-02    0.100000E+03    0.100000E+03    0.100000E+05
 0.100000E+01    0.100000E-04    0.100000E+01    0.200000E+00    0.500000E+00
 0.500000E+02
```

第１行の 7 と 11 は、このファイルで使用されている整数と実数のパラメータ数を表す。つまり、７は、第２行目以降の整数の数を表す。整数データ、実数データとダイアログとの対応は、以下のようである。

```
整数
１－３．各方向の地震波を解析で使用するか否かを表す　０：不使用　１：使用
４．ニューマークβ法でβの値を決めるパラメータ
     ０：β=0　１：β=1/4　２：β=1/6　３：β=0.3025
５．反復法における最大反復数
```

６．部材分布質量の考慮　０：考慮せず　１：考慮
７．解析種別　０：３次元解析　１：２次元解析（ｘ－ｚ平面）　２：２次元解析（ｙ－ｚ平面）

実数
１．第２段階における解析時間Ｔ（秒）
２．解析全体で使用する増分時間Δｔ（秒）
３-５．各方向で使用する加速度の最大値（gal）
６．反復計算において、次の予測加速度を計算する場合のパラメータ(通常は、１を設定)
７．反復計算における収束用閾値
８-11.Maxwell モデルを使用する場合で、フィルターを使用する場合のパラメータ

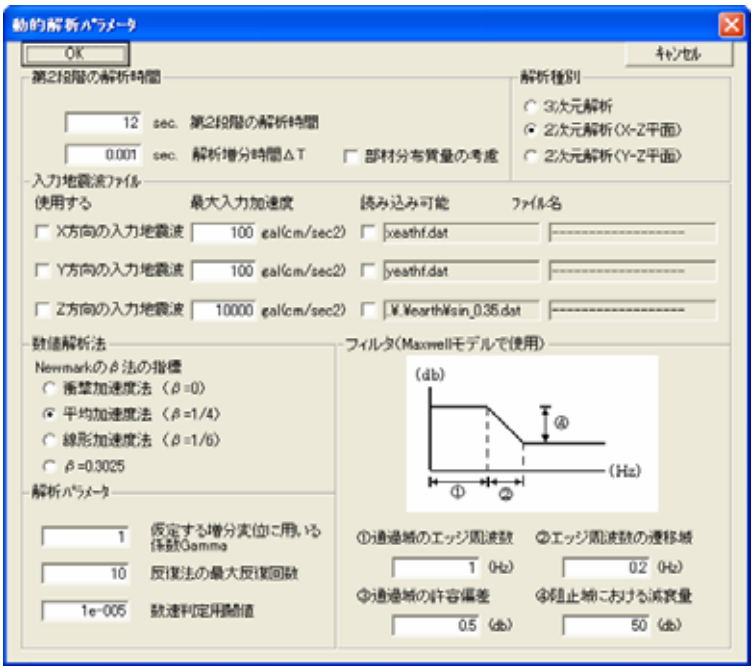

図 7-2 動的解析パラメータダイアログ

次に、上記仕様のファイルを読み込み、動的ソルバー用にデータをセットするプログラム dyctl2()を示す。上記の仕様と付き合わせて読まれると良い。容易に理解できよう。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /dyctl2
C       ----------------------------------------------------------
C     ●   動的制御ファイル No.2 を入力(ok)
C     ------------------------------------------------------------
      subroutine dyctl2(ierr,NSTEP,t,DELT,IGRA,IBETA,BETA,GAMMA,
     *         XGAL,NTIME,DLAMST,load_memb_mass,dt_M_filter,IT_ANALYS)
      IMPLICIT REAL*8(A-H,O-Z)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character fcode*6
      dimension id(100)
      real*4 fd(100)
      dimension IGRA(3),XGAL(3),DBETA(4),dt_M_filter(4)
      data DBETA/0.,4.,6.,8./
      njishin=56
      ierr=0
      ihan=0
      numb=1
      fcode='dcalcl'
      call DATSET(ihan,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif
```

```
      DELT = fd(3)
      IF(DELT.LE.0.) delt=0.005
      t=fd(2)
      nstep=t/delt
      do i=1,3
      igra(i)=id(i+1)
c                                                  地震波のチェック
      if(jdfile( njishin+i).ne.1) igra(i)=0
      xgal(i)=fd(i+3)
     enddo
     ibeta=id(5)
      gamma=fd(7)
      ntime=id(6)
      load_memb_mass=id(7)
     IT_ANALYS=id(8)
     DLAMST=fd(8)
      IF(IBETA.EQ.1) THEN
      beta=0.D0
      ELSE
     BETA=1.D0/DBETA(IBETA)
      ENDIF
      if(IBETA.eq.4) then
      BETA = 0.3025d0
      endif
c                                          Maxwell 用フィルターデータセット
     dt_M_filter(1)=fd(9)
     dt_M_filter(2)=fd(10)
     dt_M_filter(3)=fd(11)
     dt_M_filter(4)=fd(12)
      return
      end
```

7.2.5 減衰パラメータファイル

本節では、減衰パラメータを保存するファイルについて解説する。このファイルは、SPACE 中のダイアログ(図 7-3)で入力したパラメータを保存する。このファイルに関するキィワードは、damp であり、そのファイルの一例を以下に示す。

```
      5         10
      0        3        2        1        2
 0.100000E+01    0.100000E+01    0.000000E+00    0.000000E+00    0.000000E+00
 0.300000E-01    0.300000E-01    0.000000E+00    0.000000E+00    0.000000E+00
```

第 1 行の 5 と 10 は、このファイルで使用されている整数と実数のパラメータ数を表す。つまり、5 は、第 2 行目以降の整数の数を、10 は第 3 行目以降の実数の数を表す。整数データ、実数データとダイアログとの対応は以下のようである。

整数
1．モードファイルを使用するか否か（現在使用不可）
2．減衰のタイプ選択　０：質量比例型　１：剛性比例型　２：レーリー型　３：一般型レーリー減衰（現在使用不可）
3．一般型レーリー減衰で減衰定数の数（現在使用不可）
4．第１減衰定数に関するモード番号
5．第２減衰定数に関するモード番号

実数
1．第１段階における第１減衰定数
2．第１段階における第２減衰定数
3．第１段階における第３減衰定数（現在使用不可）
4．第１段階における第４減衰定数（現在使用不可）
5．第１段階における第５減衰定数（現在使用不可）
6．第２段階における第１減衰定数
7．第２段階における第２減衰定数
8．第２段階における第３減衰定数（現在使用不可）
9．第２段階における第４減衰定数（現在使用不可）
10．第２段階における第５減衰定数（現在使用不可）

図 7-3 減衰定数ダイアログ

次に、上記仕様のファイルを読み込み、動的ソルバー用にデータをセットするプログラム damctl()を示す。

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /damctl
C       ------------------------------------------------------------------
C     ●   減衰制御ファイルを入力する(ok)
C     ------------------------------------------------------------------
      subroutine damctl(ierr,NMREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH)
      IMPLICIT REAL*8(A-H,O-Z)
       common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
       character fcode*6
       dimension id(100)
       real*4 fd(100)
      dimension HH(100),QHH(100)
      nmdxx=42
       ierr=0
       ihan=0
       numb=1
      fcode='damp  '
       call datset(ihan ,fcode,id ,fd )
       if(ihan.ne.0) then
       ierr=1
      return
```

```
      endif
c      NMREAD=ID(2)
      nmread=1
      if(nmread.eq.1) then
      if(jdfile( nmdxx).ne.1) nmread=0
      if(jdfile( nmdxx+1).ne.1) nmread=0
      endif
      ITYDP=ID(3)
      NDMP=ID(5)
      NDMP2=ID(6)
      IF(NDMP.LE.0) NDMP=1
      IF(NDMP2.LE.0) NDMP2=2
      IF(ITYDP.EQ.1.OR.ITYDP.EQ.2)THEN
      NHH=1
      HH(1)=FD(2)
      QHH(1)=FD(7)
      hh(2)=0.
      qhh(2)=0.
      ELSEIF(ITYDP.EQ.3) THEN
      NHH=2
      HH(1)=FD(2)
      QHH(1)=FD(7)
      HH(2)=FD(3)
      QHH(2)=FD(8)
      ELSEIF(ITYDP.EQ.4) THEN
      NHH=ID(4)
      IF(NHH.LE.0) NHH=2
      DO I=1,NHH
      HH(i)=FD(I+1)
      QHH(i)=FD(I+6)
      enddo
      ENDIF
      RETURN
      END
```

7.2.6 固有値解析用パラメータファイル

本節では、固有値問題を解析するためのパラメータを保存するファイルに関して解説する。このファイルは、SPACE 中で設定する図 7-4 に示すダイアログで入力したパラメータを保存する。このファイルに関するキィワードは、deigen であり、その例を以下に示す。

```
       3          1
     6      100         0
 0.100000E-03
```

第１行の３と１はこのファイルで使用されている整数と実数のパラメータ数を表す。

整数
１．固有値解析で求めるモード数
２．サブスペース法における最大反復回数
３．固有値問題の解析手法の選択
　０：サブスペース法
　１：ヤコビ法
実数
１．サブスペース法における収束判定用閾値

次に、上記仕様のファイルを読み込み、ソルバー用にデータをセットするプログラム deignc()を示す。

```
C
C     ●    SUBROUTINE /deignc
C
C     ●    固有値計算定義ファイルを入力する(ok)
C
      subroutine deignc(ierr,NMODE,ITER,EPSJ,NMETHOD)
      IMPLICIT REAL*8(A-H,O-Z)
       character fcode*6
      dimension id(100)
      real*4 fd(100)
      ierr=0
      ihan=0
      numb=1
      fcode='deigen'
      call datset(ihan,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      NMODE=ID(2)
      ITER=ID(3)
      NMETHOD=ID(4)
      EPSJ=FD(2)
      RETURN
      END
```

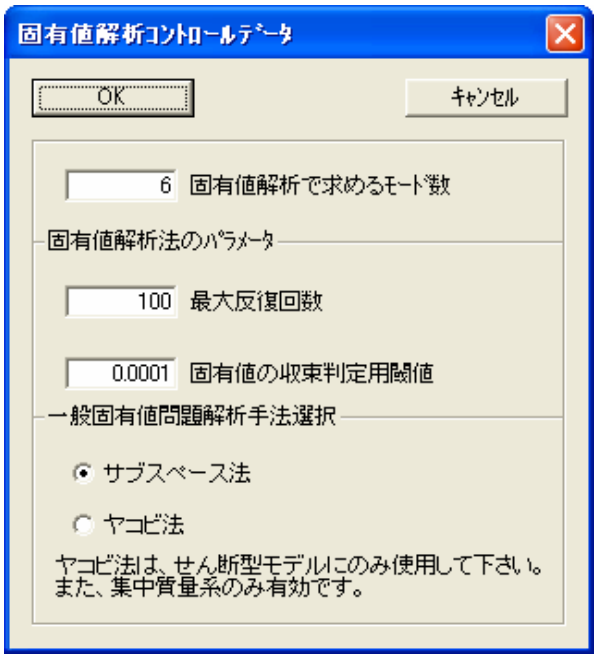

図 7-4 固有値解析コントロールデータダイアログ

7.2.7 制御データの構造体へのセット

これまでに、各種の制御用パラメータを動的ソルバーに取得するプログラムとその仕様について述べた。SPACE では、これら重要なパラメータを構造体に格納し、解析途中で書き換えられることを防ぐ。

本節では、制御ファイルの取得から各種の動的解析用パラメータの設定まで、主サブルーチン submain_dynamic_a()の中から該当する部分を抜き出し、詳細にその方法を説明する。ただし、記載するコードは、重要な部分のみであり、全てのコードは付録を参照されたい。

