第９章　新規開発モデルのシステムへの組み込み方法

9.1 はじめに

本章では、新規に部材モデルや履歴モデルを、SPACE に安全に組み込む方法について解説する。既に、SPACE の動的解析システムについては、多くの知識を得ていると思う。知識が不十分であるにもかかわらず、安易に SPACE に変更を加えると、妥当な解が得られるどころか、他のモデルも解析できなくなってしまう恐れがあるので、前節までの解説を良く読んで理解して頂きたい。

SPACE や動的解析に関する知識を利用して、新たな履歴モデルや部材モデルを組み込む手続きと注意点を示す。特に、SPACE では静的解析と動的解析において同じ構造用データを用いて解析を行っている。従って、動的解析に新しいモデルを組み込む場合は、静的解析にも同様に新規モデルを組み込んで頂きたい。また、他のモジュール、例えばプレゼンターなどにも影響を与えることになるので、システム全体を良く理解した上で変更されたい。

履歴モデルや部材モデルを組み込むには、適切な順序と方法で行う必要がある。一般的な組み込み順序は次のようである。

１）モデルを設計し、そのモデルに登録番号を付ける。
２）データの入力仕様の設計と現在の仕様との調整を行う。
３）出力仕様の設計とプレゼンターとの関係を調査する。
４）殊な構造体を必要とするかどうか調査する。構造体を設計する場合は、他の構造体との整合性を確認する。また、その構造体の定義と構造体配列の宣言と確保、及び、解放処理を行う場所を確認する。
５）新規に必要となるサブルーチンを設計し、作成する。
６）モデルの階層構造を理解し、どのレベルに組み込むかを調査する。
７）既存のサブルーチンとのインターフェイス、並びに既存のサブルーチンの追加・変更部分の調査を行う。
８）設計した新規サブルーチンを SPACE の適切な位置に組み込む。
９）ここから実際にシステムを動作させ、組み込まれた新規モデルが設計した仕様を満足するかどうか動作確認する。最初に設計したサブルーチンの引数を出力し、適切な値となっているかどうか、チェックする。
10）次に、簡単なモデルで履歴などをチェックする。

11）複雑な解析モデルを用いて、整合性をチェックする。
12）他のモデルとの整合性をチェックする。

SPACE システムは、一般に公開にされたシステムであり、常にバージョンアップが行われる。従って、個人で新規モデルを組み込む場合は、次の点に注意を払って頂ければ幸いである。

１）個人で使用する場合は、モデル番号や履歴番号は、個人用の番号を使用する。SPACE システムがバージョンアップした場合は、再度開発したプログラム組み込む必要は生じるが、個人用番号を使用すればデータなどに不具合を生じることはない。
２）新規モデルが完成したときは、SPACE に登録していただきたい。それには、まず関連サブルーチンとマニュアルを提出する。登録することで、正規のモデル番号を取得し、SPACE の新バージョンが発行されるとき正規モデルとなる。SPACE を育てるためにも、是非、新規開発モデルの正規登録にご協力ください。

9.2 せん断型モデルの履歴の組み込み方法

9.2.1 せん断型モデルの履歴の設計

本節では、せん断型モデルの履歴特性を安全に組み込む方法について説明する。このせん断型モデルの履歴特性の組み込みが、最も単純、かつ容易である。前節で説明した順序にしたがって、新しいせん断型モデルの履歴特性を組み込んでみよう。

最初に、せん断型モデルの履歴特性を設計する。履歴特性は骨格曲線と履歴ルールで表すため、それらを図で表現し、また、履歴ルールを文章で書き下すと良い。矛盾のない履歴ルールを設計しなければならない。特に、直線で状態を表す直線区分型の場合、その連結部分で接線剛性が大きく変化するときや接線剛性が負勾配を有するときは、誤差やエラーが発生しやすくなる。この段階でなんらかの対策を立てておく必要があろう。

せん断型モデルの履歴特性が設計できたとして、ここでは、仮にその名前を New_Model としよう。この履歴特性に登録番号を付けることになるが、ここでは、仮登録であるとして、101 を設定する。現在の Ver.2.10 では、以下のせん断型モデルの履歴が既に登録されており、このバージョン以降も追加の予定がある。

登録番号　履歴モデル名
1．トリリニア型：
2．最大点指向型
3．武田モデル
11. 修正バイリニア型
12. 修正 R0 モデル

9.2.2 モデルの入力仕様

次に、このモデルの入力仕様について考えてみよう。せん断型モデルでは、標準入力仕様と特別入力仕様の両者が使用されている。登録番号 1-3 の履歴モデルが標準仕様でデータを入力しており、登録番号 11-12 の履歴モデルが特別仕様でデータ入力を行っている。この特別仕様の入力方法を用いると、これらのデータを保存する構造体を別途設計しなければならず、他にも余分な手続きが必要となる。これについては、登録番号 11 の修正バイリニア型を用いて説明しよう。

まず、標準入力仕様のモデルについて説明する。データ入力を行うサブルーチン Get_structure()のなかで、要素データを入力する部分のコードを取り出す。ここで示すデータ構造で、新規履歴モデルの特性が全て設定できるようであれば、標準入力仕様となり、後の手続きが非常に単純となる。

```
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)
       .
       .
      read(5,*,err=9913,end=9918) m_type,e,g,a,rix,riy,riz,asy,asz,          ! 1
     *          am1,am2,anp,ampy,ampz,nm_type
       .
       .
c――――――――――――――――――――――――★せん断タイプで修正 R0 モデルの数をかぞえる
      if(m_type .eq. 2) then                                                ! 2
      if(nm_type.eq.RO_MODEL_NUMBER.or.
     *   nm_type.eq.TRI_MODEL_NUMBER) then
      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
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             ! 3
      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                    ! 4
      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
      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
      Element(i).AM(2)        =  am2
       endi
```

1．ここで示す要素データの入力が標準仕様の場合に相当し、この中でモデルの特性が設定できるようであれば標準仕様で良い。データ項目の内、最初の m_type は要素タイプ番号であり、せん断型の場合、2 となる。また、最後の nm_type は、履歴特性番号であり、ここでは、新規の履歴特性 New_Model として、101 を設定することになる。この 2 つのデータ以外は、全てせん断型の特性値を設定して良いことになる。

2．特別仕様となる修正バイリニアモデルや修正 RO モデルの要素の数を数える。この値を用いて特別に設計した要素に関する構造体の配列数を動的に確保することになる。

3．入力した要素データを要素に関する構造体にセットする。構造体の名前は標準の構造体 Element を用いているので、新規に設定したモデルの特性データを表す名前とは一致しない。これについては後で述べる。

４．履歴タイプの番号が構造体成分 Element(i).nm_type にセットされる。この構造体の内容が 101 であるとき、新規の New_model が動作することになる。

次に、特別入力仕様について考えよう。ここでは、要素データの標準

入力仕様を拡張して用いる場合と、特別に入力データ仕様を設計し、そのデータを読み込むサブルーチンを作成する場合とがある。

まず、入力仕様を拡張する場合について説明する。要素データは標準仕様として、静的縮合モデルでは第 2 レコードが追加されており、これと同様に必要とするデータを設計し、下記のようにプログラムを追加して対処する。

```
      read(5,*,err=9913,end=9918) m_type,e,g,a,rix,riy,riz,asy,asz,
     *          am1,am2,anp,ampy,ampz,nm_type
       .
       .
c--------------------------★新規モデル New_Model のために拡張データ入力
      if(m_type .eq. 2 .and. nm_type .eq. 101) then
      read(5,*,err=9913,end=9918) d1,d2,d3,d4,d5
      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
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,
     *   (Element(i).n_section(j),j=1,2)
      endif
```

無論、このデータが構造体element_sに保存できることが前提である。例えば、上記サブルーチン Get_structure()の中で次の太文字のように設定する。

```
      Element(i).nm_damp       =  d1        ! せん断型モデルの拡張仕様
      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   =  d2     ! せん断型モデルの拡張仕様
      Element(i).j_rigid_length   =  d3     ! せん断型モデルの拡張仕様
      Element(i).i_shear_G        =  d4     ! せん断型モデルの拡張仕様
      Element(i).j_shear_G        =  d5     ! せん断型モデルの拡張仕様
```

このように、要素構造体で設定できるデータ個数はおのずと制限されることになる。なお、このデータを使用する場合は、構造体 element_s2 もしくは、新たに設計した構造体として使用することになる。

入力仕様を変更した場合、動的解析システムでは、この構造データファイルは 2 度読まれるため、次のサブルーチン INPUTE()でも、同様に変更する必要がある。ただし、こちらは構造図描画用であるため、入力データは読み捨てても良い。

```
C     ------------------------------------------
C     ●   subroutine /INPUTE
```

```
C      ―――――――――――――――――――――――――――――――
C     ●     要素データ入力
C     ―――――――――――――――――――――――――――――――
      subroutine INPUTE()
       ・
       ・
      do 200 i=1,nelem
      read(5,*,end=1999,err=1998) mtype(i),youg(i),gsha(i),area(i),
     &         rix(i),riy(i),riz(i),asy,asz,am1,am2,
     &         anp,ampy,ampz,nm_type(i)
c――――――――――――――――――――――★新規モデル New_Model のために拡張データ入力
      if(m_type .eq. 2 .and. nm_type .eq. 101) then
      read(5,*,err=9913,end=9918) d1,d2,d3,d4,d5
      endif
      if(mtype(i) .eq. 11.or.mtype(i) .eq. 15.or.
     *   mtype(i) .eq. 21.or.mtype(i) .eq. 31)then
      read(5,*,end=1999,err=1998) rd1,rd2,sgi,sg2,iix1,iix1
      endif
      if(mtype(i) .eq. 12.or.mtype(i) .eq. 22
     *   .or.mtype(i) .eq. 32) then
      read(5,*,end=1999,err=1998) rd1,rd2,sgi,sg2,iix1,iix1,iix1
      endif
       ・
       ・
      return
      end
```

この構造データファイルは、他のモジュールでも入力しており、整合性を取るためにも、他のシステムの構造データ入力部分を変更しなければならない。それについては、他のマニュアルを参照されたい。

最後に、入力データを多数必要とする場合は、特別に入力データの仕様を設計し、そのデータを読み込むサブルーチンを作成する。ここでは、現在SPACEに組み込まれている修正バイリニア型モデルについて見てみよう。このサブルーチンでは、特別な構造体は、RO_work_s として設計されており、また、要素に関する構造体 element_s2_RO も、その内容を新たに設計したものである。この構造体 RO_work_s については後の節で説明する。構造体 element_s の替わりとなる構造体 element_s2_RO の内容は、プログラム内のコメントとして書き込まれている。この特別な入力データは、サブルーチン submain_dynamic_a()の中で、以下のようにコールされている。

```
c――――――――――――――――――――――――★修正 RO モデル領域セット
      n=Model_type.n_m_ro_model                                    ! 1
      if(n.ne.0) then
      nfix=5
      nfi=53
      call infile(nfi,nfix,ierr)                                   ! 2
      if(ierr.ne.0) then
```

```
      ierr_dat = 35
      call err_outf(ierr_dat)                                    ! 3
      return
      endif
      call RO_data_input(n,RO_work,Element,Parameter_C.n_element,ierr)   ! 4
      close(nfix)
      if(ierr.ne.0) then
      ierr_dat = 36
      call err_outf(ierr_dat)                                    ! 5
      return
      endif
      endif
      ・
      ・
```

1. 修正バイリニアと修正 RO モデルの個数をセットする。ここで、この個数がゼロでない場合は、これ以降の処理を実行し、特殊データを読み込むことになる。この個数は、サブルーチン Get_structure() の中で数えており、従って、特殊な構造体あるいはそれに付随する入力用のサブルーチンを必要とする場合は、Get_structure()の中でその個数を数えるコードを付け加える必要がある。
2. 修正バイリニアと修正 RO モデルに関する特殊データが保存されているファイルについて、サブルーチン infile()を用いて、存在のチェックとオープン処理を行う。
3. 上記の処理でエラーがあった場合の処理を行う。エラー出力をサブルーチン err_outf()を用いて、エラーコード 35 と共にダンプファイルにエラーの内容を出力する。
4. サブルーチン RO_data_input()を用いて、修正バイリニアと修正 RO モデルに関する特殊データファイルを読み込む。
5. このデータファイルの内容を読み込む際、何らかのエラーがあった場合、サブルーチン err_outf()を用いて、エラーコード 36 と共にダンプファイルにエラーの内容を出力する。

例として、具体的に特殊データファイルを読み込むサブルーチン RO_data_input()を見てみよう。ここでは、新しく定義し直した構造体 elemennt_RO_s と新しく設計した構造体 RO_work_s が使用されている。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /RO_data_input
C       ―――――――――――――――――――――――――――――――
C     ●   RO モデルデータ入力(ok)
C     ―――――――――――――――――――――――――――――――
      subroutine RO_data_input(n,RO_work,Element,n_element,ierr)
```

```
      implicit real*8(A-H,O-Z)
      include "submainx.h"
      include "submain.h"
      record / RO_work_s     /  RO_work                                  ! 1
      record / element_s2_RO / Element                                   ! 2
      dimension RO_work(*),Element(*)                                    ! 3
      integer RO_MODEL_NUMBER,TRI_MODEL_NUMBER
      real*8 BI_DEF
      integer n_ro
      data   BI_MODEL_NUMBER/11/
      data   RO_MODEL_NUMBER/12/
      BI_DEF=10.0**(-10.0)
C     ――――――――――――――――――――――――――――――――――――
C     ●      3 次元せん断弾塑性モデル（モデル No.2）
C                修正 R-O，修正 Bi-Liner 型用
C     ――――――――――――――――――――――――――――――――――――
c     要素数（モデル No.2  3 次元せん断弾塑性モデル：トリリニア型）
c    element_s 構造体と同一
c
c     structure / element_s2_RO/
c     integer  element_type   !  要素タイプ
c     integer  n_element      !  非線形要素番号
c     real*8   AK_1           !  初期剛性（内部計算）
c     real*8   dm1(4)         !  ダミー
c     real*8   AKu            !  軸方向バネ
c     real*8   dm2(2)         !  ダミー
c     real*8   Ar             !  積層ゴムの断面積
c     real*8   dm22           !  ダミー
c     integer  nm_damp        !  部材減衰の有無(1)
c     integer  nm_type        !  履歴モデルのタイプ
c     integer  No             !  R-O ファイル履歴モデル番号
c     integer  dm3(9)         !  ダミー
c     real*8   ANP            !  軸方向耐力
c     real*8   AQPv           !  y 方向耐力
c     real*8   AQPw           !  z 方向耐力
c     real*8   dmm(3)         !  ダミー
c     real*8   dmm2(4)        !  ダミー
c     end structure
C     ――――――――――――――――――――――――――――――――――――
c     RO_work    :structure
C     ――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★データ入力(ok)
      ierr=0
      read(5,*,end=999,err=998) nn
      if(nn.le.0) goto 998
      do i=1,nn
      read(5,*,end=999,err=998) (RO_work(i).gan(j), j=1,3),
     *    (RO_work(i).arf1(j),j=1,6),
     *    (RO_work(i).arf2(j),j=1,6),
     *    (RO_work(i).beta(j),j=1,6),
     *    (RO_work(i).anyu(j),j=1,6)
      enddo
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★部材の線形剛性計算(ok)
c     Element(i).AKu          ! 軸剛性
```

```
c    Element(i).AK_1          ! 線形せん断剛性
c
      ii=0
      do i=1,n_element
      if(Element(i).element_type.eq.2) then
      if( Element(i).nm_type.eq.RO_MODEL_NUMBER.or.
     *    Element(i).nm_type.eq.BI_MODEL_NUMBER)then
      ii=ii+1
      if(ii.gt.nn) goto 999
c――――――――――――――――★初期剛性の計算挿入箇所(後で積層ゴム厚をかける必要があり)
      n_ro=Element(i).No
      if(Element(i).nm_type.eq.BI_MODEL_NUMBER)then
      Element(i).AK_1
     *    =0.7*RO_work(n_ro).arf1(1)*BI_DEF**(-0.3)
      else if(Element(i).nm_type.eq.RO_MODEL_NUMBER)then
      Element(i).AK_1
     *    =Element(i).Ar*RO_work(n_ro).arf1(1)
      endif
      endif
      endif
      enddo
      return
  998 continue
      ierr=1
      return
  999 continue
      ierr=2
      return
      end
```

1. このサブルーチンの中で使用する構造体 RO_work_s を、引数として受け渡された実際に存在する領域 RO_work に、record 文を用いて割り付ける。
2. 同じく、構造体 element_s2_RO を、引数として受け渡された実際に存在する領域 Element に、record 文を用いて割り付ける。
3. 構造体である RO_work と Element が共に配列であることを宣言する。

9.2.3 モデルの出力仕様

本節では、この新しい履歴モデルに関する出力仕様について説明する。出力データは、他のモジュールで使用されており、変更する場合は、他のシステム、例えば、動的プレゼンターなどをチェックし、整合性を取る必要がある。

まずは、せん断型モデルの標準出力仕様を見てみよう。解析結果を出力するサブルーチン Out_stress()は、submain_dynamic_a()の中で、コールされている。ここでは、せん断型に関連する部分のみ示す。

```
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                                  ! 1
      record / Element_s        / Element
      record / Out_section_s    / Out_section
      record / E_model6_real_s  / E_model6_real
       •
       •
      do i=1,n_member
      if(Member(i).element_type.eq.6) then
c―――――――――――――――――――――――――――――★Maxwell モデル
       •
       •
      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              ! 2
      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
       •
       •
      else
       •
       •
      endif
      enddo
      return
      end
```

1. このサブルーチンの中で使用する構造体 Member と Element は、標準仕様で定義されている。これは、各種のモデルの出力をこのサブルーチンが受け持っているからであり、従ってせん断型として新たに設計した構造体の内容で出力することはできない。
2. せん断型モデルの出力は非常に単純で、軸力、y 軸のせん断力、z 軸のせん断力である。現在は、他のデータは全てダミーとなっており、

ここに、新たなデータ出力項目を設けて他のモジュールで使用できるようにすることも可能である。当然のこと、他のモジュール、例えば、動的プレゼンターなどでそれに対応する処理を付け加える必要がある。

9.2.4 構造体の定義

本節では、新規の履歴モデルによって、新たな構造体を作成する必要がある場合について説明する。まず、標準入力仕様で、しかも、要素に関連する element_s 構造体と部材に関連する member_s 構造体が標準の構造体の内容で良い場合、あるいは、せん断型モデルの標準型として設計した構造体 element_s2 や member_s2 を使用する場合は、新たに構造体を設計する必要はない。

まず、要素に関係する element_s と element_s2 を示そう。両者の並びで対応する変数は同じ記憶領域を占めており、構造体配列の数は要素数分設定される。同様に、member_s と member_s2 は同じ動的記憶領域に、解析モデルの部材数分設定される。前節で示した入力プログラムの中で、構造体に入力データをセットしているが、構造体 element_s2 のどの変数に対応しているか比べてみると良い。

```
C     ―――――――――――――――――――――――――――――――
C     ●   element_s 構造体
C     ―――――――――――――――――――――――――――――――
c
c    要素
      structure / element_s/
      integer  element_type   !  要素タイプ
      integer  n_element      !  非線形要素番号
      real*8   E              !  ヤング係数
      real*8   G              !  せん断係数
      real*8   A              !  断面積
      real*8   RIx            !  ねじり剛性
      real*8   RIy            !  ｙ軸断面二次モーメント
      real*8   RIz            !  ｚ軸断面二次モーメント
      real*8   ASy            !  各部材のＹ軸回りのせん断変形用等価断面積
      real*8   ASz            !  各部材のＺ軸回りのせん断変形用等価断面積
      real*8   AM(2)          !  単位長さ当たりの質量（1:第一 2:第二ステップ用）
      integer  nm_damp        !  部材減衰の有無
      integer  nm_type        !  （maxwell モデルでは、１：ｘ方向２：ｙ方向３：ｚ方向）
      integer  n_section(5)   !  断面番号
      integer  nm_section(5)  !  ファイバー数
      real*8   ANP            !  軸方向耐力
      real*8   AMPY           !  ｙ軸塑性モーメント
      real*8   AMPZ           !  ｚ軸塑性モーメント
```

```
      real*8   dmm(3)          !  ダミー
      real*8   i_rigid_length      !  i 端剛域長さ
      real*8   j_rigid_length      !  j 端剛域長さ
      real*8   i_shear_G           !  i 端せん断剛性
      real*8   j_shear_G           !  j 端せん断剛性
      end structure
C     ―――――――――――――――――――――――――――――――――
C     ●　3 次元せん断弾塑性モデル（モデル No.2）
C     ―――――――――――――――――――――――――――――――――
c    要素数（モデル No.2　3 次元せん断弾塑性モデル：トリリニア型）
c   element_s 構造体と同一
c
      structure / element_s2/
      integer  element_type   !  要素タイプ(6)
      integer  n_element      !  非線形要素番号
      real*8   AK_1           !  第一剛性
      real*8   AK_2           !  第二剛性
      real*8   AK_3           !  第三剛性
      real*8   Q_1            !  第一折れ点のせん断力
      real*8   Q_2            !  第二折れ点のせん断力
      real*8   AKu            !  軸方向バネ
      real*8   arf            !  武田モデルα(通常は 0.4)
      real*8   U_1            !  第一折れ点の変位
      real*8   U_2            !  第一折れ点の変位
      real*8   dm4            !  ダミー
      integer  nm_damp        !  部材減衰の有無(1)
      integer  nm_type        !  履歴モデルのタイプ
      integer  n_section(5)   !  断面番号
      integer  nm_section(5)  !  ファイバー数
      real*8   ANP            !  軸方向耐力
      real*8   AQPv           !  y 方向耐力
      real*8   AQPw           !  z 方向耐力
      real*8   dmm(3)         !  ダミー
      real*8   dmm2(4)        !  ダミー
      end structure
```

上の構造体 element_s2 をそのまま利用することも可能であるが、新規の履歴モデル用として、新たに設計することもできる。上に示した構造体 element_s2 を参照して作成されたい。この構造体は、要素データを保存するためのものであり、部材毎に計算途中で変化するデータを入れるものではない。これについては、構造体 member_s が用意されている。また、この構造体の全ての成分が使用可能となっておらず、上記の element_s2 の中で濃い文字で示した成分はシステムが使用する。無論、ここでは、ダミーとして定義している成分は別の変数として設定し、それらを使用することもできる。

次に、部材に関する構造体 member_s を示そう。上記の element_s と同様に、標準仕様の member_s とせん断型モデルの構造体 member_s2 がある。せん断型モデルの構造体 member_s2 では、ワーク領域は多く必要

としないので、設定していても使用していない成分がある。ここでも、新たな構造体を設計したい場合は、この構造体 member_s2 を参照して作成されたい。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   member_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / member_s/
      integer  nm_element         !  要素番号（入力した要素番号を示す）
      integer  element_type       !  要素タイプ番号
      integer  n_model            !  モデルの入れ物番号
      integer  n_model_type       !  モデル別の通し番号
      integer  n_element_type     !  要素タイプ別通し番号
      integer  analysis_3D        !  解析型 0:3D 1:2D(x-z) 2:2D(y-z)
      integer  nm_so              !  部材の層番号
      integer  nm_dll_element     !  DLL を用いた要素か（０；システム内要素、１：DLL 要素）
      integer  nm_point(2)        !  節点番号
      integer  irest(12)          !  部材両端の自由度番号表
      integer  ijp(2)             !  両端節点への結合状況（0:剛結合 1:ピン結合）
      integer  nm_analysis        !  部材解析種別（-1:弾性解析、その他：通常解析）
      integer  nm_group           !  部材グループ
      integer  nm_local_coord(2)  !  局所座標系の有無とその回転行列の番号
      integer  nm_damp            !  部材減衰の有無とその減衰行列の番号
      real*8   alength            !  長さ
      real*8   i_rigid_length     !  i 端剛域長さ
      real*8   j_rigid_length     !  j 端剛域長さ
      real*8   i_shear_G          !  i 端せん断剛性
      real*8   j_shear_G          !  j 端せん断剛性
      real*8   rot_x              !  部材主軸の回転角度（度）
      real*8   force(12)          !  部材両端の部材端力（釣合座標系）
      real*8   stress(18)         !  部材両端,中央の応力（部材座標系）
      real*8   an_stress(10)      !  部材軸力（部材座標系）
      integer  d_stat(3)          !  断面の弾塑性状態　(1) i 端 (2) j 端　(3) 中央
      real*8   an_vv(10)          !  部材軸力（計算用内部変位ｖ）
      real*8   an_ww(10)          !  部材軸力（計算用内部変位w）
      end structure
C     ―――――――――――――――――――――――――――――――――
C     ●   3 次元せん断弾塑性モデル用（モデル No.2）member_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / member_s2/
      integer  nm_element         !  要素番号
      integer  element_type       !  要素タイプ
      integer  n_model            !  モデルの入れ物番号
      integer  n_model_type       !  モデル別の通し番号
      integer  n_element_type     !  要素タイプ別番号
      integer  analysis_3D        !  解析型 0:3D 1:2D(x-z) 2:2D(y-z)
      integer  nm_so              !  部材の層番号
      integer  nm_dll_element     !  DLL を用いた要素か（０；システム内要素、１：DLL 要素）
      integer  nm_point(2)        !  節点番号
```

```
      integer  irest(12)          !  部材両端の自由度番号表
      integer  istat_v            !  y 方向:履歴特性の状態(*)
      integer  istat_w            !  z 方向:履歴特性の状態(*)
      integer  nm_analysis        !  部材解析種別
      integer  nm_group           !  部材グループ
      integer  nm_local_coord(2)  !  局所座標系の有無とその回転行列の番号
      integer  nm_damp            !  部材減衰の有無とその減衰行列の番号
      real*8   alength            !  長さ
      real*8   i_rigid_length     !  i 端剛域長さ
      real*8   j_rigid_length     !  j 端剛域長さ
      real*8   i_shear_G          !  i 端せん断剛性
      real*8   j_shear_G          !  j 端せん断剛性
      real*8   rot_x              !  部材主軸の回転角度（度）
      real*8   force(12)          !  部材両端の部材端力（釣合座標系）
      real*8   stress(6)          !  部材中央の応力（部材座標系）
      real*8   AKv_tan            !  接線剛性(*)
      real*8   AKw_tan            !  接線剛性(*)
      real*8   dmv(10)            !  y 方向耐力:履歴特性で使用(*)
      real*8   dmw(10)            !  z 方向耐力:履歴特性で使用(*)
      integer  d_stat(3)          !  断面の弾塑性状態　(1) i 端 (2) j 端　(3) 中央
      real*8   an_vv(10)          !  部材軸力（計算用内部変位ｖ）
      real*8   an_ww(10)          !  部材軸力（計算用内部変位w）
      end structure
```

標準仕様の構造体を使用する場合は、新たな構造体を必要としないが、その新規モデルに適合した構造体を設計する場合は、その構造体を新たなヘッダーファイルに作成する必要がある。このヘッダーファイルを New_submain.h としよう。読者はまずこの New_submain.h ファイルを作り、この中に新たな構造体を書き込むことになる。この構造体をどのように利用するかについては、後節で説明する。

それでは、入力データ数が多くて標準仕様では保存するデータ数が不足し、新たな構造体を設計する必要がある場合について考えよう。これも前節で示した修正バイリニアモデルで説明する。まず、新たに設計した構造体 RO_work_s を示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●　履歴モデル　RO/修正 Bi-Liner モデル：せん断要素用
C                                          Work エリア構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / RO_work_s/
      real*8   gan(3)             !  適用歪
      real*8   arf1(6)            !  歪１の骨格曲線の定義
      real*8   arf2(6)            !  歪２の骨格曲線の定義
      real*8   beta(6)            !  等価減衰定数の定義
      real*8   anyu(6)            !  履歴剛性の定義
      end structure
```

この構造体は、動的に確保する配列であることから、サブルーチンsubmain_dynamic_a()の中で以下のように確保する必要がある。ここで、確保するかしないか判別するパラメータ Model_type.n_m_ro_model は、先に説明した修正バイリニアモデルと修正 RO モデルの数である。

```
c――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      n=Model_type.n_m_ro_model
      if(n.ne.0) then
      ALLOCATE (RO_work(n))
      endif
```

手続きとしては、動的確保の他に、動的領域であることの宣言、使用後の領域解放を行うことになる。構造体を新たに設計、使用する場合は、これらの処理を必ず既存プログラムの中に書き込む必要が生じる。

9.2.5 必要となるサブルーチン

次に、この履歴特性 New_Model を SPACE に組み込むために、必要となるサブルーチンについて説明する。この必要となるサブルーチンは

１）線形剛性を求めるサブルーチン
２）非線形剛性を求めるサブルーチン
３）応力を求めるサブルーチン
４）応力をチェックし、接線剛性を求めるサブルーチン

である。ただし、この中で標準のせん断型モデルのサブルーチンを転用できる場合は、そのサブルーチンを使用すれば良い。まず、上記４つの標準的なせん断型モデルに関するサブルーチンを見てみよう。

最初は、線形剛性行列を求めるサブルーチン Cal_lin_stiff_M2()であり、次に示す。

```
C     ――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_lin_stiff_M2
C       ――――――――――――――――――――――――――――――――
C     ●   Model_No.2 3 次元せん断弾塑性モデル
C     ――――――――――――――――――――――――――――――――
      subroutine Cal_lin_stiff_M2(Member,Element,ak_linear)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s2    / Member
      record / element_s2   / Element
      dimension ak_linear(12,12)
      data    BI_MODEL_NUMBER/11/
      data    RO_MODEL_NUMBER/12/
```

```
c―――――――――――――――――――――――――――――――――――――――
c           Element             structure
c           Member              structure
c           ak_linear           real*8   線形剛性行列
c―――――――――――――――――――――――――――――――――――――――
      do i=1,12
      do j=1,12
        ak_linear(j,i) = 0.0
      end do
      end do
      ak=Element.AKu
      ak_linear(1,1)= ak
      ak_linear(1,7)=-ak
      ak_linear(7,7)= ak
      ak_linear(7,1)=-ak
      if (Element.nm_type.eq.BI_MODEL_NUMBER) then
        ak=Element.AK_1*Member.alength
      else if(Element.nm_type.eq.RO_MODEL_NUMBER) then
        ak=Element.AK_1/Member.alength
      else
        ak=Element.AK_1
      endif
      ak_linear(2,2)= ak
      ak_linear(2,8)=-ak
      ak_linear(8,8)= ak
      ak_linear(8,2)=-ak
      ak_linear(3,3)= ak
      ak_linear(3,9)=-ak
      ak_linear(9,9)= ak
      ak_linear(9,3)=-ak
c―――――――――――――――――――――――――――――――★履歴特性の初期設定

      Member.AKv_tan=ak              ! Member.AKv_tan=Element.AK_1
      Member.AKw_tan=ak              ! Member.AKw_tan=Element.AK_1
      Member.istat_v=0
      Member.istat_w=0
      return
      end
```

このサブルーチンを見れば分かるように、要素に関する構造体の中の成分 Element.AKu と Element.AK_1 に所定の値がセットされていれば、このまま使用できることになる。

次に、非線形剛性を求めるサブルーチン Cal_nonlin_stiff_M2()について調査する。このプログラムも以下に示すように線形剛性のプログラムと同様、構造体成分 Element.AKu と Member.AKv_tan、Member.AKw_tan を他のプログラムでセットしておけば転用可能となる。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_nonlin_stiff_M2
```

```
C      ―――――――――――――――――――――――――――――――――
C     ●   Model_No.2 3 次元せん断弾塑性モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_nonlin_stiff_M2(Member,Element,ak)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s2    / Member
      record / element_s2   / Element
      dimension ak(12,12)
C     ―――――――――――――――――――――――――――――――――
c           Element             structure
c           Member              structure
c           ak_linear           real*8   線形剛性行列
c―――――――――――――――――――――――――――――――――――――――――
      do i=1,12
      do j=1,12
      ak(j,i) = 0.0
      end do
      end do
      akk=Element.AKu
      ak(1,1)= akk
      ak(1,7)=-akk
      ak(7,7)= akk
      ak(7,1)=-akk
      akk=Member.AKv_tan
      ak(2,2)= akk
      ak(2,8)=-akk
      ak(8,8)= akk
      ak(8,2)=-akk
      akk=Member.AKw_tan
      ak(3,3)= akk
      ak(3,9)=-akk
      ak(9,9)= akk
      ak(9,3)=-akk
      return
      end
```

次に、応力を求めるサブルーチンを調べよう。このプログラムも、せん断型モデルでは、特殊な応力計算を行わない限り、非線形剛性行列を求めるプログラムと同様、構造体成分 Element.AKu と Member.AKv_tan、Member.AKw_tan を他のプログラムでセットしておけば転用可能となる。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress_M2
C     ―――――――――――――――――――――――――――――――――
C     ●   部材の応力計算(次元せん断弾塑性モデル（モデル No.2）)
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_stress_M2(Member,Element,vv)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s2    / Member
```

```
      record / element_s2   / Element
      dimension vv(12)
C     ―――――――――――――――――――――――――――――――――
c           Element             structure
c           Member              structure
c           vv               real*8  増分変位
C     ―――――――――――――――――――――――――――――――――
      Member.stress(1)=Member.stress(1)+Element.AKu*
     *                (vv(7)-vv(1))
      Member.stress(2)=Member.stress(2)+Member.AKv_tan*
     *                (vv(8)-vv(2))
      Member.stress(3)=Member.stress(3)+Member.AKw_tan*
     *                (vv(9)-vv(3))
      return
      end
```

以上のように、せん断モデルの履歴特性では、標準仕様の３つのサブルーチンがそのまま使用することができることが分かる。後は、応力をチェックし、接線剛性を求めるサブルーチンであるが、このサブルーチンについては、次節で説明しよう。

9.2.6 モデルの階層構造とサブルーチンの組み込み

せん断型モデルでは、submain_dynamic_a()から Check_stress()がコールされ、その中で応力の弾塑性チェックのプログラムがコールされることになる。まず、Check_stress()の中で、このせん断モデルに関係する部分を抜き出してみよう。このサブルーチンの中には、部材モデルの階層構造が見られる。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Check_stress
C     ―――――――――――――――――――――――――――――――――
C     ●   部材の塑性状態をチェックする(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Check_stress( )
c―――――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
c――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=disp_point(ires)
      vp(j)=past_disp_point(ires)
      else
      v(j)=0.
      vp(j)=0.
      endif
      enddo
c――――――――――――――――――――――――――――――――★部材両端の節点力のゼロセット
```

```
      do j=1,12
      f(j)=0.
      enddo
c―――――――――――――――――――――――――――――――★変位を釣合系から部材座標系に変換
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c―――――――――――――――――――――――――――――――★要素及びモデルのセット
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie   = Member(i).nm_element
      im   = Element(ie).n_element
      ien  = Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0)  goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
c―――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
       .
       .
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      if(N_analysis.ge.9) then
      call Cal_check_stiff_M2(Member(i),Element(ie),RO_work,
     *     vv,vpp )
      endif
      do j=1,3
      f(j)=-Member(i).stress(j)
      f(j+6)=-f(j)
      enddo
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      do j=1,12
      Member(i).force(j)=ff(j)
      enddo
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
       .
       .
  100 continue
      end do
      return
      end
```

部材モデル番号2によってせん断型弾塑性モデルのサブルーチンに制御が移る。このとき、解析種別が９以上で弾塑性解析を行う場合、このサブルーチンが実行される。引数として増分変位と増分前の変位を、釣合座標系から部材座標系に変換し、受け渡す。

次に、サブルーチン Cal_check_stiff_M2()を示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_check_stiff_M2
```

```
C      ―――――――――――――――――――――――――――――――――
C     ●   Model_No.2 3 次元せん断弾塑性モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_check_stiff_M2(Member,Element,RO_work,vv,vpp)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h "
      record / member_s2    / Member
      record / element_s2   / Element
      record / RO_work_s    /  RO_work
      dimension vv(12),vpp(12),RO_work(*)
c―――――――――――――――――――――――――――――――――――――――――
      No_rireki=Element.nm_type
      if(No_rireki/10.eq.0) then
      goto(5,10,20,30,40,50,60),No_rireki+1
    5 continue
c――――――――――――――――――――――――――――――――★規定モデル：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)
      goto 999
   10 continue
c――――――――――――――――――――――――――――――――★トリリニア：Nomal
      call Mesing_TriLiner(Member,Element,vv,vpp)
      goto 999
   20 continue
c――――――――――――――――――――――――――――――――★トリリニア：最大点指向型
      call DirecMax_TriLiner(Member,Element,vv,vpp)
      goto 999
   30 continue
c――――――――――――――――――――――――――――――――★トリリニア：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)
      goto 999
   40 continue
c――――――――――――――――――――――――――――――――★
      goto 999
   50 continue
c――――――――――――――――――――――――――――――――★
      goto 999
   60 continue
c――――――――――――――――――――――――――――――――★履歴モデル
      goto 999
      elseif(No_rireki/10.eq.1) then
      goto(101,102),No_rireki - 10
  101 continue
c――――――――――――――――――――――――――――――――★修正バイリニアモデル
      mro=Element.n_section(1)
      call Modify_Bi_Liner1(Member,Element,RO_work(mro),vv,vpp)
      goto 999
  102 continue
c――――――――――――――――――――――――――――――――★修正 RO モデル
      mro=Element.n_section(1)
      call Modify_RO1(Member,Element,RO_work(mro),vv,vpp)
      goto 999
      elseif(No_rireki/100.eq.1) then
```

