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
```