```
c------------------------------------------------------------★システムからのコントロール情報を取得(ok)
      call sysnam(FNX_file,N_analysis)
      if(N_analysis.ne.i_calnum) N_analysis=i_calnum  ! 解析番号を変更
c------------------------------------------------------------★コントロールデータの内容を取得(ok)
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
c------------------------------------------------------------★動的解析ダイアログその１のデータを取得(ok)
      call dyctl1(ierr,NINDIT,GINDIS,F1SEC,fs_st,fl_st,ifp_st,IST,
     *            JIKUZERO,G_JIKUZERO_ALPH)
c------------------------------------------------------------★動的解析ダイアログその２のデータを取得(ok)
      call dyctl2(ierr,NSTEP,F2SEC,DELT,IGRA,IBETA,BETA,GUMMA,XGAL,
     *            NNTIME,EPSDSP,load_memb_mass,dt_M_filter,IT_ANALYS)
c------------------------------------------------------------★解析結果の出力パラメータを取得(ok)
      call doutcl(ierr,IWSTP,SOUTSC,DMAXCK,No_section)
c------------------------------------------------------------★減衰ダイアログのデータを取得(ok)
      call damctl(ierr,NREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH)
c------------------------------------------------------------★制御情報の取得に失敗した場合はここで戻る
      if(ierr_dat .ne. 0 ) return
c-----------------------------------------------------------------------
c
c    ★         制御情報を構造体にセット
c
c-----------------------------------------------------------------------
      call Set_control(Control,N_analysis,NINDIT,GINDIS,
     *                 IWSTP,SOUTSC,DMAXCK,in_disp,in_stres,
     *                 IT_ANALYS,JIKUZERO,G_JIKUZERO_ALPH)       !ok (in_disp,in_stres  未定義)
       call Set_model_type(Model_type)
       call Set_newmark(Newmark_P,F1SEC,F2SEC,HH(1),HH(2),
     *                 QHH(1),QHH(2),DELT,BETA,EPSDSP,NNTIME,GUMMA,
     *                 ITYDP,NDMP,NDMP2)
       call Set_dynamic_load(Dynamic_load,IST,IGRA,
     *                       XGAL,load_memb_mass)
c-----------------------------------------------------------------------
```

入力した動的解析を制御するパラメータを構造体にセットする４つのサブプログラムを以下に示す。プログラムの内容は単純であり、構造体の内容や変数の意味は、コメントや出力文を見れば容易に理解できる。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Set_control
C       ----------------------------------------------------------
C     ●   Control 構造体の値セット(ok)
C     ------------------------------------------------------------
      subroutine Set_control(Control,IANAL,NINDIT,GINDIS,
     *           IWSTP,SOUTSC,DMAXCK,in_disp,in_stres,IT_ANALYS,
     *           JIKUZERO,G_JIKUZERO_ALPH)
```

```
C     ------------------------------------------------------------------
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record /control_s    / Control
      dimension igra(3),gra(3),ist(3)
C     ------------------------------------------------------------------
c   コントロールデータ
c      structure  / control_s/
c      integer    type_analysis        !  解析種別
c       integer    analysis_3D         !  解析型 0:3D 1:2D(x-z) 2:2D(y-z)
c      integer    init_disp            !  初期変位ありか
c      integer    init_stress          !  初期応力ありか
c      integer    init_inperfection    !  形状初期不整ありか
c      real*8     amp_inperfection     !  初期不整の大きさ
c      real*8     collapse_maxdisp     !  解析最大変位
c      real*8     out_start_sec        !  データ出力の最初の時刻（現在使用不可）
c      integer    interval_out         !  データ出力間隔
c      end structure
C     ------------------------------------------------------------------
      Control.type_analysis       =  IANAL
      Control.init_disp           =  in_disp
      Control.init_stress         =  in_stres
      Control.init_imperfection   =  NINDIT
      Control.amp_imperfection    =  GINDIS
      Control.collapse_maxdisp    =  DMAXCK
      Control.out_start_sec       =  SOUTSC
      Control.interval_out        =  IWSTP
      Control.analysis_3D         =  IT_ANALYS
       Control.jikuzero            =  JIKUZERO
       Control.jikuzero_alph       =  G_JIKUZERO_ALPH
       write(76,'(//a)') ' 構造体：Control: '
      write(76,*) Control.type_analysis,'       =  IANAL'
      write(76,*) Control.init_disp    ,'       =  in_disp'
      write(76,*) Control.init_stress  ,'       =  in_stres'
      write(76,*) Control.init_imperfection,'   =  NINDIT'
      write(76,*) Control.amp_imperfection,'    =  GINDIS'
      write(76,*) Control.collapse_maxdisp,'    =  DMAXCK'
      write(76,*) Control.out_start_sec  ,'     =  SOUTSC'
      write(76,*) Control.interval_out   ,'     =  IWSTP'
      write(76,*) Control.analysis_3D    ,'     =  IT_ANALYS'
      write(76,*) Control.jikuzero       ,'     =  JIKUZERO'
      write(76,*) Control.jikuzero_alph  ,'     =  G_JIKUZERO_ALPH'
      return
      end
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Set_Model_type
C     ------------------------------------------------------------------
C     ●   Parameter 構造体の値セット
C     ------------------------------------------------------------------
      subroutine Set_Model_type(Model_type)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / n_model_s / Model_type
c-----------------------------------------------------------------------
```

```
c  モデルパラメータ
c      structure  / n_model_s/
c       integer    n_e_models           ! 要素モデルの最大数
c       integer    no_e_model(100)      ! 要素モデルの番号
c       integer    n_div_model(100)     ! 要素モデルの分割数
c       integer    nm_div_model(100)    ! ファイバー要素の最大数
c       integer    nm_div_element(100)  ! ファイバー要素のエレメント最大数
c       integer    n_e_model(100)       ! 要素モデルの数
c       integer    n_m_model(100)       ! 部材モデルの数
c       integer    n_damp(100)          ! 部材減衰
c       integer    n_m_damp             ! 全部材減衰数
c       integer    n_m_bilinear         ! ファイバー要素バイリニア用の最大要素数
c       integer    n_m_trilinear        ! ファイバー要素トリリニア用の最大要素数
c      end structure
c      record / n_model_s / Model_type
c        モデル番号
c        Model_No.1 = 1              !  幾何学非線形型有限要素弾性モデル
c        Model_No.2 = 2              !  3 次元せん断弾塑性モデル
c        Model_No.3 = 3              !  3 次元軸力弾塑性モデル
c        Model_No.4 = 4              !  3 次元ケーブル弾塑性モデル
c        Model_No.5 = 5              !  3 次元免振モデル (MSS モデル)
c        Model_No.6 = 6              !  3 次元制震 Maxwell モデル
c        Model_No.7 = 7              !  3 次元弾塑性バネモデル(*)
c        Model_No.11 = 11            !  両端ファイバーモデル
c        Model_No.12 = 12            !  両端、中央ファイバーモデル
c        Model_No.13 = 21            !  両端 MS モデル
c        Model_No.14 = 22            !  両端、中央 MS モデル
c        Model_No.15 = 31            !  幾何学非線形+弾塑性型有限要素モデル
c        Model_No.21 = 41            !  3 次元プレテンション動作モデル
c        Model_No.50 = 1001          !  DLL 有限要素弾塑性モデル
c-------------------------------------------------------------------------
      Model_type.n_e_models =      100
       Model_type.n_m_damp =        0
       Model_type.n_m_bilinear =    0
       Model_type.n_m_trilinear =   0
       Model_type.nm_div_fmodel =   0
       Model_type.nm_div_felement = 0
       Model_type.nm_div_msmodel =  0
       Model_type.nm_div_mselement= 0
       Model_type.n_m_ro_model    = 0
       do i=1,Model_type.n_e_models
      Model_type.no_e_model(i) = 0
       end do
c-------------------------------------------------------------------------
c
c    ★          モデル No. 1 の定義 幾何学非線形型有限要素弾性モデル
c
c-------------------------------------------------------------------------
      number = 1
      Model_type.no_e_model(number)   = 1  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
```

```
c------------------------------------------------------------------------
c
c     ★          モデル No. ２の定義 3 次元せん断弾塑性モデル
c
c------------------------------------------------------------------------
      number = 2
      Model_type.no_e_model(number)   = 2  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c------------------------------------------------------------------------
c
c     ★          モデル No. ３の定義 3 次元軸力弾塑性モデル
c
c------------------------------------------------------------------------
      number = 3
      Model_type.no_e_model(number)   = 3  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c------------------------------------------------------------------------
c
c     ★          モデル No. ４の定義 3 次元ケーブル弾塑性モデル
c
c------------------------------------------------------------------------
      number = 4
      Model_type.no_e_model(number)   = 4  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c------------------------------------------------------------------------
c
c     ★          モデル No. ５の定義 3 次元免振モデル
c
c------------------------------------------------------------------------
      number = 5
      Model_type.no_e_model(number)   = 5  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c------------------------------------------------------------------------
c
c     ★          モデル No. ６の定義 3 次元制震 Maxwell モデル
c
c------------------------------------------------------------------------
      number = 6
      Model_type.no_e_model(number)   = 6  !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
```

```
      Model_type.n_damp(number)        = 1  !部材減衰ありか
c-----------------------------------------------------------------------
c
c     ★          モデル No. ７の定義 3 次元弾塑性バネモデル
c
c-----------------------------------------------------------------------
      number = 7
      Model_type.no_e_model(number)    = 7  !要素モデルの番号
      Model_type.n_div_model(number)   = 1  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-----------------------------------------------------------------------
c
c     ★          モデル No. １１の定義 両端ファイバーモデル
c
c-----------------------------------------------------------------------
      number =11
      Model_type.no_e_model(number)    = 11 !要素モデルの番号
      Model_type.n_div_model(number)   = 6  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-----------------------------------------------------------------------
c
c     ★          モデル No. １２の定義 両端、中央ファイバーモデル
c
c-----------------------------------------------------------------------
      number = 12
      Model_type.no_e_model(number)    = 12 !要素モデルの番号
      Model_type.n_div_model(number)   = 7  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-----------------------------------------------------------------------
      number =18
      Model_type.no_e_model(number)    = 13 !要素モデルの番号
c-------------------------------------------★弾性要素４モデル
c      Model_type.n_div_model(number)  = 5 !要素モデルの分割数
c-------------------------------------------★弾性要素２モデル
      Model_type.n_div_model(number)   = 3  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-----------------------------------------------------------------------
c
c     ★          モデル No. １３の定義 両端 MS モデル
c
c-----------------------------------------------------------------------
      number = 13
      Model_type.no_e_model(number)    = 21 !要素モデルの番号
      Model_type.n_div_model(number)   = 6  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
```

```
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c-------------------------------------------------------------------
c
c     ★          モデル No. １４の定義 両端、中央 MS モデル
c
c-------------------------------------------------------------------
      number = 14
      Model_type.no_e_model(number)   = 22 !要素モデルの番号
      Model_type.n_div_model(number)  = 7  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c-------------------------------------------------------------------
c
c     ★          モデル No. １５の定義 幾何学非線形+弾塑性型有限要素モデル
c
c-------------------------------------------------------------------
      number = 15
      Model_type.no_e_model(number)   = 15 !要素モデルの番号
      Model_type.n_div_model(number)  = 1  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c-------------------------------------------------------------------
c
c     ★          モデル No. １６の定義 両端アナロジーモデル
c
c-------------------------------------------------------------------
      number = 16
      Model_type.no_e_model(number)   = 31 !要素モデルの番号
      Model_type.n_div_model(number)  = 6  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c-------------------------------------------------------------------
c
c     ★          モデル No. １７の定義 両端、中央アナロジーモデル
c
c-------------------------------------------------------------------
      number = 17
      Model_type.no_e_model(number)   = 32 !要素モデルの番号
      Model_type.n_div_model(number)  = 7  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
      Model_type.n_m_model(number)    = 0  !部材モデルの数
      Model_type.n_damp(number)       = 0  !部材減衰ありか
c-------------------------------------------------------------------
c
c     ★          モデル No. １９の定義 両端ぴん、中央アナロジーモデル
c
c-------------------------------------------------------------------
      number = 19
      Model_type.no_e_model(number)   = 33 !要素モデルの番号
      Model_type.n_div_model(number)  = 3  !要素モデルの分割数
      Model_type.n_e_model(number)    = 0  !要素モデルの数
```

```
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-------------------------------------------------------------------------
c
c     ★          モデル No. ２１の定義 3 次元プレテンション動作モデル
c
c-------------------------------------------------------------------------
      number = 21
      Model_type.no_e_model(number)    = 41 !要素モデルの番号
      Model_type.n_div_model(number)   = 1  !要素モデルの分割数
      Model_type.n_e_model(number)     = 0  !要素モデルの数
      Model_type.n_m_model(number)     = 0  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
c-------------------------------------------------------------------------
c
c     ★          モデル No. １００１の定義 スペースフレーム用 DLL 有限要素弾塑性モデル
c
c-------------------------------------------------------------------------
      number = 50
      Model_type.no_e_model(number)    = 1001  !要素モデルの番号
      Model_type.n_div_model(number)   = 1  !要素モデルの分割数
      Model_type.n_e_model(number)     = 1  !要素モデルの数
      Model_type.n_m_model(number)     = 1  !部材モデルの数
      Model_type.n_damp(number)        = 0  !部材減衰ありか
       return
      end
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Set_newmark
C       ----------------------------------------------------------
C     ●   構造体 Newmark_P の値セット(ok)
C     ------------------------------------------------------------
      subroutine Set_newmark(Newmark_P,t1,t2,bet1_1,bet1_2,
     *        bet2_1,bet2_2,dt,beta,eps_v,iroop,gumma,
     *        n_damp_type,n_damp_1,n_damp_2)
C     ------------------------------------------------------------
      implicit real*8(A-H,O-Z)
       include "submain.h"
      record / newmark_s / Newmark_P
C     ------------------------------------------------------------
c     ニューマークβ法のパラメータ
c      structure  / newmark_s/
c       real*8    total_T    !  解析時間
c      real*8    f1_T       !  第１段階解析時間
c      real*8    f2_T       !  第２段階解析時間
c      integer   nn_step    !  解析ステップ数
c      integer   n2_step    !  第２段階開始ステップ数
c       integer   n_damp_type! 減衰タイプ  1:質量比例  2:剛性比例  3:レーリー型 4:その他
c      integer   n_damp_1   !  減衰定数決定用第一モード番号
c      integer   n_damp_2   !  減衰定数決定用第二モード番号
c      real*8    alf1_1     !  第１段階レーリー減衰パラメータ１
c                              ここでは減衰定数をセットする (hh1)
c      real*8    alf1_2     !  第１段階レーリー減衰パラメータ 2
c                              ここでは減衰定数をセットする (hh2)
c      real*8    alf2_1     !  第２段階レーリー減衰パラメータ１
```

```
c                                ここでは減衰定数をセットする（qhh1）
c      real*8    alf2_2     !  第２段階レーリー減衰パラメータ 2
c                                ここでは減衰定数をセットする（qhh2）
c      real*8    dt         !  増分時間Δt
c      real*8    dt2        !  増分時間Δtの２乗
c      real*8    beta       !  ニューマークのβ
c      real*8    delta      !  ニューマークのδ
c      real*8    ddt        !  δ・Δt
c      real*8    bdt        !  β・Δt2
c      real*8    ddt_1      !  (1 -δ)Δt
c      real*8    bdt_5      !  (0.5 -β)Δt2
c      real*8    dt5        !  0.5・Δt2
c      real*8    eps_v      !  収束判定閾値
c      integer   max_repeat !  収束計算のための最大反復回数
c      real*8    gumma      !  収束計算のための収束予測係数
c      end structure
c      record / newmark_s / Newmark_P
c      t1                       第１段階解析時間
c      t2                       第２段階解析時間
c      bet1_1                   第１段階減衰定数（hh1）
c      bet1_2                   第１段階減衰定数（hh2）
c      bet2_1                   第２段階減衰定数（qhh1）
c      bet2_2                   第２段階減衰定数（qhh1）
c      dt                       増分時間Δt
c      beta                     ニューマークのβ
c      eps_v                    収束判定閾値
c      iroop                    収束計算のための最大反復回数
c      gumma                    収束計算のための収束予測係数
c      n_damp_type              減衰タイプ　1:質量比例　2:剛性比例　3:レーリー型 4:その他
c      n_damp_1                 減衰定数決定用第一モード番号
c      n_damp_2                 減衰定数決定用第二モード番号
C      ------------------------------------------------------------
       delta             =  0.5
       if(beta.gt.0.3) delta = 0.6
       Newmark_P.f1_T        =  t1
       Newmark_P.f2_T        =  t2
       Newmark_P.total_T     =  t1 + t2
       Newmark_P.n2_step     =  t1/dt + 1
       Newmark_P.nn_step     =  (t1 + t2)/dt + 1
       Newmark_P.n_damp_type =  n_damp_type
       Newmark_P.n_damp_1    =  n_damp_1
       Newmark_P.n_damp_2    =  n_damp_2
       Newmark_P.alf1_1      =  bet1_1
       Newmark_P.alf1_2      =  bet1_2
       Newmark_P.alf2_1      =  bet2_1
       Newmark_P.alf2_2      =  bet2_2
       Newmark_P.dt          =  dt
       Newmark_P.beta        =  beta
       Newmark_P.delta       =  delta
       Newmark_P.eps_v       =  eps_v
       Newmark_P.max_repeat  =  iroop
       Newmark_P.gumma       =  gumma
       Newmark_P.dt2         =  dt*dt
       Newmark_P.ddt         =  delta*dt
```

```
      Newmark_P.bdt          =  beta*dt*dt
      Newmark_P.ddt_1        =  (1. - delta)*dt
      Newmark_P.bdt_5        =  (0.5- beta )*dt*dt
      Newmark_P.dt5          =  0.5*dt*dt
      write(76,'(//a)') ' 構造体：Newmark_P '
      write(76,*) Newmark_P.f1_T         ,'=  t1'
      write(76,*) Newmark_P.f2_T         ,'=  t2'
      write(76,*) Newmark_P.total_T      ,'=  t1 + t2'
      write(76,*) Newmark_P.n2_step      ,'=  t1/dt + 1'
      write(76,*) Newmark_P.nn_step      ,'=  (t1 + t2)/dt + 1'
      write(76,*) Newmark_P.n_damp_type ,'=  n_damp_type'
      write(76,*) Newmark_P.n_damp_1     ,'=  n_damp_1'
      write(76,*) Newmark_P.n_damp_2     ,'=  n_damp_2'
      write(76,*) Newmark_P.alf1_1       ,'=  bet1_1'
      write(76,*) Newmark_P.alf1_2       ,'=  bet1_2'
      write(76,*) Newmark_P.alf2_1       ,'=  bet2_1'
      write(76,*) Newmark_P.alf2_2       ,'=  bet2_2'
      write(76,*) Newmark_P.dt           ,'=  dt'
      write(76,*) Newmark_P.beta         ,'=  beta'
      write(76,*) Newmark_P.delta        ,'=  delta'
      write(76,*) Newmark_P.eps_v        ,'=  eps_v'
      write(76,*) Newmark_P.max_repeat   ,'=  iroop'
      write(76,*) Newmark_P.gumma        ,'=  gumma'
      write(76,*) Newmark_P.dt2          ,'=  dt*dt'
      write(76,*) Newmark_P.ddt          ,'=  delta*dt'
      write(76,*) Newmark_P.bdt          ,'=  beta*dt*dt'
      write(76,*) Newmark_P.ddt_1        ,'=  (1. - delta)*dt'
      write(76,*) Newmark_P.bdt_5        ,'=  (0.5- beta )*dt*dt'
      write(76,*) Newmark_P.dt5          ,'=  0.5*dt*dt'
      return
      end
C     ------------------------------------------------------------------
C     ●　SUBROUTINE /Set_dynamic_load
C     ------------------------------------------------------------------
C     ●　構造体 Dynamic_load の値セット(ok)
C     ------------------------------------------------------------------
      subroutine Set_dynamic_load(Dynamic_load,load_s,
     *                     load_d,amp_load_d,load_mass)
C     ------------------------------------------------------------------
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / dynamic_load_s / Dynamic_load
      dimension load_s(3),load_sd(3),load_d(3),amp_load_d(3)
C     ------------------------------------------------------------------
c      structure / dynamic_load_s/
c      integer    load_point(3)          !  静的節点荷重ありか(0:なし、1:データセット 2:ファイル入力)
c      real*8     dt_load_point(3)       !  静的節点荷重の増分時間
c      integer    n_load_point           !  動的節点荷重ファイルの最大個数
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
c      integer    n_load_dynamic         ! *地震荷重ファイルの最大個数
c       integer    load_mass             ! *整合質量ありか
c                           注：*印はここでは定義されず
```

```
c       end structure
c      record / dynamic_load_s / Dynamic_load
c      load_s     :integer  節点荷重の有無
c      load_d     :integer  地震荷重の有無
c      amp_load   :real*8   地震荷重の大きさ
c      load_mass  :integer  整合質量の有無
C     ------------------------------------------------------------
      do i=1,3
      Dynamic_load.load_point(i)        =  load_s(i)
      Dynamic_load.load_dynamic(i)      =  load_d(i)
      Dynamic_load.amp_load_dynamic(i)  =  amp_load_d(i)
      Dynamic_load.dt_load_dynamic(i)   =  0.
      Dynamic_load.dt_load_point(i)     =  0.
       end do
      Dynamic_load.load_mass            =  load_mass
      Dynamic_load.n_load_point         = 0
      Dynamic_load.n_load_dynamic       = 0
       write(76,'(//a)') ' 構造体：Dynamic_load'
      do i=1,3
      write(76,*) Dynamic_load.load_point(i)       ,' =  load_s(i)'
      write(76,*) Dynamic_load.load_dynamic(i)     ,' =  load_d(i)'
      write(76,*) Dynamic_load.amp_load_dynamic(i) ,' =  amp_load_d(i)'
      write(76,*) Dynamic_load.dt_load_dynamic(i) ,'  =  dt_load_dynamic'
      write(76,*) Dynamic_load.dt_load_point(i)    ,' =  dt_load_point'
       end do
      write(76,*) Dynamic_load.load_mass           ,' =  load_mass'
       write(76,*) Dynamic_load.n_load_point        ,'=  n_load_point'
       write(76,*) Dynamic_load.n_load_dynamic      ,'=  n_load_dynamic'
      return
       end
```