```
      goto(201,202),No_rireki - 100
  201 continue
c――――――――――――――――――――――――――――――――――★個人用新規モデル
      call New_model_rireki(Member,Element,vv,vpp)
      goto 999
  202 continue
c――――――――――――――――――――――――――――――――――★個人用新規モデル
      goto 999
      endif
  999 continue
      return
      end
```

このサブルーチンの中に履歴モデルに関する第２層の階層構造が見られる。この中に履歴特性番号 101 として、太文字で示すコードが追加され、新規の履歴モデルをチェックするサブルーチンが加えられている。ここでは、サブルーチン名を New_model_rireki()としている。これで、既存サブルーチンに新規モデルが組み込まれたことになる。引数として受け渡すデータは、構造体の Member と Element、部材座標系の部材両端の節点増分変位と増分前の変位である。特殊な構造体を設計した場合は、ここで引数として受け渡す必要がある。この構造体は、当然、上位のサブルーチンから受け渡される必要があり、また、その利用法は修正 RO モデルにおける RO_work を参考にされたい。その場合、構造体を設定したヘッダーファイル New_submain.h をインクルードしなければならない。

この新規サブルーチンでは、実際にせん断型モデルの履歴を追うことで、少なくとも構造体 Member 中の次の値を設定しなければならない。

> １．Member.istat が -1 のとき、初期設定行う。
> ２．接線剛性 Member.AKw_tan を設定する。
> ３．せん断ばねの応力 Member.stress を求める。

以上のように、新規せん断型モデルの組み込みは、標準入力仕様で間に合えば非常に簡単である。基本的には、応力の弾塑性チェック用サブルーチンを用意すれば良い。

9.3 ファイバーの履歴の組み込み方法

9.3.1 履歴の組み込み

本節では、新規のファイバー履歴を SPACE に組み込む方法について解説する。基本的には、せん断モデルの履歴組み込み方法と同じであるが、多少異なるので、その違いを中心に説明する。現在、SPACE に登録されている履歴モデルは、以下のようであり、履歴番号 1-8 までが使用されている。個人用としては、せん断型と同じく 101 を用いることにする。ここでは、新規のモデルを New_model_fiber とする。

１．対称バイリニア型：BiLinear()
２．対称トリリニア型：TriLinear()
３．直線コンクリート履歴モデル：Concrete()
４．曲線コンクリート履歴モデル：Concrete_e()
５．バイリニア型（移動＋等方硬化用）：BiLinear_h()
６．対称トリリニア型（移動＋等方硬化用）：TriLinear_h()
７．非対称バイリニア型：TriLinear_AS()
８．非対称トリリニア型：BiLinear_AS()

9.3.2 モデルの入力仕様

最初に、このファイバーモデルの入力仕様について考えてみよう。ファイバーの履歴モデルは全て特別仕様でデータ入力を行っており、データを保存する構造体が設計されている。

まず、データ入力を行うサブルーチン Fiber_input()の中で、要素データを入力する部分のコードを取り出す。新規履歴モデルに関するコードが太文字で追加されている。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_input
C     ―――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の入力(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Fiber_input(  )
      ierr=0
      if(it.eq.0) then
c――――――――――――――――――――――――――――――――★ファイバーデータの予備入力
      read(5,*,err=999) nm          ! 最大断面数
      write(76,'(a,i4)') ' ファイバー断面総数：',nm
      ii=0
      do i=1,nm
      read(5,*,err=999) n_m,nmm,(ddm(j),j=1,20)     ! 断面番号、エレメント数
c――――――――――――――――――――――――――――――――★断面ファイバー数のセット
      do i1=1,n_element
      itype_m = Model_type.no_e_model(Element(i1).element_type)
      if(itype_m.eq.11) then
c――――――――――――――――――――――――――――――――★モデル１１                                    ! x1
       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
       ・
       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
       ii = ii + 1
       read(5,*,err=999) n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az                    ! 1
       if(nm_type.le.10) then
       goto(901,902,903,904,905,906,907,908,909,910),nm_type                      ! 2
  901  continue
       goto 900
```

```
902  continue
     read(5,*,err=999) E_3,Q_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                ! コンクリート型
     goto 900
904  continue
     read(5,*,err=999) AK_4,Q_3,STR_3,STR_7                 ! 直線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
     goto(911,912,913,914,915,916,917,918,919,920),nm_type-10
911  continue
     goto 900
912  continue
     goto 900
913  continue
     goto 900
914  continue
     goto 900
915  continue
     goto 900
916  continue
     goto 900
917  continue
     goto 900
918  continue
     goto 900
919  continue
     goto 900
920  continue
     goto 900
     else
     goto(921,922,923,924,925,926,927,928,929,930),nm_type-100          ! 3
921  continue
     read(5,*,err=999) E_3,Q_2                              ! 個人用新規ファイバー履歴
     goto 900
922  continue
     read(5,*,err=999) E_3,Q_2                              ! 個人用新規ファイバー履歴
```

```
      goto 900
 923  continue
      goto 900
 924  continue
      goto 900
 925  continue
      goto 900
 926  continue
      goto 900
 927  continue
      goto 900
 928  continue
      goto 900
 929  continue
      goto 900
 930  continue
      goto 900
      endif
 900  continue
      enddo
      enddo
c―――――――――――――――――――――――★要素ファイバー数セット
      Model_type.nm_div_felement= ii
      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
      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―――――――――――――――――――――――★モデル１１                                  ! x2
      if(itype_m.eq.11) then
         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
      ・
      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
      read(5,*,err=999) nm
      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)
      do j=1,nmm
      read(5,*,err=999) n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az      ! 標準データ     ! 4
c―――――――――――――――――――――★部材断面ファイバー数セット
      ii = ii + 1
      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                    ! 5
 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―――――――――――――――――――――――★対称トリリニア型
      read(5,*,err=999) E_3,Q_2          ! 対称トリリニア型 等方 硬化＋移動硬化トリリニア型
      write(76,'(2i4,18e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az,
     *     E_3,Q_2,beta,beta_2
      E_model_fiber(ii).E_3     = E_3                 !  ファイバーの第三剛性 E3
      E_model_fiber(ii).Q_2     = Q_2                 !  ファイバーの第二折れ点
      E_model_fiber(ii).Ec_1    = 0.                  !  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
      E_model_fiber(ii).Beta    = 0.                  !  移動硬化用パラメータ
      E_model_fiber(ii).Beta_2  = 0.                  !  移動硬化用パラメータその２
      goto 800
 803  continue
c―――――――――――――――――――――――★直線コンクリート型
      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
      read(5,*,err=999) AK_4,Q_3,STR_3,STR_7                 ! 曲線コンクリート型
      write(76,'(2i4,18e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az,
     *     AK_4,Q_3,STR_3,STR_7
      E_model_fiber(ii).E_3     = AK_4                !  引張第二勾配
      E_model_fiber(ii).Q_2     = Q_3                 !  圧縮強度
      E_model_fiber(ii).Ec_1    = STR_3               !  最大圧縮応力点におけるひずみ量
      E_model_fiber(ii).Ec_2    = STR_7               !  弾性限界ひずみ量
      E_model_fiber(ii).Ec_3    = 0.                  !  ダミー
      E_model_fiber(ii).Qc_1    = 0.                  !  ダミー
      E_model_fiber(ii).Qc_2    = 0.                  !  ダミー
      goto 800
 805  continue
c―――――――――――――――――――――――★等方硬化＋移動硬化バイリニア型
      read(5,*,err=999) beta
       write(76,'(2i4,10e12.4)') n,nm_type,A,ry,rz,E_1,E_2,
     *                     Q_1,G,Ay,Az,beta   !等方硬化＋移動硬化バイリニア型
      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
      E_model_fiber(ii).Beta    = beta                  !  移動硬化用パラメータ
      goto 800
  806 continue
c-----------------------★等方硬化＋移動硬化トリリニア型
      read(5,*,err=999) E_3,Q_2,beta,beta_2      !
      write(76,'(2i4,18e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az,
     *     E_3,Q_2,beta,beta_2
      E_model_fiber(ii).E_3     = E_3                   !  ファイバーの第三剛性 E3
      E_model_fiber(ii).Q_2     = Q_2                   !  ファイバーの第二折れ点
      E_model_fiber(ii).Ec_1    = 0                     !  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
      E_model_fiber(ii).Beta    = beta                  !  移動硬化用パラメータ
      E_model_fiber(ii).Beta_2  = beta_2                !  移動硬化用パラメータ
      goto 800
  807 continue
c-----------------------★非対称バイリニア型
      read(5,*,err=999) Ec_1,Ec_2,Qc_1,beta
       write(76,'(2i4,15e12.4)') n,nm_type,A,ry,rz,E_1,E_2,Q_1,G,Ay,Az,
     *              Ec_1,Ec_2,Qc_1,beta                !等方硬化＋移動硬化バイリニア型
      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    = Ec_1                  !  ファイバーの圧縮側第一剛性 E1
      E_model_fiber(ii).Ec_2    = Ec_2                  !  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    = Qc_1                  !  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
      E_model_fiber(ii).Beta    = beta                  !  移動硬化用パラメータ
      goto 800
  808 continue
c-----------------------★非対称トリリニア型
      read(5,*,err=999) E_3,Q_2,Ec_1,Ec_2,Ec_3,Qc_1,Qc_2,beta,beta_2
      E_model_fiber(ii).E_3     = E_3                    !  ファイバーの第三剛性 E3
      E_model_fiber(ii).Q_2     = Q_2                    !  ファイバーの第二折れ点
      E_model_fiber(ii).Ec_1    = Ec_1                   !  ファイバーの圧縮側第一剛性 E1
      E_model_fiber(ii).Ec_2    = Ec_2                   !  ファイバーの圧縮側第二剛性 E2
      E_model_fiber(ii).Ec_3    = Ec_3                   !  ファイバーの圧縮側第三剛性 E3
      E_model_fiber(ii).Qc_1    = Qc_1                   !  ファイバーの圧縮側第一折れ点
      E_model_fiber(ii).Qc_2    = Qc_2                   !  ファイバーの圧縮側第二折れ点
      E_model_fiber(ii).Beta    = beta                   !  移動硬化用パラメータ
      E_model_fiber(ii).Beta_2  = beta_2                 !  移動硬化用パラメータ
      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-20
  811 continue
      goto 800
```

```
 812  continue
      goto 800
 813  continue
      goto 800
 814  continue
      goto 800
 815  continue
      goto 800
 816  continue
      goto 800
 817  continue
      goto 800
 818  continue
      goto 800
 819  continue
      goto 800
 820  continue
      goto 800
      else
      goto(821,822,823,824,825,826,827,828,829,830),nm_type-100                    ! 6
 821  continue
      read(5,*,err=999) E_3,Q_2                              ! 個人用新規ファイバー履歴
      goto 800
 822  continue
      read(5,*,err=999) E_3,Q_2                              ! 個人用新規ファイバー履歴
      goto 800
 823  continue
      goto 800
 824  continue
      goto 800
 825  continue
      goto 800
 826  continue
      goto 800
 827  continue
      goto 800
 828  continue
      goto 800
 829  continue
      goto 800
 830  continue
      goto 800
      endif
 800  continue
      enddo
c――――――――――――――――――――――――――――★ファイバーモデル剛性出力
      call Fiber_output(E_model_fiber(ii-nmm+1),nmm)
      enddo
c――――――――――――――――――――――――――――★ファイバー履歴特性数セット
      n_m_bilinear    = 0
      n_m_trilinear   = 0
      n_m_concrete    = 0
      n_m_analogy     = 0
      do i=1,n_member
```

```
      ie  = Member(i).nm_element
      imm = Element(ie).n_element
      im  = Member(i).n_model_type
c-----------------------★モデル１１                                    ! x3
      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
            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
     *       .or._model_fiber(nmm).nm_type.eq.101) then  ! 新規履歴モデルのための追加コード
             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
         ii  = E_model11(imm).n_section_2
         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
     *       .or._model_fiber(nmm).nm_type.eq.101) then  ! 新規履歴モデルのための追加コード
            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
       ・
       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                    ! x4
      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
```

1. サブルーチン Fiber_input()は､対応するファイルを2度読みする。
   従って、2 箇所にデータを入力するコードが存在するので、同時に
   変更しなければならない。また、このファイバーデータの仕様を変
   更する場合は、静的解析と静的・動的プレゼンターの入力用プログ

ラムも変更しなければならない。まず、ここでは、ファイバーモデルの標準入力の第 1 レコードを入力する。変数 nm_type は、ファイバーの履歴特性番号を表す。ファイバーの履歴は階層構造を構成している。

2．ファイバーの履歴特性番号毎に入力仕様が異なるため、階層構造に従って処理を分類し、入力用コードを別個に記述する。

3．個人用新規ファイバー履歴の第 2 レコードを設計し、履歴特性番号 101 によりこの位置入力用コードを追加する。

4．2 度目のファイル読み込みを行う部分であり、標準第１レコードを読み込む。

5．ファイバーの履歴特性番号毎に入力仕様が異なるため、階層構造に従って処理を分類し、入力用コードを別個に記述する。

6．個人用新規ファイバー履歴の第 2 レコードを設計し、履歴特性番号 101 によりこの位置に入力用コードを記述する。つまり、新規ファイバー用履歴特性の入力データを設定する場合、標準仕様の第 1 レコードと残りのデータを第 2 レコードとして、ここと 3.の部分で入力するコードを追加する。

x1. ここでは、現在設計されている部材モデルのどの位置で、このファイバー断面が使用されているかをチェックする。もし使用している場合は、要素構造体にファイバー断面番号をセットする。以下の処理では､部材モデル 12､13 などファイバー断面に関連するモデルは、全てここでファイバー断面とリンクを取らなければならない。詳細は、次節の部材モデルの新規設計で説明する。

x2. 上記と同じく、このファイバー断面がどの部材モデルで使用されているかチェックする。ただし、上記は全要素についてであるが、ここでは全部材についてチェックする。

x3. ここでは、断面内で使用しているファイバーの履歴を解析するために必要となるワーク領域構造体の数を数えている。ここでは、両端ファイバーモデルについてであり、プログラムコードを見ると、i 端と j 端の 2 箇所でこの構造体の数を数えている。他の部材モデルでも同様の検索を行っていることは当然のことである。現在、このワーク領域構造体は、3 種類用意されており、その種類に対応する解析モデル中に存在するファイバー数分必要となる。その 3 種類とは、

１．バイリニア型構造体：Bilinear_work(:)

２．トリリニア型構造体：Trilinear_work(:)

３．コンクリート型構造体：Concrete_work(:)

である。ここで、チェックを行っているコードの順番は、上記のバイリニア型、トリリニア型、コンクリート型の順であり、その中の番号は履歴番号である。

新規の履歴モデルが追加される場合で、上記の３つのワーク領域を使用する場合は、コメントに示したコードを追加することになる。なお、追加する履歴特性の番号は 101 であり、使用するワーク領域はトリリニア型構造体とした。新規の履歴モデルが新しいワーク領域構造体を必要とする場合は、その構造体を新たに設計し、さらに、その数をここで数える必要があり、そのコードを追加することになる。もちろん、他の部材モデルでこの新規履歴モデルを組み込む場合は、該当する部分に同様の追加コードを加える必要がある。

x4. 最後に、各種の部材モデルで必要となる３つワーク領域構造体の数が、構造体成分 model_type.n_m_bilinear などにセットされる。当然、新しいワーク領域構造体を設定した場合は、ここでその個数をセットする必要があり、そのコードを追加する。

9.3.3 モデルの出力仕様

本節では、この新しい履歴モデルに関する出力仕様について説明する。出力データは他のモジュールで使用されており、変更する場合は、他のシステム、例えば、動的プレゼンターなどをチェックし、整合性を取る必要がある。

まずは、通常の部材モデルにおける標準出力仕様を見てみよう。解析結果を出力するサブルーチン Out_stress()は、submain_dynamic_a()の中でコールされている。ここでは、両端ファイバーモデルに関連する部分のみ示す。

```
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
      •
      •
      do i=1,n_member
      if(Member(i).element_type.eq.6) then
c――――――――――――――――――――――――――――――★Maxwell モデル
      •
      •
      elseif(Member(i).element_type.eq.2) then
c――――――――――――――――――――――――――――――★3 次元せん断弾塑性モデル
      •
      •
      elseif(Member(i).element_type.eq.3) then
c――――――――――――――――――――――――――――――★3 次元軸力弾塑性モデル
      •
      •
      else
  c――――――――――――――――――――――――――――――★静的縮合モデル（ファイバーモデルなど）
      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
      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
      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)
      endif
      enddo
      return
      end
```

このサブルーチンの該当する部分をみると、部材断面の応力が構造体成分 Member().stress()にセットされていれば、ファイバーデータに全く関係しない。したがって、ここでは、新規ファイバー履歴モデルを追加したとしても変更することはない。

次に、断面のファイバー応力とひずみの出力に関連する２つのサブルーチンを検討しよう。まず出力部材の断面をチェックし、断面ファイバーデータのヘッダー部分を出力するサブルーチン out_section_check() を見てみよう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /out_section_check
C     ―――――――――――――――――――――――――――――――――
C     ●   出力部材の断面応力をチェック
C     ―――――――――――――――――――――――――――――――――
      subroutine out_section_check(Member,Element,
     *               n_member,No_section,Out_section,
     *               ifl,iflz,i_print)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / Member_s  / Member
      record / Element_s  / Element
      record / Out_section_s  / Out_section
      dimension Member(*),Element(*),No_section(*)
      dimension ifl(16),iflz(16)
C     ―――――――――――――――――――――――――――――――――
C     ●   out_section_s 構造体
C     ―――――――――――――――――――――――――――――――――
c     解析パラメータ
c       structure  / out_section_s/
c       integer    n_member        !  出力部材数
c       integer    n_section       !  出力断面数
c       integer    no_member(10)   !  部材番号
c       integer    no_section(11)  !  断面数
c       integer    no_fiber(30)    !  ファイバー数
c       end structure
c       record /out_section_s/ Out_section
      if(ifl(6).eq.0) return
      nn=No_section(1)                          ! 出力断面数                    ! 1
```

```
      if(nn.gt.0) then
      iix=0
      iiy=0
      do i=1,nn                                                    ! 2
      n=No_section(i+1)
      if(n.ge.1.and.n.le.n_member)then                             ! 3
      im=Member(n).element_type
      ie= Member(n).nm_element
      if(im.eq.11.or.im.eq.13) then                                ! 4
      iix=iix+1
      iiy=iiy+2
      Out_section.no_member(iix) = n
      Out_section.no_section(iix)= 2
      Out_section.no_fiber(iiy-1)= Element(ie).nm_section(1)
      Out_section.no_fiber(iiy)  = Element(ie).nm_section(2)
      Out_section.nm_fiber(iiy-1)= Element(ie).n_section(1)
      Out_section.nm_fiber(iiy)  = Element(ie).n_section(2)
      elseif(im.eq.18) then                                        ! 5
      iix=iix+1
      iiy=iiy+1
      Out_section.no_member(iix) = n
      Out_section.no_section(iix)= 1
      Out_section.no_fiber(iiy)= Element(ie).nm_section(1)
      Out_section.nm_fiber(iiy)= Element(ie).n_section(1)
      elseif(im.eq.12.or.im.eq.14) then                            ! 6
      iix=iix+1
      iiy=iiy+3
      Out_section.no_member(iix) = n
      Out_section.no_section(iix)= 3
      Out_section.no_fiber(iiy-2)= Element(ie).nm_section(1)
      Out_section.no_fiber(iiy-1)= Element(ie).nm_section(2)
      Out_section.no_fiber(iiy)  = Element(ie).nm_section(3)
      Out_section.nm_fiber(iiy-2)= Element(ie).n_section(1)
      Out_section.nm_fiber(iiy-1)= Element(ie).n_section(2)
      Out_section.nm_fiber(iiy)  = Element(ie).n_section(3)
      endif
      endif
      enddo
      Out_section.n_member = iix
      Out_section.n_section = iiy
      else
      Out_section.n_member = 0
      endif
c―――――――――――――――――――――――――――――★断面応力ヘッダー出力
      write(iflz(6)) Out_section.n_member                          ! 7
      if(Out_section.n_member.eq.0) return
      write(iflz(6))(Out_section.no_member(j),j=1,Out_section.n_member)
      write(iflz(6))(Out_section.no_section(j),j=1,Out_section.n_member)
      write(iflz(6)) Out_section.n_section
      write(iflz(6))(Out_section.nm_fiber(j),j=1,Out_section.n_section)
      write(iflz(6))(Out_section.no_fiber(j),j=1,Out_section.n_section)
c―――――――――――――――――――――――――――――★断面応力ヘッダー出力終了
      write(76,'(a,i4)') ' No. member: ',Out_section.n_member
      write(76,'(a,i4)') ' No. section: ',Out_section.n_section
```

```
      if(Out_section.n_member.eq.0) return
      return
      end
```

1. ユーザーが出力を要求している部材数を取得する。
2. 出力部材数分、以下の処理を行う。
3. 出力要求を行っている部材番号が解析モデルの部材の範囲かどうかチェックする。範囲外の場合はその部材を無視する。
4. 部材モデルのコード番号が 11 と 13 の場合は、両端ファイバー断面もしくは両端 MS 断面であり、その管理番号をファイバー断面出力構造体 Out_section にセットする。
5. 部材モデルのコード番号が 18 の場合は、両端ピンで中央ファイバー断面を有するモデルであり、その管理番号をファイバー断面出力構造体 Out_section にセットする。
6. 部材モデルのコード番号が 12 と 14 の場合は、両端・中央ファイバー断面もしくは両端・中央 MS 断面であり、その管理番号をファイバー断面出力構造体 Out_section にセットする。
7. 断面ファイバー応力出力ファイルのヘッダー部分を出力する。

次に、実際の断面内のファイバー応力とひずみを出力するサブルーチン Out_Fiber()を見てみよう。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_Fiber
C       ―――――――――――――――――――――――――――――――
C     ●   部材の断面応力を出力(ok)
C     ―――――――――――――――――――――――――――――――
      subroutine Out_Fiber( )
      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                                    ! 1
      i=Out_section.no_member(j)                                     ! 2
c――――――――――――――――――――――――――――――★断面応力
      ie = Member(i).nm_element
      imm= Element(ie).n_element                  ! 要素番号
      immm= Member(i).n_model_type                ! モデルタイプ別番号
      iet  = Member(i).element_type
      if(iet.eq.11) then                                             ! 3
c――――――――――――――――――――――――――――――★モデル１１
      nm_div=E_model11(imm).n_section_1
      nnm=M_model11(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_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
      .
      .
      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
      .
      .
      elseif(iet.eq.101.or.iet.eq.102) then                    ! 4
c―――――――――――――――――――――――★個人用新規部材モデル
```

```
      endif
c――――――――――――――――――――――――――――――★断面応力出力終わり
      enddo
      return
      end
```

1. ユーザーが出力を要求している部材数で以下の出力処理を行う。
2. 出力部材番号を取得する。
3. 両体ファイバーの部材モデル 11 について出力処理を行う。ここのコードを見れば理解できるように、履歴特性に関連するコードはない。そのため、新規ファイバー履歴特性を追加しても、このサブルーチンは変更する必要はない。
4. ここでは、次節以降で説明するファイバー断面を有する部材モデルを新規に作成する場合、この部分に出力コードを追加する必要がある。

9.3.4 構造体の定義

本節では、新規のファイバー用履歴モデルによって、新たな構造体を作成する必要がある場合について説明する。現在、ファイバーを含む部材モデルは全て静的縮合モデルであり、弾性はりエレメントとファイバーによって弾塑性状態を表すエレメント部分などによって構成される。弾性はりに関する情報の保存とワーク領域として、標準型の Element_s、Member_s 構造体が用いられている。ファイバーが集まった断面に関するワーク領域としては M_model_fiber_s が使用され、さらに、ファイバーに関する情報の確保とワーク領域として、構造体 E_model_fiber_s と Bilinear_work、Trilinear_work、Concrete_work が使用されている。なお、部材の弾塑性状態を表すエレメントである MS モデルとアナロジーモデルに関する構造体も、ファイバーモデルと同じ構造体を兼用している。

さて、新たにファイバーモデルの履歴特性を設計する場合であるが、標準入力仕様で、しかも解析中データが変化しないファイバー要素構造体 E_model_fiber_s（鉄骨及び鉄筋用）、及び E_model_fiberc_s（コンクリート用）と解析中データが変化するファイバー部材構造体 M_model_fiber_s の内容で十分な場合は、新たに構造体を設計する必要はない。しかし、入力データ数が多数で現在の仕様では不足する場合や、ワーク領域が不足する場合は、新たに構造体を設計する必要が発生する。

これに伴って、既存のコードを変更しなければならなくなる。これについては、後で解説する。しかし、標準仕様の構造体の内容を変更することで使用可能であれば、登録手続きは単純である。

まずは、標準仕様であるファイバー要素構造体 E_model_fiber_s と E_model_fiberc_s を示そう。両者の並びで対応する変数は同じ記憶領域を占めており、構造体配列の数はファイバー要素数分設定される。

```
C     ―――――――――――――――――――――――――――――――――
C     ●　ファイバーモデル E_model_fiber_s 構造体（バイリニアー、トリリニア用）
C     ―――――――――――――――――――――――――――――――――
      structure / E_model_fiber_s/
      integer  nm_type              !  履歴モデルの番号
      real*8   E_1                  !  ファイバーの第一剛性 E1
      real*8   E_2                  !  ファイバーの第二剛性 E2
      real*8   E_3                  !  ファイバーの第三剛性 E3
      real*8   Q_1                  !  ファイバーの第一折れ点
      real*8   Q_2                  !  ファイバーの第二折れ点
      real*8   Ec_1                 !  ファイバーの圧縮側第一剛性 E1
      real*8   Ec_2                 !  ファイバーの圧縮側第二剛性 E2
      real*8   Ec_3                 !  ファイバーの圧縮側第三剛性 E3
      real*8   Qc_1                 !  ファイバーの圧縮側第一折れ点
      real*8   Qc_2                 !  ファイバーの圧縮側第二折れ点
      real*8   G                    !  ファイバーせん断剛性 G
      real*8   A                    !  ファイバー断面積
      real*8   Ay                   !  ファイバーｙ軸せん断用断面積
      real*8   Az                   !  ファイバーｚ軸せん断用断面積
      real*8   ry                   !  中立軸から断面中心までのｙ方向距離
      real*8   rz                   !  中立軸から断面中心までのｚ方向距離
      real*8   Ary                  !  断面積掛ける距離
      real*8   Arz                  !  断面積掛ける距離
      real*8   Ary2                 !  断面積掛ける距離の２乗
      real*8   Arz2                 !  断面積掛ける距離の２乗
      real*8   Aryz                 !  断面積掛ける距離の２乗
      real*8   Beta                 !  移動硬化用パラメータ
      real*8   Beta_2               !  移動硬化用パラメータその２
      end structure
c       record / E_model_fiber_s    / E_model_fiber
c
C     ―――――――――――――――――――――――――――――――――
C     ●　ファイバーモデル E_model_fiberc_s 構造体（コンクリート用）
C     ―――――――――――――――――――――――――――――――――
      structure / E_model_fiberc_s/
      integer  nm_type              !  履歴モデルの番号
      real*8   AK_1                 !  圧縮と引張　第一勾配
      real*8   AK_2                 !  圧縮第二勾配
      real*8   AK_3                 !  圧縮第三勾配
      real*8   Q_1                  !  引張強度
      real*8   Q_2                  !  圧縮側第一折れ点の応力
      real*8   Q_3                  !  圧縮強度
      real*8   Q_4                  !  圧縮流れ点
      real*8   AK_4                 !  引張第二勾配
```

```
      real*8   dm                  !  ダミー
      real*8   dm1                 !  ダミー
      real*8   G                   !  ファイバーせん断剛性 G
      real*8   A                   !  ファイバー断面積
      real*8   Ay                  !  ファイバーｙ軸せん断用断面積
      real*8   Az                  !  ファイバーｚ軸せん断用断面積
      real*8   ry                  !  中立軸から断面中心までのｙ方向距離
      real*8   rz                  !  中立軸から断面中心までのｚ方向距離
      real*8   Ary                 !  断面積掛ける距離
      real*8   Arz                 !  断面積掛ける距離
      real*8   Ary2                !  断面積掛ける距離の２乗
      real*8   Arz2                !  断面積掛ける距離の２乗
      real*8   Aryz                !  断面積掛ける距離の２乗
      real*8   Beta                !  移動硬化用パラメータ
      real*8   Beta_2              !  移動硬化用パラメータその２
      end structure
c       record / E_model_fiberc_s    / E_model_fiber
```

上の構造体 E_model_fiber_s をそのまま利用することも可能であるが、新規の履歴モデル用として、新たに設計することもできる。上に示した構造体 E_model_fiber_s を参照して作成すれば良い。この構造体は、ファイバー要素データを保存するためのものであり、部材毎に計算途中で変化するデータを入れるものではない。これについては、ファイバー部材構造体 M_model_fiber_s が用意されている。また、この構造体の全ての成分が使用可能となっておらず、上記の M_model_fiber_s の中で濃い文字で示したエレメントはシステムが使用する。

次に、ファイバー部材構造体 M_model_fiber_s を示そう。ここでも、新たな構造体を設計したい場合は、この構造体 M_model_fiber_s を参照して作成されたい。

```
C     ―――――――――――――――――――――――――――――――
C     ●   ファイバーモデル M_model_fiber_s 構造体
C     ―――――――――――――――――――――――――――――――・
      structure / M_model_fiber_s/
      integer  n_type              !  履歴モデルの通し番号
      integer  i_elastic           !  ファイバー要素の状態（弾性の場合は０：塑性は１）
      real*8   d_eps_x             !  軸方向歪
      real*8   d_stress_x          !  軸方向応力
      real*8   d_E                 !  接線剛性
      end structure
c       record / M_model_fiber_s    / M_model_fiber
```

さらに、ファイバーモデルでは、ワーク用として履歴特性に合わせて３つの構造体が設計され、使用されている。この３つの構造体は、現在以下の８つの履歴特性で用いられている。

１．バイリニア型ワーク領域構造体：Bilinear_work_s
　　１．対称バイリニア型：**BiLinear()**
　　５．バイリニア型（移動＋等方硬化用）：**BiLinear_h()**
　　７．非対称バイリニア型：**TriLinear_AS()**

２．トリリニア型ワーク領域構造体：Trilinear_work_s
　　２．対称トリリニア型：**TriLinear()**
　　６．対称トリリニア型（移動＋等方硬化用）：**TriLinear_h()**
　　８．非対称トリリニア型：**BiLinear_AS()**

３．コンクリート型ワーク領域構造体： Concrete_work_s
　　３．直線コンクリート履歴モデル：**Concrete()**
　　４．曲線コンクリート履歴モデル：**Concrete_e()**

上記３つのワーク領域構造体を以下に示そう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●　履歴モデル　Bilinear：ファイバー要素 用　Work エリア構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / Bilinear_work_s/
      integer  i_stat              !  ファイバー要素の状態
      real*8   P1                  !  ワーク領域
      real*8   P2                  !  ワーク領域
      real*8   Sig_z               !  ワーク領域
      end structure
c       record / Bilinear_work_s    / Bilinear_work
c
C     ―――――――――――――――――――――――――――――――――
C     ●　履歴モデル　Trilinear：ファイバー要素 用　Work エリア構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / Trilinear_work_s/
      integer  i_stat              !  ファイバー要素の状態
      real*8   P1(5)               !  ワーク領域
      end structure
c       record / Trilinear_work_s    / Trilinear_work
c
C     ―――――――――――――――――――――――――――――――――
C     ●　履歴モデル　Concrete：ファイバー要素 用　Work エリア構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / Concrete_work_s/
      integer  i_stat              !  ファイバー要素の状態
      integer  ipret               !  過去の引張履歴
      integer  ipre_c              !  過去の圧縮履歴
      real*8   P1(6)               !  ワーク領域
      end structure
c       record / Concrete_work_s    / Concrete_work
```

標準仕様の構造体を使用する場合は、新たな構造体を必要としないが、その新規モデルに適合した構造体を設計する場合は、その構造体を新たなヘッダーファイルに作成する必要がある。このヘッダーファイルを New_submain.h としよう。読者はまずこの New_submain.h ファイルを作り、この中に新たな構造体を書き込むことになる。新たなヘッダーファイルは、この構造体を使用する場合に、そのサブルーチンの中にインクルードされる。

新規にファイバー要素の履歴特性を設計する際、入力データやワーク領域が先に示した構造体では対応できない場合、新たな構造体を設計し、使用することになる。その際、多くの手続きが必要となり、既存のプログラムに手続きのためのコードを追加する必要が生じる。その手続きを知るためにも、現在の構造体がどのような手続きで利用可能となっているか調査しよう。

現在、ファイバー履歴に関連する構造体として、ファイバー断面に関する構造体 2 種類、ファイバーの履歴に関する構造体 3 種類が用いられている。これらの構造体は、submain_dynamic_a()サブルーチンの中で、構造体の手続きにしたがって、宣言、動的確保、領域解放が行われている。その手続きの幾つかを以下に示す。

```
c―――――――――――――――――――――――――――――――★ファイバーモデル用エリア
      record / E_model_fiber_s   / E_model_fiber
      record / M_model_fiber_s   / M_model_fiber
c―――――――――――――――――――――――――――――――★履歴モデル用エリア
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
c―――――――――――――――――――――――――――――――★ファイバーモデル用エリア
      ALLOCATABLE :: E_model_fiber(:)
      ALLOCATABLE :: M_model_fiber(:)
c―――――――――――――――――――――――――――――――★履歴モデル用エリア
      ALLOCATABLE :: Bilinear_work(:)
      ALLOCATABLE :: Trilinear_work(:)
      ALLOCATABLE :: Concrete_work(:)
```

これらの構造体は、次に示すように解析モデルで使用する数に従って、動的に領域を確保されなければならない。当然、この構造体の数をどこかで数えておく必要がある。

```
c―――――――――――――――――――――――――――――――★ファイバーモデル領域セット
      n= Model_type.nm_div_fmodel          !部材内のファイバー要素の数
      if(n.ne.0) then
      ALLOCATE (M_model_fiber(n))
      endif
      n= Model_type.nm_div_felement        !ファイバー要素のエレメント最大数
```

```
      if(n.ne.0) then
      ALLOCATE (E_model_fiber(n))
      endif
c―――――――――――――――――――――――――――――――――★ファイバー用履歴要素
      n= Model_type.n_m_bilinear          !  バイリニアの履歴要素の数
      if(n.ne.0) then
      ALLOCATE (Bilinear_work(n))
      endif
      n= Model_type.n_m_trilinear         !  トリリニアの履歴要素の数
      if(n.ne.0) then
      ALLOCATE (Trilinear_work(n))
      endif
      n= Model_type.n_m_Concrete          !  コンクリートの履歴要素の数
      if(n.ne.0) then
      ALLOCATE (Concrete_work(n))
      Endif
```

これらの構造体は動的に確保する配列であることから、サブルーチンsubmain_dynamic_a()のなかで確保する必要がある。手続きとしては、動的確保の他に、動的領域であることの宣言、使用後の領域解放を行うことになる。構造体を新たに設計、使用する場合は、これらの処理を必ず既存プログラムの中に書き込む必要が生じる。

9.3.5 必要となるサブルーチン

次に、この履歴特性 New_model_fiber を SPACE に組み込むために、必要となるサブルーチンについて説明する。この必要となるサブルーチンは

１）線形剛性を求めるサブルーチン
２）非線形剛性を求めるサブルーチン
３）応力を求めるサブルーチン
４）応力をチェックし、接線剛性を求めるサブルーチン

である。ただし、この中で標準の静的縮合モデルに関連するサブルーチンを転用できる場合は、そのサブルーチンを使用すれば良い。まず、上記４つの標準的な静的縮合モデルに関するサブルーチンを検討しよう。例として、両端ファイバー部材モデルを取り上げる。

最初は、線形剛性行列を求めるサブルーチン Fiber_Model_GI11()であり、次に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_Model_GI11
C     ―――――――――――――――――――――――――――――――――
C     ●   線形剛性行列の計算(ok)
C     ―――――――――――――――――――――――――――――――――
```

```
C     ●   Model_No.11 両端ファイバーモデル
C     ―――――――――――――――――――――――――――――――――――
      subroutine Fiber_Model_GI11(it,ak,alength,Member,Element,
     *          E_model,E_model_fiber,M_model,M_model_fiber,
     *          Bilinear_work,Trilinear_work,Concrete_work,New_fiber_work)
      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_model
      record / E_model_fiber_s    / E_model_fiber
      record / M_model11_s        / M_model
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      record / New_fiber_work_s   / New_fiber_work
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*),New_fiber_work(*)
      dimension ak(12,12)
c―――――――――――――――――――――――――――――――――――――――★
      if(it.eq.1) then
      do i=1,30
      M_model.ff_ip(i) = 0.
      enddo
      nm_div = E_model.n_section_1
      nn     = E_model.nm_section_1 - 1
      nnm    = M_model.nm_section_1 - 1
      elseif(it.eq.2) then
      nm_div = E_model.n_section_2
      nn     = E_model.nm_section_2 - 1
      nnm    = M_model.nm_section_2 - 1
      endif
      do i=1,nm_div
      nn     =  nn  + 1
      nnm    =  nnm + 1
c―――――――――――――――――――――――――――――――――――――――★データの初期設定
      M_model_fiber(nnm).d_E        = E_model_fiber(nn).E_1
      M_model_fiber(nnm).i_elastic  = 0        !  ファイバー要素の状態（弾性の場合は０：塑性は１）
      M_model_fiber(nnm).d_eps_x    = 0.       !  軸方向歪
      M_model_fiber(nnm).d_stress_x = 0.       !  軸方向応力
c―――――――――――――――――――――――――――――――――――――――★ファイバーデータセット
      E_model_fiber(nn).Ary  = E_model_fiber(nn).A*E_model_fiber(nn).ry
      E_model_fiber(nn).Arz  = E_model_fiber(nn).A*E_model_fiber(nn).rz
      E_model_fiber(nn).Ary2 =
     *                       E_model_fiber(nn).Ary*E_model_fiber(nn).ry
      E_model_fiber(nn).Arz2 =
     *                       E_model_fiber(nn).Arz*E_model_fiber(nn).rz
      E_model_fiber(nn).Aryz =
     *                       E_model_fiber(nn).Arz*E_model_fiber(nn).ry
      nmx                    = M_model_fiber(nnm).n_type
      nm_type                = E_model_fiber(nn).nm_type
```

```
      if(nm_type.le.10) then
      goto ( 10,20,30,30,10,20,10,20),nm_type
   10 continue
c-----------------------★バイリニア型（スチール用）
      Bilinear_work(nmx).i_stat  = -1              !  ファイバー要素の状態
      goto 100
   20 continue
c-----------------------★非対称トリリニア型
      Trilinear_work(nmx).i_stat = -1              !  ファイバー要素の状態
      goto 100
   30 continue
c-----------------------★コンクリート型
      Concrete_work(nmx).i_stat  = -1              !  ファイバー要素の状態
      goto 100
      else
      goto ( 110,120),nm_type-100
  110 continue
c-----------------------★個人用ファイバーモデル
      New_fiber_work(nmx).i_stat  = -1             !  新規ファイバー要素の状態
      goto 100
  120 continue
c-----------------------★
      goto 100
  100 continue
      enddo
c-------------------------------★ファイバー要素のデータセット
      if(it.eq.1) then
      nm_div = E_model.n_section_1
      nn     = E_model.nm_section_1 - 1
      nnm    = M_model.nm_section_1 - 1
      elseif(it.eq.2) then
      nm_div = E_model.n_section_2
      nn     = E_model.nm_section_2 - 1
      nnm    = M_model.nm_section_2 - 1
      endif
      ra     = 0.
      ray    = 0.
      raz    = 0.
      raz2   = 0.
      ray2   = 0.
      rayz   = 0.
      gg     = 0.
      aa     = 0.
      do i=1,nm_div
      nn     = nn+1
      nnm    = nnm+1
      A      = E_model_fiber(nn).A
      E      = M_model_fiber(nnm).d_E
      ra     = ra   + E*A
      ray    = ray  + E*E_model_fiber(nn).Arz
      raz    = raz  + E*E_model_fiber(nn).Ary
      ray2   = ray2 + E*E_model_fiber(nn).Arz2
      raz2   = raz2 + E*E_model_fiber(nn).Ary2
      rayz   = rayz + E*E_model_fiber(nn).Aryz
```

```
      aa     = aa   + A
      gg     = gg   + A*E_model_fiber(nn).G
      enddo
      gg     = gg / aa
      if(it.eq.1) then
      M_model.d_aa_1     = aa              !  断面積の和
      M_model.d_ra_1     = ra              !  E*断面積の和
      M_model.d_ray_1    = ray             !  E*A*z
      M_model.d_raz_1    = raz             !  E*A*y
      M_model.d_raz2_1   = raz2            !  E*A*y*y
      M_model.d_ray2_1   = ray2            !  E*A*z*z
      M_model.d_rayz_1   = rayz            !  E*A*z*y
      M_model.d_gg_1     = gg              !  G*A
      M_model.d_epsi_x_1 =  0.             !  軸方向歪
      M_model.d_epsi_y_1 =  0.             !  y 軸に関する曲げ歪
      M_model.d_epsi_z_1 =  0.             !  z 軸に関する曲げ歪
      else
      M_model.d_aa_2     = aa              !  断面積の和
      M_model.d_ra_2     = ra              !  E*断面積の和
      M_model.d_ray_2    = ray             !  E*A*z
      M_model.d_raz_2    = raz             !  E*A*y
      M_model.d_raz2_2   = raz2            !  E*A*y*y
      M_model.d_ray2_2   = ray2            !  E*A*z*z
      M_model.d_rayz_2   = rayz            !  E*A*z*y
      M_model.d_gg_2     = gg              !  G*A
      M_model.d_epsi_x_2 =  0.             !  軸方向歪
      M_model.d_epsi_y_2 =  0.             !  y 軸に関する曲げ歪
      M_model.d_epsi_z_2 =  0.             !  z 軸に関する曲げ歪
      endif
c-----------------------------------★初期剛性行列の計算
      call Fiber_Model_G11(it,ak,alength,Member,Element,
     *     E_model,E_model_fiber,M_model,M_model_fiber)
      return
      end
```

このサブルーチンを見れば分かるように、ファイバー要素に関する構造体の中で、Bilinear_work、Trilinear_work、Concrete_work 以外に、新たな構造体 New_fiber_work を使用する場合は、太文字で示した部分を変更し、初期設定を行うことになる。