7.2.8 動的解析結果出力用パラメータファイル

本節では、動的解析結果を制御するパラメータを保存するファイルについて解説する。このファイルは、SPACE 中のダイアログで入力したパラメータを保存する。このファイルに関するキィワードは、doutcl であり、その例を以下に示す。

```
      12         2
      4       2       3       4       5       0       0       0       0       0
      4       0
 0.000000E+00     0.500000E+03
```

第 1 行の 12 と 2 は、このファイルで使用されている整数と実数のパラメータ数を表す。

整数
1．計算結果をファイルに出力するときの間隔である。この間隔とは、増分時間毎に 1 ステップとして計算する。例えば、２ステップとすると増分計算の２回目毎に出力する。

11. 計算結果を図形描画するときの間隔をステップ数で指定する。
2-10,12. フィバーの応答結果について出力する部材を設定する。出力するデータ量が非常に多いため、SPACE では 10 部材に制限する。ここで指定した部材が、ファイバー部材であるかどうかは、システム内でチェックし、ファイバー部材でない場合は、自動的にファイバー応力を出力しない。

実数
1．動的解析を実行する上で、システムが崩壊とみなす最大変位を表す。ただし、この最大変位は、数値計算そのものに影響を与えるものではない。解析を実行する過程で、任意節点の変位がこの値以上になると計算を中止する。
2． 出力を開始する時間（秒）、ただし、現在は使用していない。

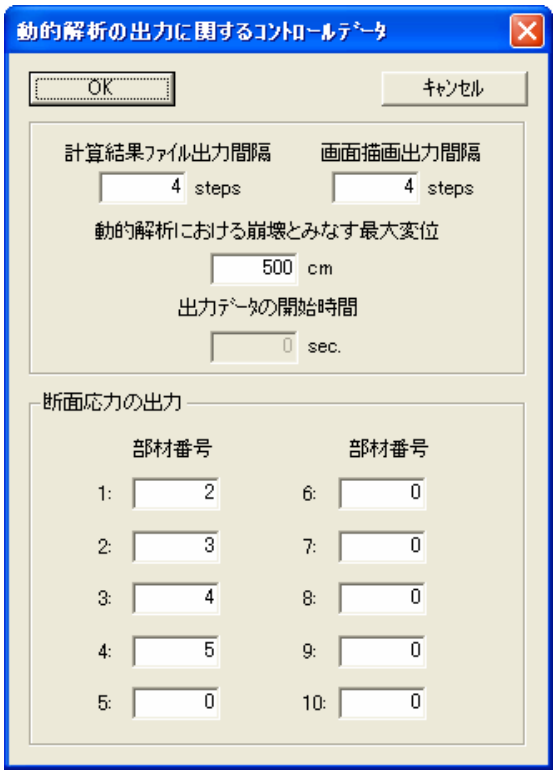

図 7-5 出力コントロールデータダイアログ

上記仕様のファイルを読み込み、動的ソルバー用にデータをセットするプログラム doutcl() を以下に示す。

```
C     ●    SUBROUTINE /doutcl
C
C     ●    ファイル出力定義ファイルを入力する(ok)
C
      subroutine doutcl(ierr,IWSTP,SOUTSC,DMAXCK,No_section)
      IMPLICIT REAL*8(A-H,O-Z)
       character fcode*6
      dimension id(100),No_section(11)
      real*4 fd(100)
      ierr=0
      ihan=0
      numb=1
      fcode='doutcl'
      call datset(ihan,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      IWSTP=ID(2)
      SOUTSC=FD(2)
      DMAXCK=FD(3)
      iix=1
      do i=1,9
      if(id(i+2).ne.0) then
      iix=iix+1
      No_section(iix)=id(i+2)
      endif
      enddo
```

```
c      write(76,'(a,15i4)') ' id:',(id(j),j=1,13)
       if(id(13).ne.0)then
       iix=iix+1
      No_section(iix)=id(13)
       endif
       No_section(1)=iix-1
c      write(76,'(a,15i4)') ' No:',(No_section(j),j=1,11)
       RETURN
       END
```

7.2.9 図形表示用パラメータファイル

本節では、図形表示のパラメータを保存するファイルに関して解説する。このファイルはSPACE中のダイアログで入力したパラメータを保存する。このファイルに関するキィワードは、perscl であり、その例を以下に示す。

```
      3         17
    2       2       2
 0.000000E+00    0.000000E+00    0.000000E+00    0.000000E+00   -0.100000E+05
 0.000000E+00    0.500000E+00    0.100000E+00    0.100000E+01    0.100000E+01
 0.100000E+03    0.200000E+00    0.100000E+01    0.100000E+02    0.100000E+01
 0.200000E+01    0.100000E+03
```

第１行の 3 と 17 は、このファイルで使用されている整数と実数のパラメータ数を表す。

```
整数
１－３．各３方向に、図形の原点移動を行うか否かを表す。最初、透視図は解析モ
    デルの原点をウインドウの中心にして描かれる。そのため構造物の節点座標の
    設定により、原点が構造物の中心にないとき図形がウインドウの隅に描かれる。
    1：移動しない　2：図形の中央に原点を移動する。
実数
１－３．透視計算するときの画面の３方向位置（現在は、原点におくように設定さ
      れており、データを変更できない）
４－６．透視計算するときの３方向視点位置
７．図形の縮尺
以下は、図形表示における各パラメータの初期値
８．加速度　　９．速度　　10．変位　　11．モード
12．サークル記号　13．矢印　　14．曲げモーメント
15．せん断力　　16．断面の応力　　17．断面のひずみ
```

上記仕様のファイルを読み込み、動的ソルバーの図形処理用にデータをセットするプログラム PERSCT()を示す。このプログラムは動的解析システムが起動した直後に、C++の MainFrame()から呼ばれる。

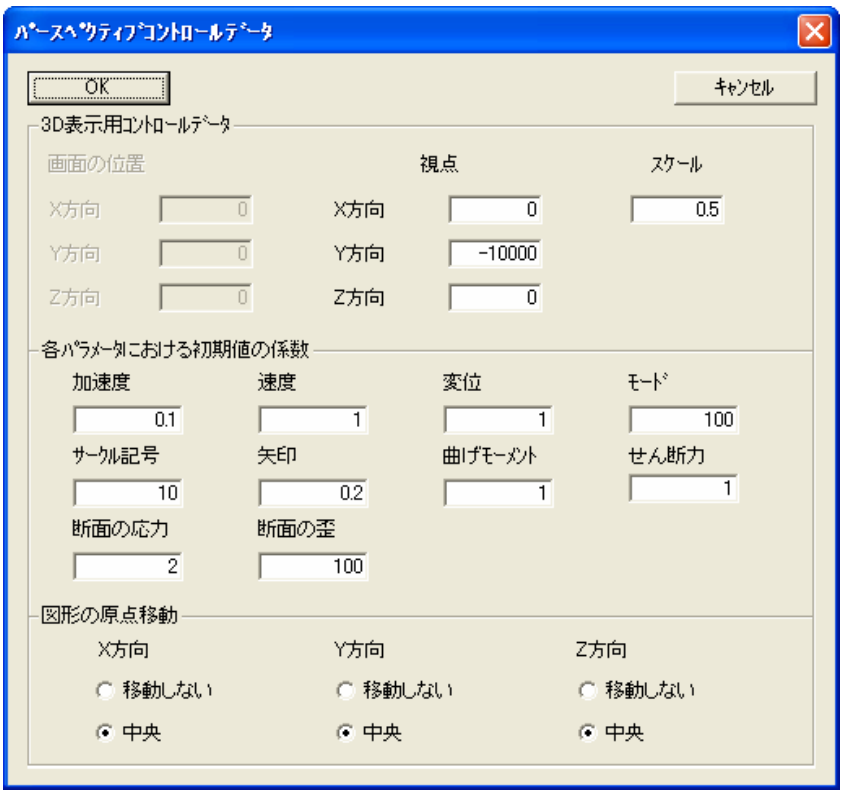

図 7-6 透視図コントロールデータダイアログ

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /PERSCT
C       ----------------------------------------------------------------
C     ●
C     ------------------------------------------------------------------
      subroutine PERSCT(Ix1,ix2,ix3,SCRNPS,VIEWPS,SCALPS,SCALEP,FMAG)
      character fcode*6
      dimension id(100),SCRNPS(3),VIEWPS(3),SCALEP(4),FMAG(6)
      real*4 fd(100)
      ierr=0
      ihan=0
      numb=1
      fcode='perscl'
      call DATSET(IHAN,fcode,ID,FD)
      if(ihan.ne.0) then
      ierr=1
      return
      endif
      ix1=id(2)
      ix2=id(3)
      ix3=id(4)
      SCRNPS(1)    = FD(2)
      SCRNPS(2)    = FD(3)
      SCRNPS(3)    = FD(4)
      VIEWPS(1)    = FD(5)
```

```
      VIEWPS(2)     = FD(6)
      VIEWPS(3)     = FD(7)
      scalps=60
      if(fd(8).ne. 0.)  SCALPS = FD(8)
      SCALEP(1)         = FD(9)
      SCALEP(2)         = FD(10)
      SCALEP(3)         = FD(11)
      SCALEP(4)         = FD(12)
      FMAG(1)           = FD(13)
      FMAG(2)           = FD(14)
      FMAG(3)           = FD(15)
      FMAG(4)           = FD(16)
      FMAG(5)           = FD(17)
      FMAG(6)           = FD(18)
10    continue
      return
      end
```

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。

SPACE 動的解析システム中の動的ソルバーは、多くの解析結果をファイルとして出力する。これらは動的プレゼンターやレポーターで処理され、理解しやすい形でユーザーに提示される。本節では、動的ソルバーが出力するファイルの仕様と出力コードを解説する。

動的ソルバーで出力するファイルは、図 7-7 に示す SPACE の中のダイアログで選択する。このダイアログの中で、書き込み可能とすると該当するファイルに結果が出力される。

図 7-7 動的解析の結果ファイルダイアログ

SPACE で選択したファイル名等の情報は、動的ソルバーの中に読み込まれ、処理される。まず、動的ソルバーに読み込まれ、処理される部分のコードを以下に示す。

最初に、ファイルの管理部分を動的ソルバーの submain_dynamic_a()から取り出す。動的ソルバーの中で、まず、サブルーチン sysnam()で、コントロールファイル名を取得する。次に、サブルーチン ctlset()で、コントロールファイルから必要な入出力ファイル名と管理パラメータを得る。この処理によって動的ソルバー内で管理すべきファイルが設

定されることになる。

```
c-------------------------------------------------★システムからのコントロール情報を取得(ok)
      call sysnam(FNX_file,N_analysis)
      if(N_analysis.ne.i_calnum) N_analysis=i_calnum  ! 解析番号を変更
       if(N_analysis.le.6) then
       ierr_dat=1
       call err_outf(ierr_dat)
       return
       endif
      ihan = 0
       ierr = 0
      NFILE=100
       ierr_dat =0
       write(damp_out,*) ' System file input ok. No. of analysis:',
     *                  N_analysis
c-------------------------------------------------★コントロールデータの内容を取得(ok)
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      if(ihan.ne.0) then
      ierr_dat = 2
       call err_outf(ierr_dat)
       return
      endif
```

動的解析処理が終了すると、オープンしたファイルを全てクローズしなければならない。クローズ処理を行った後、コントロールファイル中の該当するファイル名部分に日時を設定する。これらの処理の該当する部分を、動的ソルバーである submain_dynamic_a()から取り出す。

このプログラムの中で配列 ifl は、1 か 0 かで、ファイルがオープンしているかどうかを表す。そのため、このパラメータをチェックし、オープンしている場合は、全ての出力用ファイルをクローズする。その次に、サブルーチン ctlset()でコントロールファイルを再度入力し、fltime()で現時点の日時を所定の位置にセットする。さらに、それらのデータを、同じサブルーチン ctlset()を用いて再度コントロールファイルに書き込む。このように、サブルーチン ctlset()には、コントロールファイルを読み込む機能と書き込む機能が付いている。

```
c-------------------------------------------★ファイルのクローズ
      do i=1,16
       if(ifl(i).eq.1) close(iflz(i))
       enddo
      write(damp_out,*) ' ファイルのクローズ ok'
c-------------------------------------------★ファイルのタイムスタンプ
      ihan = 0
      NFILE=100
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      call fltime(ifl,ifly,N_analysis,iflout)
c      write(damp_out,*) ' ファイルのタイムスタンプ ok',FNX_file
```

```
      ihan = 1
      if(iflout.eq.1) call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
c      write(damp_out,*) ' データセット ok'
```

動的ソルバーにおけるファイル管理の全体像が理解できただろうか。ここからは、具体的にその中身を調べてみよう。観察するサブルーチンは、

```
sysnam()  sysfset()
CTLSET()
Flcheck()
fltime()  dstamp()
```

であり、以下にそのプログラムコードを示す。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /sysnam
C       ----------------------------------------------------------
C     ●   コントロールファイルを入力する(ok)
C     ------------------------------------------------------------
      subroutine sysnam(fn,calnum)
      character fn*100,fnsys*150
      integer   calnum
      fn=' '
      call sysfset(fnsys)                                          ! 1
       write(76,'(a)') fnsys
      open(12,FILE=fnsys)                                          ! 2
       read(12,'(i8)') calnum                                      ! 3
       read(12,'(a)') fn                                           ! 4
       write(76,'(a)') fn
      close(12)                                                    ! 5
       return
       end
C     ------------------------------------------------------------
C     ●   SUBROUTINE /sysfset
C       ----------------------------------------------------------
C     ●   システムからデータを取得(ok)
C     ------------------------------------------------------------
      subroutine sysfset(fnsys)
      CHARACTER fnss*150,fnssx*1(150),fnsys*150
      equivalence (fnss,fnssx)
      fnss=' '
      klen = getenvqq('TMP',fnss)                                  ! 6
      do 8 i=1,150
       if(fnssx(i).eq.' ') goto 9
   8  continue
   9  if(i.ne.0) goto 100
      klen = getenvqq('TEMP',fnss)                                 ! 7
      do 18 i=1,150
       if(fnssx(i).eq.' ') goto 19
```

```
   18 continue
   19 if(i.ne.0) goto 100
      i=1
      goto 10
  100 continue
      fnssx(i)='¥'
      i=i+1
   10 fnssx(i)='s'
      fnssx(i+1)='p'                                                    ! 8
      fnssx(i+2)='a'
      fnssx(i+3)='c'
      fnssx(i+4)='e'
      fnssx(i+5)='s'
      fnssx(i+6)='y'
      fnssx(i+7)='s'
      fnssx(i+8)='.'
      fnssx(i+9)='x'
      fnssx(i+10)='x'
      fnssx(i+11)='x'
      fnsys=fnss
      return
      end
C     ------------------------------------------------------------------
C     ●　SUBROUTINE /CTLSET
C     ------------------------------------------------------------------
C     ●　コントロールファイルを入力する(ok)
C     ------------------------------------------------------------------
      SUBROUTINE CTLSET(ihan,fn,titlez,idfile,nfile)
      common /comctl/ctl,ctlf
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      character fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),kdfile(100),lengf(100)
      character CTL(100)*6,ctlf(100)*10
      integer*4 idfile(100)
      character FN*100
      character BUF*120,FNxx*10,TITLEZ*50
      character bufx(100)*120,BUFY*120
      integer*1 ifn(101),ititlx(100)
      integer*4 ihan,ilen
      data fnxx/'kiteif.ctx'/
C     ------------------------------------------------------------------
      kfile=100
      if(ihan.eq.1) goto 2000                                           ! 9
      do 11 l=1,nfile                                                   ! 10
      fnfile(l)='  '
      jdfile(i)=-1
      kdfile(i)=-1
   11 continue
      title=' '
      open(7,FILE=FN,STATUS='OLD',ERR=810)                              ! 11
      goto 211
  810 continue                                                          ! 12
      write(76,'(a)') ' エラー：コントロールファイルが存在しません。'
      ihan=-1
```

```
      return
  211 continue
C     ------------------------------------------------------------------
      read(7,*) IIDAT                                                  ! 13
      do 210 iij=1,IIDAT
      read(7,'(A)',end=300) BUF
       bufx(iij)=buf
       if(buf(3:3).eq.'T'.or.buf(3:3).eq.'t') goto 23                  ! 14
      do 21 I=1,kfile
      if(BUF(6:11).EQ.CTL(I) .and. buf(14:14).ne.' ') then             ! 15
      fnfile(I)=BUF(14:63)                                             ! 16
       lengf(i)=50
       jdfile(i)=1
       if(BUF(3:3).NE.' ' ) jdfile(i)=0                                ! 17
       kdfile(i)=1
       if(buf(66:66).ne.' ') kdfile(i)=0                               ! 18
       timex(i)=buf(67:86)                                             ! 19
      goto 210
      endif
   21 continue
      goto 210
C     ------------------------------------------------ title set
   23 continue
      title = buf(14:63)                                               ! 20
      TITLEZ=TITLE
       ltitle=50
  210 continue
  300 continue
      close(7)                                                         ! 21
      do 40 i=1,nfile
       idfile(i)=jdfile(i)
   40 continue
      return
C     ------------------------------------------------
 2000 continue                                                         ! 22
      do 101 i=1,iidat                                                 ! 23
       buf=bufx(i)
       if(buf(3:3).eq.'T'.or.buf(3:3).eq.'t') goto 123                 ! 24
      do 121 j=1,kfile                                                 ! 25
      if(BUF(6:11).EQ.CTL(j))then
      BUF(14:63)= fnfile(j)
       buf(3:3)='*'
       if(jdfile(j).eq.1) buf(3:3)=' '
       buf(66:66)='*'
       if(kdfile(j).eq.1) buf(66:66)=' '
       buf(67:84)=timex(j)
      goto 100
      endif
  121 continue
      goto 100
C     ------------------------------------------------ title set
  123 continue                                                         ! 26
      buf(14:63)=title
  100 continue
```

```
      bufx(i)=buf
  101 continue
C     ------------------------------------------------
      open(7,FILE=FN,ERR=819)                                    ! 27
      goto 890
  819 ihan=-1
      write(76,'(a)')
     *  ' エラー：コントロールファイルがオープンできません'
      return
C     ------------------------------------------------
  890 continue                                                   ! 28
      write(7,'(I8)') IIDAT
      do 891 i=1,iidat
      write(7,'(A)')BUFx(i)
  891 continue
      close(7)
      return
       end
C     ------------------------------------------------------------
C     ●   SUBROUTINE /flcheck
C       ----------------------------------------------------------
C     ●    出力ファイルのセットとオープン
C     ------------------------------------------------------------
      subroutine    flcheck(iflx,ifly,iflz,ierr,nread)
       common /sf01/jdfile,ifl,iidat,lengf,ltitle,timex,fnfile,title
      CHARACTER fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),ifl(100),lengf(100)
       integer*4 iflx(16),ifly(16),iflz(16)
C     ------------------------------------------------------------
      ierr=0
       do 2 i=1,16                                               ! 29
       iflx(i)=0
       ifly(i)=0
       iflz(i)=0
 2     continue
c-------------------------------------------★ファイルのオープン
      if(nread.ne.6) then                                        ! 30
       nfl=35
      do i=1,6
       if(ifl(nfl+i).eq.1) then                                  ! 31
       iflx(i)=1
       ifly(i)=nfl+i
       iunit=10+i
      open(UNIT=iunit,FILE=fnfile(nfl+i),FORM='UNFORMATTED')
       iflz(i)=iunit
       endif
      enddo
C     ------------------------------------------------------------
      do i=9,16
       if(ifl(nfl+i).eq.1) then
       iflx(i)=1
       ifly(i)=nfl+i
       iunit=10+i
       if(i.eq.12.or.i.eq.16) then
```

```
      OPEN(UNIT=iunit,FILE=fnfile(nfl+i))
      else
      OPEN(UNIT=iunit,FILE=fnfile(nfl+i),FORM='UNFORMATTED')
      endif
      iflz(i)=iunit
      endif
      enddo
C     ------------------------------------------------------------------
      else                                                          ! 32
      nfl=35
      do i=7,8
      if(ifl(nfl+i).eq.1) then
      iflx(i)=1
      ifly(i)=nfl+i
      iunit=10+i
      OPEN(UNIT=iunit,FILE=fnfile(nfl+i))
      iflz(i)=iunit
      else
      ifl(nfl+i)=0
      endif
      enddo
      endif
      return
      end
C     ------------------------------------------------------------------
C     ●　SUBROUTINE /fltime
C     ------------------------------------------------------------------
C     ●　オープンしたファイルの時間を得る
C     ------------------------------------------------------------------
      subroutine   fltime(iflx,ifly,IANAL,iflout)
      common /sf01/jdfile,kdfile,iidat,lengf,ltitle,timex,fnfile,title
      CHARACTER fnfile(100)*50,title*50,timex*20(100)
      integer*4 jdfile(100),lengf(100),kdfile(100)
      integer*4 iflx(16),ifly(16)
C     ------------------------------------------------------------------
      iflout=0                                                      ! 33
      if(ianal .eq. 6) then                                         ! 34
       do 10 i=7,8
      if(iflx(i).eq.1) then                                         ! 35
      call dstamp(timex(ifly(i)))
      iflout=1
      endif
 10   continue
      else                                                          ! 36
       do 20 i=1,16
      if(iflx(i).eq.1) then
      call dstamp(timex(ifly(i)))
      iflout=1
      endif
 20   continue
      endif
      return
      end
C     ------------------------------------------------------------------
```

```
C     ●   SUBROUTINE /dstamp
C       ──────────────────────────────────────────
C     ●   時間を取り込む(ok)
C       ──────────────────────────────────────────
      subroutine dstamp(timex)
      character timex*20,timexx*20,clockt*9
      timexx='                    '                                    ! 37
      call date(timexx)                                                ! 38
      call time(clockt)
      timexx(10:10)='/'
      timexx(11:18)=clockt
      timexx(19:20)='  '
      timex=timexx                                                     ! 39
      return
      end
```

プログラムコード右端に付した番号にしたがって、その内容の説明を行う。

1. SPACE から情報を受け取るためのファイルを検索する。
2. 該当するファイルをオープンする。
3. 解析種別番号を入力する。
4. コントロールファイル名を入力する。
5. ファイルをクローズする。
6. Windows システム内の「TEM」フォルダの絶対パス名を取得する。
7. 上記のフォルダがない場合は、同じく「TEMP」フォルダの絶対パス名を取得する。
8. そのフォルダ名の後に、SPACE からの情報伝達用ファイル名¥spacesys.xxx を付け加える。
9. サブルーチン CTLSET()には、コントロールファイルを入力する機能と、出力する機能がある。コントロールデータは、common 領域（comcl,sf01)にセットする。ただし、特定のサブルーチン以外は、この common 領域にアクセスすることはできない。機能を判定する変数 ihan が１の場合は出力であり、文番号 2000 へ制御が移る。それ以外は以降のコードが実行される。
10. コントロールファイルの内容を入れる配列（common 領域、jdfile は読み込みチェック、kdfile は書き込みチェック用）を-1 に初期設定する。また、タイトルもクリアする。
11. コントロールファイルをオープンする。
12. ファイルがオープンできない場合は、エラーコード ihan=-1 を付けて、このサブルーチンから戻る。

13. ファイル先頭にある入力行数を読み込む。この入力行数分、以降のデータを読み込むことになる。１行分をバッファ領域に読み込む。
14. この読み込んだ１行がタイトル行であるかどうかチェックする。タイトル行である場合は、文番号 23 に制御が移ることになる。
15. キィワードを用いて、入力した１行がどのファイルに適合するかをチェックする。ここで、kfile はキィワードの総数であり、現在は 100 となっている。また、配列 CTL はキィワードが設定されている common 配列である。
16. 上記のチェックで、キィワードが適合すると、当該のファイル名をセットする。
17. 読み込み可能であるかどうかチェックし、可能である場合は、配列 jdfile を１とし、そうでない場合は 0 とする。
18. 書き込み可能であるかどうかチェックし、可能である場合は、配列 kdfile を１とし、そうでない場合は 0 とする。
19. そのファイルのタイムスタンプを timex にコピーする。
20. タイトルを入れておく変数 title にバッファ領域からコピーする。
21. コントロールファイルをクローズする。common 領域の配列から、引数の配列へデータをコピーする。これで、コントロールファイルの読み込み処理が終了する。
22. ここ以降は、管理ファイル情報（システムが起動中の場合は common 領域で管理している）のコントロールファイルへの書き込み処理が行われる。
23. ファイルの個数分、以下の処理を行い、出力するデータを整える。入力しておいたバッファ配列をバッファ変数にコピーする。
24. その行がタイトルであるかどうかチェックする。タイトルである場合は文番号 123 に制御が移る。そうでない場合は以降の処理を行う。
25. キィワードより適合するファイル番号を見付け出し、ファイル名、書き込み条件、読み込み条件、タイムスタンプを common 配列からコピーする。
26. タイトル行をコピーする。
27. コントロールファイルをオープンする。オープンできない場合は、エラーコードを付けて、処理を終了する。
28. バッファ領域に保存されているコントロールデータを、コントロールファイルに出力する。
29. このサブルーチンは、動的解析結果を出力するファイルを調査し、オープンする。まず、16 個のファイルを管理する配列をゼロクリ

アする。ここで、配列 iflx は書き込みチェック用、配列 ifly はファイル番号、配列 iflz は出力用ユニット番号である。

30. 解析種別によってオープンするファイルを変更する。解析種別 nread が 6 以外（固有値解析以外の場合）は、次のファイルをチェックする。ファイル番号 36-41、44-51 であり、47 と 51 は、ASKII ファイルとして､その他はバイナリ(UNFORMTTED)でオープンする。
31. 書き込みチェック用 common 配列 ifl を用いて、書き込み可能かどうかチェックする｡書き込み可能であれば､ファイルをオープンし、配列 iflx に 1 をセットし、さらに配列 ifly にファイル番号を、配列 iflz にオープンしたファイルのユニット番号をセットする。
32. 固有値解析の場合は、以降の処理を行う。ファイル番号 42,43 に対して、上記のチェックを行い、ファイルをオープンする。
33. このサブルーチン fltime()は、オープンしてデータを出力したファイルに対してクローズした時間をコントロールファイルに書き込む。
34. 解析が固有値解析である場合は、以降の処理を行う。ファイル番号 42、43 に対して、書き込みチェックを行う。
35. 書き込み可能となっている場合、サブルーチン dstamp()をコールして、その日時をシステムより取り込み、common 配列 timex に書き込む。
36. 動的解析に対し、上記と同様の処理を行う。
37. OS より日時を取り込むために、変数 timexx をヌルクリアする。
38. 組み込み関数 date()と time()を用いて、日付と時間を取得する。
39. 取り込んだ日時を引数である変数にコピーする。

動的ソルバーで使用している出力ファイルは、16 であり、その内現在使用しているファイル数は、13 である。以下に、その仕様を示す。

```
c------------------------------------------------------------★ファイルとユニット番号
c      キィワード   ファイル番号   ユニット番号              備考
c  1   'nofc_d'        36           11          !  部材応力により計算された反力
c  2   'hing_d'        37           12          !  不釣合力
c  3   'disp_d'        38           13          !  節点の変位
c  4   'dibm_d'        39           14          !  Inner node displacements in members
c  5   'stsp_d'        40           15          !  部材応力
c  6   'stbm_d'        41           16          !  断面応力
c  7   'mode_d'        42           17     F    !  ﾓｰﾄﾞ変位
c  8   'omeg_d'        43           18     F    !  固有周期、振動数、減衰定数、刺激係数
c  9   'shar_d'        44           19          !  各階せん断力
c  10  'acc_d '        45           20          !  相対加速度
```

```
c  11  'vel_d '       46        21          !  速度
c  12  'max_d '       47        22     F    !  最大相対加速度、速度、変位
c  13  'absa_d'       48        23          !  絶対加速度
c  14  'engy_d'       49        24          !  Energy  of motion
c  15  'ave_d '       50        25          !  Average displacements on response
c  16  'beta_d'       51        26     F    !  最大部材応力
c------------------------------------------------------------★
```

上記は、プログラムの中から抜き出したコメント行であり、左のｃはコメントを表す。次の数字は、ファイルの通し番号であり、以下、ファイルのキィワード、SPACE におけるファイル管理番号、入出力するときのユニット番号である。その次の F は、そのファイルが ASKII ファイルであることを示し、他のファイルはバイナリファイルである。最後は、備考としてファイルの中身を示している。後節以降では、全てのファイルについて、その仕様と出力プログラムコードを示して、解説する。

7.4.2 反力と不釣合力ファイル

本節では、反力と不釣合力を出力するプログラムコードとそのファイル仕様について解説する。まずは、これら当該処理を動的ソルバーである submain_dynamic_a()から取り出し、以下に示す。

```
Get_pointforce()
Out_pointforce()
```

この２つのサブルーチンは、反力と荷重に関連し、下のサブルーチンは不釣合力にも関連する。ここで、出力するデータは、まず部材端応力を釣合座標系に変換し、各節点について和を取る。最終的に得られる応力は、境界では反力となり、荷重が加わっている節点では荷重に等しくなる。これが反力と荷重としてこのファイルに出力される。次に、不釣合力は、この状態の節点に荷重を加えると当該の節点では力の釣合が取れていなければならず、残った応力は不釣合力となり、ファイルとして出力される。非線形性が大きくなると、この不釣合力が解除できず、徐々に大きくなる場合がある。

```
c------------------------------------------------------------★部材節点力（反力と荷重）の出力(ok)
      call Get_pointforce(fll_force_point,Member,n_member,
     *                    Point,n_point)
      call Out_pointforce(fll_force_point,Point,rot_local,
     *                   n_point,ifl(1),iflz(1),i_print)
```

```
c―――――――――――――――――――――――――――――――――――――――――――――★不釣合力の出力(ok)
      call Out_pointforce(fll_force_point,Point,rot_local,
     *                    n_point,ifl(2),iflz(2),i_print)
```

このサブルーチンのプログラムコードを具体的に示す。サブルーチン Get_pointforce()では、部材の両端の応力を各節点について、和を取っている。ただし、ここでは、曲げモーメントに関しては図形描画の関係から省略している。最初に、全節点について応力に関する配列をゼロクリアしている。次に、全部材について両端の応力を、各節点で和を取っている。

```
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /Get_pointforce
C       ―――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    部材節点力のセット(ok)
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Get_pointforce(fll_force_point,Member,n_member,
     *                          Point,n_point)
      include "submain.h"
      record / member_s   / Member
      record / point_s    / Point
      dimension Member(*),Point(*)
      real*8 fll_force_point(3,*)
      real*8 v(6),vv(6)
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
c           ld_point      :real*8 右辺項
c           Member        :structure
c           n_member      :integer 部材数
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_point
       do j=1,3
      fll_force_point(j,i)=0.
       enddo
       enddo
      do i=1,n_member
       i1=Member(i).nm_point(1)
       j1=Member(i).nm_point(2)
      do j=1,3
      fll_force_point(j,i1)=fll_force_point(j,i1) + Member(i).force(j)
      fll_force_point(j,j1)=fll_force_point(j,j1) + Member(i).force(j+6)
       end do
       end do
       return
       end
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /Out_pointforce
C       ―――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●    部材節点力の出力(ok)
C     ―――――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Out_pointforce(fll_force_point,Point,rot_local,
     *                          n_point,ifl,iflz,i_print)
```

```
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / point_s  / Point
      dimension Point(*),rot_local(3,3,*)
      real*8 fll_force_point(3,*),vv(6)
      real*4 v(6)
C     ------------------------------------------------------------------
c           fll_force_point      :real*8 節点不釣合力
c           Point                :structure
c           n_Point              :integer 節点数
c           i_print              :integer 出力制御変数 0:ファイル出力あり
C     ------------------------------------------------------------------
      if(i_print.ne.0) return
       if(ifl .ne.1 ) return
c-------------------------------------------------★応力
       do i=1,n_point
      ij=Point(i).local_coord
       if(ij.ne.0) then
      call trans_VT(fll_force_point(1,i),vv,rot_local(1,1,ij))
      do j=1,3
      v(j) = vv(j)
      end do
      else
      do j=1,3
      v(j) = fll_force_point(j,i)
       end do
       endif
      write(iflz ) (v(j),j=1,3)
       end do
       return
       end
```