次に、非線形剛性を求めるサブルーチン Stiff_M11()について調査する。このプログラムは、以下のように履歴モデルに関連するコードを含まない。そのため、新規履歴モデルを追加しても非線形剛性を求めるサブルーチンは影響しない。

```
C     ------------------------------------------
C     ●   SUBROUTINE /Stiff_M11
C       ------------------------------------------
C     ●   接線剛性行列の計算(ok)
C     ------------------------------------------
      subroutine Stiff_M11(im,it,n_type,ak,Member,alength,
```

```
     *       Model_type,Element,
     *     E_model11, E_model_fiber,M_model11, M_model_fiber)
C     ―――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s          / Member
      record / element_s         / Element
      record / n_model_s         / Model_type
      record / E_model11_s       / E_model11
      record / M_model11_s       / M_model11
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension ak(12,12),alength(*)
      dimension vv(12)
      dimension E_model_fiber(*),M_model_fiber(*)
c―――――――――――――――――――――――――――――――――――――★要素及びモデルのセット
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      goto(11,12,13,14,15,16,17,18,19,20),n_type
   11 continue
c―――――――――――――――――――――――――――――――――――――★弾性要素
      call Cal_nonlin_stiff_Mx(Member,Element,
     *        Member.stress(7),ak,alength(im) )
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――――――――★ファイバー要素
      it=it+1
      call Fiber_Model_G11(it,ak,alength(im),Member,Element,
     *      E_model11,E_model_fiber,
     *      M_model11,M_model_fiber)
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――――――――★せん断バネ要素
      call Shear_G(it,ak,Member)
      goto 100
   14 continue
c―――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
   15 continue
c―――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
   16 continue
c―――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
   17 continue
c―――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
   18 continue
c―――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
```

```
   19 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――★ダミー
      goto 100
   20 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――★ダミー
  100 continue
      return
      end
C     ――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_Model_G11
C     ――――――――――――――――――――――――――――――――――――――――――――――――――
C     ●   接線剛性行列の計算
C     ――――――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Fiber_Model_G11(it,ak,alength,Member,Element,
     *      E_model,E_model_fiber,M_model,M_model_fiber)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
C     ――――――――――――――――――――――――――――――――――――――――――――――――――
      record / member_s          / Member
      record / element_s         / Element
      record / E_model11_s       / E_model
      record / M_model11_s       / M_model
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension ak(12,12)
c――――――――――――――――――――――――――――――――――――――――――――――――――――★
      if(it.eq.1) then                  !  it:部材位置（１：ｉ端、 ２：ｊ端）
      aa   = M_model.d_aa_1             !  断面積の和
      ra   = M_model.d_ra_1             !  E*断面積の和
      ray  = M_model.d_ray_1            !  E*A*z
      raz  = M_model.d_raz_1            !  E*A*y
      raz2 = M_model.d_raz2_1           !  E*A*y*y
      ray2 = M_model.d_ray2_1           !  E*A*z*z
      rayz = M_model.d_rayz_1           !  E*A*z*y
      gg   = M_model.d_gg_1             !  G*A
      else
      aa   = M_model.d_aa_2             !  断面積の和
      ra   = M_model.d_ra_2             !  E*断面積の和
      ray  = M_model.d_ray_2            !  E*A*z
      raz  = M_model.d_raz_2            !  E*A*y
      raz2 = M_model.d_raz2_2           !  E*A*y*y
      ray2 = M_model.d_ray2_2           !  E*A*z*z
      rayz = M_model.d_rayz_2           !  E*A*z*y
      gg   = M_model.d_gg_2             !  G*A
      endif
      rix  = Element.RIx
      asy  = Element.ASy
      asz  = Element.ASz
c――――――――――――――――――――――――――――――――――――――――――――――――――――★
      call Fiber_GK(ak,alength,ra,ray,raz,
     *              raz2,ray2,rayz,gg,aa,rix,asy,asz)
      return
```

```
      end
```

次に、応力を求めるサブルーチンを調べよう。以下に示すサブルーチン Cal_stress_M11()の内容から、静的縮合型部材モデルでは、特殊な応力計算を行わない限り、現在のサブルーチンが転用可能となる。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress_M11（両端ファイバーモデル）
C       ――――――――――――――――――――――――――――――――
C     ●   代表的な部材モデルの応力計算：非線形剛性を用いて、材端応力を計算する
C     ――――――――――――――――――――――――――――――――――
      subroutine Cal_stress_M11(Model_type,Member,Element, ak,vv,r1,r2)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s    / Member
      record / element_s   / Element
      dimension ak(12,12) ,vv(12),r1(3,3),r2(3,3),st(12),ss(12)
C     ――――――――――――――――――――――――――――――――――
      do i=1,12
      s=0.
      do j=1,12
      s=s+ak(i,j)*vv(j)
      enddo
      st(i)=s
      enddo
c――――――――――――――――――――――――――――――★全体座標から部材座標へ変換
      call RotateL_v(1,st,r1,r2,ss)
      do i=1,6
      Member.stress(i)=-ss(i)+Member.stress(i)
      enddo
      do i=7,12
      Member.stress(i)=ss(i)+Member.stress(i)
      enddo
      return
      end
```

以上のように、静的縮合部材モデルのファイバー履歴特性では、新たな構造体を追加しなければ、標準仕様の３つのサブルーチンがそのまま使用できることが分かる。後は応力をチェックし、接線剛性を求めるサブルーチンであるが、このサブルーチンについては、次節で説明しよう。

9.3.6 モデルの階層構造とサブルーチンの組み込み

静定縮合部材モデルでは、サブルーチン submain_dynamic_a()から Check_stress()がコールされ、その中で、両端ファイバーモデルの応力の弾塑性チェックプログラム Cal_check_stiff_M11()が呼び出される。以下にその部分のコードを示す。

```
      goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
c-------------------------------★Model_No.11 両端ファイバーモデル
      call Cal_check_stiff_M11(Control,N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model11, E_model_fiber,
     *      M_model11, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
```

さらに、このサブルーチン Cal_check_stiff_M11()は、以下のような静的縮合に関する処理を行い、ファイバーの弾塑性チェックを行う。このサブルーチンから関連する部分を抜き出してみよう。

```
C     ---------------------------------------
C     ●   SUBROUTINE /Cal_check_stiff_M11
C       -------------------------------------
C     ●   代表的な部材モデルの塑性チェック(両端ファイバーモデル)
C     ---------------------------------------
      subroutine Cal_check_stiff_M11( )
C     ---------------------------------------
       .
       .
c-------------------------------★部材剛性行列の作成
      it = 0
      do i=1,n_div
c-------------------------------★内部部材の剛性行列の計算
      call Stiff_M11(i,it,n_type(i),akk(1,1,i),Member,alength,
     *       Model_type,Element,
     *       E_model11(imm), E_model_fiber,
     *       M_model11(immm), M_model_fiber)
      if(N_analysis.ne.9.and.N_analysis.ne.7.and.n_type(i).ne.3) then    !  幾何学的非線形剛性
      call Cal_geomet_stiffx(Member.an_stress(i),Member,
     *                       akk(1,1,i),alength(i))
      call Create_Kn(akk(1,1,i),Member.an_vv(i),Member.an_ww(i),
     *              EA(i),alength(i))
      endif
c-------------------------------★剛性行列の分配
      call Bnd_FEM(i,akk(1,1,i),irest_Point,ak,c,ab,iubw,n_if)
      enddo
      enddo
       .
       .
c-----------------------------------------------------------
c    ★        部材の弾塑性チェック
c-------------------------------★ファイバーチェック
      if(n_type(i).eq.2) then
      it=it+1
      call Fiber_check(Control,i,N_analysis,
     *      mem_x,it,alength(i),Member,Element,
     *      E_model11(imm),E_model_fiber,M_model11(immm),M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vvx)
      endif
```

```
      ・
      ・
      return
      end
```

このサブルーチンの中で、接線剛性を求める Stiff_M11()サブルーチンがコールされているが、この中でこの静的縮合モデルで使用されるエレメントモデルが階層構造となっている。このサブルーチンは前節で示した。

さらに、このサブルーチンの中で、静定縮合部材モデルである両端ファイバーモデルの応力チェック用サブルーチン Fiber_check()がコールされている。次は、この応力チェック用サブルーチンを見てみよう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_check11
C     ―――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の材料非線形性チェックし、応力を計算
C     ―――――――――――――――――――――――――――――――――
      subroutine Fiber_check(idiv,N_analysis,
     *      mem_x,it,Alength,Member,Element,
     *      E_model,E_model_fiber,M_model,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h "
      record / member_s           / Member
      record / element_s          / Element
      record / E_model11_s        / E_model
      record / E_model_fiber_s    / E_model_fiber
      record / M_model11_s        / M_model
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*)
      dimension vv(12)
c―――――――――――――――――――――――――――――――――――――★i 端部ファイバー
c―――――――――――――――――――――――――――――――――――――★歪のセット
      if(it.eq.1) then
      d_epsi_x_1 =  (vv(7)  - vv(1)) / Alength   !  軸方向歪
      d_epsi_y_1 =  (vv(11) - vv(5)) / Alength   !  y 軸に関する曲げ歪
      d_epsi_z_1 =  (vv(12) - vv(6)) / Alength   !  z 軸に関する曲げ歪
      if(N_analysis.eq.10.or.N_analysis.eq.8) then                ! 非線形軸方向歪
      d_epsi_x_1= d_epsi_x_1+strain_nonlinear(Member.an_vv(idiv),
     *      Member.an_ww(idiv),vv,Alength)
      endif
      if(Member.analysis_3D .eq. 1) d_epsi_z_1 =0.  ! 平面問題における面外方向の曲げ変形を無視する
```

```
      if(Member.analysis_3D .eq. 2) d_epsi_y_1 =0.  ! 平面問題における面外方向の曲げ変形を無視する
      M_model.d_epsi_x_1 =  d_epsi_x_1 + M_model.d_epsi_x_1    !  軸方向歪
      M_model.d_epsi_y_1 =  d_epsi_y_1 + M_model.d_epsi_y_1    !  y軸に関する曲げ歪
      M_model.d_epsi_z_1 =  d_epsi_z_1 + M_model.d_epsi_z_1    !  z軸に関する曲げ歪
c-----------------------------★ファイバー要素のチェック
      nm_div   = E_model.n_section_1
      nn       = E_model.nm_section_1 - 1
      nnm      = M_model.nm_section_1 - 1
      iistat   = 0                          ! 塑性率を計算するための指標
      do i=1,nm_div
      nn       = nn  + 1
      nnm      = nnm + 1
      nmx      = M_model_fiber(nnm).n_type
      nm_type  = E_model_fiber(nn).nm_type
      du =  d_epsi_x_1 +                                    ! 軸方向歪
     *      d_epsi_y_1 * E_model_fiber(nn).rz -              ! y軸に関する曲げ歪
     *      d_epsi_z_1 * E_model_fiber(nn).ry                ! z軸に関する曲げ歪
      M_model_fiber(nnm).d_eps_x = M_model_fiber(nnm).d_eps_x + du !ファイバー要素の歪
      if(N_analysis.le.8.or.Member.nm_analysis.eq.-1) then
c-----------------------------★弾性解析
c                                   ファイバー軸力計算
      M_model_fiber(nnm).d_stress_x = M_model_fiber(nnm).d_E*du +
     *                               M_model_fiber(nnm).d_stress_x
      else
      if(nm_type/10.eq.0) then
c-----------------------------★弾塑性解析
      goto ( 10,20,30,40,50,60,70),nm_type
  10  continue
c-----------------------------★バイリニア型（スチール用）
      call BiLinear()
      goto 100
  20  continue
c-----------------------------★対称トリリニア型 （スチール用）
      call TriLinear()
      goto 100
  30  continue
c-----------------------------★コンクリート型
      call Concrete()
      goto 100
  40  continue
c-----------------------------★曲線コンクリート型
      call Concrete_e()
      goto 100
  50  continue
c-----------------------------★バイリニア型（移動＋等方効果用）
      call BiLinear_h()
      goto 100
  60  continue
c-----------------------------★対称トリリニア型 （移動＋等方効果用）
      call TriLinear_h()
      goto 100
  70  continue
c-----------------------------★非対称バイリニア型
      call BiLinear_AS()
```

```
      goto 100
      elseif(nm_type/10.eq.10) then
c―――――――――――――――――――――――――★弾塑性解析
      goto (110,120),nm_type - 100
  110 continue
c―――――――――――――――――――――――――★個人用ファイバー履歴特性
      call New_model_fiber()
      goto 100
  120 continue
c―――――――――――――――――――――――――★個人用ファイバー履歴特性
      goto 100
      endif
      endif
c―――――――――――――――――――――――――★弾塑性解析終了
  100 continue
      enddo
c―――――――――――――――――――――――――――――――――★ファイバー要素応力の計算
      d_state = float(iistat)/float(nm_div)   ! 塑性した面積の計算
      if(d_state .eq.0) then
      Member.d_stat(1) = 0
      elseif(d_state .ge.0.8) then
      Member.d_stat(1) = 2
      else
      Member.d_stat(1) = 1
      endif
      nm_div  = E_model.n_section_1
      nn      = E_model.nm_section_1 - 1
      nnm     = M_model.nm_section_1 - 1
      ra      = 0.
      ray     = 0.
      raz     = 0.
      raz2    = 0.
      ray2    = 0.
      rayz    = 0.
      gg      = 0.
      aa      = 0.
      AN      = 0.
      AMy     = 0.
      AMz     = 0.
      ra_before = M_model.d_ra_1
      do i=1,nm_div
      nn      = nn  + 1
      nnm     = nnm + 1
      A       = E_model_fiber(nn).A
      E       = M_model_fiber(nnm).d_E
      ra      = ra   + E*A
      ray     = ray  + E*E_model_fiber(nn).Arz
      raz     = raz  + E*E_model_fiber(nn).Ary
      ray2    = ray2 + E*E_model_fiber(nn).Arz2
      raz2    = raz2 + E*E_model_fiber(nn).Ary2
      rayz    = rayz + E*E_model_fiber(nn).Aryz
      aa      = aa + A
      gg      = gg + A*E_model_fiber(nn).G
      ANN     = M_model_fiber(nnm).d_stress_x*E_model_fiber(nn).A
```

```
      AN      = AN + ANN
      AMy     = AMy + ANN * E_model_fiber(nn).rz
      AMz     = AMz + ANN * E_model_fiber(nn).ry
      enddo
      if(iistat.ge.nm_div) then
      ra=ra_before
      endif
      M_model.d_aa_1 = aa                 !  断面積の和
      M_model.d_ra_1 = ra                 !  E*断面積の和
      M_model.d_ray_1 = ray               !  E*A*z
      M_model.d_raz_1 = raz               !  E*A*y
      M_model.d_raz2_1 = raz2             !  E*A*y*y
      M_model.d_ray2_1 = ray2             !  E*A*z*z
      M_model.d_rayz_1 = rayz             !  E*A*z*y
      M_model.d_gg_1  = gg                !  G*A
c―――――――――――――――――――――――――――――――★j 端部ファイバー
c―――――――――――――――――――――――――――――――★歪のセット
      elseif(it.eq.2) then
       •
       •
      do i=1,nm_div
       •
       •
      if(N_analysis.le.8.or.Member.nm_analysis.eq.-1) then
c―――――――――――――――――――――――――★弾性解析
c                                   ファイバー軸力計算
      M_model_fiber(nnm).d_stress_x = M_model_fiber(nnm).d_E * du +
     *                               M_model_fiber(nnm).d_stress_x
      else
      if(nm_type/10.eq.0) then
c―――――――――――――――――――――――――★弾塑性解析
      goto ( 11,21,31,41,51,61,71),nm_type
  11  continue
c―――――――――――――――――――――――――★バイリニア型（スチール用）
      call BiLinear()
      goto 101
  21  continue
c―――――――――――――――――――――――――★対称トリリニア型 （スチール用）
      call TriLinear()
      goto 101
  31  continue
c―――――――――――――――――――――――――★コンクリート型
      call Concrete()
      goto 101
  41  continue
c―――――――――――――――――――――――――★曲線コンクリート型
      call Concrete_e(M_model_fiber()
      goto 101
  51  continue
c―――――――――――――――――――――――――★バイリニア型（移動＋等方効果用）
      call BiLinear_h()
      goto 101
  61  continue
c―――――――――――――――――――――――――★対称トリリニア型 （移動＋等方効果用）
```

```
      call TriLinear_h()
      goto 101
  71  continue
c――――――――――――――――――――――――★非対称バイリニア型
      call BiLinear_AS()
      goto 101
      elseif(nm_type/10.eq.10) then
c――――――――――――――――――――――――★弾塑性解析
      goto (111,121),nm_type - 100
  111 continue
c――――――――――――――――――――――――★個人用ファイバー履歴特性
      call New_model_fiber()
      goto 101
  121 continue
c――――――――――――――――――――――――★個人用ファイバー履歴特性
      goto 101
      endif
      endif
c――――――――――――――――――――――――★弾塑性解析終了
  101 continue
      enddo
c――――――――――――――――――――――――――――――――★ファイバー要素応力の計算
       ・
       ・
      return
      end
```

このサブルーチンの中に履歴モデルに関する第３層の階層構造が見られる。この中に履歴特性番号 101 として、太文字で示すコードがあり、新規モデルが追加されている。この中で、応力を弾塑性チェックする新規履歴モデルに関するサブルーチンが、両端ファイバーモデルであることから 2 箇所に加えられている。ここでは、サブルーチン名を New_model_fiber()としている。これで、既存サブルーチンに新規モデルが組み込まれたことになる。引数として受け渡すデータとして、構造体の Member と Element など多くの構造体や配列が、新規に登録されたサブルーチンには見られるはずである。新規サブルーチンでもデータの受け渡しには細心の注意が必要である。また、特殊な構造体を設計した場合、その構造体は引数として受け渡すことになるが、当然、上位のサブルーチンから受け渡されることになる。

この新規サブルーチンでは、実際にファイバーの履歴を追うことが主な処理であるが、少なくとも構造体 M_model_fiber と３つのワーク領域 Bilinear_work、Trilinear_work、Concrete_work かもしくは新たに設計した New_fiber_work 中で、次の値を設定しなければならない。

1. New_fiber_work (nmx).i_stat が-1 のとき、初期設定を行う。
2. ファイバーの接線剛性(M_model_fiber(nnm).d_E を設定する。
3. ファイバーの応力 M_model_fiber(nnm).d_stress_x を求める。

9.4 静的縮合部材モデルの組み込み方法

9.4.1 部材モデルの設計

本節では、新たな部材モデルを SPACE に組み込む方法について解説する。現在、SPACE には各種の静的縮合モデルが組み込まれている。しかし、そのいずれもが部材中のエレメントの位置やその数などは設計されており、ユーザーが任意に設定することはできない。そこで例題として、静的縮合モデルにおける部材中の各種モデルのエレメントを任意に並べることができ、また履歴モデルを選択できる部材モデルを設計して、SPACE に組み込むことを考えよう。静的縮合モデルを組み込む作業はそれ自体複雑で面倒である。ここでは、さらに部材内部のエレメント数が任意であることによる難しさが重なる。しかし、この例題を理解することで SPACE に関する有用な情報が多数得られることになろう。なお、このモデルは、Ver.2.20 では既に標準モデルとして組み込まれている。

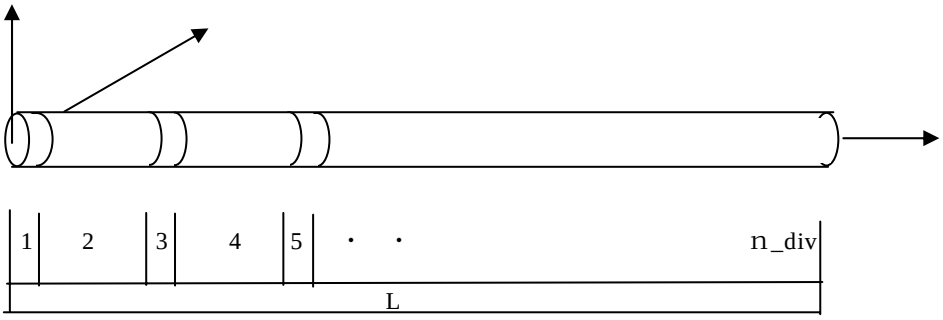


図 9-1 静的縮合モデル

ここでは、例題として新規に開発する静的縮合モデルに関する基本的な設計目標を示す。

1．部材内エレメント数 n_div は任意とし、静的縮合を行って両端の剛性や部材端力を求める。
2．部材は両端 2 節点で 1 節点 6 自由度を有する。
3．部材は、直線部材であり、内部で座標変換はないものとする。
4．幾何学的非線形性を考慮した弾塑性エレメントを含む。
5．部材内エレメントの部材モデルは、現在 SAPCE に組み込まれているモデルを使用するが、将来は新たなモデルを組み込めるように、階層構造を用いて設計する。
6．静的縮合モデルを定義する情報をファイルより読み込む。安定した静的縮合が得られる定義ファイルの設定は使用者に任せられる。したがって、モデルはそのファイルで管理することになる。
7．部材モデル番号は、51 から 70 までとし、ひとつの解析で同時に 20 モデル使用することができる。

８．プログラムの性格上、部材内のエレメント数を 100 以内に制限する。また、この部材内エレメントにおける断面特性の種類は 50 以内とする。現在は、弾性はり、ファイバー断面、MS 断面、アナロジーモデルによるばね断面の 4 種を組み込む。今後、順次増加可能となるように設計する。

このモデルを設計する上で、最も重要な点は次の２つであり、最初に、この 2 点について考える。

１．静的縮合モデルを定義する構造体の設計とその使用法
２．部材内エレメントの情報やワーク領域、例えばファイバーのワーク領域構造体の管理と、プログラムからこれら構造体へのアクセス手法

9.4.2 静的縮合モデル設定ファイルの仕様と構造体

本節では、新規に設定した部材モデルで、必要となる構造体と静的縮合モデル設定ファイルの仕様について述べる。これは前節で指摘した要点の第 1 に相当する。現在 SPACE に組み込まれている静的縮合モデルでは、サブルーチン set_model11_dat()などを用いて、部材モデルとして必要となる情報を設定している。設定しているデータとは以下のようである。

１．部材モデルのエレメント数
２．内部節点の自由度
３．内部エレメントの剛性行列半バンド幅
４．部材節点の拘束番号表あるいは自由度番号表
５．部材内エレメントの長さ
６．部材内に配置されたエレメントのタイプ

上記を考慮して静的縮合モデルを定義する構造体を設計してみよう。この構造体は、New_submain.h で定義され、必要となるサブルーチンの中にインクルードされる。この構造体の名前は S_comp_model_s とする。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   S_comp_model_s 構造体　任意静的縮合モデル
C     ―――――――――――――――――――――――――――――――――
c
c  モデルパラメータ
```

```
      structure  / S_comp_model_s /
      integer    n_div_element          ! 部材モデル中のエレメント数
      integer    i_set_ok               ! データの変換が終了か（0:変換前  1:終了）
      integer    n_out_stress           ! ファイバーデータ出力個数
      integer    nm_out_stress_x        ! 応力出力個数
      integer    n_if                   ! 内部節点自由度
      integer    n_iubw                 ! 半バンド幅
      integer    n_type_element(50)     ! エレメント型(1:弾性梁、2:ファイバー、3:MS、4:アナロジー
      integer    nm_out_stress(100)     ! 応力表示エレメント番号
      integer    nm_type_element(100)   ! エレメントモデルの番号
      integer    irest_Point(6,101)     ! 部材内の自由度
      real*8     alength(100)           ! エレメントモデルの長さ
      integer    n_out_stress_x(5)      ! 応力出力エレメント番号
      end structure
c      record / S_comp_model_s / S_comp_model
```

次に、新規静的縮合モデルの定義ファイルを設計する。この定義ファイルは以下の仕様とする。

```
1.  コメント行数
2.  上記行数分、全体コメント
3.  モデル個数、最大モデル番号
    以下のデータをモデル個数分繰り返す
4.  1 行のモデル用コメント
5.  モデル番号、部材モデル中の要素数(n_div)
6.  要素モデル番号(1 －  n_div) x 軸原点より順に設定する。
7.  要素の長さ（1 - n_div）(部材長さを 1．として、比率で設定する)
8.  次のデータを（1 －  n_div+1）繰り返す
9.  節点自由度(1 －  6 ）  x 軸原点より順に設定する。
10. 弾塑性応力出力・表示番号（1 －  n_out_stress）
11. 応力出力数、応力出力番号(1-5)
```

弾塑性応力出力・表示番号
項目数：１－n_div
番号の意味
1: ｉ端位置
2: ｊ端位置
3: 中央
4: ｉ端接合部
5: ｊ端接合部
0: 表示せず
注：弾性部材はこの値は無視される。

両端ファイバーモデルである部材モデル番号 11 で、両端にせん断変形エレメントを含まないモデルについて、上の仕様にしたがってモデル設定用ファイルを作ってみよう。

```
1
テスト用設定ファイル
1,51
両端ファイバーモデル:モデル番号 11
51,4
2,1,1,2
0.03,0.47,0.47,0.03
-1,-2,-3,-4,-5,-6
1,1,1,1,1,1
1,1,1,1,1,1
1,1,1,1,1,1
-7,-8,-9,-10,-11,-12
1, 0, 0, 2
2,1,6,0,0,0
```

静的縮合部材の節点拘束仕様として、外部節点の自由度番号は負符号を付け、拘束は 0 とする。内部節点では、1 が自由で 0 が拘束となり、10 以上の値は、他の内部節点との変位の同一視を表す。この場合、第一位のけたは、自由度番号で、第二けた以上が節点番号を表す。

上記の静的縮合モデルの定義ファイルにしたがって、構造体にデータを設定するサブルーチンを設計する。このサブルーチンでは、構造体の大きさを動的に確保するため、上記ファイルを２度読むことになる。このサブルーチン Get_S_comp_model()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_S_comp_model
C       ―――――――――――――――――――――――――――――――――――
C     ●   静的縮合部材モデルの定義ファイルを読む
C     ―――――――――――――――――――――――――――――――――――
      subroutine Get_S_comp_model(nx, nx_file,S_comp_model,Model_type ,ieerx)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "New_submain.h"
      record / n_model_s / Model_type
      record / S_comp_model_s    / S_comp_model
      dimension S_comp_model(*)
      character aa*1
C     ―――――――――――――――――――――――――――――――――――
c     nx              :1: パラメータセット 2:データ入力
c     nx_file     設定モデルの最大モデル番号 -  50 ：構造体 S_comp_model の確保用
c     1. コメント行数
c     2. 上記行数分、全体コメント
c     3. モデル個数、最大モデル番号
c        以下のデータをモデル個数分繰り返す
c     4. 1行のモデル用コメント
c     5. モデル番号、部材モデル中の要素数(n_div)
c     6. 要素モデル番号(1 -  n_div) x 軸原点より順に設定する。
C     7. 要素の長さ（1 - n_div）(部材長さを 1. として、比率で設定する)
C     8. 次のデータを（1 -  n_div+1）繰り返す
C     9. 節点自由度(1 -  6 )  x 軸原点より順に設定する。
C     10. 応力出力エレメント番号（1 -  n_out_stress）
C     11. 応力出力数、応力出力番号(1-5)
C     ―――――――――――――――――――――――――――――――――――
      ierrx=0
      if(nx.eq.1) then
C     ―――――――――――――――――――――――――――――――――――
c     ★         モデルの定義ファイルを予備入力し、構造体の値を設定する
C     ―――――――――――――――――――――――――――――――――――
      nfix=5
      nfi=65
      call infile(nfi,nfix,ierr)
      if(ierr.eq.0) then
      read(nfix,*) ii
      do i=1,ii
      read(nfix,'(a)')  aa
      enddo
      read(nfix,*) nn,nx_file
      nx_file=nx_file-50
      close(nfix)
      else
      ierr=1
```

```
      endif
      return
C     ―――――――――――――――――――――――――――――――――――
c     ★        静的縮合モデルの定義ファイルを入力する
C     ―――――――――――――――――――――――――――――――――――
      else
      nfix=5
      nfi=65
      call infile(nfi,nfix,ierr)
      read(nfix,*) ii
      do i=1,ii
      read(nfix,'(a)')  aa
      enddo
      read(nfix,*) nn, nx_file
      nx_file=nx_file-50
      do ij=1, nn
      read(nfix,'(a)')  aa
      read(nfix,*)  number, n_div
      i= number -50                                    ! モデルより 50 を減ずる
      S_comp_model(i).n_div_element = n_div
      read(nfix,*) (S_comp_model(i).nm_type_element(j),j=1,n_div)
      read(nfix,*) (S_comp_model(i).alength(j),j=1,n_div)
      do j=1,n_div+1
      read(nfix,*) (S_comp_model(i).irest_Point(k,j),k=1,6)
      enddo
      read(nfix,*) (S_comp_model(i). nm_out_stress(j),j=1, n_div)
      n_out_stress=0
      do j=1,n_div
      if(n_out_stress.lt.S_comp_model(i).nm_out_stress(j))
     +      n_out_stress=S_comp_model(i).nm_out_stress(j)
      enddo
      S_comp_model(i).n_out_stress=n_out_stress
      read(nfix,*) n_out_stress,
     *     (S_comp_model(i).n_out_stress_x(j),j=1, 5)
      S_comp_model(i).n_out_stress=n_out_stress
C     ―――――――――――――――――――――――――――――――――――
      Model_type.no_e_model(number)   = number  !要素モデルの番号
      Model_type.n_div_model(number)  = n_div   !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     ★        各要素の個数を数える
C     ―――――――――――――――――――――――――――――――――――
      S_comp_model(i).i_set_ok=0
      do k=1,50
      S_comp_model(i).n_type_element(k)=0
      enddo
      do k=1,n_div
      j= S_comp_model(i).nm_type_element(k)
      S_comp_model(i).n_type_element(j)= S_comp_model(i).n_type_element(j)+1
      enddo
      enddo
      close(nfix)
```

```
      endif
      return
      end
```

このサブルーチン Get_S_comp_model()は、submain_dynamic_a()の中でコールされ、静的縮合モデルが設定される。この静的縮合モデルに関する情報が構造データを入力するサブルーチンで必要となるため、この構造データを入力する前でコールされる必要がある。まず、構造データを予備入力し、必要なパラメータを先に取得する。この時点で、静的縮合モデルに関するデータも同様に設定することにしよう。

第２の重要な点は実際のファイルとどのようにリンクするか、また、ファイルを読む、読まないをどのように判断するかである。ここでは、構造データの中にこの新規モデルを含んでいるかどうかチェックして、定義ファイルを読むかどうかを判断する。その判断パラメータとして構造体成分 Parameter_C.n_Scomp_model を用いる。なお、この構造体成分は新しく構造体 parameter_s に追加し、構造データを予備入力するサブルーチン Get_parameters()で設定されなければならない。また、ファイル番号はコントロールファイルの中の現在使用していない番号 65 を用いることにする。ファイル名の指定などは、SPACE 管理システム（他のファイルを指定する箇所）で行うことになる。

最初に構造体 parameter_s の成分に次の 3 つの成分を追加しておこう。まず、n_S_comp_model、そして、nE_New_Element、nM_New_Element である。構造体成分 n_S_comp_model は、先に示した新規の静的縮合モデルの数であり、同じく成分 nE_New_Element と nM_New_Element は、新規モデルに含まれる要素内と部材内のエレメント数である。

構造体 parameter_s で追加した成分は、新規モデルのために新たに設計した構造体の配列の大きさを以下のように決定するために使用される。

n_S_comp_model →
S_comp_model_s
nE_New_Element →
E_Fiber_work_s
nM_New_Element →
M_ Fiber_work_s

さらに、n_model_s 構造体に追加する 2 成分で構造体の配列の大きさを決定する。

n_e_New_fiber →
E_modelx_s
n_m_New_fiber →
M_ modelx_s

上記の５つの中の下 4 つの構造体に関する動的確保に関するコードは後節で説明する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   parameter_s 構造体
C     ―――――――――――――――――――――――――――――――――
c     解析パラメータ
      structure  / parameter_s/
      integer    n_unknown       !  全自由度
      integer    n_point         !  節点数
      integer    n_element       !  要素数
      integer    n_element_dll   !  DLL 用要素数
      integer    n_member        !  部材数
      integer    n_rot_axis      !  主軸回転部材数
      integer    n_local_coord   !  局所座標系を使用する節点数
      integer    n_boundary_p    !  境界節点数
      integer    nc_member       !  部材減衰機構を有する部材数
      integer    n_member_dll    !  DLL 用部材数
      integer    n_S_comp_model  !  任意型静的縮合モデル数
      integer    nE_New_Element  !  任意型静的縮合モデルに含まれる要素エレメント数
```

```
      integer    nM_New_Element  !  任意型静的縮合モデルに含まれる部材エレメント数
      integer    n_free          !  節点当たり解析自由度数
      integer    n_dim           !  解析次元数
      integer    n_skyline       !  スカイライン行列の領域数
      integer    n_sky_ave       !  平均バンド幅
      end structure
c      record /parameter_s/ Parameter_C
```

次に、構造体 n_model_s にも次に示す２つの成分を追加する。追加する成分 n_e_New_fiber と n_m_New_fiber は、新規モデルの要素内及び部材内に存在する特殊断面(ファイバーエレメントやアナロジーエレメントなど)の総数を示す。

```
C     ―――――――――――――――――――――――――――――――
C     ●   n_model_s 構造体
C     ―――――――――――――――――――――――――――――――
c
c  モデルパラメータ
      structure  / n_model_s/
      integer    n_e_models          ! 要素モデルの最大数
      integer    no_e_model(100)     ! 要素モデルの番号
      integer    n_div_model(100)    ! 要素モデルの分割数
      integer    n_e_model(100)      ! 要素モデルの数
      integer    n_m_model(100)      ! 部材モデルの数
      integer    n_damp(100)         ! 部材減衰
      integer    n_m_damp            ! 全部材減衰数
      integer    nm_div_fmodel       ! ファイバー要素の最大数
      integer    nm_div_felement     ! ファイバー要素のエレメント最大数
      integer    n_m_bilinear        ! ファイバー要素バイリニア用の最大要素数
      integer    n_m_trilinear       ! ファイバー要素トリリニア用の最大要素数
      integer    n_m_Concrete        ! ファイバー要素コンクリート用の最大要素数
      integer    n_m_analogy         ! アナロジー要素の最大要素数
      integer    nm_div_msmodel      ! ms 要素の最大数
      integer    nm_div_mselement    ! ms 要素のエレメント最大数
      integer    n_m_ro_model        ! せん断型モデルで使用する RO モデルの総数
      integer    n_m_filter          ! Maxwell 用フィルターの設計あり、なし
      integer    n_spring            ! MSS モデルのばね要素の数
      integer    n_e_New_fiber       ! 新規モデルの要素内特殊断面の総数
      integer    n_m_ New_fiber      ! 新規モデルの部材内特殊断面の総数
      real*8     cosin(2,16)         ! MSS モデルの角度係数を入れるワークエリア
      end structure
c      record / n_model_s / Model_type
```

次に、submain_dynamic_a()の中でサブルーチン Get_S_comp_model() をコールするコードを示そう。ここでは、データを２度読みして、設定領域構造体を動的確保する。無論、ここで動的確保した構造体は、宣言、保存、解放の各処理を行う必要があるが、ここでは省略する。さらに構造体 E_Fiber_work も動的領域確保を行っているが、これについては後で説明する。

```
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★        構造体の大きさを動的確保する（その１）
c
c―――――――――――――――――――――――――――――――――――――――――
      ALLOCATE (Max_disp(Parameter_C.n_point))
      ALLOCATE (Member(Parameter_C.n_member))
      ALLOCATE (Max_stress(Parameter_C.n_member))
      ALLOCATE (Element(Parameter_C.n_element))
      ALLOCATE (Point(Parameter_C.n_point))
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★        配列の大きさを動的確保する
c
c―――――――――――――――――――――――――――――――――――――――――
      N= Parameter_C.n_S_comp_model                    ! 任意静的縮合型モデル
      if(N.ne.0) then
      nx=1
      call Get_S_comp_model(nx,nx_file,S_comp_model, Model_type ,ieerx)
      if(ieerx.ne.0) then
      write(76,*) ' 任意静的縮合型モデル用ファイルがありません ',ieerx
      return
      else
      N = Parameter_C.nE_New_Element          !  任意型静的縮合モデルに含まれる要素エレメント数
      if(N.ne.0)then
      ALLOCATE (E_Fiber_work(N))              ! 新規縮合モデルの要素エレメント数の動的確保  ①-2
      endif
      N = nx_file                             ! 任意静的縮合型モデル
      if(N.ne.0)then
      ALLOCATE (S_comp_model(N))              ! 新規縮合モデル設定領域の動的確保
      endif
      nx=2
      call Get_S_comp_model(nx,nx_file,S_comp_model, Model_type ,ieerx)
      endif
      endif
c―――――――――――――――――――――――――――――――――――――――――
      N=Parameter_C.n_point            ! 節点数
      ALLOCATE (
     *   fll_static_point(3,6,N),am_point(2,N),
     *   fll_force_point(3,N)
     *        )
```

これで、構造体 S_comp_model に新規モデルの情報がセットされたことになる。次は、この構造体を利用して静的縮合モデルを設定するサブルーチン set_modelx_dat()を設計する。このサブルーチンは構造体 S_comp_model を用いて新規部材モデルに必要なデータを作り出す。その内容は、第 5.9.4 節で説明した set_model11_dat()とほぼ同じであり、理解することは容易であろう。後節で説明する剛性を計算するサブルーチンの中で、このサブルーチンは使用される。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /set_modelx_dat
C       ―――――――――――――――――――――――――――――――――――
C     ●   部材モデルの拘束表作成(ok)  任意部材
C     ―――――――――――――――――――――――――――――――――――
      subroutine set_modelx_dat(irest_Point,n_if,n_div,iubw,
     *         Element,Member,S_comp_model,
     *         n_type,alength,gxi,gxj)
C     ―――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "New_submain.h"
      record / element_s    / Element
      record / member_s     / Member
      record / S_comp_model_s / S_comp_model
      dimension irest_Point(6,*),n_type(*),alength(*)
      real*8  gxi,gxj                 ! 剛域
c―――――――――――――――――――――――――――――――――★
c      n_div= S_comp_model.n_div_element
      do i=1,n_div
      n_type(i)= S_comp_model.nm_type_element(i)
      enddo
      al=Member.alength - gxi - gxj      ! 剛域を削除
      do i=1,n_div
      alength(i)=al* S_comp_model.alength(i)
      enddo
      nn=n_div+1
      do i=1,nn
      do j=1,6
      irest_Point(j,i)= S_comp_model.irest_Point(j,i)
      enddo
      enddo
c―――――――――――――――――――――――★ 内部変位の節点番号セット
      if(S_comp_model.i_set_ok.eq.0) then
      n_if =0
      do i=2,nn-1
      do j=1,6
      if(irest_Point(j,i).eq.1) then
      n_if = n_if +1
      irest_Point(j,i)= n_if
      else
      irest_Point(j,i)= -irest_Point(j,i)
      endif
      enddo
      enddo
      do i=2,nn-1
      do j=1,6
      if(irest_Point(j,i).lt.-10) then
      irest= -irest_Point(j,i)
      ip=irest/10
      ipp= Mod(irest,10)
      irest_Point(j,i)=irest_Point(ipp,ip)
      endif
      enddo
```

```
      enddo
      call set_iubw(irest_Point,n_div,iubw)
c―――――――――――――――――――――――――★ 構造体でデータ設定
      S_comp_model.i_set_ok=1
      S_comp_model.n_if=n_if
      S_comp_model.n_iubw=iubw
      do i=1,nn
      do j=1,6
      S_comp_model.irest_Point(j,i)= irest_Point(j,i)
      enddo
      enddo
      else
c―――――――――――――――――――――――――★ 2 度目以降の設定
      n_if= S_comp_model.n_if
      iubw= S_comp_model.n_iubw
      endif
      return
      end
```

9.4.3 部材モデルの階層構造と構造体

本節では、先に指摘した部材内エレメントの情報やワーク領域、例えばファイバーのワーク領域となる構造体の管理、あるいは構造体へのアクセス手法について考えよう。まず、SPACE で使用されている部材モデルの階層構造を示す。

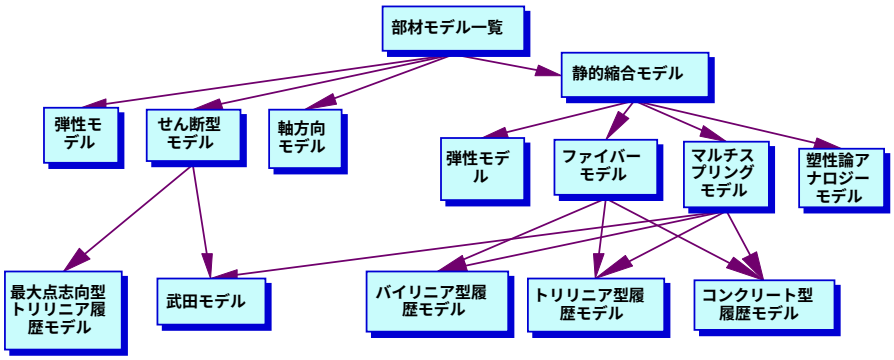


図 9-2　部材モデルの階層構造

弾塑性解析や幾何学的非線形性を考慮するために、ファイバーモデルやアナロジーモデルでは、多くのワーク領域を必要とする。このワーク領域を如何に効率よく使用するかが重要となる。ここでは、このワーク領域の使用法、特にデータへのアクセス法について考える。図 9-2 に示す部材モデル階層構造で、様々なデータ領域が構造体として定義されている。そのため、多数のモデルが混乱することなく使用でき、それに対応して設計された構造体を効率良く、しかもすばやくアクセスすること

が重要となる。

ここでは、部材モデルの階層構造に合せて、その構造体と情報の流れ、また、他の構造体へのリンク方法を検討する。例として両端ファイバーモデルとして設計された部材モデル 11 について、構造体の階層構造並びにそれらのリンク方法を以下の表にまとめた。構造体は、一般に対になっており、解析中に値が変化しない構造体と値が変化する構造体が存在する。部材に割り付けられた標準の構造体として、前者が要素構造体 Element_s であり、後者は部材に関連する構造体 Member_s である。この２つの構造体は、配列として全要素と全部材に各１つ存在することになる。表中の構造体は階層構造となっており、各部材モデルに対応する。表の最後に記述されている構造体が階層構造の最下位層となっており、各ファイバーの情報を保存する。この下位層の構造体は、効率よく配置するために要素や部材に対応せず、実際の解析モデルに合せて必要数分詰めて配置される。したがって、下位の構造体から情報を取り出すためには、上位の構造体からのリンク情報が必要となる。この表では、リンク情報を構造体から抜き出して表示し、その使用例を載せている。

表 9-1 部材モデル：両端ファイバーモデル

解析時に値が変化しない構造体　　解析時に値が変化する構造体

標準部材構造体

```
部材              Element_s                    Member_s
リンク情報     Element_s 構造体
          integer  n_element          !  非線形要素番号
リンク情報     Member_s 構造体
          integer  n_model            !  モデルの入れ物番号
          integer  n_model_type       !  モデル別の通し番号
例
        imm= Element.n_element      →    E_model11(imm)
        ie   = Member.nm_element    →    Element(ie)
        immm= Member.n_model_type   →    M_model11(immm).d_ra_1
```

静的縮合型部材モデル１１

```
                E_model11_s                  M_model11_s
リンク情報     E_model11_s 構造体
      integer  nm_section_1         !  i 端断面のファイバー開始番号
      integer  n_section_1          !  i 端断面のファイバー数
      integer  nm_section_2         !  j 端断面のファイバー開始番号
      integer  n_section_2          !  j 端断面のファイバー数
リンク情報     M_model11_s 構造体
      integer  nm_section_1         !  i 端断面のファイバー開始番号
```

```
      integer  n_section_1          !  i 端断面のファイバー数
      integer  nm_section_2         !  j 端断面のファイバー開始番号
      integer  n_section_2          !  j 端断面のファイバー数
例
       nm_div   = E_model.n_section_1  →  do i=1,nm_div
       nn       = E_model.nm_section_1 -  1 →  nm_type=E_model_fiber(nn).nm_type
       nnm      = M_model.nm_section_1 - 1  →  nmx=M_model_fiber(nnm).n_type

断面モデル

ファイバーデータ  E_model_fiber_s                  M_model_fiber_s
リンク情報    E_model_fiber_s 構造体
        integer  nm_type              !  履歴モデルの番号
リンク情報    M_model_ fiber_s 構造体
        integer  n_type               !  履歴モデルの通し番号
例
       nm_type=E_model_fiber(nn).nm_type →  goto ( 10,20,30,40,50,60,70,80),nm_type
       nmx=M_model_fiber(nnm).n_type  →    Bilinear_work(nmx).i_stat
                                           Concrete_work(nmx).P1(2)
```

```
ファイバー要素のワーク領域
ワーク領域                                  Bilinear_work_s
                                            Trilinear_work_s
                                            Concrete_work_s
```

新規の静的縮合部材モデルでは、部材内の要素数は任意であり、既存の部材モデルのように、Member_s と Element_s からリンクが張られている M_model11_s や E_model11_s のように構造体を特定することができない。そこで、他の構造体を用いてリンクを張る必要がある。まず、新規モデルで必要となる断面データやワーク領域に関する構造体を、以下のように設計する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 E_modelx_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / E_modelx_s/
      integer  nm_section         !  指定断面のファイバー開始番号
      integer  n_section          !  指定断面のファイバー数
      end structure
c       record / E_modelx_s    / E_modelx
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 ファイバー断面用  M_modelx_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / M_modelx_s/
```

```
      integer  nm_section        !  指定断面のファイバー開始番号
      real*8  d_aa               !  断面積の和
      real*8  d_ra               !  E*断面積の和
      real*8  d_ray              !  E*A*z
      real*8  d_raz              !  E*A*y
      real*8  d_raz2             !  E*A*y*y
      real*8  d_ray2             !  E*A*z*z
      real*8  d_rayz             !  E*A*z*y
      real*8  d_gg               !  G*A
      real*8  d_epsi_x           !  軸方向歪
      real*8  d_epsi_y           !  y 軸に関する曲げ歪
      real*8  d_epsi_z           !  z 軸に関する曲げ歪
      real*8  disp_p(12)         !  指定断面両端の変位
      end structure
c      record / M_modelx_s    / M_modelx
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 Mss 断面用 M_model_mss_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / M_model_mss_s/
      integer  nm_section        !  指定断面のファイバー開始番号
      real*8  d_aa               !  断面積の和
      real*8  d_ra               !  E*断面積の和
      real*8  d_ray              !  E*A*z
      real*8  d_raz              !  E*A*y
      real*8  d_raz2             !  E*A*y*y
      real*8  d_ray2             !  E*A*z*z
      real*8  d_rayz             !  E*A*z*y
      real*8  d_gg               !  G*A
      real*8  d_epsi_x           !  軸方向歪
      real*8  d_epsi_y           !  y 軸に関する曲げ歪
      real*8  d_epsi_z           !  z 軸に関する曲げ歪
      real*8  disp_p(12)         !  指定断面両端の変位
      end structure
c      record / M_model_mss_s    / M_model_mss
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 アナロジーモデル用 M_model_ang_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / M_model_ang_s/
      integer  nm_section          !  指定断面のファイバー開始番号
      real*8  fax                  !  関数φx
      real*8  fay                  !  関数φy
      real*8  faz                  !  関数φz
      real*8  c_n                  !  ｎ軸の移動
      real*8  c_my                 !  My 軸の移動
      real*8  c_mz                 !  Mz 軸の移動
      real*8  H                    !
      real*8  d_epsi_x             !  軸方向歪
      real*8  d_epsi_y             !  y 軸に関する曲げ歪
      real*8  d_epsi_z             !  z 軸に関する曲げ歪
      real*8  d_damy               !  ダミー
```

```
      real*8  disp_p(12)            !  指定断面両端の変位
      end structure
c      record / M_model_ang_s    / M_model_ang
```

上記の構造体は、この新規モデルの各ファイバー断面に対応しており、部材内エレメント数が任意であるため、Element_s と Member_s から直接アクセスすることはできない。そこで、次の 2 つの構造体を用いて間接的にアクセスする。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 E_Fiber_Work  構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / E_Fiber_Work_s/
      integer  n_Fiber_section      !  指定断面の特殊断面番号
      integer  nm_section           !  特殊断面の E_modelx_s 用番号
      end structure
c       record / E_Fiber_Work_s    / E_Fiber_Work
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.51 M_Fiber_Work  構造体
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / M_Fiber_Work_s/
      integer  nm_section           !  特殊断面の M_modelx_s 用番号
      real*8   an_vv                !  内部エレメントのｖ方向相対変位
      real*8   an_ww                !  内部エレメントのw方向相対変位
      real*8   an_stress            !  内部エレメントの軸力
      real*8   ff_ip(6)             !  内部エレメントの不釣合力
      end structure
c       record / M_Fiber_Work_s    / M_Fiber_Work
```

構造体 E_Fiber_Work_s()は、この部材モデルの内部エレメントの特殊断面番号（含ファイバー断面、ただし弾性はりの場合は 0 をセット）と E_modelx_s へのリンク情報を保持し、その個数は、解析モデルの要素に含まれる新規モデルの全内部エレメント数となる。また、構造体 M_Fiber_Work_s()も動的配列であり、その大きさは、解析モデルの部材に含まれる新規モデルの全内部エレメント数となる。この中で、内部エレメントの変位や不釣合力は内部の釣合式を解くときに、あるいはｖ方向、ｗ方向の相対変位とエレメント軸力は幾何学的非線形性に関する剛性行列を計算するときに必要となる。

この新しく設定した動的配列と構造体を用いて、新規モデルに関する構造体の階層構造とリンク方法を以下の表にまとめる。

表 9-2 部材モデル：任意型モデル 51-70

解析時に値が変化しない構造体　　解析時に値が変化する構造体

標準部材構造体

```
部材            Element_s                    Member_s
リンク情報    Element_s 構造体
        integer  n_element          !  非線形要素番号
        integer  n_section(1)       !  部材内要素開始番号
        integer  n_section(2)       !  部材内要素数
リンク情報    Member_s 構造体
        integer  n_model            !  モデルの入れ物番号
        integer  n_model_type       !  モデル別の部材内要素開始番号
例
      imm  = Element.n_section(1)  →   E_Fiber_work(imm)
      ie   = Member.nm_element     →   Element(ie)
      immm = Member.n_model_type   →   M_Fiber_work(immm).d_ra
```

部材内要素情報

```
              E_Fiber_work_s        M_ Fiber_work_s
リンク情報    E_Fiber_work_s 構造体
     integer  n_Fiber_section   !  指定断面の特殊断面番号
     integer  nm_section        !  特殊断面の番号
リンク情報    M_Fiber_work_s 構造体
     integer  nm_section        !  特殊断面の番号
例
      nn  = E_Fiber_work.nm_section  →  nm_type= E_modelx(nn).nm_section
      nnm = M_Fiber_work.nm_section  →  nmx= M_modelx(nnm).nm_section
```

静的縮合型部材モデル 51

```
              E_modelx_s                  M_modelx_s
リンク情報    E_modelx_s 構造体
     integer  nm_section        !  断面のファイバー開始番号
     integer  n_section         !  断面のファイバー数
リンク情報    M_modelx_s 構造体
     integer  nm_section        !  断面のファイバー開始番号
例
      nm_div  = E_modelx.n_section  →  do i=1,nm_div
      nn      = E_modelx.nm_section – 1  →  nm_type=E_model_fiber(nn).nm_type
      nnm     = M_modelx.nm_section - 1  →  nmx=M_model_fiber(nnm).n_type
```

断面モデル

```
ファイバーデータ  E_model_fiber_s              M_model_fiber_s
リンク情報    E_model_fiber_s 構造体
        integer  nm_type            !  履歴モデルの番号
リンク情報    M_model_ fiber_s 構造体
        integer  n_type             !  履歴モデルの通し番号
```

例

```
nm_type=E_model_fiber(nn).nm_type →  goto ( 10,20,30,40,50,60,70,80),nm_type
nmx=M_model_fiber(nnm).n_type  →     Bilinear_work(nmx).i_stat
                                     Concrete_work(nmx).P1(2)
```

ファイバー要素のワーク領域
ワーク領域　　**Bilinear_work_s**
Trilinear_work_s
Concrete_work_s

表 9-2 に示すような構造体の階層構造とリンク情報を用いて、この新規の任意型静的縮合モデルを構築することになる。新規モデルの構築は、多くの構造体を必要とするが、その全てが動的に領域確保することになる。新規モデルで必要とする構造体配列の大きさを決めるパラメータと動的確保を行う位置について、表 9-3 にまとめる。表中の＊印は他のモデルと共用であり、他は新規モデル専用の構造体である。また、構造体配列の数欄の要素内エレメント総数と部材内エレメント総数とは、新規モデルに対する数を表し、要素内特殊断面エレメント総数と部材内特殊断面エレメント総数とは、新規モデル中における特殊断面を使用しているエレメントを抽出した数を表す。

表 9-3 構造体とリンク情報

構造体	配列の大きさを決めるパラメータ	動的確保の位置	構造体配列の数
Element_s	Parameter_C.n_element	M_alloc(1)	全要素数*
E_Fiber_work_s	**Parameter_C.nE_New_Element**	M_alloc(1)	要素内エレメント総数
E_modelx_s	**Model_type.n_e_New_fiber**	M_alloc(2)	要素内特殊断面エレメント総数
E_model_fiber_s	Model_type.nm_div_felement	M_alloc(3)	要素内特殊断面エレメント総数*
Member_s	Parameter_C.n_member	M_alloc(1)	全部材数*
M_ Fiber_work_s	**Parameter_C.nM_New_Element**	M_alloc(2)	部材内エレメント総数
M_modelx_s	**Model_type.n_m_ New_fiber**	M_alloc(2)	部材内特殊断面エレメント総数
M_model_fiber_s	Model_type.nm_div_fmodel	M_alloc(3)	部材内特殊断面エレメント総数*
Bilinear_work_s	Model_type.n_m_bilinear	M_alloc(5)	バイリニアファイバーの総数*
Trilinear_work_s	Model_type.n_m_trilinear	M_alloc(5)	トリリニアファイバーの総数*
Concrete_work_s	Model_type.n_m_Concrete	M_alloc(5)	コンクリートファイバーの総数*

構造体	備考
Element_s	入力データの要素数で総数が決定、コードを追加する必要がない。

E_Fiber_work_s	新しい構造体であるため、個数を数えるコードを新たに追加する必要がある。①
E_modelx_s	新しい構造体であるため、個数を数えるコードを新たに追加する必要がある。②
E_model_fiber_s	既存の個数を数えるコードに、新規モデル用のコードを加える。③
Member_s	入力データの部材数で総数が決定、コードを追加する必要がない。
M_ Fiber_work_s	新しい構造体であるため、個数を数えるコードを新たに追加する必要がある。④
M_modelx_s	新しい構造体であるため、個数を数えるコードを新たに追加する必要がある。⑤
M_model_fiber_s	既存の個数を数えるコードに、新規モデル用のコードを加える。⑥
Bilinear_work_s	既存の個数を数えるコードに、新規モデル用のコードを加える。⑦
Trilinear_work_s	既存の個数を数えるコードに、新規モデル用のコードを加える。⑧
Concrete_work_s	既存の個数を数えるコードに、新規モデル用のコードを加える。⑨

次節では、これらの構造体配列の大きさをどのように決定しているか、またどのようにデータをセットするか、特にリンク情報をどのように設定し、利用するかにについて解説する。大切なのは、これらの構造体配列を動的確保する場合、その個数とどのタイミングで実際に確保すべきかを決定することである。表中右側に示す①、②などは新規モデルに関する構造体の設定する箇所を示す印であり、実際のプログラムコード内に付された印と対応する。また、①-1 などは構造体配列の個数の計算部分、①-2 は動的領域の確保、①-3 はリンク情報の設定位置を示す。

9.4.4 部材モデルの入力仕様

本節では、入力関連で新規の静的縮合モデルで必要となるコードを検討する。まず構造データを予備入力するサブルーチン Get_parameters() について考えよう。ここでは２つの変更点がある。ひとつは静的縮合設定ファイルを読むか否かの判断を行うために、新規静的縮合部材モデル 51-70 を使用しているかチェックするコードを追加することであり、他の一つはこのモデルに含まれるファイバー断面の数を数えるコードである。太文字で示した部分が上記２つの作業を行うコードである。なお、このサブルーチン内では、ここでは必要のない入力データは読み捨てている。

構造データの入力仕様を少し変更することとし、新規静的縮合部材モデルの要素データを入力する部分で、標準データ入力の後の第 2 レコードを以下の仕様とする。内部エレメント数は、新規静的縮合モデルの内部に含まれる全エレメントの数であり、ファイバー断面番号で、弾性部材など特殊な断面を含まない場合は 0 とする。

１．この部材に含まれる内部エレメント数
２．ファイバー断面番号（１－ 内部エレメント数）

追加したコードと共にサブルーチン Get_parameters()を見てみよう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_parameters
C       ―――――――――――――――――――――――――――――――――
C     ●   構造データを予備入力し、構造用のパラメータを設定する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Get_parameters(Parameter_C,ierr)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s  / Parameter_C
      character amoji*1
      dimension n_section(100)
C     ―――――――――――――――――――――――――――――――――
C     ―――――――――――――――――――――――――――――――――
c     ★   タイトル入力
C     ―――――――――――――――――――――――――――――――――
      ierr=0
      read(5,*,err=999) n
      if(n.ne.0) then
      do J=1,n
      read(5,'(a)') amoji
      end do
      endif
C     ―――――――――――――――――――――――――――――――――
c     ★   制御データ入力とパラメータセット
C     ―――――――――――――――――――――――――――――――――
      read(5,*,err=999) node,nelem,memb,nrbound,locod,njiku
      Parameter_C.n_point      =node
      Parameter_C.n_element    =nelem
      Parameter_C.n_member     =memb
      Parameter_C.n_boundary_p =nrbound
      Parameter_C.n_local_coord=locod
      Parameter_C.n_rot_axis   =njiku
      Parameter_C.n_free       =6
c
      write(76,'(//a)') ' 構造体：Parameter_C'
      write(76,*) Parameter_C.n_point      ,'=node'
      write(76,*) Parameter_C.n_element    ,'=nelem'
      write(76,*) Parameter_C.n_member     ,'=memb'
      write(76,*) Parameter_C.n_boundary_p ,'=nrbound'
      write(76,*) Parameter_C.n_local_coord,'=locod'
      write(76,*) Parameter_C.n_rot_axis   ,'=njiku'
      write(76,*) Parameter_C.n_free       ,'=6'
C     ―――――――――――――――――――――――――――――――――
c     ★   これ以降のコードを追加する
C     ―――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――★節点データ入力
      do i=1,node
      read(5,*,err=9912,end=9918) ii,x,y,z,idm
```

```
      end do
c―――――――――――――――――――――――――――――――★節点局所座標入力
      if(locod.ne.0) then
      do i=1,locod
      read(5,*,err=9912,end=9918) j,tlx,tly,tlz
      end do
      end if
c―――――――――――――――――――――――――――――――★境界拘束条件入力
c――――――――――――――――――――――★境界拘束条件入力
      if(nrbound.ne.0) then
      do i=1,nrbound
      read(5,*,err=9912,end=9918) j,ir1,ir2,ir3,ir4,ir5,ir6
      end do
      end if
c―――――――――――――――――――――――――――――――★線形要素モデルデータ入力
      Parameter_C.n_S_comp_model=0
      Parameter_C.nE_New_Element =0
      do  i=1,nelem
      read(5,*,err=9913,end=9918) m_type,e,g,a,rix,riy,riz,asy,asz,
     *          am1,am2,anp,ampy,ampz,nm_type
c――――――――――――――――――――――★新規の静的縮合モデルを数える
      mmx=(m_type-1)/10
      if(mmx .eq. 5 .or. mmx .eq. 6) then
      Parameter_C.n_S_comp_model= Parameter_C.n_S_comp_model + 1
      read(5,*,err=9913,end=9918) nxx,(n_section(j),j=1,nxx)
      Parameter_C.nE_New_Element= Parameter_C.nE_New_Element + nxx                ! ①-1
      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
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,j1,j2
      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
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,j1,j2,j3
      endif
c――――――――――――――――――――――★要素番号 31 に対してデータ入力
      if(m_type .eq. 31) then
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,j1,j2
      endif
c――――――――――――――――――――――★要素番号 32 に対してデータ入力
      if(m_type .eq. 32) then
      read(5,*,err=9913,end=9918) rd1,rd2,sg1,sg2,j1,j2,j3
      endif
      end do
      return
  999 continue
      write(76,'(a)')' 構造データファイルの制御データにエラーが存在する'
      ierr=1
      return
 9912 continue
      write(76,'(a)')' 構造データファイルの節点データにエラーが存在する'
      ierr=1
      return
```

```
 9913 continue
      write(76,'(a)')' 構造データファイルの要素データにエラーが存在する'
      ierr=1
 9918 continue
      end
```

ここでは、構造体 S_comp_model を動的確保すべきかどうか判定するためパラメータ Parameter_C. n_S_comp_model と、構造体 E_Fiber_work の大きさを決めるパラメータ Parameter_C.nE_New_Element を設定している（①-1)。

構造データのファイル仕様が変更になったので、これに伴い、予備入力サブルーチン Get_parameters()と同様に、Get_structure()も変更しなくてはならない。以下に、このサブルーチンの変更部分についてのみ記す。追加したコードでは、新規の静的縮合モデルにおけるファイバー断面の番号を動的配列 n_Fiber_section にセットする。このとき、構造体成分 Element(i).n_section(1)に要素断面番号の初期値がセットされ、また、Element(i).n_section(2)にはその部材のエレメント個数がセットされる。

```
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,
     *           S_comp_model,E_Fiber_work)

      include "New_submain.h"
      record / E_Fiber_work_s / E_Fiber_work
      record / S_comp_model_s / S_comp_model
       .
      dimension  n_section(100),E_Fiber_work(*),S_comp_model(*)
       .
      nnx=0
      do  i=1,nelem
       .
      read(5,*,err=9913,end=9918) m_type,e,g,a,rix,riy,riz,asy,asz,
     *          am1,am2,anp,ampy,ampz,nm_type
       .
c―――――――――――――――――――――――――★新規の静的縮合モデルの第２レコード
      mmx=(m_type-1)/10
      if(mmx .eq. 5 .or. mmx .eq. 6) then
      read(5,*,err=9913,end=9918) nxx,(n_section(j),j=1,nxx)
c―――――――――――――――――――――――――★エレメント数チェック
      if(nxx.ne. S_comp_model(m_type-50).n_div_element)
     *  write(76,*) ‘ エレメント個数エラー  ‘, m_type, nxx
      do j=1,nxx
```

```
      E_Fiber_work(j+nnx).n_Fiber_section = n_section(j)
      enddo
      Element(i).n_section(1) =nnx+1              ! 新規モデルの要素内エレメントの最初の番地
      Element(i).n_section(2) =nxx                ! 新規モデルの要素内エレメントの個数
      nnx = nnx + nxx
      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
      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
c―――――――――――――――――――――★要素データを構造体にセット
      •
      •
c――――――――――――――――――――――――――★モデルタイプ別の要素数の計算（ゼロセット）
      do i=1,Model_type.n_e_models
      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
      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)
     *   Parameter_C.n_element_dll = Parameter_C.n_element_dll+1
      end do
      if(ierr.ne.0) goto 9914
c―――――――――――――――――――――★新規モデルの要素内エレメント数を数える
      isum=0
      do j=51,70
      if(Model_type.n_e_model(j).ne.0) then
      do k=1, Model_type.n_div_model(j)
       if(S_comp_model(j-50).nm_type_element(k).ne.1) then          ! 弾性はりエレメントを除く
       isum=isum+ Model_type.n_e_model(j)
       endif
      enddo
      endif
      enddo
      Model_type.n_e_New_fiber =isum                                  ! ②-1
```

```
      •
c―――――――――――――――――――――――――――――――★部材データ入力
      Parameter_C.nM_New_ Element = 0
      do i=1,memb
      read(5,*,err=9915,end=9918) ii,i1,i2,ie,ian,ig,iso,ii1,ii2,
     *          rigid_i,rigid_j,shear_i,shear_j
      •
      •
      if(ie.le. nelem ) then
      Member(ii).element_type = Element(ie).element_type
      mmx=(Member(ii).element_type -1)/10
      if(mmx .eq. 5 .or. mmx .eq. 6) then
      nxx= Element(ie).n_section(2)                          ! 新規モデルに含まれるエレメント数
      Parameter_C.nM_New_ Element = Parameter_C.nM_New_Element + nxx          ! ④-1
      shear_i=0
      shear_j=0
      elseif(Member(ii).element_type eq. 50 ) then
      Member(ii).nm_dll_element=1                            ! DLL 部材を数える
      endif
      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―――――――――――――――――――――――――――――――★モデルタイプ別部材数
      Parameter_C.n_member_dll=0
      DO 30 ii=1,memb
      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)
     *   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
c―――――――――――――――――――――――――★新規モデルの部材内エレメント数を数える
      isum=0
```

```
      do j=51,70
      if(Model_type.n_e_model(j).ne.0) then
      do k=1, Model_type.n_div_model(j)
       if(S_comp_model(j-50).nm_type_element(k).ne.1) then        ! 弾性はりエレメントを除く
       isum=isum+ Model_type.n_m_model(j)
       endif
      enddo
      endif
      enddo
      Model_type.n_m_New_fiber =isum                                          ! ⑤-1
c―――――――――――――――――――――――――――――――★モデル別通し番号計算
      do 32 i=1,Model_type.n_e_models
      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―――――――――――――――――――――――★新規モデル 51-70 までの通し番号設定
      j=1
      do ii=1,memb
            if(Member(ii).n_model.ge.51 .and. Member(ii).n_model.le.70) then
            Member(ii).n_model_type = j
            j=j+Model_type.n_div_model(Member(ii).n_model)
            endif
      enddo
      ・
      ・
```

先に示したように構造体 E_Fiber_work の動的確保は、サブルーチン Get_prarameters()でその大きさを設定した後、Get_structur()をコールする前に行わなければならない。このコードについては前節で示した。新たに設計した他の構造体の動的確保は以下に示される。ここでは省略するが、これら新規に設計した構造体に対し、構造体の宣言、保存、解放処理を忘れてはならない。

```
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★        計算用構造体の大きさを動的確保する（その２）
c
c―――――――――――――――――――――――――――――――――――――――――
      ・
      ・
c―――――――――――――――――――――――――――――★Model_No.33 両端ピン、中央アナロジーモデル
      n= Model_type.n_e_model(19)        !要素モデルの数
      if(n.ne.0) then
      ALLOCATE (E_model33(n))
      endif
```

```
      n= Model_type.n_m_model(19)        !部材モデルの数
      if(n.ne.0) then
      ALLOCATE ( M_model33(n))
      endif
c――――――――――――――――――――――――――――――――――――★新規モデル Model_No.51-70
      n= Model_type.n_e_New_fiber        !要素モデルの数
      if(n.ne.0) then
      ALLOCATE (E_modelx(n))                                          ! ②-2
      Endif
      n= Model_type.n_m_New_fiber        ! 部材モデルの数
      if(n.ne.0) then
      ALLOCATE (M_modelx(n))                                          ! ④-2
      endif
      n=Parameter_C.nM_New_Element         ! 数新規モデルにおける部材エレメント数
      if(n.ne.0) then
      ALLOCATE (M_Fiber_work(n))                                      ! ⑤-2
      endif
```

ファイバーデータを入力するサブルーチン Fiber_input()では、ファイバーデータの入力と部材データとのリンクのための処理が三箇所で行われる。新規モデルでは、ファイバーデータの仕様変更はないため入力部分の変更はない。ただし、部材モデルが追加されることによって、三箇所で処理に変更が加えられる。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_input
C     ――――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の入力(ok)
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,
     *      M_Fiber_work, E_Fiber_work,E_modelx,M_modelx,Parameter_C)
       .
       .
      include "New_submain.h"
      record / parameter_s    / Parameter_C
      record / M_Fiber_work_s / M_Fiber_work
      record / E_Fiber_work_s / E_Fiber_work
      record / M_modelx_s     / M_modelx
      record / E_modelx_s     / E_modelx
      dimension M_Fiber_work(*),E_Fiber_work(*),M_modelx(*),E_modelx(*)
       .
       .
      ierr=0
      if(it.eq.0) then
c――――――――――――――――――――――――――――――――――――★ファイバーデータの予備入力
      read(5,*,err=999) nm          ! 最大断面数
```

```
      write(76,'(a,i4)') ' ファイバー断面総数：',nm
      ii=0
      nmmx=0
      do i=1,nm
      read(5,*,err=999) n_m,nmm,(ddm(j),j=1,20)     ! 断面番号、エレメント数
c-----------------------------★断面ファイバー数のセット
      do i1=1,n_element
      itype_m = Model_type.no_e_model(Element(i1).element_type)
      if(itype_m.eq.11) then
c-----------------------★モデル１１                              ! x1
       ・
      endif
c-----------------------★モデル１２
      if(itype_m.eq.12) then
       ・
      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-----------------------★新規モデル 51-70
      nx_itype=(itype_m-1)/10                                  ! 1
      if(nx_itype.eq.5 .or. nx_itype.eq.6) then
        nmx = Element(i1).n_section(1)-1
        nnx = Element(i1).n_section(2)
        do k=1,nnx                                             ! 2
         if(E_Fiber_work(nmx+k).n_Fiber_section.eq.n_m) then    ! 3
           nmmx = nmmx + 1
           E_Fiber_work(nmx+k).nm_section = nmmx          ! ①-3
           E_modelx(nmmx).nm_section = ii+1               ! ④-3
```

```
            E_modelx(nmmx).n_section = nmm                    ! ④-3
          endif
        enddo
      endif
      enddo
c―――――――――――――――――――――――――★
      do j=1,nmm
      ii = ii + 1                                             ! ①-1
      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
      ・
      enddo
      enddo
c―――――――――――――――――――――――――★ファイバー数セット
      Model_type.nm_div_felement= ii
      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
      nmmx=0
      do i=1,n_member
      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―――――――――――――――――――――――――★モデル１１                                    ! x2
      if(itype_m.eq.11) then
      ・
      endif
c―――――――――――――――――――――――――★モデル１２
      if(itype_m.eq.12) then
      ・
      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―――――――――――――――――――――★新規モデル 51-70
      nx_itype=(itype_m-1)/10
        nnx = Element(i1).n_section(2)
        nmx = Element(i1).n_section(1) -1
        nnmx = Member(i).n_model_type -1
        do k=1,nnx                                                    !4
         if(E_Fiber_work(nmx+k).n_Fiber_section.ne.0) then
              nmmx=nmmx+1
              nmmxx=E_Fiber_work(nmx+k).nm_section
              M_Fiber_work(nnmx+k).nm_section = nmmx      ! ④-3
              M_modelx(nmmx).nm_section = jj+1            ! ⑤-3
              jj = jj + E_modelx(nmmxx).n_section         ! ⑥-1
         endif
        enddo
      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
      read(5,*,err=999) nm
      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)
      do j=1,nmm
      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
       (E_model_fiber(ii-nmm+1),nmm)
      enddo
c―――――――――――――――――――――★ファイバー履歴特性数セット
      n_m_bilinear    = 0
      n_m_trilinear   = 0
      n_m_concrete    = 0
      n_m_analogy     = 0
      do i=1,n_member
      ie  = Member(i).nm_element
      imm = Element(ie).n_element
      im  = Member(i).n_model_type
c―――――――――――――――――――――★モデル１１                                    ! x3
      itype_m = Model_type.no_e_model(Element(ie).element_type)
      if(itype_m.eq.11) then
       ・
      endif
c―――――――――――――――――――――★モデル１２
      if(itype_m.eq.12) then
```

```
        ・
      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―――――――――――――――――――――★新規モデル 51-70
      nx_type_m= (itype_m-1)/10
      if(nx_type_m.eq.5 .or. nx_type_m.eq.6) then                    ! 5
        nmx  = Element(ie).n_section(1)-1
        nnmx = Member(i).n_model_type -1
        nnx =  Element(ie).n_section(2)
        do k=1,nnx
         if(E_Fiber_work(nmx+k).n_Fiber_section.ne.0) then
           ii  = E_modelx(inmm).n_section
          imm = E_Fiber_work(nmx+k).nm_section
          inmm= M_Fiber_work(nnmx+k).nm_section
          nmm = E_modelx(imm).nm_section - 1
          nnmm = M_modelx(inmm).nm_section - 1
          do j=1,ii                                                  ! 6
            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                      ! ⑦-1
            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                    ! ⑧-1
               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                     ! ⑨-1
               M_model_fiber(nnmm).n_type = n_m_concrete
            elseif(E_model_fiber(nmm).nm_type.eq.11.or.
     *          E_model_fiber(nmm).nm_type.eq.12.or.
     *          E_model_fiber(nmm).nm_type.eq.13) then
                   n_m_analogy  = n_m_analogy + 1
               M_model_fiber(nnmm).n_type = n_m_analogy
             endif
           enddo
         endif
         enddo
         endif
c―――――――――――――――――――――――★
      enddo
      Model_type.n_m_bilinear   = n_m_bilinear                                ! x4
      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
```

新しく追加したコードの説明に入る前に、このプログラム右の x1 から x4 で示される４箇所の処理内容について復習しておこう。

x1. ここでは、現在設計されている部材モデルのどの位置で、このファイバー断面が使用されているのかをチェックする。もし使用している場合は、要素構造体 Element にファイバー断面番号をセットする。

x2. 上記と同じく、このファイバー断面がどの部材モデルで使用されているかチェックする。ただし、上記は、全要素についてであるが、ここでは、全部材についてチェックする。

x3. 断面内で使用しているファイバー履歴の解析で必要となるワーク領域構造体配列の数を数えている。ここでは、両端ファイバーモデルについてであり、プログラムコードを見ると、ｉ端とｊ端の 2 箇所でこの構造体の数を数えている。他の部材モデルでも同様の検索を以降のコードで行っていることは当然のことである。現在、このワ

ーク領域構造体は 3 種類用意されており、各構造体は対応する解析モデル中に存在するファイバー数分必要となる。その 3 種類とは、

１．バイリニア型構造体：Bilinear_work(:)

２．トリリニア型構造体：Trilinear_work(:)

３．コンクリート型構造体：Concrete_work(:)

である。ここで、チェックを行っているコードの順番は、上記のバイリニア型、トリリニア型、コンクリート型の順であり、その中の番号は履歴番号である。

x4. 最後に、各種の部材モデルで必要となるの３つのワーク領域構造体の数が、構造体 model_type.n_m_bilinear などにセットされる。

次に、新しく追加したコードについて説明しよう。

1. 部材モデル番号が 51-70 であるかどうかチェックする。そうであれば、以下の処理を行う。まず、その部材に含まれる要素数と構造体 E_Fiber_work の先頭番地を取得する。
2. 部材に含まれるエレメント数分、以下の処理を行う。
3. そのエレメントの断面番号が、入力した断面番号と一致するかどうかチェックし、もし、その範囲であれば断面内のファイバー数と先頭番地を構造体にセットする。
4. 部材に含まれるエレメント数分、以下の処理を行う。この断面の先頭番地を構造体にセットする。
5. ここでは、その部材が部材モデル番号 51-70 であるかどうかチェックする。もし、同じであれば、構造体 E_Fiber_work と M_Fiber_work の先頭番地を取得する。
6. 履歴のワーク領域である構造体 n_m_bilinear、n_m_trilinear、n_m_concrete、n_m_analogy の数を数える。

9.4.5 部材モデルの出力仕様

本節では、新規の静的縮合モデルに関するデータ出力を考えてみよう。関連する出力項目として、部材応力とファイバー応力との 2 種類があり、これらを順に検討する。最初は、部材に関する応力を出力するサブルーチン Out_stress()を示す。このサブルーチンの中で、太文字で示した箇所が、新規モデルのために追加・変更するコードである。

```
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"
      dimension M_model(*)
      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, 3,3,3,3,3,3,3,3,3,3,
     7           3,3,3,3,3,3,3,3,3,3, 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/
       •
       •
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
      elseif(Member(i).element_type.eq.2) then
c――――――――――――――――――――――――――――★3 次元せん断弾塑性モデル
       •
      elseif(Member(i).element_type.eq.3) then
c――――――――――――――――――――――――――――★3 次元軸力弾塑性モデル
       •
      elseif(Member(i).element_type.eq.4 ) then
c――――――――――――――――――――――――――――★3 次元ブレースモデル
       •
      elseif(Member(i).element_type.ge.5.and.
     *   Member(i).element_type.le.10) then
c――――――――――――――――――――――――――――★Model No.5-10
        •
c――――――――――――――――――――――――――――★Model No. 18,19
      elseif(Member(i).element_type.eq.18.or.
     *       Member(i).element_type.eq.19) then
       •
      else
c――――――――――――――――――――――――――――★有限要素モデル
      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
 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
 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
```

このサブルーチンから分かるように、新規の部材モデル番号は 51-70 であることから、else 以下の処理となる。構造体成分 Member(i).stress に適切なデータがセットされていれば、このプログラムを変更しなくても良い。

次に、断面内のファイバーの応力とひずみを出力するサブルーチンについて検討しよう。サブルーチン Out_Fiber()で、関連する部分を抜き出して表示する。ここでは、部材モデルごとに出力しており、新たな部

材モデルに対しては、該当する部分を作る必要がある。

```
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,
     *      S_comp_model, E_modelx, M_modelx,
     *      E_fiber_work, M_fiber_work)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
c―――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      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)
       dimension E_modelx(*),M_modelx(*),S_comp_model(*)
       dimension E_fiber_work(*),M_fiber_work(*)
       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)
c-------------------------------★断面応力
       ie = Member(i).nm_element
       imm= Element(ie).n_element              ! 要素番号
       immm= Member(i).n_model_type            ! モデルタイプ別番号
       iet  = Member(i).element_type
       if(iet.eq.11) then
c-----------------------★モデル１１
       .
       elseif(iet.eq.12) then
c-----------------------★モデル１２
       .
       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
       .
       elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
c-----------------------★新規モデル 51-70
```

```
      i_comp= iet -  50                                                  ! 1
      n_out_stress  = S_comp_model(i_comp).n_out_stress
      nmmx= Member(i).n_model_type - 1                  ! M_Fiber_work の開始番号   2
      nmx = Element(ie).n_section(1) - 1                ! E_Fiber_work の開始番号
      do m=1, n_out_stress
      kk = S_comp_model(i_comp).nm_out_stress(m)         ! 出力エレメント番号       3
      if(kk.ne.0) then
      immx = E_Fiber_work(nmx+kk).nm_section             ! エレメント番号           4
      immmx= M_Fiber_work(nmmx+kk).nm_section            ! 内部エレメント番号
      nm_div=E_modelx(immx).n_section                                    ! 5
      nnm=M_modelx(immmx).nm_section -  1
      do k=1,nm_div
      v(k)=M_model_fiber(k+nnm).d_stress_x                               ! 6
      enddo
      write(iflz(6)) (v(k),k=1,nm_div)                  ! 応力           ! 7
      do k=1,nm_div
      v(k)=M_model_fiber(k+nnm).d_eps_x                                  ! 8
      enddo
      vf(1) = M_modelx(immmx).d_epsi_x                   !  軸方向歪
      vf(2) = M_modelx(immmx).d_epsi_y                   !  y 軸に関する曲げ歪
      vf(3) = M_modelx(immmx).d_epsi_z                   !  z 軸に関する曲げ歪
      write(iflz(6))(vf(k),k=1,3),(v(k),k=1,nm_div)      ! ひずみ       ! 9
      endif
      enddo
      endif
c――――――――――――――――――――――――――――――★断面応力出力終わり
      enddo
      return
      end
```

このサブルーチンでは、部材モデル毎に処理が分けられている。新規に設計した静的縮合モデルに対して出力部分のコードが追加されている。右に書かれているコメント番号に従って説明する。特に階層構造となっている構造体へのアクセス方法をここで理解されたい。次節以降で説明する他の処理でも同様の方法を用いてワーク用構造体にアクセスする。

1. 部材モデル番号が 51-70 であるかどうかチェックする。そうであれば、以下の出力処理を行う。新規部材モデル番号からモデル設定用構造体の番号を求め、その番号より部材の中で、応力とひずみを出力するエレメント数を取得する。
2. 構造体 M_Fiber_work の先頭番地と、同じく構造体 E_Fiber_work の先頭番地を取得する。
3. 応力とひずみを出力するエレメント数分、以下の処理を行う。まず、そのエレメント番号 kk を取得する。そのエレメント番号が 0 の場合は出力処理を行わない。