サブルーチン Out_pointforce()では、まず、出力すべきか否かについてチェックする。変数 i_print は、ファイルにデータを出力する解析ステップであるかどうかを示し、また、ifl はこのファイルがオープンしているかどうかを示す。この 2 つの変数は、ユーザーが選択するパラメータ、つまり、出力するステップ間隔と出力を希望するか否か、によって決まる。

次に、各節点における応力を出力するわけであるが、2 つの点について注意されたい。ひとつは、計算は全て倍精度で行っているが、ファイルが大きくなることを考慮して、このファイルには全て単精度で出力する。そのため、出力は以下の write 文で、単精度の配列 v で、しかもバイナリデータとして出力している。iflz は、このファイルのユニット番号である。

```
write(iflz ) (v(j),j=1,3)
```

他の注意点は、節点の応力は、釣合座標系で求められており、この

ままで描画すると、局所座標系を使用している節点では、異なった描画を行わなければならない。そこで、この節点応力を釣合座標系から、全体座標系に座標変換しなければならない。そこで、当該の節点が局所座標系を使用しているか否かを示すPoint(i).local_coordをチェックし、使用している場合は、サブルーチン trans_VT()を用いて全体座標系に変換する。このように、描画することを考慮して、全節点同じ座標系である全体座標系でデータを出力する。後の節で説明する節点に関する全ての物理量は、このような処理を行い、全体座標系に変換してファイルに出力する。

このファイルの仕様は、以下のような出力で構成される。3 方向の単精度のバイナリーデータが全節点数分連続で１つのユニットとなり、あとは、ユーザーが設定した出力ステップ毎に、この 1 ユニットが出力される。

7.4.3 節点変位、速度、加速度ファイル

本節では、節点変位、速度、加速度に関する出力ファイルの仕様及びその出力プログラムコードについて解説する。このファイル出力に関するプログラムコードは、submain_dynamic_a()で以下のようにコールされる。

```
c————————————————————————————————★変位、速度、加速度を出力
      call Out_disp_vel_acc(Point,n_point,Parameter_C,
     *     past_disp_point, past_vel_point, past_acc_point,
     *     rot_local,ifl,iflz,vacc,i_print)
```

以下の２つのサブルーチンについて内容を示す。

```
Out_disp_vel_acc()
trans_VT()
```

最初のサブルーチンは、各ステップにおける節点変位、速度、相対加速度、絶対加速度を出力する。同時に、4 つのファイルに出力するため、多少プログラムは長くなっているが、全て同じ仕様で出力されているため、理解することは困難ではない。まず、前節と同様にパラメータ i_print を用いて、このステップで出力すべきかどうかチェックする。次に、節点が局所座標を使用しているかどうかで、2 つに処理を分けている。前半は、全節点局所座標を使用せずの場合であり、後半は、1 箇所でも局所座標を使用している場合である。

最初は、節点変位で、3 方向を同時に、全節点についてファイル出力

する。以下は全て同じ仕様、つまり、単精度のバイナリーで出力されている。ここでは、まず、単精度の配列ｖをゼロクリアし、次に、節点の境界条件Point(i).irest(j)を参照し、境界で変位がゼロの場合はそのままとし､一方境界でない場合は､自由度番号を示すPoint(i).irest(j)を用いてpast_disp_point(ires)から該当するデータを取り出す。これが一連の処理であり、次の速度、加速度も同様な操作を行って、ファイルにデータを出力する。プログラムコードで、太文字の部分がそれに該当する。この中で、配列 vacc は、そのステップの 3 方向地震加速度である。出力する write 文の前で、そのファイルがオープンされているかどうかチェックされており、また、write(ifl(13))などの ifl は、このファイルのユニット番号が収められている。

次に、後半部分である局所座標系を含む場合であるが、構造体成分である Point(i).local_coord をチェックし、その節点が局所座標系であるかどうかチェックする。使用していない場合は、前記と同様の処理を行い、また、局所座標系を使用している場合は、該当する変位を一旦、倍精度の変位配列 vv に格納する。さらに、その変位をサブルーチン trans_VT()によって全体座標系の変位に変換する。最後に、倍精度変位配列から単精度配列ｖに格納し直し､バイナリー型でファイルに出力する。後の速度、加速度は、変位に対する処理方法と全く同一である。

この４つのファイル仕様は、全て同じで、以下のような出力で構成される。3 方向の単精度のバイナリーデータが全節点数分連続で１つのユニットとなり、あとは、ユーザーが設定した出力ステップ毎に、この１ユニットが出力される。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /Out_disp_vel_acc
C       ----------------------------------------------------------
C     ●    節点の変位、速度、加速度を出力(ok)
C     ------------------------------------------------------------
      subroutine Out_disp_vel_acc(Point,n_point,Parameter_C,
     *     past_disp_point, past_vel_point, past_acc_point,
     *     rot_local,ifl,iflz,vacc,i_print)
       implicit real*8(A-H,O-Z)
       include "submain.h"
       record / point_s  / Point
      record / parameter_s /Parameter_C
       dimension Point(*)
       dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
       dimension rot_local(3,3,*),vacc(6),vv(6),vvv(6)
       dimension ifl(16),iflz(16)
       real*4    v(6)
C     ------------------------------------------------------------
```

```
c     Max_disp        :structure 最大値
c     n_point         :integer   節点数
c     Point           :structure
c     past_disp_point :real*8 計算結果の変位
c     past_vel_point  :real*8 計算結果の速度
c     past_acc_point  :real*8 計算結果の加速度
c     vacc            :real*8 地震加速度
c     i_print         :integer 出力制御変数 0:ファイル出力あり
C     ------------------------------------------------------------
      if(i_print.ne.0) return
c-----------------------------------------------------★局所座標系なし
      if(Parameter_C.n_local_coord.eq.0) then
c-----------------------------------------------------★変位 3（この番号は、出力管理番号）
      do i=1,n_point
      do j=1,3
      ires= Point(i).irest(j)                    !節点の境界条件を示す。0：固定、0 以外：自由度番号
      v(j)=0.
      if(ires.ne.0) v(j) = past_disp_point(ires) !釣合系における節点変位
      end do
      if(ifl(3).eq.1) write(iflz(3)) (v(j),j=1,3)
      end do
c-----------------------------------------------------★速度 11
c      write(76,*) ' 速度：', n_point
      do i=1,n_point
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j) = past_vel_point(ires)      !釣合系における節点速度
      end do
      if(ifl(11).eq.1) write(iflz(11)) (v(j),j=1,3)
      end do
c-----------------------------------------------------★相対加速度 10
c-----------------------------------------------------★絶対加速度 13
c      write(76,*) ' 加速度：', n_point
      do i=1,n_point
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j) = past_acc_point(ires)      !釣合系における節点相対加速度
      end do
      if(ifl(10).eq.1) write(iflz(10)) (v(j),j=1,3)
      do j=1,3
      v(j)=v(j)+vacc(j)            !相対加速度に地震加速度を足して、絶対加速度とする
      enddo
      if(ifl(13).eq.1) write(iflz(13)) (v(j),j=1,3)
      end do
c-----------------------------------------------------★局所座標系あり
      else
c-----------------------------------------------------★変位 3
      do i=1,n_point
      ij=Point(i).local_coord                        !各節点の局所座標系管理番号
      if(ij.eq.0) then                          !局所座標系管理番号が０の場合は、使用せず
      do j=1,3
      ires= Point(i).irest(j)
```

```
      v(j)=0.
      if(ires.ne.0) v(j) = past_disp_point(ires)
      end do
      if(ifl(3).eq.1) write(iflz(3)) (v(j),j=1,3)
      else                                    !局所座標系管理番号が０以外の場合の処理
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_disp_point(ires)
      end do
      call trans_VT(vv,vvv,rot_local(1,1,ij))       !節点変位を局所座標系から全体座標系に変換
      do j=1,3
      v(j)=vvv(j)                                   !各節点の変位を単精度配列に入れ替える
      enddo
      if(ifl(3).eq.1) write(iflz(3)) (v(j),j=1,3)
      endif
      end do
c-------------------------------------------------★速度 11
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.eq.0) then
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j) = past_vel_point(ires)
      end do
      if(ifl(11).eq.1) write(iflz(11)) (v(j),j=1,3)
      else
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_vel_point(ires)
      enddo
      call trans_VT(vv,vvv,rot_local(1,1,ij))
      do j=1,3
      v(j)=vvv(j)
      enddo
      if(ifl(11).eq.1) write(iflz(11)) (v(j),j=1,3)
      endif
      end do
c-------------------------------------------------★相対加速度 10
c-------------------------------------------------★絶対加速度 13
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.eq.0) then
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j) = past_acc_point(ires)
      end do
      if(ifl(10).eq.1) write(iflz(10)) (v(j),j=1,3)
      do j=1,3
      v(j)=v(j)+vacc(j)
      enddo
```

```
      if(ifl(13).eq.1) write(iflz(13)) (v(j),j=1,3)
      else
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_acc_point(ires)
      end do
      call trans_VT(vv,vvv,rot_local(1,1,ij))
      do j=1,3
      v(j)=vvv(j)
      enddo
      if(ifl(10).eq.1) write(iflz(10)) (v(j),j=1,3)
      do j=1,3
      v(j)=vvv(j)+vacc(j)
      enddo
      if(ifl(13).eq.1) write(iflz(13)) (v(j),j=1,3)
      endif
      end do
      endif
      return
      end
C     ------------------------------------------------------------
C     ●   SUBROUTINE /trans_VT
C       ----------------------------------------------------------
C     ●   荷重ベクトル（釣合系）を全体系に座標変換する
C     ------------------------------------------------------------
      subroutine trans_VT(p,q,rot)
      implicit real*8(A-H,O-Z)
      real*8 p(3),q(3)
      dimension rot(3,3)
      do i=1,3
      sum = 0.
      do j=1,3
      sum=sum + rot(i,j)*p(j)
      end do
      q(i)=sum
      end do
      return
      end
```

サブルーチン trans_VT()は、節点における物理データを座標変換するルーチンであり、ここで配列 rot は、釣合座標系より全体座標系へ変換する回転行列である。

7.4.4 部材応力ファイル

本節では、部材の応力を出力するファイルの仕様とその出力サブルーチンについて解説する。このファイル出力に関するプログラムコードは、submain_dynamic_a()で以下のようにコールされる。

```
c------------------------------------------------------★部材応力を出力
      call Out_stress(Member,Element,E_model6_real,M_model11,M_model12,
     *                M_model13,M_model15,M_model21,M_model22,
     *                M_model31,M_model32,M_model33,
     *                n_member,ifl,iflz,i_print,Out_section)
```

以下の１つのサブルーチンについて内容を示す。

Out_stress ()

このサブルーチンでは、各部材の両端及び中央の応力を出力する。部材両端以外での応力の出力は、各部材モデルによって定義されている。その定義は、配列 mxtype と myytype で行われている。mxtype の配列番号は部材モデル番号となっており、その値は、myytype の配列番号を指す。配列 myytype の値は、ひとつの部材に対する出力レコード数を表す。現在は、２レコードと３レコードが用いられており、前者は、i 端、j 端の応力を、後者は i 端、j 端の応力に中央の断面応力を付加して、単精度のバイナリーデータとして出力する。当然、このデータを読み込むことになる動的プレゼンターなどでは、この仕様と同じでなくてはならない。この仕様を変更する場合は、他のサブシステムの該当する部分も変更しなければならないので、特に注意されたい。

標準的な 1 レコードの出力は、

```
write(iflz(5)) istat,(v(k),k=1,4)
```

で表され、各断面について、最初のデータは塑性状態を、次に 4 つのデータは、順に、軸力、ｙ軸の曲げモーメント、ｚ軸の曲げモーメント、塑性関数値を表す。ただし、これらの値の意味は、部材モデルによって異なる。塑性状態の仕様は、0 が弾性、1 と 2 は塑性状態を表し、プレゼンターでは 1 は黄色表示、2 は赤表示となる。ファイバー部材モデルでは、1 は断面の一部が塑性域に達したとき、2 は全断面の 80%が塑性状態になったときを表すように設定されている。

プログラムには、次の順序で部材モデルの応力を出力するコードが記述されている。

１．Maxwell モデル
２．3 次元せん断弾塑性モデル
３．3 次元軸力弾塑性モデル
４．3 次元ブレースモデル
５．Model No.5-10
６．モデル 18, 19（両端ピンのファイバーモデルとアナロギーモデル）
７．その他のモデルで標準型

以下に示すプログラムを参照しながら、上記のモデルで特異な部分について説明する。

１）の Maxwell モデルでは、プレゼンターで各種の情報を提示する関係で、標準型のデータ出力とはかなり異なっている。ここでは、2 レコード出力するが、端部の応力ではなく、部材全体の応力、変位を表している。第 1 レコードでは、軸力、y 軸せん断力、z 軸せん断力、Maxwell モデルのダンパー軸力を表す。ダンパー軸力は、1000 分の１の値としている。これは、他の情報としてこの領域が使用されるとき 0 に近い値となるようにするためである。無論、プレゼンターで Maxwell モデルのダンパー軸力として図化する場合は、1000 倍して用いている。なお、塑性状態はダミーとして 0 を設定している。第 2 レコードは、ダンパー変位、ばね部分の変位、フィルター濾過後の制震装置速度、ダンパー速度を表す。上記の理由と同じに、ダンパー変位とダンパー速度は、各々、10000 分１、1000 分の１にしてセットしている。

２）の 3 次元せん断型弾塑性モデルでは、2 レコードを出力しているが、両者とも同じ情報である。塑性状態はダミーであり、4 つのデータは順次、軸力、 z 軸せん断力、 y 軸せん断力であり、最後はダミー情報で 0 としている。

３）の 3 次元軸力弾塑性モデルでは、2 レコード出力となっており、第 1 レコードは、塑性状態、軸力、 z 軸せん断力、 y 軸せん断力、最後はダミーであり、第 2 レコードは、軸力の替わりに軸力変位を 10000 分１にしてセットし、あとは第 1 レコードと同じである。

４）の 3 次元ブレースモデルでは、2 レコード出力となっており、第 1 レコードは、塑性状態、軸力、 z 軸せん断力、 y 軸せん断力、最後はダミーであり、第 2 レコードは、軸力の替わりに軸力変位を 10000 分１にしてセットし、あとは第 1 レコードと同じである。 ５）のモデル NO.5-10 では、2 レコード出力となっており、同一情報を出力する。塑性状態はダミーであり、続いて、軸力、z 軸せん断力、y 軸せん断力、最後はダミーである。

６）は、両端ピンの部材モデルであり、3 レコード出力となっている。第 1 と第 2 レコードは i 端、j 端の応力であり、第 3 レコードは部材中央の応力となっている。第 1、第 2 レコードは、標準と同じで、塑性状態、軸力、y 軸曲げモーメント、z 軸曲げモーメント、塑性関数であり、ここで塑性関数を計算して求めている。第 3 レコードは、部材中央の曲げモーメントで、最後は軸力の変位を 10000 分の１にセットして、出力している。 ７）はその他のモデルであり、一般的なファイバーモ

デル、アナロジーモデルの応力出力コードを表す。レコード数は、先に示した配列 myytype の値を用いる。

以上の仕様を元に、出力用コードを見てみよう。容易に理解できるはずである。

```
C     ―――――――――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_stress
C       ―――――――――――――――――――――――――――――――――――――――――――――
C     ●   部材両端と中央の応力を出力(ok)
C     ―――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Out_stress(Member,Element,E_model6_real,M_model11,
     *                      M_model12,M_model13,M_model15,M_model21,
     *                      M_model22,M_model31,M_model32,M_model33,
     *                      n_member,ifl,iflz,i_print,Out_section)
C     ―――――――――――――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / Member_s         / Member
      record / Element_s        / Element
      record / Out_section_s    / Out_section
      record / E_model6_real_s  / E_model6_real
      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
      dimension ifl(16),iflz(16)
      dimension Member(*),Element(*),E_model6_real(*)
      dimension M_model11(*),M_model12(*),M_model13(*)
      dimension M_model15(*),M_model21(*),M_model22(*)
      dimension M_model31(*),M_model32(*),M_model33(*)
      dimension mxtype(100),myytype(4)
      data mxtype/1,1,1,1,1,1,1,1,1,1,  1,3,1,3,1,1,3,3,3,1,
     3            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     5            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     7            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     9            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1/
      data myytype/2,4,3,5/
      real*4    v(6),vv(100)
C     ―――――――――――――――――――――――――――――――――――――――――――――――
c     i_print         :integer 出力制御変数 0:ファイル出力あり
c――――――――――――――――――――――――――――――――――――――――――――――★応力 5
      if(i_print.ne.0) return
       if(ifl(5).ne.0) then
       do i=1,n_member
       if(Member(i).element_type.eq.6) then
c――――――――――――――――――――――――――――――――――――――――――――――★Maxwell モデル
```