4. 構造体 M_Fiber_work の先頭番地と、同じく構造体 E_Fiber_work より、構造体 E_model と M_model の該当番地を取得する。
5. 構造体の成分 E_model(immx).n_section より、その断面のファイバー総数を取得し、同様に構造体の成分 M_model(immmx).nm_section より、該当するファイバーの番地を取得する。さらに、以下の処理をファイバー数分行う。
6. ファイバー軸方向応力を出力用の単精度配列 v にセットする。
7. 当該のファイバー応力をファイルに出力する。
8. ファイバー軸方向ひずみを出力用の単精度配列 v にセットする。さらに、断面に関する軸方向ひずみと y 軸、z 軸に関する曲げひずみを出力用の単精度配列 vf にセットする。
9. 当該断面に関するひずみをファイルに出力する。

9.4.6 線形剛性

本節では、新しい静的縮合モデルを SPACE に組み込むために、必要となるサブルーチンについて説明する。この必要となるサブルーチンは

１）線形剛性を求めるサブルーチン
２）非線形剛性を求めるサブルーチン
３）応力を求めるサブルーチン
４）応力をチェックし、接線剛性を求めるサブルーチン

である。以降の節から部材モデルに関する上記４つのサブルーチンを検討しよう。最初は、線形剛性について考える。階層が非常に深くなっているので注意して欲しい。また、先に示した構造体のリンク手法がここで使われている。この手法は後から説明する３つのサブルーチンでも同様に使用されている。ここでは、特にこの構造体のリンク手法と部材モデルの階層構造について理解されたい。線形の剛性行列を求めるサブルーチン Cal_stiff_linear()では、太文字で示した部分がコードを追加する箇所となっている。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stiff_linear
C       ―――――――――――――――――――――――――――――――
C     ●   線形剛性行列の計算(ok)
C     ―――――――――――――――――――――――――――――――
      subroutine Cal_stiff_linear(Model_type,Element,Member,Parameter_C,
```

```
     *      ak_linear, E_model11,E_model_fiber,
     *      M_model11, M_model_fiber,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,
     *      Bilinear_work,Trilinear_work,Concrete_work,
     *      work1_elememt,work2_elememt,work1_member,work2_member,
     *      S_comp_model, E_modelx, M_modelx,
     *      E_fiber_work, M_fiber_work)
C    ------------------------------------------
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / parameter_s / Parameter_C
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      dimension Member(*),Element(*)
      dimension ak_linear(12,12,*),cosin(2,32)
c-------------------------------★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
       .
c-----------------------------------------------------
      n_member = Parameter_C.N_member
      do i=1,n_member
      mem = i
      iet  = Member(i).element_type
      ie = Member(i).nm_element
      iett=(iet-1)/10
      ien= Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
       .
       .
c-------------------------------★Model_No.1 通常の有限要素弾性モデル
      goto 100
      elseif(iett.eq.1)then
      goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
       .
       .
c-------------------------------★Model_No.51-70 任意要素静的縮合モデル
```

```
      goto 100
      elseif(iett.eq.5.or. iett.eq.6)then
      call Cal_lin_stiff_Mxx(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i),E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,
     *      S_comp_model,E_fiber_work,M_fiber_work)
      goto 100
      endif
      goto 100
 9999 continue
  100 continue
      end do
      return
      end
```

このサブルーチンでは、部材モデルで階層構造を構成している。新規部材モデルを追加したため、その対応を取る必要がある。ここでは、新規モデルで必要となる５つの構造体を仮引数とし、それらの構造体の定義文が保存されているヘッダーファイルNew_submain.hをインクルードする。部材モデルで階層となっているため、新規モデルのコード番号51-70 までを条件として、サブルーチン Cal_lin_stiff_Mxx()をコールする。このとき、サブルーチンの引数として 5 つの構造体が引き渡す。計算が終了し、このサブルーチンから戻るときこの新規モデルの線形剛性が受け渡される。

新規静的縮合モデルの線形剛性を計算するサブルーチンを示す。このサブルーチンは、新たに設計しなければならないが、その内容は他の静的縮合モデルとほとんど同じであり、作成することは容易である。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_lin_stiff_Mxx
C       ―――――――――――――――――――――――――――――――――
C     ●   代表的な部材モデルの剛性 任意要素
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_lin_stiff_Mxx(Model_type,Member,Element,ak,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,
     *      S_comp_model,E_fiber_work,M_fiber_work)                    ! 1
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"                                          ! 2
      record / member_s          / Member
      record / element_s         / Element
      record / n_model_s         / Model_type
      record / E_model_fiber_s   / E_model_fiber
```

```
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
c―――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s     / S_comp_model                              ! 3
      record / E_modelx_s         / E_modelx
      record / M_modelx_s         / M_modelx
      record / E_fiber_work_s     / E_fiber_work
      record / M_fiber_work_s     / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)                       ! 4
      dimension E_fiber_work(*),M_fiber_work(*)
      dimension ak(12,12),akk(12,12)
      real*8, ALLOCATABLE :: c(:,:),ab(:,:),ba(:),alength(:)
      integer, ALLOCATABLE :: irest_Point(:,:),n_type(:)
C     ―――――――――――――――――――――――――――――――――――――
      iet = Member.n_model                        ! モデルタイプ番号
      ix_model= iet-50                            ! 任意要素モデル(51-69)
      nmmx= Member.n_model_type - 1               ! M_Fiber_work の開始番号     ! 5
      nmx = Element.n_section(1) - 1              ! E_Fiber_work の開始番号
      n_div= Element.n_section(2)                 ! 部材分割数
      n_if = 6*(n_div-1)                          ! 内部自由度
c―――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   irest_Point(6,n_div+1),n_type(n_div),alength(n_div)
     *         )
c―――――――――――――――――――――――――――――――★節点拘束表の作成
c                                                 未知数等をセット
      call set_modelx_dat(irest_Point,n_if,n_div,iubw,                        ! 6
     *         Element,Member,S_comp_model(ix_model),
     *         n_type,alength,
     *  Member.i_rigid_length,    ! i 端剛域
     *  Member.j_rigid_length)    ! j 端剛域
c―――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   c(0:iubw,n_if),ab(n_if,12),ba(n_if)
     *         )
c―――――――――――――――――――――――――――――――★剛性行列のゼロクリア
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      do i=1,n_if
      do j=0,iubw
      c(j,i)=0.
      enddo
      enddo
      do i=1,12
      do j=1,n_if
      ab(j,i)=0.
      enddo
      enddo
c―――――――――――――――――――――――――――――――★ワークベクトルのゼロクリア
```

```
      do i=1,n_div                                                    ! 7
      M_Fiber_Work(nmmx+i).an_stress=0.
      M_Fiber_Work(nmmx+i).an_vv=0.
      M_Fiber_Work(nmmx+i).an_ww=0.
      do j=1,6
      M_Fiber_Work(nmmx+i).ff_ip(j) = 0.
      enddo
      enddo
c-------------------------------★部材剛性行列の作成
      do i=1,n_div
         if(E_Fiber_work(nmx+i).n_Fiber_section.eq.0) then
            E_Fiber_work(nmx+i).nm_section=1
            M_Fiber_work(nmmx+i).nm_section=1
         endif
           imm = E_Fiber_work(nmx+i).nm_section        ! エレメント番号      ! 8
           immm= M_Fiber_work(nmmx+i).nm_section       ! 内部エレメント番号  ! 9
      call Stiff_Mx_I(i,n_type(i),akk,Member,alength,
     *       Model_type,Element,
     *       E_modelx(imm), E_model_fiber,                              ! 10
     *       M_modelx(immm), M_model_fiber,
     *       Bilinear_work,Trilinear_work,Concrete_work)
      call Bnd_FEM(i,akk,irest_Point,ak,c,ab,iubw,n_if)
      enddo
c-------------------------------★部材剛性行列の縮合
      call Typical_member_model(c,ab,ak, n_if,n_if,iubw,iubw,ba,ier)
c-------------------------------★両端の剛域処理
      call Deal_Rigid_element(ak,Member.i_rigid_length,
     *                        Member.j_rigid_length)
c-------------------------------★動的記憶領域の解放
      DEALLOCATE (
     *   c ,ab ,ba,irest_point,n_type,alength
     *        )
      return
      end
```

ここでは、部材モデル内の各エレメントの線形剛性を求め、さらに両端の節点変位に対応する剛性行列を、静的縮合を行うことによって求める。各部材内エレメントの線形剛性を求めるサブルーチンは、Stiff_Mx_I()である。

このサブルーチンは、他の静的縮合モデルとほとんど同じであり、太文字部分が追加・変更する箇所である。プログラムの右側のコメント番号にしたがって説明する。

1. 新しく設計した構造体５つが引数として受け渡される。
2. 新しく設計した構造体の定義文がヘッダーファイルに保持されており、そのファイル New_submain.h をインクルードしている。
3. 構造体の割付を行う。
4. 新しい構造体を整合配列として宣言する。

5. 構造体 M_Fiber_work と E_Fiber_work の開始番号をセットする。また、この部材内部のエレメント数をセットする。
6. 先に設計した新規の任意型静的縮合モデルを設定するサブルーチン set_modelx_dat()をコールし、モデル情報を取得する。
7. この新規モデルが静的縮合を行う際、必要となる内部節点とエレメントに関するワーク領域をゼロクリアする。
8. 部材内部のエレメント番号を構造体 E_Fiber_work().nm_section から取得する。ただし、この値が 0 の場合は 1 をセットする。これは、値 0 を用いると構造体の配列で設定外を指定する可能性があるためである。
9. 部材内部の部材番号を構造体 M_Fiber_work().nm_section から取得する。ただし、この値が 0 の場合は 1 をセットする。これは、前記と同様に値 0 を用いると構造体の配列で設定外を指定する可能性があるためである。
10. 線形の剛性行列を計算するサブルーチン Stiff_Mx_I()をコールする。そのとき、引数として２つの構造体を受け渡す。その配列番号は、8、9 で求められている。

次に、線形の剛性行列を計算するサブルーチンを検討しよう。このサブルーチンでも階層構造が見られる。これは、部材内のエレメントがこの階層の中から任意に選択可能であることを示す。今後、ここにエレメントの新規モデルを追加することで、さらに、この部材モデルの応用範囲が拡大する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Stiff_Mx_I
C       ―――――――――――――――――――――――――――――――――
C     ●   線形剛性行列の計算
C     ―――――――――――――――――――――――――――――――――
      subroutine Stiff_Mx_I(im,n_type,ak,Member,alength,
     *     Model_type,Element,
     *     E_modelx,E_model_fiber,M_modelx,M_model_fiber,
     *     Bilinear_work,Trilinear_work,Concrete_work)
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s          / Member
      record / element_s         / Element
      record / n_model_s         / Model_type
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
```

```
      record / E_model_fiber_s    / E_model_fiber
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      dimension ak(12,12),alength(*)
      dimension E_model_fiber(*),M_model_fiber(*)
c―――――――――――――――――――――――――――――――★要素及びモデルのセット
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      goto(11,12,13,14,15,16,17,18,19,20),n_type
   11 continue
c―――――――――――――――――――――――――――――――★弾性要素
       call Cal_lin_stiff_Mx(Member,Element,
     *      ak,alength(im) )
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――★ファイバー要素
      call Fiber_Model_GIx(ak,alength(im),Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――★MS 要素
      call MS_Model_GIx(ak,alength(im),Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
      goto 100
   14 continue
c―――――――――――――――――――――――――――――――★アナロジー要素
      call Analogy_Model_GIx(ak,Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber)
      goto 100
   15 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
   16 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
   17 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
   18 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
   19 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
```

```
   20 continue
c――――――――――――――――――――――――――――――――――★ダミー
  100 continue
      return
      end
```

ここでは、各種の断面モデルに対応して線形の剛性行列を計算サブルーチンが階層構造となって並んでいる。後は、これらのサブルーチンを設計して追加すれば良い。これらのサブルーチンは、他の部材モデルで使用されたものとほとんど同じであるため作成することは容易である。ここでは、ファイバーモデルの初期設定と線形剛性行列を計算するサブルーチン Fiber_Model_GIx()について示す。この中にも各ファイバーに対する履歴特性が下位の階層として構成されている。実際に剛性行列を作成するサブルーチン Fiber_Model_Gx()は次節で示す。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_Model_GIx
C     ――――――――――――――――――――――――――――――――――
C     ●   線形剛性行列の計算(ok)
C     ――――――――――――――――――――――――――――――――――
C     ●   Model_No.51-70
C     ――――――――――――――――――――――――――――――――――
      subroutine Fiber_Model_GIx(ak,alength,Member,Element,
     *           E_modelx,E_model_fiber,M_modelx,M_model_fiber,
     *           Bilinear_work,Trilinear_work,Concrete_work)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s           / Member
      record / element_s          / Element
      record / E_modelx_s         / E_modelx
      record / E_model_fiber_s    / E_model_fiber
      record / M_modelx_s         / M_modelx
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*)
      dimension ak(12,12)
c――――――――――――――――――――――――――――――――――★
      nm_div = E_modelx.n_section
      nn     = E_modelx.nm_section - 1
      nnm    = M_modelx.nm_section - 1
      do i=1,nm_div
      nn     =  nn  + 1
      nnm    =  nnm + 1
c――――――――――――――――――――――――――――――――――★データの初期設定
```

```
      M_model_fiber(nnm).d_E       = E_model_fiber(nn).E_1
      M_model_fiber(nnm).i_elastic  = 0        !  ファイバー要素の状態（弾性の場合は０：塑性は１）
      M_model_fiber(nnm).d_eps_x    = 0.       !  軸方向歪
      M_model_fiber(nnm).d_stress_x = 0.       !  軸方向応力
c―――――――――――――――――――――――――★ファイバーデータセット
      E_model_fiber(nn).Ary  = E_model_fiber(nn).A*E_model_fiber(nn).ry
      E_model_fiber(nn).Arz  = E_model_fiber(nn).A*E_model_fiber(nn).rz
      E_model_fiber(nn).Ary2 =
     *                       E_model_fiber(nn).Ary*E_model_fiber(nn).ry
      E_model_fiber(nn).Arz2 =
     *                       E_model_fiber(nn).Arz*E_model_fiber(nn).rz
      E_model_fiber(nn).Aryz =
     *                       E_model_fiber(nn).Arz*E_model_fiber(nn).ry
      nmx                    = M_model_fiber(nnm).n_type
      nm_type                = E_model_fiber(nn).nm_type
      goto ( 10,20,30,30,10,20,10,20),nm_type
  10  continue
c―――――――――――――――――――――――――★バイリニア型（スチール用）
      Bilinear_work(nmx).i_stat  = -1             !  ファイバー要素の状態
      goto 100
  20  continue
c―――――――――――――――――――――――――★非対称トリリニア型
      Trilinear_work(nmx).i_stat = -1             !  ファイバー要素の状態
      goto 100
  30  continue
c―――――――――――――――――――――――――★コンクリート型
      Concrete_work(nmx).i_stat  = -1             !  ファイバー要素の状態
      goto 100
  100 continue
      enddo
c―――――――――――――――――――――――――――――――――――――――★ファイバー要素のデータセット
      nm_div = E_modelx.n_section
      nn     = E_modelx.nm_section - 1
      nnm    = M_modelx.nm_section - 1
      ra     = 0.
      ray    = 0.
      raz    = 0.
      raz2   = 0.
      ray2   = 0.
      rayz   = 0.
      gg     = 0.
      aa     = 0.
      do i=1,nm_div
      nn     = nn+1
      nnm    = nnm+1
      A      = E_model_fiber(nn).A
      E      = M_model_fiber(nnm).d_E
      ra     = ra   + E*A
      ray    = ray  + E*E_model_fiber(nn).Arz
      raz    = raz  + E*E_model_fiber(nn).Ary
      ray2   = ray2 + E*E_model_fiber(nn).Arz2
      raz2   = raz2 + E*E_model_fiber(nn).Ary2
      rayz   = rayz + E*E_model_fiber(nn).Aryz
      aa     = aa   + A
```

```
      gg     = gg   + A*E_model_fiber(nn).G
      enddo
      gg     = gg / aa
      M_modelx.d_aa     = aa              !  断面積の和
      M_modelx.d_ra     = ra              !  E*断面積の和
      M_modelx.d_ray    = ray             !  E*A*z
      M_modelx.d_raz    = raz             !  E*A*y
      M_modelx.d_raz2   = raz2            !  E*A*y*y
      M_modelx.d_ray2   = ray2            !  E*A*z*z
      M_modelx.d_rayz   = rayz            !  E*A*z*y
      M_modelx.d_gg     = gg              !  G*A
      M_modelx.d_epsi_x =  0.             !  軸方向歪
      M_modelx.d_epsi_y =  0.             !  y 軸に関する曲げ歪
      M_modelx.d_epsi_z =  0.             !  z 軸に関する曲げ歪
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★初期剛性行列の計算
      call Fiber_Model_Gx(ak,alength,Member,Element,
     *      E_modelx,E_model_fiber,M_modelx,M_model_fiber)
      return
      end
```

9.4.7 非線形剛性

本節では、非線形剛性について考える。基本的には、線形の剛性行列を計算するサブルーチンと同じである。非線形剛性を求めるサブルーチン Get_nonlinear_stiff()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_nonlinear_stiff
C       ―――――――――――――――――――――――――――――――――――――――――
C     ●   接線剛性行列の計算(ok)
C     ―――――――――――――――――――――――――――――――――――――――――
      subroutine Get_nonlinear_stiff(N_analysis,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
     *     MSS_work, S_comp_model, E_modelx, M_modelx,
     *     E_fiber_work, M_fiber_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member)
C     ―――――――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
```

```
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
c―――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
       ・
       ・
C     ―――――――――――――――――――――――――――――――――――
      if(N_analysis.eq.7) return     !   線形解析;7
      do  9999 i=1,n_member
      if(Member(i).nm_analysis .eq. -1) goto 9999 ! 各部材の解析種別のチェック(-1:線形解析)
c―――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=past_disp_point(ires)
      else
      v(j)=0.
      endif
      enddo
c―――――――――――――――――――――――――――――――――★変位を釣合系から部材座標系に変換
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
c―――――――――――――――――――――――――――――――――★要素及びモデルのセット
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie = Member(i).nm_element
      im = Element(ie).n_element
      imm = Member(i).n_element_type
      ien= Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0)  goto 9998   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
       ・
       ・
c―――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
      goto 100
      elseif(iett.eq.1)then
      goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
       ・
       ・
c―――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素静的縮合モデル
      goto 100
      elseif(iett.eq.5.or. iett.eq.6)then
      call Cal_nonlin_stiff_Mxx(N_analysis,
     *      Model_type,Member(i),Element(ie),
```

```
     *      ak,ier,E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      S_comp_model,E_fiber_work,M_fiber_work)
      goto 100
      endif
      goto 100
 9998 continue
  100 continue
c――――――――――――――――――――――――――――――――――――★部材の接線剛性を釣合系に変換
      call Rotate_K(ak,rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),ak_nonlinear(1,1,i))
 9999 continue
      return
      end
```

このサブルーチンでも、部材モデルで階層構造を構成している。新規部材モデルを追加したため、その対応を取る必要がある。ここでは、新規モデルで必要となる５つの構造体を仮引数とし、それらの構造体の定義文が保持されているヘッダーファイルNew_submain.hをインクルードする。部材モデルで階層となっているため、新規モデルのコード番号51-70 までを条件として、サブルーチン Cal_nonlin_stiff_Mxx()をコールする。このとき、サブルーチンの引数として5つの構造体を引き渡す。計算が終了し、このサブルーチンから戻るときこの新規モデルの接線剛性行列が受け渡される。

次に、接線剛性行列を求めるサブルーチン Cal_nonlin_stiff_Mxx()を示す。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_nonlin_stiff_Mxx
C       ――――――――――――――――――――――――――――――――――――
C     ●   代表的な部材モデルの接線剛性
C     ――――――――――――――――――――――――――――――――――――
      subroutine Cal_nonlin_stiff_Mxx(N_analysis,
     *            Model_type,Member,Element, ak,ier,
     *     E_modelx,E_model_fiber,M_modelx,M_model_fiber,
     *     S_comp_model,E_fiber_work,M_fiber_work)
C     ――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / E_model_fiber_s   / E_model_fiber
      record / M_model_fiber_s   / M_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
c――――――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
```

```
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
      dimension ak(12,12),akk(12,12)
      real*8,  ALLOCATABLE :: c(:,:),ab(:,:),ba(:),alength(:),EA(:)
      integer, ALLOCATABLE :: irest_Point(:,:),n_type(:)
C     ――――――――――――――――――――――――――――――――――――
      iet = Member.n_model                          ! モデルタイプ番号
      ix_model= iet-50                              ! 任意モデル(51-69)
      nmmx= Member.n_model_type - 1                 ! M_Fiber_work の開始番号
      nmx = Element.n_section(1) - 1                ! E_Fiber_work の開始番号
      n_div= Element.n_section(2)                   ! 部材分割数
      n_if = 6*(n_div-1)                            ! 内部自由度
c―――――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   irest_Point(6,n_div+1),n_type(n_div),alength(n_div),EA(n_div)
     *         )
c―――――――――――――――――――――――――――――――――★節点拘束表の作成
c                                                   未知数等をセット
      call set_modelx_dat(irest_Point,n_if,n_div,iubw,
     *         Element,Member,S_comp_model(ix_model),
     *         n_type,alength,
     *  Member.i_rigid_length,    ! i 端剛域
     *  Member.j_rigid_length)    ! j 端剛域
c―――――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   c(0:iubw,n_if),ab(n_if,12),ba(n_if )
     *         )
c―――――――――――――――――――――――――――――――――★剛性行列のゼロクリア
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      do i=1,n_if
      do j=0,iubw
      c(j,i)=0.
      enddo
      enddo
      do i=1,12
      do j=1,n_if
      ab(j,i)=0.
      enddo
      enddo
      EAx=Element.A*Element.E
c―――――――――――――――――――――――――――――――――★部材剛性行列の作成
      do i=1,n_div
      imm = E_Fiber_work(nmx+i).nm_section          ! 要素番号
      immm= M_Fiber_work(nmmx+i).nm_section         ! 内部要素番号
      EA=EAx
      if(n_type(i).eq.2) then       ! ファイバー：他のモデルでも必要なときはここに加える
```

```
      EA=M_modelx(immm).d_ra
      endif
      call Stiff_Mx(i,n_type(i),akk,Member,alength,
     *       Model_type,Element,
     *       E_modelx(imm), E_model_fiber,
     *       M_modelx(immm), M_model_fiber)
      if(N_analysis.ne.9.and.N_analysis.ne.7.and.n_type(i).le.3) then        !  幾何学的非線形剛性
      call Cal_geomet_stiffx(M_Fiber_Work(nmmx+i).an_stress,Member,akk,alength(i))
      call Create_Kn(akk, M_Fiber_Work(nmmx+i).an_vv, M_Fiber_Work(nmmx+i).an_ww,
     *              EA,alength(i))
      endif
      call Bnd_FEM(i,akk,irest_Point,ak,c,ab,iubw,n_if)
      enddo
c―――――――――――――――――――――――――――――――★部材剛性行列の縮合
      call Typical_member_model(c,ab,ak, n_if,n_if,iubw,iubw,ba,ier)
c―――――――――――――――――――――――――――――――★両端の剛域処理
      call Deal_Rigid_element(ak,Member.i_rigid_length,
     *                        Member.j_rigid_length)
c―――――――――――――――――――――――――――――――★動的記憶領域の解放
      DEALLOCATE (
     *   c ,ab ,ba,irest_point,n_type,alength,EA
     *        )
      return
      end
```

接線剛性行列を求めるサブルーチン Cal_nonlin_stiff_Mxx()において、前節と同様に、階層構造を有する構造体にアクセスしている。下位層の構造体 E_modelx と M_modelx までのリンクは、前節で説明しており、ここでも再度確認されたい。ここでは、幾何学的非線形剛性を作成するサブルーチンで必要となる各エレメントの軸方向剛性 EA、軸力 N、y 方向と z 方向の相対変位について考えよう。まず、軸方向剛性 EA は、断面内に塑性域が生じると変化する。そこで、エレメントがファイバーモデルの場合、他のサブルーチンで計算した当該エレメントの剛性と断面積の積を表す M_modelx(immm).d_ra を軸剛性として使用する。また、そのエレメントの軸力、y 方向と z 方向の相対変位も他のサブルーチンでセットした構造体 M_Fiber_Work の各成分から取得する。これらの値の設定は、次節で説明する。最後に、この幾何学的非線形剛性は、部材長さを有するエレメントモデルに適用されて求められる。そこで、新規のエレメントモデルを組み込む場合はこの点に注意されたい。

次は、弾塑性剛性行列を求めるサブルーチン Stiff_Mx()について以下に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Stiff_Mx
C       ―――――――――――――――――――――――――――――――――
C     ●   接線剛性行列の計算(ok)
```

```
C     ―――――――――――――――――――――――――――――――――
      subroutine Stiff_Mx(im,n_type,ak,Member,alength,
     *       Model_type,Element,
     *       E_modelx,E_model_fiber,M_modelx,M_model_fiber)
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s          / Member
      record / element_s         / Element
      record / n_model_s         / Model_type
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension ak(12,12),alength(*)
      dimension vv(12)
      dimension E_model_fiber(*),M_model_fiber(*)
c―――――――――――――――――――――――――――――――★要素及びモデルのセット
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      goto(11,12,13,14,15,16,17,18,19,20),n_type
   11 continue
c―――――――――――――――――――――――――――――――★弾性要素
      call Cal_lin_stiff_Mx(Member,Element,ak,alength(im) )
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――★ファイバー要素
      it=it+1
      call Fiber_Model_Gx(ak,alength(im),Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber)
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――★MS 要素
      call MS_Model_Gx(ak,alength(im),Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber)
      goto 100
   14 continue
c―――――――――――――――――――――――――――――――★アナロジー要素
      iep=M_model_fiber(nnm).i_elastic
      call Analogy_Model_Gx(iep,ak,Member,Element,
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,nn)
      goto 100
   15 continue
c―――――――――――――――――――――――――――――――★ダミー
      goto 100
   16 continue
c―――――――――――――――――――――――――――――――★ダミー
```

```
      goto 100
   17 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   18 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   19 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   20 continue
c――――――――――――――――――――――――――――――★ダミー
  100 continue
      return
      end
```

弾塑性の剛性行列を求めるサブルーチン Fiber_Model_Gx()について以下に示す。この中のサブルーチン Fiber_GX()は既に第５章で説明したのでそちらを参照されたい。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_Model_Gx
C     ――――――――――――――――――――――――――――――――――
C     ●   接線剛性行列の計算
C     ――――――――――――――――――――――――――――――――――
C     ●   Model_No.51
C     ――――――――――――――――――――――――――――――――――
      subroutine Fiber_Model_Gx(ak,alength,Member,Element,
     *            E_modelx,E_model_fiber,M_modelx,M_model_fiber)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s          / Member
      record / element_s         / Element
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension ak(12,12)
c――――――――――――――――――――――――――――――★
      rix  = Element.RIx
      asy  = Element.ASy
      asz  = Element.ASz
c――――――――――――――――――――――――――――――★
      call Fiber_GK(ak,alength,
     *     M_modelx.d_ra, M_modelx.d_ray, M_modelx.d_raz,
     *               M_modelx.d_raz2,M_modelx.d_ray2,
     *               M_modelx.d_rayz,M_modelx.d_gg,
     *                 aa,rix,asy,asz)
      return
      end
```

9.4.8 部材の応力チェック

本節では、部材の弾塑性チェックを行い、部材内エレメンの力ー変位の履歴、あるいはファイバーの応力ーひずみ履歴を追跡する。また、その時点での部材接線剛性を求める。まず、部材の弾塑性チェックを行うサブルーチン Check_stress()を検討しよう。太文字部分がコードを追加する箇所である。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Check_stress
C     ―――――――――――――――――――――――――――――――――
C     ●   部材の塑性状態をチェックする(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Check_stress(Control,N_analysis,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
     *     MSS_work,
     *     Bilinear_work,Trilinear_work,Concrete_work,RO_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member ,
     *     S_comp_model,E_modelx,M_modelx,
     *     E_fiber_work,M_fiber_work)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record /control_s    / Control
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      c――――――――――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
       .
       .
c――――――――――――――――――――――――――――――――――――――――――――――――
```

```
      do i=1,n_member
c-------------------------------★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=disp_point(ires)
      vp(j)=past_disp_point(ires)
      else
      v(j)=0.
      vp(j)=0.
      endif
      enddo
c-------------------------------★部材両端の節点力のゼロセット
      do j=1,12
      f(j)=0.
      enddo
c-------------------------------★変位を釣合系から部材座標系に変換
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c-------------------------------★要素及びモデルのセット
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie   = Member(i).nm_element
      im   = Element(ie).n_element
      ien  = Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0)  goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
      .
      .
c-------------------------------★Model_No.1 通常の有限要素弾塑性モデル
      goto 100
      elseif(iett.eq.1)then
      goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
      .
      .
c-------------------------------★Model_No.11 両端ファイバーモデル
      goto 100
      elseif(iett.eq.5.or. iett.eq.6)then
      call Cal_check_stiff_Mx(Control,N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f,
     *      S_comp_model,E_fiber_work,M_fiber_work)
c-------------------------★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      do j=1,12
      Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
```

```
      endif
      goto 100
 9999 continue
c――――――――――――――――――――――――――――――――――★部材の接線剛性を釣合系に変換
  100 continue
      end do
      return
      end
```

このサブルーチンにおいても、部材モデルに関する階層構造となっている。適切な階層で、新規部材モデルに対する新しいサブルーチンをコールし、対応する新規のサブルーチン Cal_check_stiff_Mx()を作成する必要がある。このサブルーチンを以下に示す。この中で、引数として新たな構造体を受け渡す。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_check_stiff_Mx
C       ――――――――――――――――――――――――――――――――――――
C     ●   代表的な部材モデルの塑性チェック
C     ――――――――――――――――――――――――――――――――――――
      subroutine Cal_check_stiff_Mx(Control,N_analysis,
     *      mem_x,Model_type,Member,Element,
     *      E_modelx, E_model_fiber,
     *      M_modelx, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vp,f_p,
     *      S_comp_model,E_fiber_work,M_fiber_work)
C     ――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / E_model_fiber_s    / E_model_fiber
      record / M_model_fiber_s    / M_model_fiber
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension ak(12,12),f_p(12)
      dimension vv(12),vp(12),vvx(12)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*)
      c――――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
```

```
      real*8, ALLOCATABLE :: c(:,:),ab(:,:),alength(:),EA(:)
      real*8, ALLOCATABLE :: bav(:),akk(:,:,:),f1(:),f2(:)
      integer, ALLOCATABLE :: irest_Point(:,:),n_type(:)
C     ――――――――――――――――――――――――――――――――――――
      iet = Member.n_model                          ! モデルタイプ番号
      ix_model= iet-50                              ! 任意要素モデル(51-69)
      nmmx= Member.n_model_type - 1                 ! M_Fiber_work の開始番号
      nmx = Element.n_section(1) - 1                ! E_Fiber_work の開始番号
      n_div= S_comp_model(ix_model).n_div_element   ! 部材分割数
      n_if = 6*(n_div-1)                             ! 内部自由度
c―――――――――――――――――――――――――――――――――――――――――――
c
c    ★        部材の剛性行列の設定
c
c―――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   irest_Point(6,n_div+1),n_type(n_div),alength(n_div),EA(n_div),
     *    akk(12,12,n_div)
     *         )
c――――――――――――――――――――――――――――――――――――――――――――――★節点拘束表の作成
c                                                   未知数等をセット
      call set_modelx_dat(irest_Point,n_if,n_div,iubw,
     *         Element,Member,S_comp_model(ix_model),
     *         n_type,alength,
     *  Member.i_rigid_length,    ! i 端剛域
     *  Member.j_rigid_length)    ! j 端剛域
c――――――――――――――――――――――――――――――――――――――――――――――★動的記憶領域の確保
      ALLOCATE (
     *   c(0:iubw,n_if),ab(n_if,12),bav(n_if),f1(n_if),f2(n_if)
     *         )
c――――――――――――――――――――――――――――――――――――――――――――――★剛性行列のゼロクリア
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      do i=1,n_if
      do j=0,iubw
      c(j,i)=0.
      enddo
      enddo
      do i=1,12
      do j=1,n_if
      ab(j,i)=0.
      enddo
      enddo
      do i=1,12
      f_p(i)=0.
      enddo
      do i=1,n_if
      f1(i)=0.
      enddo
      EAx=Element.A*Element.E
```

```
c―――――――――――――――――――――――――――――――★部材剛性行列の作成
      do i=1,n_div
      imm = E_Fiber_work(nmx+i).nm_section              ! エレメント番号
      immm= M_Fiber_work(nmmx+i).nm_section             ! 内部エレメント番号
      EA=EAx
      if(n_type(i).eq.2) then       ! ファイバー：他のモデルでも必要なときはここに加える
      EA=M_modelx(immm).d_ra
      endif
c―――――――――――――――――――――――――――――――★内部部材の剛性行列の計算
      call Stiff_Mx(i,n_type(i),akk(1,1,i),Member,alength,
     *       Model_type,Element,
     *       E_modelx(imm), E_model_fiber,
     *       M_modelx(immm), M_model_fiber)
      if(N_analysis.ne.9.and.N_analysis.ne.7.and.n_type(i).ne.3) then    !  幾何学的非線形剛性
      call Cal_geomet_stiffx(M_Fiber_Work(nmmx+i).an_stress,Member,
     *                       akk(1,1,i),alength(i))
      call Create_Kn(akk(1,1,i), M_Fiber_Work(nmmx+i).an_vv, M_Fiber_Work(nmmx+i).an_ww,
     *              EA,alength(i))
      endif
c―――――――――――――――――――――――――――――――★剛性行列の分配
      call Bnd_FEM(i,akk(1,1,i),irest_Point,ak,c,ab,iubw,n_if)
      enddo
c―――――――――――――――――――――――――――――――――――――――――――――
c
c    ★         部材内部の変位と不釣合力の計算
c
c―――――――――――――――――――――――――――――――――――――――――――――
c―――――――――――――――――――――――――――――――★両端変位を剛域内部の変位に変換処理
      call Deal_Rigid_element_v(vv,Member.i_rigid_length,
     *                          Member.j_rigid_length)
c―――――――――――――――――――――――――――――――★部材内部変位の計算(c 行列の分解計算)
      do i=0,n_if-1
      k=i/6
      j = i - (k)*6
      f2(i+1) = M_Fiber_work(nmmx+k+1).ff_ip(j+1)                                    ! 1
      enddo
      call Typical_member_v(c,ab,f2,
     *                      n_if,n_if,iubw,iubw,ier,vv,bav)
c―――――――――――――――――――――――――――――――★部材内部、両端節点力の計算
      do i=1,n_div
      call Typical_member_p_force(akk(1,1,i),irest_Point(1,i),
     *                            vv,bav ,f_p,f1)
      enddo
c―――――――――――――――――――――――――――――――★部材内部、両端節点力から不釣合力へ
      do i=0,n_if-1
      k=i/6
      j = i - (k)*6
      M_Fiber_work(nmmx+k+1).ff_ip(j+1) = f1(i+1)                                    ! 2
      enddo
c―――――――――――――――――――――――――――――――★部材両端節点力への縮合
      call Typical_member_f(c,ab,f1,f_p,n_if,n_if,iubw,iubw)     ! f1 はデータが変更される
c―――――――――――――――――――――――――――――――★部材節点力の両端剛域処理
      call Deal_Rigid_element_f(f_p,Member.i_rigid_length,
     *                          Member.j_rigid_length)
```

```
c――――――――――――――――――――――――――――――――――――――
c
c    ★         部材の弾塑性チェック
c
c――――――――――――――――――――――――――――――――――――――
c―――――――――――――――――――――――――――★部材内部応力のチェック
      do i=1,n_div
      imm = E_Fiber_work(nmx+i).nm_section             ! 要素番号
      immm= M_Fiber_work(nmmx+i).nm_section            ! 内部要素番号
c――――――――――――――――――――――――★変位の取りだし
      ii=0
      do j=1,2
      do k=1,6
      ii=ii+1
      irest=irest_Point(k,i+j-1)
      if(irest.lt.0) then
      vvx(ii)=vv(-irest)
      elseif(irest.gt.0) then
      vvx(ii)=bav(irest)
      else
      vvx(ii)=0.
      endif
      enddo
      enddo
      goto(11,12,13,14,15,16,17,18,19,20), n_type(i)                            ! 3
   11 continue
c―――――――――――――――――――――――――――★弾性要素
      goto 100
   12 continue
c―――――――――――――――――――――――――――★ファイバー要素
      call Fiber_checkx(Control,i,N_analysis,                                   ! 4
     *      mem_x,alength(i),Member,Element,
     *      E_modelx(imm),E_model_fiber,M_modelx(immm),M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vvx,
     *      M_Fiber_Work(nmmx+i).an_vv, M_Fiber_Work(nmmx+i).an_ww,
     *      S_comp_model (ix_model).nm_out_stress(i))
      goto 100
   13 continue
c―――――――――――――――――――――――――――★MS 要素
      call MS_checkx(N_analysis,
     *      mem_x,alength(i),Member,Element,
     *      E_modelx(imm),E_model_fiber,M_modelx(immm),M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vvx,
     *      M_Fiber_Work(nmmx+i).an_vv, M_Fiber_Work(nmmx+i).an_ww,
     *      S_comp_model (ix_model).nm_out_stress(i))

      goto 100
   14 continue
c―――――――――――――――――――――――――――★アナロジー要素
      call Analogy_checkx(N_analysis,
     *      mem_x,alength(i),Member,Element,
     *      E_modelx(imm),E_model_fiber,M_modelx(immm),M_model_fiber,vvx,
     *      M_Fiber_Work(nmmx+i).an_vv, M_Fiber_Work(nmmx+i).an_ww,
     *      S_comp_model (ix_model).nm_out_stress(i))
```

```
      goto 100
   15 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   16 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   17 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   18 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   19 continue
c――――――――――――――――――――――――――――――★ダミー
      goto 100
   20 continue
c――――――――――――――――――――――――――――――★ダミー
  100 continue
c――――――――――――――――――――――――――――――★部材の軸力計算（幾何剛性作成用）
      call nonlinear_stress_N(akk(1,1,i),vvx,fnn)
      M_Fiber_Work(nmmx+i).an_stress = M_Fiber_Work(nmmx+i).an_stress + fnn        ! 5
c――――――――――――――――――――――――――――★部材内部変位を足しこむ
      M_Fiber_Work(nmmx+i).an_vv = M_Fiber_Work(nmmx+i).an_vv +(vvx(8) - vvx(2))   ! 6
      M_Fiber_Work(nmmx+i).an_ww = M_Fiber_Work(nmmx+i).an_ww +(vvx(9) - vvx(3))
      enddo
c――――――――――――――――――――――――――――――★動的記憶領域の解放
      DEALLOCATE (
     *   c ,ab ,irest_point,n_type,alength,EA,
     *   bav,akk,f1,f2    )
      return
      end
```

このプログラム自身は作成しなければならないが、内容は他のプログラムとほぼ同じである。ここでは、他と異なる箇所を説明する。非線形剛性を求める部分は前節で説明した。その他の部分について解説する。

1. 前回の部材内節点に対応した不釣合力を配列 f1 にコピーする。
2. 新しく計算した不釣合力を、M_Fiber_work(nmmx+k+1).ff_ip(j+1) にコピーする。
3. ここでも、部材内のエレメントに階層構造が見られる。現在は、弾性はり、ファイバーモデル、MS モデル、アナロジーモデルが登録されている。
4. モデル番号 2 では、ファイバーモデルの履歴特性チェックを行うサブルーチン Fiber_checkx()をコールする。
5. エレメントの増分軸力を計算し、増分前の軸力に足しこむ。