```
c     Maxwell モデルの出力は、次のような特殊な仕様である。
c     使用する場合は、気をつけること
c     ダンパー変位とダンパー速度を 0.001 倍して送り出す。
C     ------------------------------------------------------------------
      istat = 0                                 ! 現在ダミー　塑性状態
      ien= Member(i).n_model_type
      v(1)=Member(i).stress(1)                  ! 軸力
      v(3)=Member(i).stress(2)                  ! ｙ軸せん断力
      v(2)=Member(i).stress(3)                  ! ｚ軸せん断力
      v(4)=E_model6_real(ien).fn*0.001          ! ダンパー軸力
      write(iflz(5)) istat,(v(k),k=1,4)
      v(3)=E_model6_real(ien).uud               ! フィルター濾過後の制震装置速度
      v(2)=E_model6_real(ien).u2                ! ばね部分の変位
      v(1)=E_model6_real(ien).u1* 0.00001       ! ダンパー変位
      v(4)=E_model6_real(ien).u1d*0.001         ! ダンパー速度
      write(iflz(5)) istat,(v(k),k=1,4)

      elseif(Member(i).element_type.eq.2) then
c-----------------------------------------------★3 次元せん断弾塑性モデル
      istat = 0                           ! 現在ダミー　塑性状態
      v(1)=Member(i).stress(1)            ! 軸力
      v(3)=Member(i).stress(2)            ! ｙ軸せん断力
      v(2)=Member(i).stress(3)            ! ｚ軸せん断力
      v(4)=0.
      write(iflz(5)) istat,(v(k),k=1,4)
      write(iflz(5)) istat,(v(k),k=1,4)

      elseif(Member(i).element_type.eq.3) then
c-----------------------------------------------★3 次元軸力弾塑性モデル
      istat = Member(i).d_stat(3)         ! 塑性状態
      v(1)=Member(i).stress(1)            ! 軸力
      v(3)=Member(i).stress(2)            ! ｙ軸せん断力
      v(2)=Member(i).stress(3)            ! ｚ軸せん断力
      v(4)=0.
      write(iflz(5)) istat,(v(k),k=1,4)
      iee = Member(i).n_model_type
      v(1)=Member(i).stress(16)*0.00001         ! 軸力変位
      write(iflz(5)) istat,(v(k),k=1,4)

      elseif(Member(i).element_type.eq.4 ) then
c-----------------------------------------------★3 次元ブレースモデル
      istat = Member(i).d_stat(3)         ! 塑性状態
      v(1)=Member(i).stress(1)            ! 軸力
      v(3)=Member(i).stress(2)            ! ｙ軸せん断力
      v(2)=Member(i).stress(3)            ! ｚ軸せん断力
      v(4)=0.
      write(iflz(5)) istat,(v(k),k=1,4)
c     v(1)=Member(i).u_past*0.00001         ! 軸力変位    stress(8)=u_past
      v(1)=Member(i).stress(8)*0.00001
      write(iflz(5)) istat,(v(k),k=1,4)

      elseif(Member(i).element_type.ge.5.and.
     *   Member(i).element_type.le.10) then
c-----------------------------------------------★Model No.5-10
```

```
      istat = 0                          ! 現在ダミー　塑性状態
      v(1)=Member(i).stress(1)           ! 軸力
      v(3)=Member(i).stress(2)           ! ｙ軸せん断力
      v(2)=Member(i).stress(3)           ! ｚ軸せん断力
      v(4)=0.
      write(iflz(5)) istat,(v(k),k=1,4)
      write(iflz(5)) istat,(v(k),k=1,4)

      elseif(Member(i).element_type.eq.18.or.
     *       Member(i).element_type.eq.19) then
c-----------------------------------------------★Model No. 18,19
      i_t=Member(i).element_type
      im=mxtype(i_t)
      ns=myytype(im)
      ie = Member(i).nm_element
      immm= Member(i).n_model_type               ! モデルタイプ別番号
      do j=1,2
      istat = 0
      jj=6*(j-1)
      v(1)=Member(i).stress(jj+1)          ! 軸力
      v(2)=Member(i).stress(jj+5)          ! ｙ軸モーメント
c      v(3)=-Member(i).stress(jj+6)        ! ｚ軸モーメント
      v(3)=-Member(i).stress(jj+6)         ! ｚ軸モーメント
c      v(4)=fy(j,i)
      rrxx=0.                              ! 現在ダミー　塑性関数値
      if(Element(ie).ANP.ne.0.)
     *      rrxx=(Member(i).stress(jj+1)/Element(ie).ANP)**2
      rrx=0.
      if(Element(ie).AMPY.ne.0.)
     *      rrx=(Member(i).stress(jj+11)/Element(ie).AMPY)**2
      if(Element(ie).AMPZ.ne.0.)
     *      rrx=rrx+(Member(i).stress(jj+12)/Element(ie).AMPZ)**2
      v(4)=rrxx+Dsqrt(rrx)
      write(iflz(5)) istat,(v(k),k=1,4)
      enddo
      j=3
      istat = Member(i).d_stat(1)
      jj=6*(j-1)
      v(2)=Member(i).stress(jj+5)                ! ｙ軸モーメント
      v(3)=Member(i).stress(jj+6)                ! ｚ軸モーメント
      v(4)=Member(i).stress(jj+4)*0.00001        ! 軸力変位
      write(iflz(5)) istat,(v(k),k=1,4)

      else
c-----------------------------------------------★その他のモデル
      i_t=Member(i).element_type
      im=mxtype(i_t)
      ns=myytype(im)
      ie = Member(i).nm_element
      immm= Member(i).n_model_type               ! モデルタイプ別番号
      do j=1,2
      istat = Member(i).d_stat(j)
      jj=6*(j-1)
      v(1)=Member(i).stress(jj+1)          ! 軸力
```

```
      v(2)=Member(i).stress(jj+5)          ! ｙ軸モーメント
      v(3)=-Member(i).stress(jj+6)         ! ｚ軸モーメント
c       v(4)=fy(j,i)
      rrxx=0.                              ! 現在ダミー　塑性関数値
      if(Element(ie).ANP.ne.0.)
     *      rrxx=(Member(i).stress(jj+1)/Element(ie).ANP)**2
      rrx=0.
      if(Element(ie).AMPY.ne.0.)
     *      rrx=(Member(i).stress(jj+5)/Element(ie).AMPY)**2
      if(Element(ie).AMPZ.ne.0.)
     *      rrx=rrx+(Member(i).stress(jj+6)/Element(ie).AMPZ)**2
      v(4)=rrxx+Dsqrt(rrx)
      write(iflz(5)) istat,(v(k),k=1,4)
      enddo
      if(ns.gt.2) then
      do j=3,ns
      istat = Member(i).d_stat(j)
      jj=6*(j-1)
      v(1)=Member(i).stress(jj+1)          ! 軸力
      v(2)=Member(i).stress(jj+5)          ! ｙ軸モーメント
      v(3)=-Member(i).stress(jj+6)         ! ｚ軸モーメント
c     v(4)=fy(j,i)
      rrxx=0.                              ! 現在ダミー　塑性関数値
      if(Element(ie).ANP.ne.0.)
     *      rrxx=(Member(i).stress(jj+1)/Element(ie).ANP)**2
      rrx=0.
      if(Element(ie).AMPY.ne.0.)
     *      rrx=(Member(i).stress(jj+5)/Element(ie).AMPY)**2
      if(Element(ie).AMPZ.ne.0.)
     *      rrx=rrx+(Member(i).stress(jj+6)/Element(ie).AMPZ)**2
      v(4)=rrxx+Dsqrt(rrx)
      write(iflz(5)) istat,(v(k),k=1,4)
      enddo
      endif

      endif
      enddo

      endif
      return
      end
```

7.4.5 断面ファイバー応力ファイル

本節では、ファイバーの応力とひずみを出力するファイルの仕様とその出力サブルーチンについて解説する。このファイル出力に関するプログラムコードは、submain_dynamic_a()で以下のようにコールされる。

```
c-------------------------------------------------★部材応力を出力
      call Out_Fiber(Member,Element,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,
*                E_model_fiber,M_model_fiber,
*                n_member,ifl,iflz,i_print,Out_section)
```

このサブルーチンでは、各部材の両端及び中央のファイバー応力とひずみを出力する。部材両端以外のファイバー応力の出力は、各部材モデルによって定義されている。その定義は、配列 mxtype と myytype で行われている。mxtype の配列番号は部材モデル番号となっており、その値は、myytype の配列番号を指す。配列 myytype の値は、ひとつの部材に対する出力レコード数を表す。現在は、2 レコードと 3 レコードが用いられており、前者はｉ端、ｊ端の応力を、後者はｉ端、ｊ端の応力に中央のファイバー応力を加えて、単精度のバイナリーデータとして出力する。当然、このデータを読み込むことになる動的プレゼンターなどでは、この仕様と同じでなくてはならない。この仕様を変更する場合は、他のサブシステムの該当する部分も変更しなければならないので、特に注意されたい。

標準的なレコードの出力は、

```
write(iflz(6)) (v(k),k=1,nm_div)
write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)
```

で表され、nm_div は断面におけるファイバー分割数を表す。第 1 レコードは、ファイバーの応力を表し、第 2 レコードの vf は断面の図芯位置でのひずみを表し、配列内では 1 が軸方向ひずみ、2 がｙ軸に関する曲げひずみ、3 がｚ軸に関する曲げひずみである。次の配列 V は各ファイバーの軸方向ひずみである。

サブルーチン Out_Fiber()のプログラムコードは、部材モデル毎に記述されているが、その内容はほとんど同じである。従って、ここでは、最初の 2 つのモデルについて記述し、説明する。全てのコードは、付録を参照されたい。

プログラムコード内で、Out_section.n_member は、ファイバーデータを出力する部材数であり、また、Out_section.no_member は、出力する部材番号がセットされている。この部材番号を頼りに、部材のモデル番号、モデルタイプ別要素番号 imm、モデルタイプ別部材番号 immm が取り出される。モデル番号によって、モデルが 11 番であるとすると、E_model11(imm)、M_model11(immm)の構造体にアクセスすることが可能となる。ここで、断面内のファイバー分割数とそのファイバーの最初

の番号を取り出す。この値を用いて、先に計算されてセットされているファイバーの応力 M_model_fiber(k+nnm).d_stress_x を単精度の配列に取り出し、所定のファイルに出力する。同様に、ファイバーのひずみ M_model_fiber(k+nnm).d_eps_x を取り出す。また、断面の軸方向ひずみとｙ、ｚ軸に関する曲げひずみを取り出し、単精度配列 vf にセットする。このエレメントの断面図芯位置でのひずみとファイバーの軸方向ひずみが第 2 レコードとして出力される。

モデル 11 では、部材両端にファイバー断面が存在するため、上記のファイバー応力とひずみがさらに 2 レコード出力されることになる。また、次にプログラムコードを見ると分かるように、モデル 12 では、部材両端と中央にファイバー断面が存在するため、ｉ端、ｊ端及び中央の上記データが出力されることになる。

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Out_Fiber
C       ----------------------------------------------------------------
C     ●   部材の断面応力を出力(ok)
C     ------------------------------------------------------------------
      subroutine Out_Fiber(Member,Element,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,
     *             E_model_fiber,M_model_fiber,
     *             n_member,ifl,iflz,i_print,Out_section)
       implicit real*8(A-H,O-Z)
       include "submain.h"
       include "submainx.h"
       record / Member_s       / Member
       record / Element_s      / Element
       record / Out_section_s  / Out_section
      record / M_model11_s    / M_model11
      record / E_model11_s    / E_model11
      record / M_model12_s    / M_model12
      record / E_model12_s    / E_model12
      record / M_model13_s    / M_model13
      record / E_model13_s    / E_model13
      record / M_model15_s    / M_model15
      record / E_model15_s    / E_model15
      record / M_model21_s    / M_model21
      record / E_model21_s    / E_model21
      record / M_model22_s    / M_model22
      record / E_model22_s    / E_model22
      record / M_model31_s    / M_model31
      record / E_model31_s    / E_model31
      record / M_model32_s    / M_model32
      record / E_model32_s    / E_model32
```

```
      record / M_model33_s    / M_model33
      record / E_model33_s    / E_model33
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
       dimension ifl(16),iflz(16)
       dimension Member(*),Element(*),M_model11(*),E_model11(*),
     *          M_model12(*),E_model12(*),M_model13(*),E_model13(*),
     *          M_model15(*),E_model15(*)
      dimension M_model21(*),E_model21(*),M_model22(*),E_model22(*)
      dimension M_model31(*),E_model31(*),M_model32(*),E_model32(*)
      dimension M_model33(*),E_model33(*)
      dimension mxtype(100),myytype(4)
       data mxtype/1,1,1,1,1,1,1,1,1,1,  1,3,1,3,1,1,3,3,3,1,
     3            1,3,1,1,1,1,1,1,1,1,  1,3,1,1,1,1,1,1,1,1,
     5            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     7            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     9            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1/
      data myytype/2,4,3,5/
      real*4    v(100),vf(3)

c     i_print         :integer 出力制御変数 0:ファイル出力あり
c-------------------------------------------------★応力 5
      if(i_print.ne.0) return
       if(ifl(6).eq.0) return
c-------------------------------------------------★断面応力出力

      if(Out_section.n_member.eq.0) return

      do j=1,Out_section.n_member
       i=Out_section.no_member(j)               ! 出力部材番号
c-------------------------------------------------★断面応力
       ie = Member(i).nm_element                ! 要素番号
      imm= Element(ie).n_element                ! モデルタイプ別要素番号
       immm= Member(i).n_model_type             ! モデルタイプ別部材番号
       iet  = Member(i).element_type            ! モデル番号
      if(iet.eq.11) then
c-------------------------------------★モデル１１
       nm_div=E_model11(imm).n_section_1            ! 断面内のファイバー要素分割数
       nnm=M_model11(immm).nm_section_1 - 1         ! ファイバー要素の最初の番号 - 1
       do k=1,nm_div
       v(k)=M_model_fiber(k+nnm).d_stress_x
      enddo
      write(iflz(6)) (v(k),k=1,nm_div)
       do k=1,nm_div
      v(k)=M_model_fiber(k+nnm).d_eps_x
       enddo
       vf(1) = M_model11(immm).d_epsi_x_1           !  軸方向歪
      vf(2) = M_model11(immm).d_epsi_y_1            !  y軸に関する曲げ歪
      vf(3) = M_model11(immm).d_epsi_z_1            !  z軸に関する曲げ歪

      write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)

      nm_div=E_model11(imm).n_section_2
```

```
        nnm=M_model11(immm).nm_section_2 - 1
        do k=1,nm_div
        v(k)=M_model_fiber(k+nnm).d_stress_x
       enddo
        write(iflz(6)) (v(k),k=1,nm_div)
        do k=1,nm_div
       v(k)=M_model_fiber(k+nnm).d_eps_x
        enddo
        vf(1) = M_model11(immm).d_epsi_x_2            !  軸方向歪
       vf(2) = M_model11(immm).d_epsi_y_2             !  y軸に関する曲げ歪
       vf(3) = M_model11(immm).d_epsi_z_2             !  z軸に関する曲げ歪

       write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)

        elseif(iet.eq.12) then
c-------------------------------------★モデル１２
        nm_div=E_model12(imm).n_section_1
        nnm=M_model12(immm).nm_section_1 - 1
        do k=1,nm_div
        v(k)=M_model_fiber(k+nnm).d_stress_x
       enddo
       write(iflz(6)) (v(k),k=1,nm_div)
        do k=1,nm_div
       v(k)=M_model_fiber(k+nnm).d_eps_x
        enddo
        vf(1) = M_model12(immm).d_epsi_x_1            !  軸方向歪
       vf(2) = M_model12(immm).d_epsi_y_1             !  y軸に関する曲げ歪
       vf(3) = M_model12(immm).d_epsi_z_1             !  z軸に関する曲げ歪

       write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)

        nm_div=E_model12(imm).n_section_2
        nnm=M_model12(immm).nm_section_2 - 1
        do k=1,nm_div
        v(k)=M_model_fiber(k+nnm).d_stress_x
       enddo
       write(iflz(6)) (v(k),k=1,nm_div)
        do k=1,nm_div
       v(k)=M_model_fiber(k+nnm).d_eps_x
        enddo
        vf(1) = M_model12(immm).d_epsi_x_2            !  軸方向歪
       vf(2) = M_model12(immm).d_epsi_y_2             !  y軸に関する曲げ歪
       vf(3) = M_model12(immm).d_epsi_z_2             !  z軸に関する曲げ歪
       write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)
        nm_div=E_model12(imm).n_section_c
        nnm=M_model12(immm).nm_section_c - 1
        do k=1,nm_div
        v(k)=M_model_fiber(k+nnm).d_stress_x
       enddo
       write(iflz(6)) (v(k),k=1,nm_div)
        do k=1,nm_div
       v(k)=M_model_fiber(k+nnm).d_eps_x
        enddo
        vf(1) = M_model12(immm).d_epsi_x_c            !  軸方向歪
```

```
      vf(2) = M_model12(immm).d_epsi_y_c          !  y軸に関する曲げ歪
      vf(3) = M_model12(immm).d_epsi_z_c          !  z軸に関する曲げ歪
      write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)
      elseif(iet.eq.15) then
c-------------------------------------★モデル１５
      elseif(iet.eq.13) then
c-------------------------------------★モデル２１
       elseif(iet.eq.14) then
c-------------------------------------★モデル２２
      elseif(iet.eq.16) then
c-------------------------------------★モデル３１
       elseif(iet.eq.17) then
c-------------------------------------★モデル３２
       elseif(iet.eq.18.or.iet.eq.19) then
c-------------------------------------★モデル１３
       endif
c-----------------------------------------------★断面応力出力終わり
      enddo
       return
       end
```