6．エレメントの増分相対変位(v,w)を、増分前の相対変位に足しこむ。

ファイバーモデルの履歴を追跡するサブルーチン Fiber_checkx()を以下に示す。このサブルーチンには、各ファイバーの履歴特性に関するサブルーチンが階層構造となって構成されている。現在、SPACE に登録されているモデルは８つであり、今後履歴モデルを追加する場合は、ここに該当するサブルーチンを挿入することになる。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_checkx
C     ―――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の材料非線形性チェックし、応力を計算
C     ―――――――――――――――――――――――――――――――――
      subroutine Fiber_checkx(Control,idiv,N_analysis,
     *      mem_x,Alength,Member,Element,
     *      E_modelx,E_model_fiber,M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,an_vv,an_ww,
     *       n_dstat)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record /control_s    / Control
      record / member_s          / Member
      record / element_s         / Element
      record / E_modelx_s        / E_modelx
      record / E_model_fiber_s   / E_model_fiber
      record / M_modelx_s         / M_modelx
      record / M_model_fiber_s   / M_model_fiber
      record / Bilinear_work_s   / Bilinear_work
      record / Trilinear_work_s  / Trilinear_work
      record / Concrete_work_s   / Concrete_work
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*)
      dimension vv(12)
c―――――――――――――――――――――――――――――――★i 端部ファイバー
c―――――――――――――――――――――――――――――――★歪のセット
      d_epsi_x_1 =  (vv(7)  - vv(1)) / Alength   !  軸方向歪
      d_epsi_y_1 =  (vv(11) - vv(5)) / Alength   !  y軸に関する曲げ歪
      d_epsi_z_1 =  (vv(12) - vv(6)) / Alength   !  z軸に関する曲げ歪
      if(N_analysis.eq.10.or.N_analysis.eq.8) then                  ! 非線形軸方向歪
      d_epsi_x_1= d_epsi_x_1+strain_nonlinear(an_vv,
     *           an_ww,vv,Alength)
      endif
      if(Member.analysis_3D .eq. 1) d_epsi_z_1 =0.  ! 平面問題における面外方向の曲げ変形を無視する
      if(Member.analysis_3D .eq. 2) d_epsi_y_1 =0.  ! 平面問題における面外方向の曲げ変形を無視する
      M_modelx.d_epsi_x =  d_epsi_x_1 + M_modelx.d_epsi_x     !  軸方向歪
      M_modelx.d_epsi_y =  d_epsi_y_1 + M_modelx.d_epsi_y     !  y軸に関する曲げ歪
      M_modelx.d_epsi_z =  d_epsi_z_1 + M_modelx.d_epsi_z     !  z軸に関する曲げ歪
```

```
c――――――――――――――――――――――――――――――――★ファイバー要素のチェック
      nm_div   = E_modelx.n_section
      nn       = E_modelx.nm_section - 1
      nnm      = M_modelx.nm_section - 1
      iistat   = 0                              ! 塑性率を計算するための指標
      do i=1,nm_div
      nn       = nn  + 1
      nnm      = nnm + 1
      nmx      = M_model_fiber(nnm).n_type
      nm_type  = E_model_fiber(nn).nm_type
      du =  d_epsi_x_1 +                                        ! 軸方向歪
     *      d_epsi_y_1 * E_model_fiber(nn).rz -                  ! y 軸に関する曲げ歪
     *      d_epsi_z_1 * E_model_fiber(nn).ry                    ! z 軸に関する曲げ歪
      if(N_analysis.le.8.or.Member.nm_analysis.eq.-1) then
c――――――――――――――――――――――――★弾性解析
c                                   ファイバー軸力計算
      M_model_fiber(nnm).d_stress_x = M_model_fiber(nnm).d_E*du +
     *                               M_model_fiber(nnm).d_stress_x
      else
      if(nm_type/10.eq.0) then
c――――――――――――――――――――――――★弾塑性解析
      goto ( 10,20,30,40,50,60,70,80),nm_type
  10  continue
c――――――――――――――――――――――――★バイリニア型（スチール用）
      call BiLinear(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).Q_1,du,
     *              M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1)
      if(Bilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      if(Bilinear_work(nmx).i_stat.ne.0) then
      endif
      goto 100
  20  continue
c――――――――――――――――――――――――★対称トリリニア型 （スチール用）
      call TriLinear(M_model_fiber(nnm).d_E,Trilinear_work(nmx).i_stat,
     *               E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *               E_model_fiber(nn).E_3,
     *               E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *               du,M_model_fiber(nnm).d_stress_x,
     *               Trilinear_work(nmx).P1(1))
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100
  30  continue
c――――――――――――――――――――――――★コンクリート型
      call Concrete(M_model_fiber(nnm).d_E,Concrete_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).E_3,E_model_fiber(nn).Ec_3,
     *              E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *              E_model_fiber(nn).Ec_1,E_model_fiber(nn).Ec_2,
     *              du,  M_model_fiber(nnm).d_stress_x,
     *              Concrete_work(nmx).p1(1),Concrete_work(nmx).P1(2),
     *              Concrete_work(nmx).ipret,Concrete_work(nmx).P1(3),
     *              Concrete_work(nmx).p1(4),Concrete_work(nmx).P1(5),
     *              Concrete_work(nmx).ipre_c)
```

```
      if(Concrete_work(nmx).i_stat.ne.0.and.Concrete_work(nmx).i_stat
     *            .ne.3) iistat = iistat + 1
      goto 100
   40 continue
c――――――――――――――――――――――――★曲線コンクリート型
      call Concrete_e(M_model_fiber(nnm).d_E,Concrete_work(nmx).i_stat,
     *             E_model_fiber(nn).E_1,E_model_fiber(nn).E_3,
     *             E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *             du,  M_model_fiber(nnm).d_stress_x,
     *             Concrete_work(nmx).P1(1),Concrete_work(nmx).P1(2),
     *             Concrete_work(nmx).ipret,Concrete_work(nmx).ipre_c,
     *             Concrete_work(nmx).P1(3),Concrete_work(nmx).P1(4),
     *             Concrete_work(nmx).P1(5),E_model_fiber(nn).Ec_1,
     *             Concrete_work(nmx).P1(6),E_model_fiber(nn).Ec_2)
      if(Concrete_work(nmx).i_stat.eq.0)then
      if(du.lt.E_model_fiber(nn).Ec_1)then
      iistat = iistat + 1
      endif
      elseif(Concrete_work(nmx).i_stat.ne.3)then
      iistat = iistat + 1
      endif
      goto 100
   50 continue
c――――――――――――――――――――――――★バイリニア型（移動＋等方効果用）
      call BiLinear_h(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *             E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *             E_model_fiber(nn).Q_1,du,
     *             M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1,
     *             E_model_fiber(nn).Beta)
      if(Bilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100
   60 continue
c――――――――――――――――――――――――★対称トリリニア型（移動＋等方効果用）
      call TriLinear_h(M_model_fiber(nnm).d_E,
     *              Trilinear_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).E_3,
     *              E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *              du,M_model_fiber(nnm).d_stress_x,
     *             Trilinear_work(nmx).P1(1),Trilinear_work(nmx).P1(2),
     *             E_model_fiber(nn).Beta, E_model_fiber(nn).Beta_2)
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100
   70 continue
c――――――――――――――――――――――――★非対称バイリニア型
      call BiLinear_AS(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *             E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *             E_model_fiber(nn).Q_1,E_model_fiber(nn).Ec_1,
     *             E_model_fiber(nn).Ec_2,E_model_fiber(nn).Qc_1,du,
     *             M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1,
     *             Bilinear_work(nmx).P2,Bilinear_work(nmx).Sig_z,
     *             E_model_fiber(nn).Beta)
      if(Bilinear_work(nmx).i_stat.eq.2.or.
     *   Bilinear_work(nmx).i_stat.eq.3) iistat = iistat + 1
```

```
      goto 100
   80 continue
c――――――――――――――――――――――――――★非対称トリリニア型
      call TriLinear_AS(M_model_fiber(nnm).d_E,
     *               Trilinear_work(nmx).i_stat,
     *               E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *               E_model_fiber(nn).E_3,E_model_fiber(nn).Ec_1,
     *               E_model_fiber(nn).Ec_2,E_model_fiber(nn).Ec_3,
     *               E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *               E_model_fiber(nn).Qc_1,E_model_fiber(nn).Qc_2,
     *               du,M_model_fiber(nnm).d_stress_x,
     *              Trilinear_work(nmx).P1(1),Trilinear_work(nmx).P1(2),
     *              Trilinear_work(nmx).P1(3),
     *               E_model_fiber(nn).Beta, E_model_fiber(nn).Beta_2)
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100
      elseif(nm_type/10.eq.10) then
c――――――――――――――――――――――――――★弾塑性解析
      goto (110,120),nm_type - 100
   110 continue
c――――――――――――――――――――――――――★個人用ファイバー履歴特性
c     call New_model_fiber()
      goto 100
   120 continue
c――――――――――――――――――――――――――★個人用ファイバー履歴特性
      goto 100
      endif
c――――――――――――――――――――――――――★弾塑性解析終了
      endif
   100 continue
      M_model_fiber(nnm).d_eps_x = M_model_fiber(nnm).d_eps_x + du !ファイバー要素の歪
      enddo
c――――――――――――――――――――――――――――――――――★ファイバー要素応力の計算
      if(n_dstat.ge.1.and.n_dstat.le.3) then      ! n_dstat:塑性状態を表す位置（0:表示しない）
      d_state = float(iistat)/float(nm_div)   ! 塑性した面積の計算
      if(d_state .eq.0) then
      Member.d_stat(n_dstat) = 0
      elseif(d_state .ge.0.8) then
      Member.d_stat(n_dstat) = 2
      else
      Member.d_stat(n_dstat) = 1
      endif
      endif
      nm_div  = E_modelx.n_section
      nn      = E_modelx.nm_section - 1
      nnm     = M_modelx.nm_section - 1
      ra      = 0.
      ray     = 0.
      raz     = 0.
      raz2    = 0.
      ray2    = 0.
      rayz    = 0.
      gg      = 0.
      aa      = 0.
```

```
      AN      = 0.
      AMy     = 0.
      AMz     = 0.
      do i=1,nm_div
      nn      = nn  + 1
      nnm     = nnm + 1
      A       = E_model_fiber(nn).A
      E       = M_model_fiber(nnm).d_E
      ra      = ra   + E*A
      ray     = ray  + E*E_model_fiber(nn).Arz
      raz     = raz  + E*E_model_fiber(nn).Ary
      ray2    = ray2 + E*E_model_fiber(nn).Arz2
      raz2    = raz2 + E*E_model_fiber(nn).Ary2
      rayz    = rayz + E*E_model_fiber(nn).Aryz
      aa      = aa + A
      gg      = gg + A*E_model_fiber(nn).G
      ANN     = M_model_fiber(nnm).d_stress_x*E_model_fiber(nn).A
      AN      = AN + ANN
      AMy     = AMy + ANN * E_model_fiber(nn).rz
      AMz     = AMz + ANN * E_model_fiber(nn).ry
      enddo
      nn=E_modelx.nm_section
      call jikuzero_control(Control,ra,E_model_fiber(nn).E_1,
     *                      E_model_fiber(nn).A)
      M_modelx.d_aa = aa              !  断面積の和
      M_modelx.d_ra = ra              !  E*断面積の和
      M_modelx.d_ray = ray            !  E*A*z
      M_modelx.d_raz = raz            !  E*A*y
      M_modelx.d_raz2 = raz2          !  E*A*y*y
      M_modelx.d_ray2 = ray2          !  E*A*z*z
      M_modelx.d_rayz = rayz          !  E*A*z*y
      M_modelx.d_gg   = gg            !  G*A
      return
      end
```

9.4.9 部材の応力

本節では、部材応力について考える。部材応力を計算するサブルーチン Cal_stress()では、部材モデル毎に計算を実行している。したがって、新規の静的縮合部材モデルに対して、部材モデル番号 51-70 で処理を行うサブルーチン Cal_stress_Mx()を挿入する。SPACE では、部材両端の応力は、接線剛性を用いて計算する簡易的な方法を使用している。そのため、サブルーチン Cal_stress_Mx()は容易に作成することができる。また、ここで、部材中央の応力を求めている。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress
C       ―――――――――――――――――――――――――――――――――
C     ●   部材内応力の計算(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_stress(Member,n_member,Model_type,Element,
```

```
     *     past_disp_point,past_vel_point,est_ddisp_point,rot_memb,
     *     E_model6_real,ak_nonlinear)
C     ―――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / e_model6_real_s / E_model6_real
      dimension Member(*),Element(*),E_model6_real(*)
      dimension rot_memb(3,3,2,*),ak_nonlinear(12,12,*)
      dimension past_disp_point(*),past_vel_point(*),est_ddisp_point(*),
     *          v(12),vv(12),vp(12),vpp(12),ak(12,12)
c―――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
c――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      vp(j)=past_disp_point(ires)
      v(j)=est_ddisp_point(ires)
      else
      v(j)=0.
      vp(j)=0.
      endif
      enddo
c――――――――――――――――――――――――――――――――★変位を釣合系から部材座標系に変換
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c――――――――――――――――――――――――――――――――★要素及びモデルのセット
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie = Member(i).nm_element
      im = Element(ie).n_element
      imm = Member(i).n_element_type
      ien= Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
c――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
      call Cal_stress_M1(Member(i),Element(ie),
     *     ak_nonlinear(1,1,i),v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
   12 continue
c――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      goto 100
   13 continue
c――――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
      goto 100
   14 continue
c――――――――――――――――――――――――――――――――★Model_No.4 3 次元ケーブル弾塑性モデル
```

```
      goto 100
   15 continue
c――――――――――――――――――――――――――――――――――★Model_No.5 3 次元免振モデル
      goto 100
   16 continue
c――――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル(ok)
      goto 100
   17 continue
c――――――――――――――――――――――――――――――――――★Model_No.7 3 次元プレテンション動作モデル
      goto 100
   18 continue
c――――――――――――――――――――――――――――――――――★Model_No.8
      goto 100
   19 continue
c――――――――――――――――――――――――――――――――――★Model_No.9
      goto 100
   20 continue
c――――――――――――――――――――――――――――――――――★Model_No.10
      goto 100

      elseif(iett.eq.1)then
      goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
c――――――――――――――――――――――――――――――――――★Model_No.11 両端ファイバーモデル
      goto 100
  112 continue
c――――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
      goto 100
  113 continue
c――――――――――――――――――――――――――――――――――★Model_No.13 両端 MS モデル
      goto 100
  114 continue
c――――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
      goto 100
  115 continue
c――――――――――――――――――――――――――――――――――★Model_No.15 幾何学非線形+弾塑性型有限要素モデ
      goto 100
  116 continue
c――――――――――――――――――――――――――――――――――★Model_No.16
      goto 100
  117 continue
c――――――――――――――――――――――――――――――――――★Model_No.17
      goto 100
  118 continue
c――――――――――――――――――――――――――――――――――★Model_No.18
      goto 100
  119 continue
c――――――――――――――――――――――――――――――――――★Model_No.19
      goto 100
  120 continue
c――――――――――――――――――――――――――――――――――★Model_No.20
      goto 100
      elseif(iett.eq.5.or. iett.eq.6)then
c――――――――――――――――――――――――――――――――――★Model_No.51-70
```

```
      call Cal_stress_Mx( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      do j=1,6
      k=j+12
      Member(i).stress(k)=(Member(i).stress(j)+
     *                  Member(i).stress(j+6))*0.5
      enddo
      goto 100
      endif
      goto 100
 9999 continue
  100 continue
      end do
      return
      end
```

部材両端の応力を計算するサブルーチンは、接線剛性を用いて計算する簡易型で行う。そのため、以下のように容易に作成可能である。

```
C     ――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress_Mx
C       ――――――――――――――――――――――――――――――――
C     ●   代表的な部材モデルの応力計算：非線形剛性を用いて、材端応力を計算する
C     ――――――――――――――――――――――――――――――――
      subroutine Cal_stress_Mx(Model_type,Member,Element, ak,vv,r1,r2)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / member_s    / Member
      record / element_s   / Element
      dimension ak(12,12) ,vv(12),r1(3,3),r2(3,3),st(12),ss(12)
C     ――――――――――――――――――――――――――――――――
      do i=1,12
      s=0.
      do j=1,12
      s=s+ak(i,j)*vv(j)
      enddo
      st(i)=s
      enddo
c――――――――――――――――――――――――――――――――★全体座標から部材座標へ変換
      call RotateL_v(1,st,r1,r2,ss)
      do i=1,6
      Member.stress(i)=-ss(i)+Member.stress(i)
      enddo
      do i=7,12
      Member.stress(i)=ss(i)+Member.stress(i)
      enddo
      return
      end
```

以上で、新規静的縮合モデルの組み込みは終了である。ただし、他のモジュール、例えば動的プレゼンターや静的ソルバーなどに、新規静的

縮合モデルの仕様を追加し、動的解析システムと適合させなければならない。それについては各マニュアルで説明する。

9.4.10 部材モデルの組み込みの復習

全節では、新たな部材モデルを SPACE に組み込む方法について解説した。ここでは、復習のために前節で示した部材モデルを静的解析システムとプレゼンテーションに組み込む手順を示そう。

静的解析システムは、動的解析システムとほぼ同じシステムを用いている。そこで、新しい部材モデルを組み込む方法を確実にものにするため、この静的解析システムに部材モデルを組み込む手順を示す。部材モデルは、前節と同じとする。

1．新しいヘッダーファイルを主サブルーチン submain.f に組み込む。
2．新しいヘッダーファイルNew_submain.hで定義した新しい構造体の宣言、確保を行うコードを挿入する。
3．既構造体に項目を追加する。
4．このモデルのために新たに設計したサブルーチンをプロジェクトに挿入する。

```
Get_S_comp_model.f
Cal_check_stiff_Mx.f
Cal_lin_stiff_Mxx.f
Cal_nonlin_stiff_Mxx.f
Cal_stress_Mx.f
Fiber_checkx.f
Fiber_Model_GIx.f
Fiber_Model_Gx.f
Stiff_Mx.f
Stiff_Mx_I.f
set_modelx_dat.f
```

5．以下の主サブルーチン submain.f 内サブルーチンの引数を変更する。

```
Get_structure
Fiber_input
Cal_stiff_linear
Check_stress
Out_Fiber
Get_nonlinear_stiff
```

６．以下の既設のサブルーチン内を変更する。

inpute()
Get_parameters()
Get_structure()
Fiber_input()
Cal_stiff_linear()
Check_stress()
Out_Fiber()
Get_nonlinear_stiff()
Cal_stress()

７．ファイバー入力の条件に新規モデルを加える。

８．静的解析では､収束過程でサブルーチン Cal_unb_stress()を用いて不釣合力を計算する。この不釣合い力を計算するサブルーチンは、動的解析ではないので新たに設計する必要がある。以下に示すように太文字の部分を変更する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_unb_stress
C     ―――――――――――――――――――――――――――――――――
C     ●   部材の不釣合力を計算する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_unb_stress(N_analysis,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
     *     MSS_work,
     *     Bilinear_work,Trilinear_work,Concrete_work,RO_work,
     *     work1_elememt,work2_elememt, work1_member,
     *     work2_member,ld_point,
     *     S_comp_model,E_modelx,M_modelx,
     *     E_fiber_work,M_fiber_work)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s    / Member
      record / element_s   / Element
```

```
      record / n_model_s   / Model_type
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
c―――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
c―――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
c
c―――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
c      record /element2_s / Element
      record / RO_work_s          /  RO_work
c―――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
c      record /element3_s / Element
c      record /member3_s  / Member
C     record / N_Buckling_s           / N_Buckling
c―――――――――――――――――――――――――――――――★Model_No.4 3 次元ケーブル弾塑性モデル
c      record /element4_s / Element
c      record /member4_s  / Member
c―――――――――――――――――――――――――――――――★Model_No.5 3 次元免振モデル
c      record /element5_s / Element
c      record /member5_s  / Member
       record / MSS_work_s         /  MSS_work
c―――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル
c      record /element6_s         / Element
       record / E_model6_real_s   / E_model6_real
c―――――――――――――――――――――――――――――――★Model_No.7 3 次元バネモデル
c      record /element7_s / Element
c      record /member7_s  / Member
c      record / E_model7_real_s    / E_model7_real
c―――――――――――――――――――――――――――――――★Model_No.11 両端ファイバーモデル
       record / E_model11_s    / E_model11
       record / M_model11_s    / M_model11
c―――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
       record / E_model12_s    / E_model12
       record / M_model12_s    / M_model12
c―――――――――――――――――――――――――――――――★Model_No.13 両端ピン、中央ファイバーモデル
       record / E_model13_s    / E_model13
       record / M_model13_s    / M_model13
c―――――――――――――――――――――――――――――――★Model_No.15 両端ファイバーFEM モデル
       record / E_model15_s    / E_model15
       record / M_model15_s    / M_model15
c―――――――――――――――――――――――――――――――★Model_No.21 両端 MS モデル
       record / E_model21_s    / E_model21
       record / M_model21_s    / M_model21
c―――――――――――――――――――――――――――――――★Model_No.22 両端、中央 MS モデル
       record / E_model22_s    / E_model22
       record / M_model22_s    / M_model22
c―――――――――――――――――――――――――――――――★Model_No.31 両端アナロジーモデル
```

```
      record / E_model31_s    / E_model31
      record / M_model31_s    / M_model31
c-----------------------------★Model_No.32 両端、中央アナロジーモデル
      record / E_model32_s    / E_model32
      record / M_model32_s    / M_model32
c-----------------------------★Model_No.33 両端、中央アナロジーモデル
      record / E_model33_s    / E_model33
      record / M_model33_s    / M_model33
c-----------------------------★Model_No.51 3 次元プレテンション動作モデル
c     record / E_model51_s    / E_model51
c     record / M_model51_s    / M_model51

c-----------------------------★ファイバーモデル用エリア
      record / E_model_fiber_s   / E_model_fiber
      record / M_model_fiber_s   / M_model_fiber
c-------------------------------------------------------------
     dimension E_model_fiber(*),M_model_fiber(*)
     dimension Member(*),Element(*),E_model6_real(*)
     dimension MSS_work(*),RO_work(*)
     dimension E_model11(*),M_model11(*)
     dimension E_model12(*),M_model12(*)
     dimension E_model13(*),M_model13(*)
     dimension E_model15(*),M_model15(*)
     dimension E_model21(*),M_model21(*)
     dimension E_model22(*),M_model22(*)
     dimension E_model31(*),M_model31(*)
     dimension E_model32(*),M_model32(*)
     dimension E_model33(*),M_model33(*)
     dimension ak_nonlinear(12,12,*),rot_memb(3,3,2,*)
     dimension work1_elememt(*),work2_elememt(*),
    *          work1_member(*), work2_member(*)
     dimension past_disp_point(*),disp_point(*)
     dimension vv(12),vp(12),vpp(12),v(12),ak(12,12)
     dimension f(12),ff(12)
     real*8 ld_point(*)
c-------------------------------------------------------------
c       Model_No.1 = 1             !  通常の有限要素弾塑性モデル
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
c       Model_No.6 = 6             !  3 次元制震 Maxwell モデル
c       Model_No.7 = 7             !  3 次元プレテンション動作モデル
c
c    要素数
c      structure / element_s/
c      integer  element_type   !  要素タイプ
c      integer  n_element      !  非線形要素番号
c      real*8   E              !  ヤング係数
c      real*8   G              !  せん断係数
c      real*8   A              !  断面積
c      real*8   RIx            !  ねじり剛性
c      real*8   RIy            !  ｙ軸断面二次モーメント
c      real*8   RIz            !  ｚ軸断面二次モーメント
```

```
c      real*8   AM              !  単位長さ当たりの質量
c      integer  nm_damp         !  部材減衰の有無
c      end structure
c     record /element_s/ Element
c     ALLOCATABLE ::Element(:)
c     ALLOCATE (Element(n_element))
c
c
c     部材数
c      structure / member_s/
c      integer  nm_element         !  要素番号
c      integer  element_type       !  要素タイプ
c      integer  n_element_type     !  要素タイプ別番号
c      integer  nm_dll_element     !  DLL を用いた要素か（０；システム内要素、１：DLL 要素）
c      integer  nm_point(2)        !  節点番号
c      integer  irest(12)          !  部材両端の自由度番号表
c      integer  nm_analysis        !  部材解析種別
c      integer  nm_group           !  部材グループ
c      integer  nm_local_coord(2)  !  局所座標系の有無とその回転行列の番号
c      integer  nm_damp            !  部材減衰の有無とその減衰行列の番号
c      real*4   alength            !  長さ
c      real*4   rot_x              !  部材主軸の回転角度（度）
c      real*8   force(12)          !  部材両端の部材端力
c      end structure
c     record / member_s / Member
c     ALLOCATABLE :: Member (:)
c     ALLOCATE (Member (n_member))
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(20)     ! 要素モデルの番号
c      integer    n_div_model(20)    ! 要素モデルの分割数
c      integer    nm_div_model(20)   ! 要素モデル内のサブ要素の分割数
c      integer    n_e_model(20)      ! 要素モデルの数
c      integer    n_m_model(20)      ! 部材モデルの数
c      integer    n_damp             ! 部材減衰ありか
c       end structure
c      record / n_model_s / Model_type
c
c
c      ak_nonlinear          real*8    接線剛性行列
c      Member                structure
c      n_member              integer   部材数
c      Model_type            structure
c      Element               structure
c      past_disp_point(*)    real*8    増分前の変位
c      rot_memb              real*8    回転行列
c      E_model1_int          integer   要素モデル１
c      E_model1_real         real*8    要素モデル１
c      M_model1_int          integer   部材モデル１
c      M_model1_real         real*8    部材モデル１
c      E_model2_int          integer   要素モデル２
c      E_model2_real         real*8    要素モデル２
```

```
c      M_model2_int          integer   部材モデル２
c      M_model2_real         real*8    部材モデル２
c      E_model3_int          integer   要素モデル３
c      E_model3_real         real*8    要素モデル３
c      M_model3_int          integer   部材モデル３
c      M_model3_real         real*8    部材モデル３
c      E_model4_int          integer   要素モデル４
c      E_model4_real         real*8    要素モデル４
c      M_model4_int          integer   部材モデル４
c      M_model4_real         real*8    部材モデル４
c      E_model5_int          integer   要素モデル５
c      E_model5_real         real*8    要素モデル５
c      M_model5_int          integer   部材モデル５
c      M_model5_real         real*8    部材モデル５
c      E_model6_int          integer   要素モデル６
c      E_model6_real         real*8    要素モデル６
c      M_model6_int          integer   部材モデル６
c      M_model6_real         real*8    部材モデル６
c      work1_elememt         real*8    DLL 用ワークエリア
c      work2_elememt         real*8    DLL 用ワークエリア
c      work1_member          real*8    DLL 用ワークエリア
c      work2_member          real*8    DLL 用ワークエリア
c-------------------------------------------------------------
       do i=1,n_member
c      write(76,'(a,2i4)') ' memb : ',i,Member(i).element_type
c      kk=10
c      write(76,'(a,i4,3f12.3)') 'non c',kk,Member(kk).stress(7),
c     *  Member(kk).stress(8)
c-------------------------------------★部材両端の変位取得
       do j=1,12
       ires=Member(i).irest(j)
       if(ires.gt.0) then
c      write(76,*) j,'  =  ',ires
       v(j)=disp_point(ires)
       vp(j)=past_disp_point(ires)
       else
       v(j)=0.
       vp(j)=0.
       endif
       enddo
c      write(76,'(a,i3,12e12.5)') 'vv',i,(v(j),j=1,12)
c      write(76,'(a,i3,12e12.5)') 'vp',i,(vp(j),j=1,12)
c-------------------------------------★部材両端の節点力のゼロセット
       do j=1,12
       f(j)=0.
       enddo
c-------------------------------------★変位を釣合系から部材座標系に変換

       call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
       call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c        write(76,'(a,i3,12e12.5)') 'vv',i,(vv(j),j=1,12)
c        write(76,'(a,i3,12e12.5)') 'vp',i,(vpp(j),j=1,12)
c-------------------------------------★要素及びモデルのセット
       mem = i
```

```
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie   = Member(i).nm_element
      im   = Element(ie).n_element
      ien  = Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0)  goto 9999   ! DLL 要素
c      write(76,'(a,i4)') ' check member:',i
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
c-------------------------------★Model_No.1 通常の有限要素弾塑性モデル
      call Cal_check_unb_M1(ak_nonlinear(1,1,i),v,ff)
      goto 101
   12 continue
c-------------------------------★Model_No.2 3 次元せん断弾塑性モデル
      call Cal_check_unb_Mx(ak_nonlinear(1,1,i),v,ff)
c     do j=1,3
c     f(j)=-Member(i).stress(j)
c     f(j+6)=-f(j)
c     enddo
c     call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
   13 continue
c-------------------------------★Model_No.3 3 次元軸力弾塑性モデル
      call Cal_check_unb_M3(ak_nonlinear(1,1,i),v,ff)
      goto 101
   14 continue
c-------------------------------★Model_No.4 3 次元ケーブル弾塑性モデル
      call Cal_check_unb_Mx(ak_nonlinear(1,1,i),v,ff)
c     f(1)=-Member(i).stress(1)
c     f(7)=-f(1)
c      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
   15 continue
c-------------------------------★Model_No.5 3 次元免振モデル
      call Cal_check_unb_Mx(ak_nonlinear(1,1,i),v,ff)
c     do j=1,3
c     f(j)= -Member(i).stress(j)
c     f(j+6)=-f(j)
c     enddo
c      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
   16 continue
c-------------------------------★Model_No.6 3 次元制震 Maxwell モデル(ok)
c      call  Check_maxwelldamp(E_model6_real(ien),Element(ie))

      goto 100
   17 continue
c-------------------------------★Model_No.7 3 次元プレテンション動作モデル

c      call Cal_check_stiff_M7(Member(i),Element(ie),
c     *     E_model1_int(im),E_model1_real(im),
c     *     M_model1_int(imm),M_model1_int(imm),
c     *     vv,ak )
```

```
c      do j=1,12
c      f(j)=0.
c      enddo
c      do j=1,3
c      f(j)=-Member(i).stress(j)
c      f(j+6)=-f(j)
c      enddo
c       call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
c       do j=1,12
c      Member(i).force(j)=ff(j)
c       enddo
       goto 100
   18 continue
c-------------------------------★Model_No.8

       goto 100
   19 continue
c-------------------------------★Model_No.9

       goto 100
   20 continue
c-------------------------------★Model_No.10

       goto 100


       elseif(iett.eq.1)then
c        write(76,'(a,i3,12e12.5)') 'iet',iet
       goto(111,112,113,114,115,116,117,118,119,120),iet-10
  111 continue
c-------------------------------★Model_No.11 両端ファイバーモデル
       call Cal_check_unb_M11(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model11, E_model_fiber,
     *      M_model11, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
       call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
       goto 101
  112 continue
c-------------------------------★Model_No.12 両端、中央ファイバーモデル
       call Cal_check_unb_M12(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model12, E_model_fiber,
     *      M_model12, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
       call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
       goto 101
  113 continue
c-------------------------------★Model_No.13 両端 MS モデル
       call Cal_check_unb_M21(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model21, E_model_fiber,
     *      M_model21, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
```

```
c――――――――――――――――――――――――――★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  114 continue
c――――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
       call Cal_check_unb_M22(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model22, E_model_fiber,
     *      M_model22, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
c――――――――――――――――――――――――――★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  115 continue
c――――――――――――――――――――――――――――――――――★Model_No.15 幾何学非線形+弾塑性型有限要素モデ
      call Cal_check_unb_M15(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model15, E_model_fiber,
     *      M_model15, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
c――――――――――――――――――――――――――★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  116 continue
c――――――――――――――――――――――――――――――――――★Model_No.16 両端アナロジーモデル
      call Cal_check_unb_M31(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model31, E_model_fiber,
     *      M_model31, M_model_fiber,
     *      vv,vpp,f)
c――――――――――――――――――――――――――★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  117 continue
c――――――――――――――――――――――――――――――――――★Model_No.17 両端、中央アナロジーモデル
      call Cal_check_unb_M32(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model32, E_model_fiber,
     *      M_model32, M_model_fiber,
     *      vv,vpp,f)
c――――――――――――――――――――――――――★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  118 continue
c――――――――――――――――――――――――――――――――――★Model_No.18 両端ピン、中央ファイバーモデル
      call Cal_check_unb_M13(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model13, E_model_fiber,
     *      M_model13, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f)
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  119 continue
c――――――――――――――――――――――――――――――――――★Model_No.19 両端ピン、中央アナロジーモデル
```

```
      call Cal_check_unb_M33(N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_model33, E_model_fiber,
     *      M_model33, M_model_fiber,
     *      vv,vpp,f)
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 101
  120 continue
c————————————————————————————★Model_No.20

      goto 100

      elseif(iett.eq.5.or. iett.eq.6)then
      call Cal_check_unb_Mxx(Control,N_analysis,
     *      mem,Model_type,Member(i),Element(ie),
     *      E_modelx,E_model_fiber,
     *      M_modelx,M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vpp,f,
     *      S_comp_model,E_fiber_work,M_fiber_work)
c————————————————————————★部材の両端節点力を釣合系に変換
      call RotateL_v(2,f,rot_memb(1,1,1,i),rot_memb(1,1,2,i),ff)
      goto 100

      endif
      goto 100
 9999 continue

c————————————————————————————★Model_No.DLL
c      call Cal_check_stiff_dll(mem,Member(i),Element(ie),
c     *   work1_elememt,work2_elememt,work1_member,work2_member,
c     *     vv,ak)
      goto 100
  101 continue
c————————————————————————————★不釣合力の足しこみ
c     write(76,'(a,i4,12f12.4)') ' ff',i,(ff(j),j=1,12)
      call get_pointforce_ldx(ff,ld_point,Member(i))

c————————————————————————————★部材の接線剛性を釣合系に変換
  100 continue

      end do
      return
      end
```

上のサブルーチンで使用する新たなサブルーチンを以下のように設計する。

```
C     ————————————————————————————————————
C     ●   SUBROUTINE /Cal_check_unb_Mxx
C     ————————————————————————————————————
C     ●    代表的な部材モデルの不釣合い力チェック(ok)
C     ————————————————————————————————————
```

```
      subroutine Cal_check_unb_Mxx(N_analysis,
     *      mem_x,Model_type,Member,Element,
     *      E_modelx, E_model_fiber,
     *      M_modelx, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work,vv,vp,f_p,
     *      S_comp_model,E_fiber_work,M_fiber_work)
c-----------------------------------------------------------------
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
c-------------------------------★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work(*)
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension ak(12,12),f_p(12)
      dimension vv(12),vp(12),vvx(12)
      dimension Bilinear_work(*),Trilinear_work(*)
      dimension Concrete_work(*)

      real*8, ALLOCATABLE :: c(:,:),ab(:,:),alength(:),EA(:)
      real*8, ALLOCATABLE :: bav(:),akk(:,:,:),f1(:),f2(:)
      integer, ALLOCATABLE :: irest_Point(:,:),n_type(:)
c-----------------------------------------------------------------
C     ---------------------------------------
      iet = Member.n_model                        ! モデルタイプ番号
      ix_model= iet-50                            ! 任意要素モデル(51-69)
      nmmx= Member.n_model_type -1               ! M_Fiber_work の開始番号
      nmx = Element.n_section(1) -1              ! E_Fiber_work の開始番号
      n_div= S_comp_model(ix_model).n_div_element  ! 部材分割数
      n_if = 6*(n_div-1)                           ! 内部自由度
      write(76,'(a,4i6)') ' model:',iet,n_div,imm,immm
c-----------------------------------------------------------------
c
c    ★        部材の剛性行列の設定
c
c-----------------------------------------------------------------
c-------------------------------★動的記憶領域の確保
      ALLOCATE (
     *   irest_Point(6,n_div+1),n_type(n_div),alength(n_div),
     *    akk(12,12,n_div)
     *         )
c-------------------------------★節点拘束表の作成
```

```
c                                                    未知数等をセット
      call set_modelx_dat(irest_Point,n_if,n_div,iubw,
     *         Element,Member,S_comp_model(ix_model),
     *         n_type,alength,
     *  Member.i_rigid_length,    ! i端剛域
     *  Member.j_rigid_length)    ! j端剛域
c-------------------------------★動的記憶領域の確保
      ALLOCATE (
     *   c(0:iubw,n_if),ab(n_if,12),bav(n_if),f1(n_if),f2(n_if)
     *         )
c-------------------------------★剛性行列のゼロクリア
      do i=1,12
      do j=1,12
      ak(j,i)=0.
      enddo
      enddo
      do i=1,n_if
      do j=0,iubw
      c(j,i)=0.
      enddo
      enddo
      do i=1,12
      do j=1,n_if
      ab(j,i)=0.
      enddo
      enddo
      do i=1,12
      f_p(i)=0.
      enddo
      do i=1,n_if
      f1(i)=0.
      enddo
      EAx=Element.A*Element.E
c-------------------------------★部材剛性行列の作成
      do i=1,n_div
      imm = E_Fiber_work(nmx+i).nm_section            ! エレメント番号
      immm= M_Fiber_work(nmmx+i).nm_section           ! 内部エレメント番号
      EAA=EAx
      if(n_type(i).eq.2) then       ! ファイバー：他のモデルでも必要なときはここに加える
      EAA=M_modelx(immm).d_ra
      endif
c-------------------------★内部部材の剛性行列の計算
      call Stiff_Mx(i,n_type(i),akk(1,1,i),Member,alength,
     *       Model_type,Element,
     *       E_modelx(imm), E_model_fiber,
     *       M_modelx(immm), M_model_fiber)
      if(N_analysis.ne.9.and.N_analysis.ne.7.and.n_type(i).ne.3) then    !  幾何学的非線形剛性
      call Cal_geomet_stiffx(M_Fiber_Work(nmmx+i).an_stress,Member,
     *                       akk(1,1,i),alength(i))
      call Create_Kn(akk(1,1,i), M_Fiber_Work(nmmx+i).an_vv,
     *                                M_Fiber_Work(nmmx+i).an_ww,
     *                                EAA,alength(i))
      endif
c-------------------------★剛性行列の分配
```

```
      call Bnd_FEM(i,akk(1,1,i),irest_Point,ak,c,ab,iubw,n_if)
      enddo
c――――――――――――――――――――――――――――――――――――――――
c
c    ★        部材内部の変位と不釣合力の計算
c
c――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――★両端変位を剛域内部の変位に変換処理
      call Deal_Rigid_element_v(vv,Member.i_rigid_length,
     *                          Member.j_rigid_length)
c――――――――――――――――――――――――★部材内部変位の計算(c 行列の分解計算)
      do i=0,n_if-1
      k=i/6
      j = i -(k)*6
      f2(i+1) = M_Fiber_work(nmmx+k+1).ff_ip(j+1)                              ! 1
      enddo
      call Typical_member_v(c,ab,f2,
     *                      n_if,n_if,iubw,iubw,ier,vv,bav)
c――――――――――――――――――――――――★部材内部、両端節点力の計算
      do i=1,n_div
      call Typical_member_p_force(akk(1,1,i),irest_Point(1,i),
     *                            vv,bav ,f_p,f1)
      enddo
c――――――――――――――――――――――――★部材内部、両端節点力から不釣合力へ
      do i=0,n_if-1
      k=i/6
      j = i -(k)*6
      M_Fiber_work(nmmx+k+1).ff_ip(j+1) = f1(i+1)                              ! 2
      enddo
c――――――――――――――――――――――――★部材両端節点力への縮合
      call Typical_member_f(c,ab,f1,f_p,n_if,n_if,iubw,iubw)     ! f1 はデータが変更される
c――――――――――――――――――――――――★部材節点力の両端剛域処理
      call Deal_Rigid_element_f(f_p,Member.i_rigid_length,
     *                          Member.j_rigid_length)
c――――――――――――――――――――――――――――――――★動的記憶領域の解放
      DEALLOCATE (
     *   c ,ab ,irest_point,n_type,alength,
     *   bav,akk,f1,f2   )
      return
      end
```

9.5 剛床モデル

9.5.1 はじめに

SPACE（Ver.3.00）では、立体骨組の静的解析や動的解析を行っている。その際、通常の立体骨組構造では、床の扱いをどのようにするかが問題となる。以前のバージョンでは、床をブレースに置換して解析を行っていた。ここでは、立体骨組構造で良く用いられる剛床仮定を SPACE に組み込む手法について理論的に検討する。さらに、この節以降では、実際のプログラムを用いて剛床仮定を SPACE に組み込む方法を示すことにする。剛床の変換行列は、局所座標変換とは異なり、軸力と曲げモーメントが連成するため、釣合座標系を得るため全ての座標変換を変更する必要があり、多くのサブルーチンに影響する。そのため剛床仮定の機能を組み込むために、SPACE を大幅に変更することになる。ここでは、これらを全て検討しよう。

剛床仮定を SPACE に組み込むためには、以下の部分に変更を加えることになる。

１．入力仕様
２．座標変換式の保存
３．座標変換式、特に局所座標系との関係
４．剛性行列の釣合座標系への変換
５．質量行列の釣合座標系への変換
６．右辺項の釣合座標系への変換
７．釣合座標系から部材座標系に変換
８．釣合座標系から全体座標系に変換

9.5.2 座標変換

立体骨組の中で、床の剛性が他の部分に比較して著しく硬いとき、この床を剛床として扱うことができる。本節では、剛床モデルの理論的な処理手法について述べる。

ここでは、床は水平レベルにあるものとする。また、この床は剛であり、そのため、その床内にある代表点で床内の任意の節点の床面内の変位を表すことができるものとする。ただし、面外の変位に対しては、床内の節点は独自に挙動するものとする。

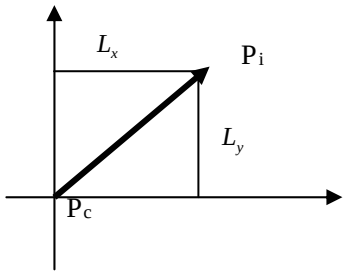
まず、代表節点と他の節点の関係を検討する。剛床モデルとして、代表節点と他の節点が長さ L の剛部材で連結しているとする。ここで、部材の代表節点側を c 端、他の節点を i 端とし、この剛部材の x 方向長さを L_x、y 方向長さを L_y とする。その剛部材をはさんで、全体座標系で表された両端の増分変位ベクトルを以下のように表す。

$$\left.\begin{aligned}\{\Delta u_i\}^T &= \{\Delta u_i, \Delta v_i, \Delta w_i, \Delta\theta_{xi}, \Delta\theta_{yi}, \Delta\theta_{zi}\} \\ \{\Delta u_c\}^T &= \{\Delta u_c, \Delta v_c, \Delta w_c, \Delta\theta_{xc}, \Delta\theta_{y_c}, \Delta\theta_{zc}\}\end{aligned}\right\} \cdots\cdots\cdots\cdots (9.1)$$

ここで、$\{\Delta u_i\}$は、任意節点側の増分変位であり、$\{\Delta u_c\}$は代表節点 c 側の増分変位である。また、同様に、全体座標系で表された両端の増分節点力ベクトルを次式で示す。

$$\left.\begin{aligned}\{\Delta f_i\}^T &= \{\Delta N_i, \Delta Q_{yi}, \Delta Q_{zi}, \Delta M_{xi}, \Delta M_{yi}, \Delta M_{zi}\} \\ \{\Delta f_c\}^T &= \{\Delta N_c, \Delta Q_{yc}, \Delta Q_{zc}, \Delta M_{xc}, \Delta M_{yc}, \Delta M_{zc}\}\end{aligned}\right\} \cdots\cdots\cdots (9.2)$$

剛床上の節点変位は、その床上の代表節点の変位で表すことになる。剛床上の他の節点変位は、この代表点における 2 方向の変位と、代表点の z 軸回りの回転による x 方向と y 方向の変位を考慮する。



$$\begin{Bmatrix}\Delta u_i \\ \Delta v_i \\ \Delta\theta_{zi}\end{Bmatrix} = \begin{bmatrix}1 & 0 & -L_y \\ 0 & 1 & L_x \\ 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}\Delta u_c \\ \Delta v_c \\ \Delta\theta_{zc}\end{Bmatrix}$$

$$\{\Delta u_i\} = [R_u]\{\Delta u_c\} \qquad \cdots\cdots\cdots (9.3)$$

ここで、両者共に全体座標系で表された変位とする。また、上式の変位変換式では、z 軸の回転に関する幾何学的非線形性を考慮していない。

次に、節点力間の関係を示す。

$$\begin{Bmatrix}\Delta P_{xi} \\ \Delta P_{yi} \\ \Delta M_{zi}\end{Bmatrix} = \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ L_y & -L_x & 1\end{bmatrix}\begin{Bmatrix}\Delta P_{xc} \\ \Delta P_{yc} \\ \Delta M_{zc}\end{Bmatrix}$$

$$\{\Delta P_i\} = [R_f]\{\Delta P_c\} \qquad \cdots\cdots\cdots (9.4)$$

ここでも、両者の節点力は全体座標系で表されているものとする。上記した 2 つの変換行列には、次のような特異な関係が存在する。

$$[R_u][R_f]^T = [I]$$

$$\begin{bmatrix}1 & 0 & -L_y \\ 0 & 1 & L_x \\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & L_y \\ 0 & 1 & -L_x \\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix} \qquad \cdots\cdots\cdots (9.5)$$

したがって、

$$[R_u]^{-1} = [R_f]^T;\quad [R_f]^{-1} = [R_u]^T \qquad \cdots\cdots\cdots (9.6)$$

という関係が存在し、これらの関係は後で利用することになる。

剛性行列を全体座標系から釣合座標系に変換してみよう。ただし、ここでは、局所座標系は使用しないものとする。次式は、全体座標系で表された釣合式である。

$$\{\Delta P_i\} = [k]\{\Delta u_i\} \qquad \cdots\cdots\cdots(9.7)$$

式(9.3)と(9.4)を利用して、釣合座標系に変換する。

$$\left[\bar{R}_f\right]\{\Delta P_c\} = [k]\left[\bar{R}_u\right]\{\Delta u_c\} \qquad \cdots\cdots\cdots(9.8)$$

上式の左より、$\left[\bar{R}_f\right]^{-1}$を掛け、式(9.6)を用いると、

$$\{\Delta P_c\} = \left[\bar{R}_u\right]^T [k]\left[\bar{R}_u\right]\{\Delta u_c\} \qquad \cdots\cdots\cdots(9.9)$$

となり、釣合座標系における剛性行列が次のように得られる。

剛性行列の変換式

$$\left[\bar{k}\right] = \left[\bar{R}_u\right]^T [k]\left[\bar{R}_u\right]$$
$$\left[\bar{R}_u\right] = \begin{bmatrix} R_u & 0 \\ 0 & I \end{bmatrix} \qquad \cdots\cdots\cdots(9.10)$$

上式で、i 端は剛床位置にあり、 j 端は一般の節点としている。両者共に剛床位置である場合は、式(9.10)中の単位行列の替わりに剛床変換行列を用いることになる。ここで、$[R_u]$は以下のようである。

$$[R_u] = \begin{bmatrix} 1 & & & & & & -L_y \\ & 1 & & & & & L_x \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \end{bmatrix} \qquad \cdots\cdots\cdots(9.11)$$

上式のL_xとL_yは、剛床代表点から当該節点までの x 方向、y 方向の長さを表す。

質量行列も、剛性行列と同様に全体座標系から釣合座標系に変換可能である。ここでも局所座標系は使用しないものとする。釣合座標系で表される質量行列は、次式で表される。

質量行列の変換式

$$\left[\bar{m}\right] = \left[\bar{R}_u\right]^T [m]\left[\bar{R}_u\right]$$
$$\left[\bar{R}_u\right] = \begin{bmatrix} R_u & 0 \\ 0 & I \end{bmatrix} \qquad \cdots\cdots\cdots(9.12)$$

右辺項ベクトルを釣合座標系に変換しよう。動的解析の右辺項は、ニューマークβ法を適用すると次式となる。

右辺項の変換式

$$\{f\}=\{G\}+\{g\} \quad \cdots\cdots\cdots(9.13)$$

$$\{G(y_{n+1},\Delta y_{n+1})\}=-\left\{\bar{f}_d\right\}-\left[K_T(y_n)\right]\{\Delta y_{n+1}\}+\left[K\right]\{y_{n+1}\} \quad \cdots\cdots\cdots(9.14)$$

$$\{g\}=-\left[M\right]\left[I\right]\left\{\ddot{u}_g\right\}+\{P_S\}-\{Q(y_n)\}-\left[\bar{C}\right]\{a\}-\left[K\right]\{\bar{b}\} \quad \cdots\cdots\cdots(9.15)$$

式(9.13)に示されるように、加速度は釣合座標系であるため、式(9.14)の線形剛性行列と接線剛性行列は釣合座標系に変換しておく必要がある。次に、式(9.15)の第１と２項は、各節点において全体座標系で求めた後、釣合座標系に変換する。したがって、ここでの質量行列は、全体座標系である必要がある。

第３項は、部材座標系の節点力から釣合座標系の節点力ベクトルに変換する。第４と５項は、$\{a\}$と$\{b\}$は釣合座標系で求められているため、減衰行列$[C]$と接線剛性行列$[K_T]$は、釣合座標系で求めておく必要がある。

9.5.3 入力仕様

本節では、剛床仮定に対する入力仕様を決めるが、現在の仕様を大きく変更することはできない。そこで、節点に関するデータ入力を次のように定義する。

節点データの変更点は、以下のようである。

１．節点の３次元座標の後のデータ入力項目を利用する。
２．局所座標系を使用する。ここで、剛床用の変換行列を局所座標変換行列と同じ領域にセットする。ただし、変換行列は、局所座標変換行列と異なるので変換サブルーチンは異なる。
３．節点の同一視を利用して、剛床の代表点の変位を参照する。

以下に、節点に関する入力仕様を示し、変更部分を提示する。

１）節点座標

この入力項目で剛床仮定用のデータ設定を行う

節点の３次元座標をセットする。節点数 NODE だけ繰り返す。

I	POSIT(1,I)	POSIT(2,I)	POSIT(3,I)	NDOUT(I)

I：節点番号
POSIT(1,I)：節点の X 座標（cm）
POSIT(2,I)：節点の Y 座標（cm）

POSIT(3,I)：節点の Z 座標（cm）

NDOUT(I)：通常節点：0

剛床用節点：0 以外の剛床グループ番号

剛床番号：0 は自由節点

正番号　剛床グループ番号

負番号　その絶対値番号の剛床グループの代表節点

この剛床の代表節点が設定されていない場合は、最も小さな節点番号の節点を代表節点とする。また、グループ内に 2 節点以上ない場合は、剛床処理は行わない。

２）局所座標系

節点に局所座標系を考慮する場合、その節点について、全体座標系から局所座標軸への回転角をセットする。このレコードは局所座標系を考慮する節点数(LOCOD*3) だけ繰り返す。**LOCOD=0 の場合は以下のデータを設定してはならない。**

剛床上の節点は、局所座標系を用いてはならない

I	TLCX(I)	TLCY(I)	TLCZ(I)

I：局所座標系を考慮する節点番号

TLCX(I)：X 軸回りの回転角 θx（度）

TLCY(I)：Y 軸回りの回転角 θy（度）

TLCZ(I)：Z 軸回りの回転角 θz（度）

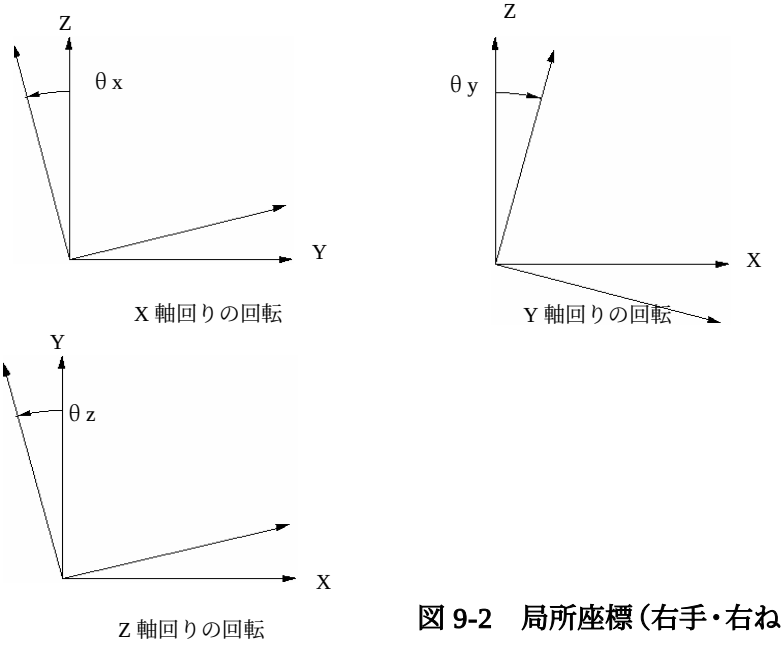


図 9-2　局所座標（右手・右ねじの法則）

３）節点の拘束指標

節点の拘束条件の指標を入力し、拘束節点数(NRBOUND) だけ繰り返す。

I	IRE(1,I)	IRE(2,I)	IRE(3,I)	IRE(4,I)	IRE(5,I)	IRE(6,I)

I：拘束する節点番号

IRE(1,I)：X 方向の変位の拘束指標

IRE(2,I)：Y 方向の変位の拘束指標

IRE(3,I)：Z 方向の変位の拘束指標

IRE(4,I)：X 軸回りの回転の拘束指標

IRE(5,I)：Y 軸回りの回転の拘束指標

IRE(6,I)：Z 軸回りの回転の拘束指標

節点の拘束指標
０：自由
－１：固定
－(10×K+L)：
他節点の変位と同一視

拘束指標は、次のようである。

IRE(*,I)=　0：自由
IRE(*,I)=－1：固定

他節点の自由度の変位と同一視する場合は、次のように拘束指標をセットする。

IRE(*,I)=－(10×K+L)：節点 K の自由度 L と同変位

ここで、方向 L とは、

=1：X 方向、=2：Y 方向、=3：Z 方向

=4：X 軸回りの回転、=5：Y 軸回りの回転、=6：Z 軸回りの回転

剛床仮定を用いる節点は、システムが自動的に、x 方向変位、Y 方向変位、z 軸回りの回転に、他節点との同一視を行う拘束指標を設定する

9.5.4 他節点の変位と同一視を利用

剛床処理を行う場合、他節点の変位との同一視処理を利用する。従って、剛床が存在する場合は、他節点の変位との同一視を行うために、システム内で、当該節点の拘束条件に変更を加える。まず、剛床上の節点で、全体座標系での変位は、

$$\{u\}=\{u \quad v \quad w \quad \theta_x \quad \theta_y \quad \theta_z\}$$

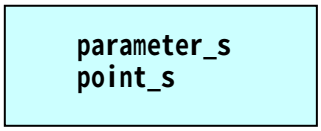
で表され。第 1、2、6 項目の変位を、代表点の変位の対応する変位と同一視する。これをシステム内のプログラムで変更する。例えば、代表節点が 10 で、剛床節点が 20 とすると、

```
20  -101   -102    0       0    0    -106
```

となる。

9.5.5 構造体の拡張

剛床処理を行うため、次のように２つの構造体の仕様拡張を行う。この構造体は、



であり、次のように変更される。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   parameter_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     解析パラメータ
      structure  / parameter_s/
      integer    n_unknown       !  全自由度
      integer    n_point         !  節点数
      integer    n_element       !  要素数
      integer    n_element_dll   !  DLL 用要素数
      integer    n_S_comp_model  !  任意型静的縮合モデル数
      integer    nE_New_Element  !  任意型静的縮合モデルに含まれる要素エレメント数
      integer    nM_New_Element  !  任意型静的縮合モデルに含まれる部材エレメント数
      integer    n_member        !  部材数
      integer    n_rot_axis      !  主軸回転部材数
      integer    n_local_coord   !  局所座標系を使用する節点数
      integer    n_boundary_p    !  境界節点数
      integer    nc_member       !  部材減衰機構を有する部材数
      integer    n_member_dll    !  DLL 用部材数
      integer    n_free          !  節点当たり解析自由度数
      integer    n_dim           !  解析次元数
      integer    n_skyline       !  スカイライン行列の領域数
      integer    n_sky_ave       !  平均バンド幅
      integer    n_gouyuka       !  剛床の有無
      end structure
c      record /parameter_s/ Parameter_C
c
C     ―――――――――――――――――――――――――――――――――
C     ●   point_s 構造体
C     ―――――――――――――――――――――――――――――――――
c
c     節点
      structure  / point_s/
      real*8   coord(3)          ! 3 次元の節点座標
      integer  irest(6)          ! 節点自由度の拘束表
      integer  local_ coord      ! 局所座標系の有無（ある場合は、回転行列の通し番号）
      integer  n_group_gouyuka   ! 剛床のグループ番号
      real*8   coord_local(3)    ! 局所座標系（全体座標系に対する角度）
      real*8   disp_initial(3)   ! 初期変位
      real*8   mass_1            ! 第一ステップ質量（ダミー）
      real*8   mass_2            ! 第二ステップ質量（ダミー）
      end structure
c     record /point_s/ Point
```

構造体要素の新規挿入

この２つの要素の仕様を拡張する

```
c     ALLOCATABLE ::Point(:)
c     ALLOCATE (Point(n_point))
c
```

上記２つの構造体要素の仕様を次のように拡張する。

```
integer   local_ coord      ! 局所座標系の有無：剛床節点の有無
      剛床節点の場合は-1 をセットする。
real*8    coord_local(3)    ! 剛床節点の場合、次の座標をセットする。
        coord_local(1)：Lx
        coord_local(2)：Ly
intger   n_group_gouyuka：剛床グループ番号
```

9.5.6 剛床データの設定

仕様に従って、剛床データが入力されるとき、まず、その入力データから、以下のサブルーチンを用いて解析に必要となる情報を構造体に設定する。最初に、構造データを入力するサブルーチンを以下のように変更する。

```
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,
     *           S_comp_model,E_Fiber_work)
       .
       .
c――――――――――――――――――――――――――――――――★節点データ入力
      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
      write(damp_out,'(i4,3f12.3,i4)')ii,x,y,z,idm  ! ii:節点番号 idm:剛床グループ番号
      Point(ii).coord(1)  = x
      Point(ii).coord(2)  = y
      Point(ii).coord(3)  = z
      Point(ii).n_group_gouyuka = idm
      end do
```

次のサブルーチンでは、剛床処理に必要となる情報を構造体に設定する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_gouyuka
C     ―――――――――――――――――――――――――――――――――
C     ●     剛床データの設定
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_gouyuka(Parameter_C,Point )
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s  / Parameter_C
      record / point_s      / Point
      dimension Point(*)
     integer, ALLOCATABLE :: nx_yuka(:)
      node= Parameter_C.n_point
c――――――――――――――――――――――――★剛床の有無と剛床グループ番号の最大値チェック
      Parameter_C.n_gouyuka = 0
      n_g=0
      do i=1,node
      if(Point(i).n_group_gouyuka .ne. 0) then
      if(n_g.le.iabs(Point(i).n_group_gouyuka))
     * n_g = iabs(Point(i).n_group_gouyuka)
      endif
      enddo
      if(n_g.eq.0) return
      Parameter_C.n_gouyuka = 1
c――――――――――――――――――――――――★剛床の代表点検索
      ALLOCATE (nx_yuka(n_g))
      do i=1,n_g
      nx_yuka(i)=0
      enddo
      do i=1,node
       if(Point(i).n_group_gouyuka .ne. 0) then
        n= Point(i).n_group_gouyuka
        if(n.gt.0 ) then
          if(nx_yuka(n) .eq. 0) nx_yuka(n)=-1
        else
          nx_yuka(-n)=I
          Point(j). n_group_gouyuka= 0
        endif
      endif
      enddo
c――――――――――――――――――――――――★
      do i=1,n_g
      if(nx_yuka(i) .eq. – 1) then
       do j=1,node
        if(Point(j).n_group_gouyuka .eq. i) then
          nx_yuka(i)=j
          Point(j). n_group_gouyuka= 0
          goto 100
        endif
       enddo
       endif
100   continue
       enddo
```

```
c――――――――――――――――――――――★
      do i=1,n_g
      nx= nx_yuka(i)
      if(nx.gt.0) then
       do j=1,node
       if(i.eq. Point(j).n_group_gouyuka) then
         Point(j).coord_local(1) = Point(nx).coord(2) - Point(j).coord(2)
         Point(j).coord_local(2) = Point(j).coord(1)  - Point(nx).coord(1)
         nxx = -nx*10
          Point(i).irest(1)= nxx -1
          Point(i).irest(2)= nxx -2
          Point(i).irest(6)= nxx -6
       endif
       enddo
      endif
      enddo
      DEALLOCATE (nx_yuka)
      return
      end
```

上のコードで、Ly **は最初から負符号を付けてセットしているので注意されたい**。このため、理論式とコードの符号が異なることになる。

9.5.7 ベクトルの変換

節点における変位と節点力を変換するサブルーチンを示す。まず、釣合座標系の変位を全体座標系に変換するサブルーチンは、次式

$$u_i = u_c - L_y\theta_{zc};\quad u_c = u_i + L_y\theta_{zi}$$
$$v_i = v_c + L_x\theta_{zc};\quad v_c = v_i - L_x\theta_{zi}$$

を用いる。

```
C     ――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /Set_gtrans_u
C     ――――――――――――――――――――――――――――――――――
C     ●     変位ベクトルの剛床座標変換
C     ――――――――――――――――――――――――――――――――――
      subroutine Set_gtrans_u(ik,u,aly,alx )
C     ――――――――――――――――――――――――――――――――――
C     ik=1 釣合座標系から全体座標系に変換
C     ――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      dimension u(6)
      if(ik.eq.1) then
      u(1)=u(1) + aly*u(6)
      u(2)=u(2) + alx*u(6)
      else
      u(1)=u(1) - aly*u(6)
      u(2)=u(2) - alx*u(6)
```

```
      endif
      return
      end
```

次に、節点力を変換するサブルーチンを以下に示す。全体座標系の節点力から釣合座標系に変換するサブルーチンは、次式

$$M_{cz} = M_z - L_y P_{xi} + L_x P_{yi};\quad M_{ci} = M_c + L_y P_{xc} - L_x P_{yc}$$

を用いる。

```
C     ―――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /Set_gtrans_f
C     ―――――――――――――――――――――――――――――――――
C     ●     力ベクトルの剛床座標変換
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_gtrans_f(ik,f,aly,alx )
C     ―――――――――――――――――――――――――――――――――
C     ik=1 釣合座標系から全体座標系に変換
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      dimension f(6)
      if(ik.eq.1) then
      f(6)=f(6) - aly*f(1) - alx*f(2)
      else

      f(6)=f(6) + aly*f(1) + alx*f(2)
      endif
      return
      end
```

9.5.8 剛性行列の変換

本節では、剛性剛列などの座標変換について述べる。変換する行列は全体座標系で与えられているものとし、釣合座標系に変換する式の詳細は、マニュアル理論編を参照されたい。変換後の剛性は

$$\left[\bar{k}\right] = [k] + [k_L]$$

$$[k_L] = \begin{bmatrix}
0 & & & k_{1,1}L_{1y} + k_{1,2}L_{1x} & & & & & k_{1,7}L_{2y} + k_{1,8}L_{2x} \\
& 0 & & k_{2,1}L_{1y} + k_{2,2}L_{1x} & & & & & k_{2,7}L_{2y} + k_{2,8}L_{2x} \\
& & 0 & k_{3,1}L_{1y} + k_{3,2}L_{1x} & & & & & k_{3,7}L_{2y} + k_{3,8}L_{2x} \\
& & & 0 & k_{4,1}L_{1y} + k_{4,2}L_{1x} & & & & k_{4,7}L_{2y} + k_{4,8}L_{2x} \\
& & & & 0 & k_{5,1}L_{1y} + k_{5,2}L_{1x} & & & k_{5,7}L_{2y} + k_{5,8}L_{2x} \\
& & & & & k_{6,1}L_{1y} + k_{6,2}L_{1x} & & & k_{6,7}L_{2y} + k_{6,8}L_{2x} \\
& & & & & k_{7,1}L_{1y} + k_{7,2}L_{1x} & 0 & & k_{7,7}L_{2y} + k_{7,8}L_{2x} \\
& & & & & k_{8,1}L_{1y} + k_{8,2}L_{1x} & & 0 & k_{8,7}L_{2y} + k_{8,8}L_{2x} \\
& & & & & k_{9,1}L_{1y} + k_{9,2}L_{1x} & & & 0 \quad k_{9,7}L_{2y} + k_{9,8}L_{2x} \\
& & & & & k_{10,1}L_{1y} + k_{10,2}L_{1x} & & & 0 \quad k_{10,7}L_{2y} + k_{10,8}L_{2x} \\
& & & & & k_{11,1}L_{1y} + k_{11,2}L_{1x} & & & 0 \quad k_{11,7}L_{2y} + k_{11,8}L_{2x} \\
& & & & & k_{12,1}L_{1y} + k_{12,2}L_{1x} & & & k_{12,7}L_{2y} + k_{12,8}L_{2x}
\end{bmatrix} +$$

$$
\begin{bmatrix}
0 & & & & & & & & & & & \\
 & 0 & & & & & & & & & & \\
 & & 0 & & & & & & & & & \\
 & & & 0 & & & & & & & & \\
 & & & & 0 & & & & & & & \\
 & & & & & 0 & & & & & & \\
k_{1,1}L_{1y} & k_{1,2}L_{1y} & k_{1,3}L_{1y} & k_{1,4}L_{1y} & k_{1,5}L_{1y} & k_{1,6}L_{1y} & k_{1,7}L_{1y} & k_{1,8}L_{1y} & k_{1,9}L_{1y} & k_{1,10}L_{1y} & k_{1,11}L_{1y} & k_{1,12}L_{1y} \\
+k_{2,1}L_{1x} & +k_{2,2}L_{1x} & +k_{2,3}L_{1x} & +k_{2,4}L_{1x} & +k_{2,5}L_{1x} & +k_{2,6}L_{1x} & +k_{2,7}L_{1x} & +k_{2,8}L_{1x} & +k_{2,9}L_{1x} & +k_{2,10}L_{1x} & +k_{2,11}L_{1x} & +k_{2,12}L_{1x} \\
 & & & & & & 0 & & & & & \\
 & & & & & & & 0 & & & & \\
 & & & & & & & & 0 & & & \\
 & & & & & & & & & 0 & & \\
 & & & & & & & & & & 0 & \\
k_{7,1}L_{2y} & k_{7,2}L_{2y} & k_{7,3}L_{2y} & k_{7,4}L_{2y} & k_{7,5}L_{2y} & k_{7,6}L_{1y} + & k_{7,7}L_{2y} & k_{7,8}L_{2y} & k_{7,9}L_{2y} & k_{7,10}L_{2y} & k_{7,11}L_{2y} & k_{7,12}L_{2y} \\
+k_{8,1}L_{2x} & +k_{8,2}L_{2x} & +k_{8,3}L_{2x} & +k_{8,4}L_{2x} & +k_{8,5}L_{2x} & k_{8,6}L_{1x} & +k_{8,7}L_{2x} & +k_{8,8}L_{2x} & +k_{8,9}L_{2x} & +k_{8,10}L_{2x} & +k_{8,11}L_{2x} & +k_{8,12}L_{2x}
\end{bmatrix} +
$$

$$
\begin{bmatrix}
0 & & & & & 0 & & & & & 0 & \\
 & 0 & & & & 0 & & & & & 0 & \\
 & & 0 & & & 0 & & & & & 0 & \\
 & & & 0 & & 0 & & & & & 0 & \\
 & & & & 0 & 0 & & & & & 0 & \\
 & & & & & L_{1y}^2k_{1,1} + 2L_{1y}k_{1,2}L_{1x} + L_{1x}^2k_{2,2} & & & & & L_{1y}k_{1,7}L_{2y} + L_{1y}k_{2,7}L_{2x} + L_{1x}k_{1,8}L_{2y} + L_{1x}k_{2,8}L_{2x} & \\
 & & & & & 0 & 0 & & & & 0 & \\
 & & & & & 0 & & 0 & & & 0 & \\
 & & & & & 0 & & & 0 & & 0 & \\
 & & & & & 0 & & & & 0 & 0 & \\
 & & & & & 0 & & & & & 0 & \\
 & & & & & L_{2y}k_{7,1}L_{1y} + L_{2y}k_{8,1}L_{1x} + L_{2x}k_{7,2}L_{1y} + L_{2x}k_{8,2}L_{1x} & & & & & L_{2y}^2k_{7,7} + 2L_{2y}k_{7,8}L_{2x} + L_{2x}^2k_{8,8} &
\end{bmatrix}
$$

で与えられる。

剛性行列を変換するサブルーチンを以下に示す。

```
C     ------------------------------------------------------------
C     ●    SUBROUTINE /Set_gtrans_k
C     ------------------------------------------------------------
C     ●     剛床データの設定
C     ------------------------------------------------------------
      subroutine Set_gtrans_k(ak,aly1,alx1, aly2,alx2 )
C     ------------------------------------------------------------
      implicit real*8(A-H,O-Z)
      dimension ak(12,12),u1(12),u2(12),v(3)
      do j=1,12
      u1(j)=aly1*ak(j,1)+alx1*ak(j,2)
      u2(j)=aly2*ak(j,7)+alx2*ak(j,8)
      enddo
      v(1)=aly1*u1(1)+alx1*u1(2)
      v(2)=aly1*u2(1)+alx1*u2(2)
      v(3)=aly2*u2(7)+alx2*u2(8)
      do j=1,12
      ak(6,j)=ak(6,j)+u1(j)
```

```
      ak(12,j)=ak(12,j)+u2(j)
      ak(j,6)=ak(j,6)+u1(j)
      ak(j,12)=ak(j,12)+u2(j)
      enddo
      ak(6,6)=ak(6,6)+v(1)
      ak(6,12)=ak(6,12)+v(2)
      ak(12,6)=ak(12,6)+v(2)
      ak(12,12)=ak(12,12)+v(3)
      return
      end
```

9.5.9 質量行列の変換

質量行列の変換は、一般的には、剛性行列と同じで良い。ただし、集中質量系では、全体座標系における質量行列が対角行列であり、しかも、回転慣性項を無視している。この集中質量の変換の詳細は、マニュアル理論編を参照されたい。

最終的に、変換後の質量行列は以下のようである。

$$[\bar{m}]=\begin{bmatrix} m_{11} & & & & & -m_{11}L_y \\ & m_{22} & & & & m_{22}L_x \\ & & m_{33} & & & \\ & & & 0 & & \\ & & & & 0 & \\ -m_{11}L_y & m_{22}L_x & & & & L_y m_{11}L_y + L_x m_{22}L_x \end{bmatrix}$$

このように、剛床を考慮すると、集中質量系であっても、回転慣性項を有することになる。

剛床を考慮する集中質量行列を求めるサブルーチンを以下の示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_gtrans_m
C     ―――――――――――――――――――――――――――――――――
C     ●    剛床データの設定
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_gtrans_m(am1,am,aly1,alx1)
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      dimension am(6,6)
      do I=1,6
      do j=1,6
      am(I,j)=0.
      enddo
      enddo
      am(1,1)=am1
```

```
      am(2,2)=am1
      am(3,3)=am1
      am(1,6)=am1*aly1
      am(2,6)=am1*alx1
      am(6,1)=am(1,6)
      am(6,2)=am(2,6)
      am(6,6)=am1*(aly1*aly1+alx1*alx1)
      return
      end
```

9.5.10 動的解析システムへの組み込み

9.5.10.1 係数行列などの変換

剛床仮定の処理を SPACE の動的解析に組み込むことにする。先に示したように、まず、構造体を拡張する。その後、構造データを入力するサブルーチンを変更する。これについては、既に先の節で説明した。

次に、動的解析の主サブルーチンの中で、構造データを入力した後、剛床に関する情報処理を行うサブルーチンをコールする。

```
c――――――――――――――――――――――――――――――――――――――
c
c     ★          構造・荷重データを入力し、データの設定を行う
c
c――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――★基本構造データを入力(ok)
c     write(damp_out,*) ' Get_structure in'
      nfix=5
      nfi=1
      call infile(nfi,nfix,ierr)
      if(ierr.ne.0) then
       ierr_dat =10
      call err_outf(ierr_dat)
      return
      endif
      call Get_structure(Point,Member,Element,Parameter_C,
     *                   Model_type,ierr,
     *                   S_comp_model,E_Fiber_work)
      close(nfix)
      if(ierr .ne. 0) then
       ierr_dat =iabs(ierr)
c                         err No. 12-19 使用
      call err_outf(ierr_dat)
      return
      endif
c     write(damp_out,*) ' Get_structure ok'
c――――――――――――――――――――――――――――――――――――――
c
c     ★          剛床の設定
c
c――――――――――――――――――――――――――――――――――――――
      call Set_gouyuka(Parameter_C,Point)
```

次に、剛性行列や質量行列に関する座標変換をみてみよう。まず、主サブルーチンの中で、次のように座標変換を行うサブルーチンをコールする。

```
c――――――――――――――――――――――――――――――――★部材の線形剛性計算(ok)
      call Cal_stiff_linear(Model_type,Element,Member,Parameter_C,
     *      ak_linear,E_model11,E_model_fiber,M_model11,M_model_fiber,
     *      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,
     *      Bilinear_work,Trilinear_work,Concrete_work,
     *      work1_elememt,work2_elememt,work1_member,work2_member,
     *      S_comp_model, E_modelx, M_modelx,
     *      E_fiber_work, M_fiber_work)
c     write(damp_out,*) ' Cal_stiff_linear Ok'
c――――――――――――――――――――――――――――――――★剛性の釣合座標系への変換(ok)
      call Rotate_stiffness(Parameter_C,ak_linear,rot_memb,Point,Member)
c     write(damp_out,*) ' Rotate_stiffness Ok'
c――――――――――――――――――――――――――――――――★部材の減衰行列計算(ok)
      if(Parameter_C.nc_member .ne. 0) then
      call Cal_damp_linear(Element,Member,Parameter_C,ac_member,
     *           E_model6_real,work1_elememt,
     *           work2_elememt,work1_member,work2_member)
c     write(damp_out,*) ' Cal_damp_linear Ok'
c――――――――――――――――――――――――――――――――★部材減衰行列の釣合座標系への変換(ok)
      call Rotate_damp(Parameter_C.n_member,ac_member,rot_memb,Member,Point,Parameter_C)
c     write(damp_out,*) ' Rotate_damp Ok'
      end if
c――――――――――――――――――――――――――――――――★節点集中質量セット(ok)
      call Set_mass(Point,Parameter_C,am_point)
c     write(damp_out,*) ' Set_mass Ok'
```

太文字で示したこれらのサブルーチンは以下のように変更される。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Rotate_stiffness
C     ――――――――――――――――――――――――――――――――――――
C     ●   部材行列の釣合系への座標変換(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Rotate_stiffness(Parameter_C,ak_linear,rot_memb,Point,Member)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record /parameter_s  / Parameter_C
      record / point_s     / Point
      record / member_s    / Member
      dimension Point(*),Member(*)
      dimension bk(12,12),rot_memb(3,3,2,*)
      dimension ak_linear(12,12,*)
      dimension alx(2),aly(2)
c――――――――――――――――――――――――――――――――――――――――――――
c     n_member                  : integer   部材数
```

```
c     ak_member(12,12,n_member)   : real*8    線形剛性行列
c     rot_memb(12,12,2,*)         : real*8    座標変換行列
c-----------------------------------------------------------------
      n_member = Parameter_C.N_member
c      write(76,*) ' 座標変換後の剛性：',n_member
      if(Parameter_C.n_gouyuka .eq.0) then

      do i=1,n_member
      call Rotate_K(ak_linear(1,1,i),rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),bK)
      do  j=1,12
      do  k=1,12
      ak_linear(j,k,i)=bk(j,k)
      end do
      end do
      end do

      else
      do i=1,n_member
      call Rotate_K(ak_linear(1,1,i),rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),bK)
c-------------------------------- 剛床座標変換
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       n_gouyuka=1
       aly(k)= Point(i1).coord_local(1)
        alx(k)= Point(i1).coord_local(2)
       else
        aly(k)=0.
        alx(k)=0.
       endif
      enddo
      if(n_gouyuka.ne.0) call Set_gtrans_k(bk,aly(1),alx(1), aly(2),alx(2) )
      do  j=1,12
      do  k=1,12
      ak_linear(j,k,i)=bk(j,k)
      end do
      end do
c      write(76,*) ' member: ',i
c     do j=1,12
c      write(76,'(i4,12e10.3)')j,(bk(j,k),k=1,12)
c     enddo
      end do

      endif
      return
      end
```

部材減衰行列の変換用サブルーチン Rotate_damp()は、以下のように変更する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Rotate_damp
C       ―――――――――――――――――――――――――――――――――
C     ●   部材減衰行列の釣合座標系への変換(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Rotate_damp(n_member,ac_member,rot_memb,Member,Point,Parameter_C)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension bk(12,12),rot_memb(3,3,2,*)
      dimension ac_member(12,12,*)
      record /parameter_s  / Parameter_C
      record / point_s     / Point
      record / member_s / Member
      dimension Member(*),Point(*)
      dimension alx(2),aly(2)
C     ―――――――――――――――――――――――――――――――――
c     n_member                  : integer   部材数
c     ac_member(12,12,n_member)   : real*8    減衰剛性行列
c     rot_memb(12,12,2,*)         : real*8    座標変換行列
c     Member                    : structure
C     ―――――――――――――――――――――――――――――――――
      if(n_member .eq.0) return
      if(Parameter_C.n_gouyuka .eq.0) then

      do i=1,n_member
      ij=Member(i).nm_damp
      if(ij.ne.0) then
      call Rotate_K(ac_member(1,1,ij),rot_memb(1,1,1,i),
     *             rot_memb(1,1,2,i),bK)
      do  j=1,12
      do  k=1,12
      ac_member(j,k,ij)=bk(j,k)
      end do
      end do
      end if
      end do

      else
      do i=1,n_member
      ij=Member(i).nm_damp
      if(ij.ne.0) then
      call Rotate_K(ac_member(1,1,ij),rot_memb(1,1,1,i),
     *             rot_memb(1,1,2,i),bK)
c―――――――――――――――――――――――― 剛床座標変換
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       n_gouyuka=1
        aly(k)= Point(i1).coord_local(1)
        alx(k)= Point(i1).coord_local(2)
       else
        aly(k)=0.
        alx(k)=0.
```

```
        endif
       enddo
       if(n_gouyuka.ne.0) call Set_gtrans_k(bk,aly(1),alx(1), aly(2),alx(2) )
       do  j=1,12
       do  k=1,12
       ac_member(j,k,ij)=bk(j,k)
       end do
       end do
       endif
       end do

       endif
       return
       end
```

次は、ニューマークβ法における左辺係数行列を作成する部分を見てみよう。ます、主サブルーチンを以下に示す。

```
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★        第１と第２段階解析の最初にスカイライン行列を作成、分解する
c             [F] = [M] + μ1[C] + μ2[K]
c―――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――★左辺係数行列の計算(ok)
c――――――――――――――――――――――――――――――――★ステップ番号のセット(ok)
      if(istep.eq.1.or.istep.eq.Newmark_P.n2_step) then
      n_istep=1
      if(istep.eq.Newmark_P.n2_step) n_istep=2
c――――――――――――――――――――――――――――――――★スカイライン行列のゼロセット(ok)
      n_skyline=Parameter_C.n_skyline
      call Set_sky_zero(gskym,n_skyline)
c     write(damp_out,*) ' Set_sky_zero ok'
c――――――――――――――――――――――――――――――――★集中質量系の行列への足し込み
c――――――――――――――――――――――――――――――――★レーリー減衰を含む
      call Build_sky_mm(n_istep,gskym,
     *          Point,n_point, am_point , rot_local,
     *          n_local_coord ,Newmark_P, max_h_sky)
c     write(damp_out,*) ' Build_sky_mm ok'
c――――――――――――――――――――――――――――――――★部材の整合質量系の行列への足し込み(ok)
c――――――――――――――――――――――――――――――――★レーリー減衰を含む
c――――――――――――――――――――――――――――――――★部材の整合質量行列計算(ok*)
      if(Dynamic_load.load_mass .ne. 0) then
      call Cal_mass_linear(n_istep,Element,Member,Parameter_C,am_member,
     *           work1_elememt,work2_elememt,work1_member,work2_member,
     *           Dynamic_load.load_mass)
c     write(damp_out,*) ' Cal_mass_linear ok'
c――――――――――――――――――――――――――――――――★整合質量の釣合座標系への変換(ok*)
      call Rotate_mass(n_istep,Element,Member,n_member,am_member,
     *                rot_memb,Dynamic_load.load_mass,Point)
c      write(damp_out,*) ' Rotate_mass ok'
c――――――――――――――――――――――――――――――――★整合質量系の足し込み(ok*)
      call Build_sky_m(n_istep,gskym,n_skyline, Member,n_member,
```

```
     *                 am_member ,Newmark_P, max_h_sky,Element)
      endif
c       write(damp_out,*) ' Build_sky_m ok'
c――――――――――――――――――――――――――――★部材減衰系の足し込み(ok)
c――――――――――――――――――――――――――――★Maxwell 線形減衰を含む
      if(Parameter_C.nc_member .ne. 0) then
       call Build_sky_c(gskym,Member,n_member,
     *                ac_member ,Newmark_P, max_h_sky)
c        write(damp_out,*) ' Build_sky_c ok'
      endif
c――――――――――――――――――――――――――――★線形剛性の足し込み(ok)
c――――――――――――――――――――――――――――★レーリー減衰を含む
      call Build_sky_k(n_istep,gskym,n_skyline, Member,n_member,
     *                      Ak_linear, Newmark_P, max_h_sky)
c       write(damp_out,*) ' Build_sky_k ok'
c――――――――――――――――――――――――――――――――――★行列の LDU 分解(ok)
      n_skyline=Parameter_C.n_skyline
      call decomp_sky(n_skyline,n_unknown,n_unknown,max_h_sky,
     *               gskym,gskym_d,nwork,twork,iexit)
c     write(damp_out,'(a,12f12.3)')' gskym *'
c――――――――――――――――――――――――――――――――――★分解成功か？
```