7.4.6 モード変位と固有周期ファイル

本節では、固有値解析で求めた固有値と固有ベクトルを、出力仕様に合わせて振動数、固有周期、振動モード、刺激係数をファイルに出力する。SPACE では、固有値解析においてサブスペース法とヤコビ法を使い分けており、各々予備計算を含めて解析方法が異なっている。ただし、出力ファイルの仕様は同じである。ここで出力するファイルは次の２つ、

```
7   'mode_d'        42          17      F     !  ﾓｰﾄﾞ変位
8   'omeg_d'        43          18      F     !  固有周期、振動数、減衰定数、刺激係数
```

であり、どちらも ASKII ファイルである。固有周期や刺激係数を保存するファイル omeg_d は、第 7.3.8 節で示されている。ここでは、ファイル mode_d について説明する。ファイルは ASKII ファイルであり、単純な構成となっている。計算モード次数がｎ次である場合、2*n 個のグループで出力されており、第１段階、第２段階の順に、同じ仕様で連続して出力されている。ここでは、第１階の出力仕様を用いて説明する。第１行目はコメントであり、モード番号が示されている。２行目以降は、同じ仕様で、左から、節点番号、次に当該の節点に対する６個の自由度に関する全体座標系におけるモード変位（$u,v,w,\theta_x,\theta_y,\theta_z$）である。

```
 ***** MODE  1 *****
1   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
2   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
3   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
4   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
5   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
6   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
7   0.00000   0.05368   0.00351  -0.00015   0.00000   0.00000
8   0.00000   0.05414  -0.00873   0.00005   0.00000   0.00000
```

さらに、プログラムの中で、出力ユニット番号が 76 のファイルは、固有値解析の結果を出力する EOUTPUT である。また、サブスペース法による結果の出力は、Out_Eigen()、ヤコビ法では、Out_Eigen_J()サブルーチンである。ヤコビ法では一般固有値問題から標準固有値問題への変換を行っており、得られた固有ベクトルは、Trans_Eigen_V()サブルーチンを用いて変換している。これらのサブルーチンは、固有値問題に関する主サブルーチンである submain_dynamic_b()から以下のようにコールされている。

```
      c―――――――――――――――――――――――――★結果の出力
      call Out_Eigen(n_step,n_unknown,Parameter_C,Eigen_d,
     *          Point,Newmark_P,Eigen_Value,Eigen_Vector,Omega,
```

```
     *           gskymm,max_h_sky,rot_local,disp_point_m,
     *           tw_a,tw_b,ifl(7),iflz(7),ifl(8),iflz(8))

c----------★固有ベクトルの変換
      call Trans_Eigen_V(Eigen_Vector,n_unknown,gskymm,vec_w2,ier)

c----------★結果の出力
      call Out_Eigen_J(n_step,n_unknown,Parameter_C,Eigen_d,
     *           Point,Newmark_P,Eigen_Value,Eigen_Vector,Omega,
     *           gskymm,rot_local,disp_point_m,
     *           vec_w3,vec_w4,ifl(7),iflz(7),ifl(8),iflz(8))
```

このサブルーチンでは出力仕様に従って、少し計算を行っており、その部分について解説する。ただし、Out_Eigen_J()のプログラムコードは、Out_Eigen()とほとんど同じであり、ここでは、説明、記述共に省略する。サブルーチン Out_Eigen_J()を見たい方は付録を参照されたい。それでは、以下の説明を参照しながら、プログラムを順次理解していこう。

１．モード変位を保存する配列 disp_point_m をゼロクリアする。
２．振動モードの最大値を検索し、最大値を１とする規準化を行う。ここで、n_mode は解析モード数であり、振動モードは配列 Eigen_Vector に保存されている。
３．振動モードの最大値を１にセットする。
４．振動数 Omega を固有値 Eigen_Value から計算する。
５．レーリー減衰等に関する係数計算、ここでは、SPACE の仕様に従って第１と第２の２段階で係数を求める。
６．現在の仕様では、1：質量比例型、2：剛性比例型、3：レーリー型の３種が選択可能であり、各々に従って係数を計算する。
７．ここでは、振動数、固有周期等のタイトルをファイル'omeg_d'に出力する。
８．次に、刺激係数を求める。この計算は多少長いプログラムコードが必要となるので、注意して読まれたい。
  8.1 これ以後の一連の計算を解析モード数繰り返す。
  8.2 固有ベクトルをワーク領域である配列 disp に保存する。
  8.3 ３方向の刺激係数をゼロにセットする。
  8.4 サブルーチン Multm()を用いて、質量行列と固有ベクトルの行列積を求め、得られたベクトルを配列 w にセットする。
  8.5 上記の配列 w と固有ベクトルのスカラー積を取り、刺激係数の分母 t_dmd を計算する。

8.6 刺激係数の分子を計算するために、上記の配列 w について方向別に和をとる。ただし、節点で釣合座標系（局所座標系）を用いている場合は、全体座標系に変換する必要があり、その処理を行った後、3 方向の和をとる。

8.7 最後に、分母である t_dmd がゼロでない場合は、3 方向の刺激係数を t_dmd で割ることによって、各々の刺激係数を求める。

９．得られた刺激係数をチェックするために、各固有ベクトルに対応する刺激係数を掛け、各節点の 3 方向毎に、全モードについて和を取る。無論、固有ベクトルが局所座標系を使用している場合は、全体座標系に座標変換する。全モードについてこの計算を行えば、その結果は各節点の 3 方向毎に全て１となるはずであるが、サブスペース法では、求められるモード数に制限があるため、全モードで処理することが不可能であり、１とはならない。ただし１に近い値が得られることになる。また、ヤコビ法では、全モードで求めるため、この値は１となる。この値は固有値解析や関連する処理が正確であることの検証となる。

10. 各振動数における減衰定数を、レーリー減衰などで定義した係数から求める。また、固有周期を計算する。

11. 得られた情報、つまりモード番号毎に、振動数、固有周期、減衰定数、3 方向の刺激係数をファイル'omeg_d'に出力する。

12. 振動モードをファイル'mode_d'に出力する。このとき、節点の座標が局所座標を使用している場合は、全体座標系に変換する。

13. 最後に、先に計算した刺激係数の検証結果を EOUTOUT ファイルに出力する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_Eigen
C     ―――――――――――――――――――――――――――――――――
C     ●    振動数と固有ベクトルの計算と出力
C     ―――――――――――――――――――――――――――――――――
      subroutine Out_Eigen(n_step,n_unknown,Parameter_C,Eigen_d,
     *           Point,Newmark_P,Eigen_Value,Eigen_Vector,Omega,
     *           gskymm,max_h_sky,rot_local,disp_point_m,
     *           disp,w,ifl1,iflz1,ifl2,iflz2)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record /eigen_d_s    / Eigen_d
      record /newmark_s    / Newmark_P
      record /Parameter_s  / Parameter_C
      record /point_s      / Point
      dimension  Point(*)
```

```
      dimension  Eigen_Value(*),Eigen_Vector(n_unknown,*)
      dimension  Omega(*),gskymm(*),max_h_sky(0:*),rot_local(3,3,*)
      dimension  disp(*),w(*),disp_point_m(3,3,*)      !ワーク領域
      dimension xx(3),yy(3),Shigeki(3),tdisp(6)

      n_point=Parameter_C.n_point                      ! 節点数
      n_local_coord = Parameter_C.n_local_coord
      n_mode = Eigen_d.n_modes                         ! モード数
      load_mass=Eigen_d.load_mass
      pi = 3.14159265D0
c―――――――――――――――――――――――★モード変位領域をゼロクリア
      do i=1,n_point                                                  ! 1
      do j=1,3
      do k=1,3
      disp_point_m(k,j,i)=0.
      enddo
      enddo
      enddo
c―――――――――――――――――――――――★振動モードの最大値検索
      do 10 imode=1,n_mode                                            ! 2
      xmax = 0.D0
      do i=1,n_unknown
      if (dabs(Eigen_Vector(i,imode)).gt.xmax)
     &           xmax = dabs(Eigen_Vector(i,imode))
      enddo
c―――――――――――――――――――――――★振動モードの最大を１にセット
      if(xmax.ne.0.) then                                             ! 3
      do i=1,n_unknown
      Eigen_Vector(i,imode) = Eigen_Vector(i,imode)/xmax
      enddo
      endif
c―――――――――――――――――――――――★振動数計算
      Omega(imode) = dsqrt(Eigen_Value(imode))                        ! 4
   10 enddo
c―――――――――――――――――――――――★レーリー減衰等に関する係数計算
      A0 = 0.0D0                                                      ! 5
      A1 = 0.0D0
      n_damp_1=Newmark_P.n_damp_1         ! 第１指定モード番号
      n_damp_2=Newmark_P.n_damp_2         ! 第２指定モード番号
      n_damp_type=Newmark_P.n_damp_type
c―――――――――――――――――――――――★ステップで減衰定数を変更
      if(n_step.eq.1) then
      h1=Newmark_P.alf1_1
      h2=Newmark_P.alf1_2
      else
      h1=Newmark_P.alf2_1
      h2=Newmark_P.alf2_2
      endif
c―――――――――――――――――――――――★質量比例型
      if (n_damp_type.eq.1 ) then                                     ! 6
      A0 = 2.0*h1*Omega(n_damp_1)
c―――――――――――――――――――――――★剛性比例型
      elseif (n_damp_type.EQ.2 ) then
      A1 = 2.0*h1/Omega(n_damp_1)
```

```
c――――――――――――――――――――――★レーリー型
      elseif (n_damp_type.EQ.3 ) then
      AA = Omega(n_damp_2)*Omega(n_damp_2)-
     *     Omega(n_damp_1)*Omega(n_damp_1)
        if ( AA.EQ.0.0D0 ) then
          write(76,'(''**** AA IS 0 !! ****  '')')
        else
          A0 = 2.0*Omega(n_damp_1)*Omega(n_damp_2)
     *           *(h1*Omega(n_damp_2)-h2*Omega(n_damp_1))/AA
          A1 = 2.0*(h2*Omega(n_damp_2)-h1*Omega(n_damp_1))/AA
        endif
      endif
c――――――――――――――――――――――――――★振動数、固有周期等のタイトル出力
      if (ifl2.eq.1 ) write(iflz2,535) n_mode                                  ! 7
  535 format(I6/2X,'MODE       FREQ.        PERIOD           DAMPING',
     * '         BETA(x)          BETA(y)',
     *          '         BETA(z)')
  536 format(//2X,'MODE       FREQ.        PERIOD           DAMPING',
     * '         BETA(x)          BETA(y)',
     *          '         BETA(z)')
c――――――――――――――――――――――――――★刺激係数、固有周期等計算          ! 8
      do 100 imode=1,n_mode                                                    ! 8.1
      do i=1,n_unknown
      disp(i) = Eigen_Vector(i,imode)                                          ! 8.2
      enddo
c――――――――――――――――――――――――――★刺激係数のゼロセット
      do j=1,3
      Shigeki(j)=0.                                                           ! 8.3
      enddo
      call Multm(load_mass,gskymm,disp,w,max_h_sky,n_unknown)                  ! 8.4
      t_dmd = 0.0
      do i = 1, n_unknown
      t_dmd = t_dmd + disp(i)*w(i)                                             ! 8.5
      enddo
c――――――――――――――――――――――――――★刺激係数
      if(n_local_coord.eq.0) then                                              ! 8.6
         do i= 1, n_point                                  ! 局所座標がない場合の処理
         do j=1,3
          irest =Point(i).irest(j)
         if(irest.gt.0) Shigeki(j) = w(irest) + Shigeki(j)
         enddo
          enddo
      else                                               ! 局所座標がある場合の処理
        do i= 1, n_point
         local_coord=Point(i).local_coord
         if(local_coord.eq.0) then
           do j=1,3
           irest =Point(i).irest(j)
          if(irest.gt.0)  Shigeki(j) = w(irest) + Shigeki(j)
           enddo
         else
           do j=1,3
           xx(j)=0.
           irest =Point(i).irest(j)
```

```
            if(irest.gt.0)  xx(j) = w(irest)
            enddo
             call trans_VT8(xx,yy,rot_local(1,1,local_coord))
            do j= 1, 3
            Shigeki(j) = yy(j) + Shigeki(j)
            enddo
           endif
          enddo

        endif
        if(t_dmd.ne.0.)then
        do j=1,3
        Shigeki(j)=Shigeki(j)/t_dmd                                    ! 8.7
        enddo
        endif
c――――――――――――――――――――――――★刺激係数のチェック
        if(n_local_coord.eq.0) then                                    ! 9
           do i= 1, n_point
           do j=1,3
           yy(j)=0.
            irest =Point(i).irest(j)
           if(irest.gt.0) yy(j) = disp(irest)
           enddo
        do j=1,3
        do k=1,3
        disp_point_m(k,j,i)=disp_point_m(k,j,i)+Shigeki(j)*yy(k)
        enddo
        enddo
           enddo
        else
          do i= 1, n_point
           local_coord=Point(i).local_coord
           if(local_coord.eq.0) then
             do j=1,3
              yy(j)=0.
             irest =Point(i).irest(j)
             if(irest.gt.0) yy(j) = disp(irest)
             enddo
           else
             do j=1,3
             xx(j)=0.
             irest =Point(i).irest(j)
             if(irest.gt.0)  xx(j) = disp(irest)
             enddo
             call trans_VT8(xx,yy,rot_local(1,1,local_coord))
           endif

        do j=1,3
        do k=1,3
        disp_point_m(k,j,i)=disp_point_m(k,j,i)+Shigeki(j)*yy(k)
        enddo
        enddo

        enddo
```

```
      endif
c―――――――――――――――――――――――――★振動数、固有周期等の出力
      hh0  = 0.0D0                                                    ! 10
      if (ifl2.eq.1 ) then
      if (n_damp_type.eq.1.and.Omega(imode).ne.0.0D0 )
     *                 hh0 = A0/(2.0*Omega(imode))
      if (n_damp_type.eq.2 ) hh0 = A1*Omega(imode)/2.0
      if (n_damp_type.eq.3.and.Omega(imode).ne.0.0D0 )
     *                 hh0 = (A0/Omega(imode)+A1*Omega(imode))/2.0
      T_period = 0.0D0
      if (Omega(imode).ne.0.0D0 ) T_period = 2.0*PI/Omega(imode)
      write(76,536)
      write(76,'(I5,6e16.8)') imode,Omega(imode),T_period,
     *                        hh0,(Shigeki(j),J=1,3)
      write(iflz2,'(I5,6e16.8)') imode,Omega(imode),T_period,
     *                        hh0,(Shigeki(j),J=1,3)                  ! 11
      endif
      if ( ifl1.EQ.1 ) then
c―――――――――――――――――――――――――★振動モードの出力
      write(iflz1,1000) imode                                         ! 12
 1000 FORMAT(5X,'***** MODE',I3,' *****')
  200  format(I5,6F10.5)
 1001 FORMAT(5X,'***** MODE',I3,' *****'/
     *'N_point    u         v         w  ',
     *          '    theta_x   theta_y   theta_z')
       write(76,1001) imode
      do i=1,n_unknown
      disp(i) = Eigen_Vector(i,imode)
      enddo
      if(n_local_coord.eq.0) then
         do i= 1, n_point
         do j=1,6
          tdisp(j)=0.
          irest =Point(i).irest(j)
         if(irest.gt.0) tdisp(j) = disp(irest)
         enddo
      write(iflz1,200) i,(tdisp(j),j=1,6)
      write(76,200) i,(tdisp(j),j=1,6)
          enddo
      else
        do i= 1, n_point
         local_coord=Point(i).local_coord
         if(local_coord.eq.0) then
           do j=1,6
           tdisp(j)=0.
           irest =Point(i).irest(j)
          if(irest.gt.0)  tdisp(j) = disp(irest)
           enddo
            write(iflz1,200) i,(tdisp(j),j=1,6)
            write(76,200) i,(tdisp(j),j=1,6)
        else
           do j=1,6
           tdisp(j)=0.
           irest =Point(i).irest(j)
```

```
            if(irest.gt.0)  tdisp(j) = disp(irest)
            enddo
            call trans_VT8(tdisp(1),xx,rot_local(1,1,local_coord))
            call trans_VT8(tdisp(4),yy,rot_local(1,1,local_coord))
              write(iflz1,200) i,(xx(j),j=1,3),(yy(j),j=1,3)
              write(76,200) i,(xx(j),j=1,3),(yy(j),j=1,3)
           endif
          enddo
        endif
        endif
  100 continue
c―――――――――――――――――――――――――★刺激係数のチェック出力
       write(76,'(//a)') ' 刺激係数と振動モードチェック'                    ! 13
       write(76,1002)
 1002 format('N_point','               x_direction',
     *                '               y_direction',
     *                '              ud_direction'/
     *       '             u         v         w   ',
     *              '      u         v         w   ',
     *              '      u         v         w')
       do i= 1, n_point
       write(76,'(i6,10f10.4)') i,((disp_point_m(k,j,i),k=1,3),j=1,3)
       enddo
       return
       end
```

7.4.7 最大変位、加速度、速度ファイル

本節では、計算過程で得た最大変位、最大速度、最大加速度を出力するファイルの仕様とそのプログラムコードについて説明する。上記の最大値は、増分時間毎の計算過程で常にチェックしており、最大値を入れ替えている。しかし、変位や速度、加速度のファイルへの出力は、使用者が設定した間隔で成されており、プレゼンターなどでこの時刻歴ファイルを用いて最大値を求めた場合と、ここでチェックしている最大値とは多少異なる場合がある。

最大値のチェックは、以下のサブルーチンコールで行われ、そのサブルーチンの内容は、後で説明する。

```
c―――――――――――――――――――――――――――――――★変位、速度、加速度の最大値セット(ok)
       call Get_max_disp(Max_disp,Parameter_C.n_point,Point,
      *      past_disp_point, past_vel_point, past_acc_point,
      *      d_max_v,id_max_v,vacc)
```

動的解析が全て終了した後、最大値を出力するサブルーチンコールが行われ、得られた変位、速度、加速度の最大値が節点ごとに ASKII ファイルとして出力される。このファイルの一部を以下に示す。このファイルは、同じ仕様で４つのグループに分けられており、相対加速

度、絶対加速度、速度、変位の順に出力されている。その仕様は、以下に示すように、最初の１行がコメント行であり、２行目以降は、左よりNODE:、節点番号、次に続く３個が負方向の３方向最大値、次の３つが正方向の最大値である。以下の例は全体座標系における x、y、z 方向の相対加速度を示す。

```
     ***** MAXIMUM RELATION ACCELERATION *****
NODE:  1  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  2  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  3  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  4  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  5  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  6  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
NODE:  7  0.4359E+03  0.0000E+00  0.1074E+02 -0.4708E+03  0.0000E+00 -0.1590E+02
NODE:  8  0.4394E+03  0.0000E+00  0.1434E+03 -0.4833E+03  0.0000E+00 -0.1437E+03
NODE:  9  0.4424E+03  0.0000E+00  0.7903E+02 -0.4865E+03  0.0000E+00 -0.7530E+02
NODE: 10  0.4442E+03  0.0000E+00  0.1544E+02 -0.4879E+03  0.0000E+00 -0.1925E+02
```

これらのサブルーチンは、いずれも submain_dynamic_a()から、以下のように呼ばれている。

```
c―――――――――――――――――――――――――――――★最大変位、速度、加速度の出力
      call Out_max_disp(Max_disp,Parameter_C.n_point,
     *                  Point,ifl(12),iflz(12))
```

上記２つのサブルーチンを具体的に示す。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_max_disp
C       ――――――――――――――――――――――――――――――――――――
C     ●   最大変位最大変位、最大加速度を求める(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Get_max_disp(Max_disp,n_point,Point,
     *      past_disp_point, past_vel_point, past_acc_point,
     *      d_max_v,id_max_v,vacc)
C     ――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / point_s  / Point
     record / max_disp_s    / Max_disp
     dimension Max_disp(*),Point(*),vacc(3)
     dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
C     ――――――――――――――――――――――――――――――――――――
c     Max_disp        :structure 最大値
c     n_point         :integer   節点数
c     Point           :structure
c     past_disp_point :real*8 計算結果の変位
c     past_vel_point  :real*8 計算結果の速度
c     past_acc_point  :real*8 計算結果の加速度
```

```
c     vacc(3)         :real*8 地震加速度
C     ―――――――――――――――――――――――――――――――――――――
       d_max_v=0.
      do i=1,n_point                                                  ! 1
      do j=1,3
      ires= Point(i).irest(j)
      if(ires.ne.0) then                                              ! 2
      aa = past_disp_point(ires)
      if(abs(aa).gt.d_max_v)then
      d_max_v=abs(aa)
      id_max_v=i
      endif
      if(Max_disp(i).disp_point(j).lt.aa) Max_disp(i).disp_point(j)=aa  ! 3
      if(Max_disp(i).disp_point(j+3).gt.aa) Max_disp(i).disp_point(j+3)=aa  ! 4
      aa = past_vel_point(ires)
      if(Max_disp(i).vel_point(j).lt.aa) Max_disp(i).vel_point(j)=aa   ! 5
      if(Max_disp(i).vel_point(j+3).gt.aa) Max_disp(i).vel_point(j+3)=aa
      aa = past_acc_point(ires)
      if(Max_disp(i).acc_point(j).lt.aa) Max_disp(i).acc_point(j)=aa   ! 6
      if(Max_disp(i).acc_point(j+3).gt.aa) Max_disp(i).acc_point(j+3)=aa
      aa = past_acc_point(ires)+vacc(j)
      if(Max_disp(i).ab_acc_point(j).lt.aa)                           ! 7
     *     Max_disp(i).ab_acc_point(j)=aa
      if(Max_disp(i).ab_acc_point(j+3).gt.aa)
     *     Max_disp(i).ab_acc_point(j+3)=aa
      else
      aa = vacc(j)
      if(Max_disp(i).ab_acc_point(j).lt.aa)                           ! 8
     *     Max_disp(i).ab_acc_point(j)=aa
      if(Max_disp(i).ab_acc_point(j+3).gt.aa)                         ! 9
     *     Max_disp(i).ab_acc_point(j+3)=aa
    end if
    end do
    end do
    return
    end
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_max_disp
C       ―――――――――――――――――――――――――――――――――――――
C     ●   最大変位、最大速度、最大加速度を出力する(ok)
C     ―――――――――――――――――――――――――――――――――――――
      subroutine Out_max_disp(Max_disp,n_point,Point,
     *                        ifl,iflz)
      implicit real*8(A-H,O-Z)
    include "submain.h"
    record / point_s  / Point
    record / max_disp_s    / Max_disp
    dimension Max_disp(*),Point(*)
C     ―――――――――――――――――――――――――――――――――――――
c     Max_disp        :structure 最大値
c     n_point         :integer   節点数
c     Point           :structure
C     ―――――――――――――――――――――――――――――――――――――
      if(ifl.ne.1) return                                             ! 10
```

```
      write(iflz,10)                                                    ! 11
   10 format(5X,'***** MAXIMUM RELATION ACCELERATION *****')
      do i=1,n_point
      write(iflz,25) I,(Max_disp(i).acc_point(j),j=1,6)                 ! 12
      enddo
   25 format('NODE:',I3,6E12.4)
C
      write(iflz,15)                                                    ! 13
   15 format(5X,'***** MAXIMUM ABSOLUTE ACCELERATION *****')
      do i=1,n_point
      write(iflz,27) I,(Max_disp(i).ab_acc_point(j),j=1,6)              ! 14
      enddo
   27 format('NODE:',I3,6E12.4)
C
      write(iflz,30)                                                    ! 15
   30 format(5X,'***** MAXIMUM VELOCITY *****')
      do i=1,n_point
      write(iflz,45) I,(Max_disp(i).vel_point(j),j=1,6)                 ! 16
      enddo
   45 format('NODE:',I3,6E12.4)
C
      write(iflz,50)                                                    ! 17
   50 format(5X,'***** MAXIMUM DISPLACEMENT *****')
      do i=1,n_point
      write(iflz,65) I,(Max_disp(i).disp_point(j),j=1,6)                ! 18
      enddo
   65 format('NODE:',I3,6E12.4)
C
    return
    end
```

1. この時刻における節点全体の最大変位を求めるために、最大値を入れる変数 d_max_y をゼロクリアする。
2. その節点における３方向変位の拘束状態を調べ、拘束なしであれば以降の処理を行う。最大変位とその位置である節点番号 id_max_y を調査する。
3. ここからは時刻歴における各節点の最大値をチェックする。まず、変位について行う。負の方向の最大変位をチェックし、最大値ならば値を変更する。
4. 正の方向の最大変位をチェックし、最大値ならば値を変更する。
5. 負の方向の最大速度をチェックし、最大値ならば値を変更する。同じく、正の方向の最大速度をチェックし、最大値ならば値を変更する。
6. 負の方向の最大相対加速度をチェックし、最大値ならば値を変更する。同じく、正の方向の最大相対加速度をチェックし、最大値ならば値を変更する。

7. 負の方向の最大絶対加速度をチェックし、最大値ならば値を変更する。同じく、正の方向の最大絶対加速度をチェックし、最大値ならば値を変更する。次に、その節点の相対加速度に３方向別に地震加速度を加え、絶対加速度を計算する。
8. 節点拘束がある場合は、節点に働く負方向地震加速度をチェックし、最大値ならば値を変更する。
9. 同じく、正方向地震加速度をチェックし、最大値ならば値を変更する。
10. 次に、サブルーチン Out_max_disp()について説明する。まず、出力要求がない場合はこのサブルーチンから直ちに戻る。
11. 最初に、相対加速度の最大値を出力する。出力ユニット番号は iflz である。ここでは、相対加速度であることのコメント行を出力する。
12. 全節点について、最大相対加速度を仕様に従って出力する。最初は節点番号、次の３つが負方向の最大値、最後の３つが正方向の最大値である。
13. ここでは、絶対加速度であることのコメント行を出力する。
14. 全節点について、最大絶対加速度を仕様に従って出力する。最初は節点番号、次の３つが負方向の最大値、最後の３つが正方向の最大値である。
15. 速度であることのコメント行を出力する。
16. 全節点について、最大速度を仕様に従って出力する。最初は節点番号、次の３つが負方向の最大値、最後の３つが正方向の最大値である。
17. 変位であることのコメント行を出力する。
18. 全節点について、最大変位を仕様に従って出力する。最初は節点番号、次の３つが負方向の最大値、最後の３つが正方向の最大値である。

7.4.8 最大応力ファイル

本節では、計算過程で得た最大応力を出力するファイルの仕様とそのプログラムコードについて説明する。上記の最大値は、増分時間毎の計算過程で常にチェックしており、最大値を入れ替えている。しかし、応力のファイルへの出力は、使用者が設定した間隔で成されており、プレゼンターなどでこの時刻歴ファイルを用いて最大値を求めた場合と、ここでチェックしている最大値とは多少異なる場合がある。

最大値のチェックは、最初に示したサブルーチンコールで行われ、その内容は、後で説明する。また、動的解析が全て終了した後、次に示すサブルーチンコールが行われ、得られた応力の最大値が節点ごとにASKII ファイルとして出力される。

```
c―――――――――――――――――――――――――――――――★応力等の最大値セット
      call Get_max_stress(Member,n_member,Max_stress)
c―――――――――――――――――――――――――――――――★最大応力等の出力
      call Out_max_stress(Max_stress,Parameter_C.n_member,
     *                    ifl(16),iflz(16))
```

これらのサブルーチンは、いずれも submain_dynamic_a()から呼ばれている。以下に、２つのサブルーチンの内容を示す。

```
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_max_stress
C       ―――――――――――――――――――――――――――――――――――
C     ●   最大応力を求める(ok)
C     ―――――――――――――――――――――――――――――――――――――
      subroutine Get_max_stress(Member,n_member,Max_stress)
      implicit real*8(A-H,O-Z)
       include "submain.h"
       record / member_s  / Member
       record / max_stress_s    / Max_stress
       dimension Member(*),Max_stress(*)
C     ―――――――――――――――――――――――――――――――――――――
c     Max_stress       :structure 最大値
c     n_member         :integer   節点数
C     ―――――――――――――――――――――――――――――――――――――
       do i=1,n_member                                                  ! 1
       do j=1,18                                                        ! 2
      aa = Member(i).stress(j)
       if(Max_stress(i).stress_max_member(j).lt.aa)
     *           Max_stress(i).stress_max_member(j)=aa                  ! 3
       if(Max_stress(i).stress_min_member(j).gt.aa)
     *           Max_stress(i).stress_min_member(j)=aa                  ! 4
       end do
       end do
       return
       end
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_max_stress
C       ―――――――――――――――――――――――――――――――――――
C     ●   最大応力を出力する(ok)
C     ―――――――――――――――――――――――――――――――――――――
       subroutine Out_max_stress(Max_stress,n_member,
     *                           ifl,iflz)
       implicit real*8(A-H,O-Z)
       include "submain.h"
       record / max_stress_s    / Max_stress
       dimension Max_stress(*)
```

```
      dimension ff(10)
C     ――――――――――――――――――――――――――――――――――――
c     Max_stress        :structure 最大値
c     n_member          :integer   節点数
C     ――――――――――――――――――――――――――――――――――――
      if(ifl.ne.1) return
      WRITE(iflz,510)                                                  ! 5
  510 FORMAT(5X,'***** MAXIMUM fay *****')
       do i=1,5
      ff(i)=0.
      enddo
      DO i=1, n_member                                                 ! 6
        WRITE(iflz,530) I, (ff(j), j=1,5)
  530   FORMAT(I5,5e12.4)
      enddo
C
      WRITE(iflz,511)
  511 FORMAT(5X,'***** MAXIMUM AXIAL FORCE *****')
      DO i=1, n_member                                                 ! 7
        WRITE(iflz,531) i, Max_stress(i).stress_max_member(1),
     *                  Max_stress(i).stress_min_member(1)
  531   FORMAT(I5,2e12.4)
      enddo
        WRITE(iflz,512)
  512 FORMAT(5X,'***** MAXIMUM BENDING MOMENT *****')
      DO i=1, n_member                                                 ! 8
        WRITE(iflz,532) i, Max_stress(i).stress_max_member(5),
     *                     Max_stress(i).stress_max_member(6),
     *                     Max_stress(i).stress_max_member(11),
     *                     Max_stress(i).stress_max_member(12),
     *                     Max_stress(i).stress_max_member(17),
     *                     Max_stress(i).stress_max_member(18),
     *                     Max_stress(i).stress_max_member(17),
     *                     Max_stress(i).stress_max_member(18),
     *                     Max_stress(i).stress_max_member(17),
     *                     Max_stress(i).stress_max_member(18)
         WRITE(iflz,532) i, Max_stress(i).stress_min_member(5),
     *                     Max_stress(i).stress_min_member(6),
     *                     Max_stress(i).stress_min_member(11),
     *                     Max_stress(i).stress_min_member(12),
     *                     Max_stress(i).stress_min_member(17),
     *                     Max_stress(i).stress_min_member(18),
     *                     Max_stress(i).stress_min_member(17),
     *                     Max_stress(i).stress_min_member(18),
     *                     Max_stress(i).stress_min_member(17),
     *                     Max_stress(i).stress_min_member(18)
  532   FORMAT(I5,10e12.4)
      enddo
C
       return
       end
```

1．応力の最大値をチェックするために、部材全部について以下の処理を行う。
2．部材両端と中央点における 6 つの合応力について、最大値をチェックする。
3．応力の最小値を求める。
4．応力の最大値を求める。
5．現在は、塑性関数値の最大値は求めていないため、ここでは、全部材について、ゼロを出力する。
6．軸力の最小値と最大値を全部材について出力する。
7．全部材について、両端及び中央の両軸に関する曲げモーメントの最小値と最大値を出力する。