最初に、集中質量系の係数行列への組み込みを行うサブルーチンを変更しよう。このサブルーチンは、

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Build_sky_mm
C       ――――――――――――――――――――――――――――――――――
C     ●    集中質量行列のスカイライン行列への組み込み(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine Build_sky_mm(n_istep,gskym,
     *          Point,n_point, am_point , rot_local,
     *          n_local_coord ,Newmark_P, max_h_sky)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension am_point(2,*),gskym(*)
      dimension max_h_sky(0:*),rot_local(3,3,*)
      record / newmark_s   / Newmark_P
      record / point_s     / Point
      dimension Point(*)
      dimension amm(6,6)
c――――――――――――――――――――――――――――――――――――――――――――
c     n_istep                 :integer   第一ステップか第二ステップか
c     gskym(n_skyline)        :real*8    スカイライン行列
c     Point                   :structure
c     n_point                 :integer   節点数
c     am_point(2,n_point)     :real*8   節点集中質量
c     rot_local(3,3, *)       :real*8   全体座標系から局所座標への回転行列
c     n_local_coord           :integer    局所座標系を用いている節点数
c     Newmark_P               :structure
c     max_h_sky(n_unknown+1)  :integer     スカイライン行列の各列の高さ
c
```

```
c     集中系では、座標変換は必要でない。
c
c――――――――――――――――――――――――――――――――――――――――
      if(n_istep.eq.1) then
      ik=1
      par=1. + Newmark_P.ddt*Newmark_P.alf1_1
      else
      ik=2
      par=1. + Newmark_P.ddt*Newmark_P.alf2_1
      endif

      do i=1,n_point
      am=par*am_point(ik,i)
      if(Point(i).n_group_gouyuka.ne.0) then
      call Set_gtrans_m(am,amm, Point(i).coord_local(1), Point(i).coord_local(2))
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
       i3=max_h_sky(irest)
      gskym(i3)=gskym(i3)+am
      endif
      enddo
      irest = Point(i).irest(6)
      if(irest.gt.0) then
       i3=max_h_sky(irest)
      gskym(i3)=gskym(i3)+amm(6,6)
      endif
       i1= Point(i).irest(1)
       if(i1.gt.0) then
       i2= Point(i).irest(6)
       if(i2.gt.0.and.i2.le.i1) then
        i3=max_h_sky(i1)-(i1-i2)
        gskym(i3)=gskym(i3)+amk(1,6)
       elseif(i2.gt.0.and.i1.le.i2) then
        i3=max_h_sky(i2)-(i2-i1)
        gskym(i3)=gskym(i3)+amk(1,6)
      endif
      endif
       i1= Point(i).irest(2)
       if(i1.gt.0) then
        i2= Point(i).irest(6)
        if(i2.gt.0.and.i2.le.i1) then
        i3=max_h_sky(i1)-(i1-i2)
        gskym(i3)=gskym(i3)+amk(2,6)
       elseif(i2.gt.0.and.i1.le.i2) then
        i3=max_h_sky(i2)-(i2-i1)
        gskym(i3)=gskym(i3)+amk(2,6)
      endif
      endif
      else
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      i3=max_h_sky(irest)
```

```
      gskym(i3)=gskym(i3)+am
      endif
      enddo
      endif
      end do
      return
      end
```

次は、整合質量行列を座標変換するサブルーチンを変更する。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Rotate_mass
C       ―――――――――――――――――――――――――――――――――――
C     ●   部材質量行列の釣合系への座標変換(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Rotate_mass(nx,Element,Member,n_member,
     *                       am_member,rot_memb,load_mass,Point)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / point_s     / Point
      dimension Point(*),alx(2),aly(2)
      dimension Member(*),Element(*)
      dimension bk(12,12),rot_memb(3,3,2,*)
      dimension am_member(12,12,*)
C     ―――――――――――――――――――――――――――――――――――
c     n_member                    : integer   部材数
c     am_member(12,12,n_member)   : real*8    質量整合行列
c     rot_memb(12,12,2,*)         : real*8    座標変換行列
C     ―――――――――――――――――――――――――――――――――――
      if(load_mass .eq. 0 ) return
c――――――――――――――――――――――――――――――――★質量行列の計算
      if(nx.eq.1) then
      ik=1
      else
      ik=2
      endif
      do i=1,n_member
      ie = Member(i).nm_element
      if(Element(ie).am(ik) .ne. 0.0) then
      call Rotate_K(am_member(1,1,i),rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),bK)
c――――――――――――――――――――――――― 剛床座標変換
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       n_gouyuka=1
       aly(k)= Point(i1).coord_local(1)
       alx(k)= Point(i1).coord_local(2)
       else
       aly(k)=0.
```

```
        alx(k)=0.
        endif
       enddo
       if(n_gouyuka.ne.0) call Set_gtrans_k(bk,aly(1),alx(1),
      *                                     aly(2),alx(2) )
       do  j=1,12
       do  k=1,12
       am_member(j,k,i)=bk(j,k)
       end do
       end do
       endif
       end do
       return
       end
```

最後に、接線剛性を計算するサブルーチン Get_nonlinear_stiff()であるが、これは静的解析で変更部分を既に示しておいた。これをそのまま使用することになる。このサブルーチンコールは、

```
c――――――――――――――――――――――――――――――★接線剛性の計算(ok)
      call Get_nonlinear_stiff(Control.type_analysis,Point,Parameter_C,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
     *     MSS_work,S_comp_model,E_modelx, M_modelx,
     *     E_fiber_work, M_fiber_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member)
```

であり、サブルーチン自身は以下のようである。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_nonlinear_stiff
C     ――――――――――――――――――――――――――――――――――――
C     ●   接線剛性行列の計算(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Get_nonlinear_stiff(N_analysis,Point,Parameter_C,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
      *     MSS_work,S_comp_model,E_modelx, M_modelx,
      *     E_fiber_work, M_fiber_work,
      *     work1_elememt,work2_elememt, work1_member, work2_member)
       implicit real*8(A-H,O-Z)
       include "submain.h"
       include "submainx.h"
       include "New_submain.h"
       record / member_s    / Member
      record / point_s     / Point
      record /parameter_s  / Parameter_C
       record / element_s   / Element
       record / n_model_s   / Model_type
c―――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
       record / S_comp_model_s    / S_comp_model
       record / E_modelx_s        / E_modelx
       record / M_modelx_s        / M_modelx
       record / E_fiber_work_s    / E_fiber_work
       record / M_fiber_work_s    / M_fiber_work
       dimension E_modelx(*),M_modelx(*),S_comp_model(*)
       dimension E_fiber_work(*),M_fiber_work (*)
      dimension Point(*),aly(2),alx(2)

        •
        •
  100 continue
c―――――――――――――――――――――――――――――――――★部材の接線剛性を釣合系に変換

       call Rotate_K(ak,rot_memb(1,1,1,i),
     *             rot_memb(1,1,2,i),ak_nonlinear(1,1,i))
c―――――――――――――――――――――――――――――――――★剛床の変換
       if(Parameter_C.n_gouyuka .ne.0) then
       n_gouyuka=0
        do k=1,2
        i1=Member(i).nm_point(k)
        if(Point(i1).n_group_gouyuka.ne.0) then
        n_gouyuka=1
         aly(k)= Point(i1).coord_local(1)
         alx(k)= Point(i1).coord_local(2)
        else
         aly(k)=0.
         alx(k)=0.
        endif
       enddo
       if(n_gouyuka.ne.0) call Set_gtrans_k(ak_nonlinear(1,1,i),aly(1),alx(1), aly(2),alx(2) )
       endif
```

9.5.10.2 変位ベクトルの変換

本節では、動的解析において、釣合座標系から部材座標系に変換する場合について説明する。この場合、まず剛床に関する変換を行い、その後、全体座標系から部材座標系に変換しなければならない。ここでは、SPACE で使用している多くの座標変換部分を取り出し、プログラムに変更を加えることになる。変位に関連して変更するサブルーチンは、以下のようである。

```
Set_preset_disp()
Cal_stress()
Check_stress()
Out_disp_vel_acc()
```

ここでは、これらのサブルーチンの変更部分を示すことにする。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_preset_disp
C     ――――――――――――――――――――――――――――――――――
C     ●   節点の変位、速度、加速度を出力(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine Set_preset_disp(ihan,n_point,past_disp_point,
     *                           F_disp,Point,rot_local,Parameter_C)
      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(*)
      dimension rot_local(3,3,*),v(6),vv(6)
      real*4    F_disp(3,*)
      if(ihan.ne.0) goto 900
      do i=1,n_point
      do j=1,3
      F_disp(j,i)=0.
      enddo
      enddo
      return
  900 continue
c――――――――――――――――――――――――――――――――★局所座標系なし
      if(Parameter_C.n_local_coord.eq.0 .and.
     * Parameter_C.n_gouyuka .eq.0) then
c――――――――――――――――――――――――――――――――★変位 3
      do i=1,n_point
      do j=1,3
      ires= Point(i).irest(j)
      F_disp(j,i)=0.
      if(ires.ne.0) F_disp(j,i) = past_disp_point(ires)
      end do
```

```
c     write(76,'(a,i4,6f12.3)') ' set:',i,(F_disp(j,i),j=1,3)
      end do

c-------------------------------★座標変換あり
      else
c-------------------------------★変位 3
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.ne.0) then
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j ) = past_disp_point(ires)
      end do
      call trans_VT(v,vv,rot_local(1,1,ij))
      do j=1,3
      F_disp(j,i)=vv(j)
      enddo
c-------------------------------★座標変換なし
      elseif(Point(i).n_group_gouyuka.eq.0) then
c-------------------------------★変位 3
      do j=1,3
      ires= Point(i).irest(j)
      F_disp(j,i)=0.
      if(ires.ne.0) F_disp(j,i) = past_disp_point(ires)
      end do
c-------------------------------★剛床座標系あり
      else
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j ) = past_disp_point(ires)
      end do
      j=6
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j ) = past_disp_point(ires)
      call Set_gtrans_u(1,v,Point(i).coord_local(1),Point(i).coord_local(2))
      do j=1,3
      F_disp(j,i)=v(j)
      enddo

      endif
      end do

      endif
      return
      end
```

```
C     ------------------------------------------
C     ●   SUBROUTINE /Cal_stress
C     ------------------------------------------
C     ●   部材内応力の計算(ok)
```

```
C     ――――――――――――――――――――――――――――――――――――――――
      subroutine Cal_stress(Member,Point,n_member,Model_type,Element,
     *     past_disp_point,disp_point,rot_memb,
     *     E_model6_real,ak_nonlinear,N_analysis)
C     ――――――――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      record / point_s     / Point
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / E_model6_real_s   / E_model6_real
c―――――――――――――――――――――――――――――――――――――――――――――
      dimension Member(*),Element(*),E_model6_real(*)
      dimension Point(*)
      dimension rot_memb(3,3,2,*),ak_nonlinear(12,12,*)
      dimension past_disp_point(*),disp_point(*),
     *          v(12),vv(12),vp(12),vpp(12),ak(12,12),vx(12),vpx(12)
       ・
       ・
      do i=1,n_member
c     write(76,'(a,i4)') ' mem: z',i
c――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      vp(j)=past_disp_point(ires)
      v(j)=disp_point(ires)
      vpx(j)=vp(j)
      vx(j)=v(j)
      else
      v(j)=0.
      vp(j)=0.
      vpx(j)=0.
      vx(j)=0.
      endif
      enddo
c      write(76,'(a,12f12.6)')' v ',(v(j),j=1,12)
c      write(76,'(/i4,12f12.4)') i,(Member(i).stress(j),j=1,12)
c――――――――――――――――――――――――――――――――★変位を釣合系から部材座標系に変換
c―――――――――――――――――――★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,vx(k1),Point(i1).coord_local(1),
     *                          Point(i1).coord_local(2))
       call Set_gtrans_u(1,vpx(k1),Point(i1).coord_local(1),
     *                          Point(i1).coord_local(2))
       endif
      enddo
c―――――――――――――――――――★座標変換
      call RotateL_v(1,vx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
```

```
      call RotateL_v(1,vpx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp
c―――――――――――――――――――――――――――――――――★要素及びモデルのセット
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie = Member(i).nm_element
      im = Element(ie).n_element
      imm = Member(i).n_element_type
      ien= Member(i).n_model_type
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
c―――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
      call Cal_stress_M1(Member(i),Element(ie),
     *     ak_nonlinear(1,1,i),v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),
     *      Point)
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      call Cal_stress_M2(Member(i),Element(ie),vv)
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
      call Cal_stress_M3(Member(i),Element(ie),
     *     ak_nonlinear(1,1,i),v,vv,vvp,
     *     rot_memb(1,1,1,i),rot_memb(1,1,2,i),
     *      Point)
      goto 100
   14 continue
       .
       .
```

上記のサブルーチン Cal_stress()中で、各部材モデルの応力を計算するサブルーチンにおいて、次のように座標変換する必要がある。代表的なサブルーチンを以下に示すが、他のサブルーチンにおいても、同様に変更する必要がある。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress_M1
C       ――――――――――――――――――――――――――――――――――
C     ●   部材の応力計算(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine Cal_stress_M1(Member,Element,ak,vv,r1,r2,Point)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / point_s     / Point
      dimension Point(*)
      dimension ak(12,12),vv(12),r1(3,3),r2(3,3),st(12),ss(12)
```

```
      do i=1,12
      s=0.
      do j=1,12
      s=s+ak(i,j)*vv(j)
      enddo
      st(i)=s
      enddo
c―――――――――――――――――――――――――――――――★全体座標から部材座標へ変換
c――――――――――――――――――――★剛床座標変換
       do k=1,2
        i1=Member.nm_point(k)
        if(Point(i1).n_group_gouyuka.ne.0) then
        k1=6*(k-1)+1
        call Set_gtrans_f(1,st(k1),Point(i1).coord_local(1),
     *                                Point(i1).coord_local(2))
        endif
      enddo
c――――――――――――――――――――★全体座標から部材座標
      call RotateL_v(1,st,r1,r2,ss)
      do i=1,6
      Member.stress(i)=-ss(i)+Member.stress(i)
      enddo
      do i=7,12
      Member.stress(i)=ss(i)+Member.stress(i)
      enddo
c     write(76,'(6f12.2)') (Member.stress(j),j=1,6)
c
c―――――――――――――――――――――――――――――――★非線形応力の計算(ok)
c
c      ann=ak(1,1)/Member.alength*( (vp(8)-vp(2))*(vv(8)-vv(2))+
c     *           (vp(9)-vp(3))*(vv(9)-vv(3)) )
c     Member.stress(1)=Member.stress(1)+ ann
c     Member.stress(7)=Member.stress(7)+ ann

      return
      end
```

次に、サブルーチン Check_stress()の変更点は以下のようである。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Check_stress
C     ―――――――――――――――――――――――――――――――――
C     ●   部材の塑性状態をチェックする(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Check_stress(Control,N_analysis, Point,
     *     ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,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,
      *     MSS_work,
      *     Bilinear_work,Trilinear_work,Concrete_work,RO_work,
      *     work1_elememt,work2_elememt, work1_member, work2_member,
      *     S_comp_model,E_modelx,M_modelx,
      *     E_fiber_work,M_fiber_work)
       implicit real*8(A-H,O-Z)
       include "submain.h"
       include "submainx.h"
       include "New_submain.h"
       record /control_s    / Control
       record / member_s    / Member
       record / point_s     / Point
       record / element_s   / Element
       record / n_model_s   / Model_type
       record / Bilinear_work_s    / Bilinear_work
       record / Trilinear_work_s   / Trilinear_work
       record / Concrete_work_s    / Concrete_work
c―――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
       record / S_comp_model_s    / S_comp_model
       record / E_modelx_s        / E_modelx
       record / M_modelx_s        / M_modelx
       record / E_fiber_work_s    / E_fiber_work
       record / M_fiber_work_s    / M_fiber_work
       dimension E_modelx(*),M_modelx(*),S_comp_model(*)
       dimension E_fiber_work(*),M_fiber_work (*)
       dimension Point(*)
       dimension vx(12),vpx(12)
        .
        .
       do i=1,n_member
c      write(76,'(a,2i4)') ' memb : ',i,Member(i).element_type
c       kk=10
c       write(76,'(a,i4,3f12.3)') 'non c',kk,Member(kk).stress(7),
c      *  Member(kk).stress(8)
c―――――――――――――――――――――――――――――――――★部材両端の変位取得
       do j=1,12
       ires=Member(i).irest(j)
       if(ires.gt.0) then
       v(j)=disp_point(ires)
       vp(j)=past_disp_point(ires)
       vx(j)=v(j)
       vpx(j)=vp(j)
       else
       v(j)=0.
       vp(j)=0.
       vx(j)=0.
       vpx(j)=0.
       endif
       enddo
```

```
c        write(76,'(a,i3,12e12.5)') 'v',i,(v(j),j=1,12)
c       write(76,'(a,i3,12e12.5)') 'vp',i,(vp(j),j=1,12)
c――――――――――――――――――――――――――――――――――――★部材両端の節点力のゼロセット
      do j=1,12
      f(j)=0.
      enddo
c――――――――――――――――――――――――――――――――――――★変位を釣合系から部材座標系に変換
c――――――――――――――――――――――――――――――――――――★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,vx(k1),Point(i1).coord_local(1),Point(i1).coord_local(2))
       call Set_gtrans_u(1,vpx(k1),Point(i1).coord_local(1),Point(i1).coord_local(2))
       endif
      enddo
c――――――――――――――――――――――――――――――――――――★座標変換
      call RotateL_v(1,vx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vpx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c        write(76,'(a,i3,12e12.5)') 'vv',i,(vv(j),j=1,12)
c       write(76,'(a,i3,12e12.5)') 'vpp',i,(vpp(j),j=1,12)
      .
      .
```

```
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      write(76,*) ' 変位：', n_point
c――――――――――――――――――――――――――――――――――★局所座標系なし
      if(Parameter_C.n_local_coord.eq.0 .and.
     *          Parameter_C.n_gouyuka .eq.0) then
c――――――――――――――――――――――――――――――――――★変位 3
      do i=1,n_point
      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)
c     write(76,'(i4,3e12.4,2i4)') i,(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)
      end do
      if(ifl(11).eq.1) write(iflz(11)) (v(j),j=1,3)

c     write(76,'(i4,6e12.4)') i,(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)
      end do
      if(ifl(10).eq.1) write(iflz(10)) (v(j),j=1,3)
c     write(76,'(i4,6e12.4)') i,(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)
c     write(76,'(i4,6e12.4)') i,(v(j),j=1,3)
      end do
c――――――――――――――――――――――――――――――――――★座標変換あり
      else
c――――――――――――――――――――――――――――――――――★変位 3
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.ne.0) then
c――――――――――――――――――――――――――――――――――★局所座標系あり
      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))
      do j=1,3
      v(j)=vvv(j)
      enddo
c―――――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .eq.0) then
      do j=1,3
      ires= Point(i).irest(j)
      v(j)=0.
      if(ires.ne.0) v(j) = past_disp_point(ires)
      end do
      else
c―――――――――――――――――――――――――――――――★剛床座標系あり
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_disp_point(ires)
      end do
      j=6
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_disp_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
      do j=1,3
      v(j)=vv(j)
      enddo
      endif
      if(ifl(3).eq.1) write(iflz(3)) (v(j),j=1,3)
      end do
c―――――――――――――――――――――――――――――――★速度 11
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.ne.0) then
c―――――――――――――――――――――――――――――――★局所座標系あり
      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
c―――――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .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
      else
c―――――――――――――――――――――――――――――――★剛床座標系あり
```

```
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_vel_point(ires)
      enddo
      j=6
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_vel_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
      do j=1,3
      v(j)=vv(j)
      enddo
      endif
      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
      do i=1,n_point
      ij=Point(i).local_coord
      if(ij.ne.0) then
      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
c―――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .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
      else
c―――――――――――――――――――――――――――――★剛床座標系あり
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_acc_point(ires)
      end do
      j=6
```

```
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_acc_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
      do j=1,3
      v(j)=vv(j)
      enddo
      if(ifl(10).eq.1) write(iflz(10)) (v(j),j=1,3)
      do j=1,3
      v(j)=vv(j)+vacc(j)
      enddo
      endif
      if(ifl(13).eq.1) write(iflz(13)) (v(j),j=1,3)
      end do
      endif
      return
      end
```

```
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,rot_local)
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(6)
      dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
      dimension vv(6),vvv(6)
      dimension rot_local(3,3,*)
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
c――――――――――――――――――――――――――――――――――★変位最大値
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_disp_point(ires)
```

```
      enddo

      ij=Point(i).local_coord
      if(ij.ne.0) then
c―――――――――――――――――――――――――――――――――★局所座標系あり
      call trans_VT(vv,vvv,rot_local(1,1,ij))
      do j=1,3
      vv(j)=vvv(j)
      enddo
c―――――――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .ne.0) then
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      j=6
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_disp_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
      endif
c―――――――――――――――――――――――――――――――――★最大値チェック
      do j=1,3
      aa = vv(j)
      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
      if(Max_disp(i).disp_point(j+3).gt.aa)
     *                     Max_disp(i).disp_point(j+3)=aa
      enddo
c―――――――――――――――――――――――――――――――――★速度最大値
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_vel_point(ires)
      enddo

      ij=Point(i).local_coord
      if(ij.ne.0) then
c―――――――――――――――――――――――――――――――――★局所座標系あり
      call trans_VT(vv,vvv,rot_local(1,1,ij))
      do j=1,3
      vv(j)=vvv(j)
      enddo
c―――――――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .ne.0) then
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      j=6
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_vel_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
```

```
      endif
c―――――――――――――――――――――――――――――――――★最大値チェック
      do j=1,3
      aa = vv(j)
      if(Max_disp(i).vel_point(j).lt.aa) Max_disp(i).vel_point(j)=aa
      if(Max_disp(i).vel_point(j+3).gt.aa) Max_disp(i).vel_point(j+3)=aa
      enddo
c―――――――――――――――――――――――――――――――――★加速度最大値
      do j=1,3
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_acc_point(ires)
      enddo

      ij=Point(i).local_coord
      if(ij.ne.0) then
c―――――――――――――――――――――――――――――――――★局所座標系あり
      call trans_VT(vv,vvv,rot_local(1,1,ij))
      do j=1,3
      vv(j)=vvv(j)
      enddo
c―――――――――――――――――――――――――――――――――★座標変換なし
      elseif(Point(i).n_group_gouyuka .ne.0) then
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      j=6
      ires= Point(i).irest(j)
      vv(j)=0.
      if(ires.ne.0) vv(j) = past_acc_point(ires)
      call Set_gtrans_u(1,vv,Point(i).coord_local(1),
     *                       Point(i).coord_local(2))
      endif
c―――――――――――――――――――――――――――――――――★最大値チェック
      do j=1,3
      aa=vv(j)
      if(Max_disp(i).acc_point(j).lt.aa) Max_disp(i).acc_point(j)=aa
      if(Max_disp(i).acc_point(j+3).gt.aa) Max_disp(i).acc_point(j+3)=aa
      enddo

c―――――――――――――――――――――――――――――――――★絶対加速度最大値
チェック
      do j=1,3
      aa = vv(j)+vacc(j)
      if(vv(j).ne.0.) then
      if(Max_disp(i).ab_acc_point(j).lt.aa)
     *     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)
     *     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
      end if
```

```
      end do
      end do

      return
      end
```

9.5.10.3 応力と節点力ベクトルの変換

最初に、静的荷重を設定するサブルーチンを変更する。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_point_load
C       ―――――――――――――――――――――――――――――――――
C     ●   節点荷重データを釣合系の座標に変換し、セットする(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_point_load(fll_static_point,Parameter_C,
     *                Dynamic_load,Point,fld_static,rot_local)
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / parameter_s    / Parameter_C
      record /Dynamic_load_s/ Dynamic_load
      record / point_s        / Point
      dimension Point(*),fll_static_point(3,6,*),fld_static(3,*)
      dimension rot_local(3,3,*),p(3),q(3) ,pp(6)
       •
       •
      do i=1,3
      if(Dynamic_load.load_point(i) .ne. 0) then
      do j=1,n_point
      if(Point(j).local_coord .eq. 0 .and.
     *  Point(j).n_group_gouyuka.eq.0) then
      do k=1,6
      ires=Point(j).irest(k)
      if(ires .ne. 0)
     *  fld_static(i,ires) = fld_static(i,ires)+fll_static_point(i,k,j)
      end do
c―――――――――――――――――――――――――――――★局所座標
      elseif(Point(j).local_coord .ne. 0 )then
      do k=1,3
      p(k) = fll_static_point(i,k,j)
      end do
      ip=Point(j).local_coord
      call trans_V(p,q,rot_local(1,1,ip))
      do k=1,3
      ires=Point(j).irest(k)
      if(ires .gt. 0)
     *  fld_static(i,ires) = fld_static(i,ires)+q(k)
      end do

      do k=1,3
      p(k) = fll_static_point(i,k+3,j)
```

```
      end do
      ip=Point(j).local_coord
      call trans_V(p,q,rot_local(1,1,ip))
      do k=1,3
      ires=Point(j).irest(k+3)
      if(ires .gt. 0)
     *  fld_static(i,ires) = fld_static(i,ires)+q(k)
      end do
c-----------------------------★剛床座標
      else
      do k=1,6
      pp(k) = fll_static_point(i,k,j)
      end do
      call Set_gtrans_f(2,pp,
     *     Point(j).coord_local(1),Point(j).coord_local(2))
      do k=1,6
      ires=Point(j).irest(k)
      if(ires .ne. 0)
     *  fld_static(i,ires) = fld_static(i,ires)+pp(k)
      end do
      endif
      end do

      endif
      end do
      write(76,'(//a,i4)') ' 静的荷重: ',Parameter_C.n_unknown
      write(76,*) 'load_point(1)',Dynamic_load.load_point(1)
      write(76,*) 'load_point(2)',Dynamic_load.load_point(2)
      write(76,*) 'load_point(3)',Dynamic_load.load_point(3)
      do j=1,Parameter_C.n_unknown
      write(76,'(i4,3f10.2)')j,(fld_static(i,j),i=1,3)
      end do
      return
      end
```

次に、右辺項ベクトルを作成する部分に関するサブルーチンを検討しよう。右辺項ベクトルは、

$$\{f\} = \{G\} + \{g\}$$
$$\{G(y_{n+1}, \Delta y_{n+1})\} = -\left\{\bar{f}_d\right\} - \left[K_T(y_n)\right]\{\Delta y_{n+1}\} + [K]\{y_{n+1}\}$$
$$\{g\} = -[M][I]\{\ddot{u}_g\} + \{P_S\} - \{Q(y_n)\} - \left[\bar{C}\right]\{a\} - [K]\left\{\bar{b}\right\}$$

である。ここで、再度、右辺項の変換規則について整理して置こう。なお、線形の剛性行列、接線剛性行列、整合質量行列、部材減衰行列は、剛床変換を含めた釣合座標系に変換された後、保存される。部材両端の節点力も釣合座標系に変換した後、保存される。静的荷重ベクトルも釣合座標系に変換された後、保存される。一方、集中質量は、対角行列であり、行列の形式で保存されていない。作成する場合は、全体座標系で

求める。

右辺項を求める場合、注意すべき点は慣性項である。質量行列に掛け算する地震加速度は、全体座標系で求められているため、集中質量系では、全体座標系の質量行列と掛け算した後、慣性項を釣合座標系に変換する。また、整合質量系は、既に釣合座標系に変換されているため、地震加速度ベクトルを全体座標系から釣合座標系に変換した後、掛け算して慣性項を求める。以下に、変換規則をまとめる。

１） Maxwell モデルの非線形項$\{\bar{f}_d\}$は、まず、釣合座標系の変位を全体座標系に変換した後、部材座標系に変換し、これよりモデルの非線形項を計算する。その後、非線形項を全体座標系に変換した後、剛床変換を用いて釣合座標系に変換する。

２） 接線剛性と線形剛性は、共に剛性行列を計算するときに釣合座標系に変換されており、このまま座標変換する必要はない。

３） 慣性項$[M][I]\{\ddot{u}_g\}$は、集中質量、整合質量共に全体座標系で求めておき、地震加速度と掛け算した後、慣性力を釣合座標系に変換する。

４） 擬似的静的荷重$\{\bar{f}_d\}$は、荷重ベクトルが既に釣合座標系に変換されているため、反復内では、座標変換を行う必要がない。

５） 不釣合い力$\{Q(y_n)\}$は、部材両端の節点力を求めた後、一端全体座標系に変換し、その後、剛床がある場合、釣合座標系に変換する。

６） 線形の剛性行列は、既に釣合座標系に変換されているため、そのまま、変位ベクトルと掛け算すればよい。

７） 質量に関連する減衰項は、集中質量系では、一端、$\{a\}$ベクトルを全体座標系に変換し、その後、質量行列と掛け算を行い、その結果を釣合座標系に変換する。整合質量行列は、既に釣合座標系に変換されているため、そのまま、$\{a\}$ベクトルと掛け算を行えば良い。

最初に、主サブルーチンの中で、右辺項ベクトルを求めるルーチンを以下に示す。

```
c------------------------------------------------------------------
c
c    ★          動的解析（反復解法）
c
c------------------------------------------------------------------
c-------------------------------★右辺の定数ベクトルゼロセット(ok)
      call Clear_vec(n_unknown,ld_point )
c        write(damp_out,*) ' Clear_vec ok'
c-------------------------------★Work(a,b)ベクトルのセット(ok)
```

```
      call Set_a_b_vec(n_unknown,a_vector,b_vector,Newmark_P,
     *                 past_vel_point, past_acc_point)
c-------------------------------★部材節点力のセット(ok)
      call Get_pointforce_ld(ld_point,Member,n_member)
c                   write(damp_out,'(a,6f16.5)') 'Get_pointforce_ld ok'
c      write(76,'(10e12.4)')(ld_point(i),i=1,n_unknown)
c-------------------------------★地震加速度セット(ok)
      acc1=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)
      acc2=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      acc3=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
c      write(damp_out,'(a,4f10.3,i4)')'Get_Acc ok'
c-------------------------------★集中質量に関する慣性項
      call Add_earth1_ld(n_istep,acc1,acc2,acc3,ld_point,
     *          Point,n_point,am_point,rot_local,Parameter_C)
c         write(damp_out,*) ' Add_earth1_ld ok'
c-------------------------------★整合質量に関する慣性項
      call Add_earth2_ld(acc1,acc2,acc3,ld_point,
     *    Member,n_member,am_member,rot_local,Parameter_C,Dynamic_load)
c      write(damp_out,*) ' Add_earth2_ld ok'
c-------------------------------★節点荷重のセット(ok)
      p1=Get_Ps(T,1,fdd_point,Dynamic_load)
      p2=Get_Ps(T,2,fdd_point,Dynamic_load)
      p3=Get_Ps(T,3,fdd_point,Dynamic_load)
c     write(damp_out,'(a,4f10.3)') ' Get_Ps ok'
      call Add_point_ld(p1,p2,p3,ld_point,n_unknown,
     *       Dynamic_load,fld_static)
c      write(damp_out,*) ' Add_point_ld ok'
c-------------------------------★線形減衰項計算(ok)
c-------------------------------★集中質量(ok)
c      call Add_damp1_ld(n_istep,ld_point,Point,n_point,
c     *       past_disp_point,past_vel_point,past_acc_point,
c     *       am_point,Newmark_P,Parameter_C,rot_local)
      call Add_damp1_ld_ex(n_istep,ld_point,Point,n_point,
     *       a_vector,am_point,Newmark_P,Parameter_C,rot_local)
c     write(damp_out,*) ' Add_damp1_ld ok'
c-------------------------------★整合質量(ok)
c      call Add_damp2_ld(n_istep,ld_point,Member,n_member,
c     *       past_disp_point,past_vel_point,past_acc_point,
c     *       am_member,Newmark_P,Element,Dynamic_load.load_mass)
      call Add_damp2_ld_ex(n_istep,ld_point,Member,n_member,a_vector,
     *       am_member,Newmark_P,Element,Dynamic_load.load_mass)
c     write(damp_out,*) ' Add_damp2_ld ok'
c-------------------------------★部材減衰(ok)
c-------------------------------★Maxwell 線形減衰を含む
c      call Add_damp3_ld(n_istep,ld_point,Member,n_member,
c     *       past_disp_point,past_vel_point,past_acc_point,
c     *       ac_member,Newmark_P,Model_type.n_m_damp)
      call Add_damp3_ld_ex(n_istep,ld_point,Member,n_member,
     *       ac_member,a_vector,Element,rot_memb,E_model6_real,
     *       Newmark_P,Model_type.n_m_damp,Point)
c     write(damp_out,*) ' Add_damp3_ld ok'
c-------------------------------★線形剛性項計算(ok)
c                                                 レーリー減衰も含む
      call Add_stiff1_ld(n_istep,ld_point,Member,n_member,
```

```
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ak_linear,Newmark_P)
c      write(damp_out,*) ' Add_stiff1_ld ok'
c--------------------------------★線形剛性によるレーリー減衰（解析２）
c      call Add_stiff1_ld_ex(n_istep,ld_point,Member,n_member,
c     *       a_vector,ak_linear,Newmark_P)
c     write(damp_out,*) ' Add_stiff1_ld ok'
c--------------------------------★接線剛性に関する増分ベクトル（解析２）
c      call Add_tan_stiff_ld(ld_point,
c     *       Member,n_member,b_vector,ak_nonlinear)
c     write(damp_out,*) ' Add_stiff1_ld ok'
c--------------------------------★Δｔ秒後の変位と速度を予測(ok)
      n_err_roop = 0
 9991 continue
      nx =0
      call Estimate_disp_vel(nx, n_unknown,
     *       est_disp_point, est_vel_point, est_ddisp_point,
     *       past_disp_point, past_vel_point, past_acc_point,
     *       result_disp_point, result_vel_point,
     *       past_dacc_point,result_acc_point,Newmark_P)
c     write(damp_out,*) ' Estimate_disp_vel ok'
c-----------------------------------------------------------
c
c    ★        反復計算開始
c
c-----------------------------------------------------------
      n_roop=Newmark_P.max_repeat
      do iroop=1,n_roop
c     write(damp_out,*) ' 反復回数：' ,iroop
c--------------------------------★反復に関連する右辺ベクトルのゼロセット(ok)
      call Clear_vec(n_unknown,ld_point_repeat)
c         write(damp_out,*) ' Clear_vec ok'
c--------------------------------★線形剛性に関するベクトル(ok)
      call Add_stiff2_ld(ld_point_repeat,
     *       Member,n_member,est_disp_point,ak_linear)
c     write(damp_out,*) ' Add_stiff2_ld ok'
c--------------------------------★接線剛性に関する増分ベクトル(ok)
      call Add_tan_stiff_ld(ld_point_repeat,
     *       Member,n_member,est_ddisp_point,ak_nonlinear)
c     write(damp_out,*) ' Add_tan_stiff_ld ok'
c--------------------------------★線形剛性に関するベクトル(解析２)
c      call Add_stiff2x_ld(ld_point_repeat,
c     *       Member,n_member,result_acc_point,ak_linear,Newmark_P)
c     write(damp_out,*) ' Add_stiff2_ld ok'
c--------------------------------★接線剛性に関する増分ベクトル(解析２)
c      call Add_tan_stiff2_ld(ld_point_repeat,
c     *       Member,n_member,result_acc_point,ak_nonlinear,Newmark_P)
c     write(damp_out,*) ' Add_tan_stiff_ld ok'
c--------------------------------★Maxwell 型モデルの計算(ok)
      call Add_fdd_ld(ld_point_repeat,E_model6_real,Element,
     *       Member,n_member,est_vel_point,rot_memb)
c     write(damp_out,*) ' Add_fdd_ld ok'
c--------------------------------★右辺２項の和を取る(ok)
      call add_vec(n_unknown,ld_point_repeat,ld_point)
```

```
c     write(damp_out,*) ' add_vec ok'
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★線形方程式を解く(ok)
c      write(76,'(a)')' ld_point_repeat'
c      write(76,'(10e12.4')(ld_point_repeat(i),i=1,n_unknown)
      n_skyline=Parameter_C.n_skyline
      call solve_sky(n_skyline,n_unknown,n_unknown,
     *            max_h_sky,gskym,gskym_d,
     *            nwork,twork,ld_point_repeat,result_acc_point)
c      write(76,'(a)')' result_acc_point'
c      write(76,'(10e12.4')(result_acc_point(i),i=1,n_unknown)
c     write(damp_out,*) ' solve_sky ok'
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★β法に基づき加速度より変位と速度を計算(ok)
      call Cal_disp_vel(n_unknown,
     * result_disp_point, result_vel_point, result_acc_point,
     * past_disp_point, past_vel_point, past_acc_point,
     * Newmark_P)
c     write(damp_out,*) ' Cal_disp_vel ok'
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★収束したかチェック(ok)
      if(ICheck_error(iroop,n_point,Point,n_unknown,result_disp_point,
     *     est_disp_point, Newmark_P) .eq. 0) goto 9980
c     write(damp_out,*) ' Check_error ok'
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――★次の増分値を予測(ok)
      nx=iroop
      call Estimate_disp_vel(nx, n_unknown,
     *       est_disp_point, est_vel_point, est_ddisp_point,
     *       past_disp_point, past_vel_point, past_acc_point,
     *       result_disp_point, result_vel_point,
     *       past_dacc_point,result_acc_point, Newmark_P)
c     write(damp_out,*) ' Estimate_disp_vel ok'
      end do
```

上記のルーチンの中で変更しなければならないサブルーチンは、以下の
ようである。

```
Get_pointforce_ld()
Add_earth1_ld()
Add_earth2_ld()
Add_damp1_ld_ex
Add_damp3_ld_ex
```

これらのサブルーチンを以下のように変更する。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Get_pointforce_ld
C       ――――――――――――――――――――――――――――――――――――
C     ●   部材節点力のセット(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Get_pointforce_ld(ld_point,Member,n_member,
     *           Parameter_C,Point)
```

```
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      record / point_s     / Point
      record /parameter_s  / Parameter_C
      dimension Member(*),Point(*)
      real*8 ld_point(*),ff(12)
C     ―――――――――――――――――――――――――――――――――――
c           ld_point      :real*8 右辺項
c           Member        :structure
c           n_member      :integer 部材数
C     ―――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――――――★座標変換なし
      if(Parameter_C.n_gouyuka .eq.0) then
      do i=1,n_member
c                       Maxwell  Model の応力は、他で考慮するのでここでは、無視する。
      if(Member(i).element_type.ne.6) then
      do j=1,12
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - Member(i).force(j)
      end do
      endif
c     write(76,'(i3,12e10.3)') i,(Member(i).force(j),j=1,12)
      end do
      else
c――――――――――――――――――――――――――――――――――――――★剛床座標変換あり
      do i=1,n_member
      if(Member(i).element_type.ne.6) then
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) n_gouyuka=1
       enddo
      if(n_gouyuka .ne. 0) then
c――――――――――――――――――――――――――― 剛床座標変換あり
      do j=1,12
      ff(j)=Member(i).force(j)
      enddo
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
        k1=6*(k-1)+1
        call Set_gtrans_f(2,ff(k1),
     *     Point(i1).coord_local(1),Point(i1).coord_local(2))
        endif
      enddo
      do j=1,12
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - ff(j)
      end do
c――――――――――――――――――――――――――― 剛床座標変換なし
      else
      do j=1,12
      i1 = Member(i).irest(j)
```

```
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - Member(i).force(j)
c     if(i1.gt.0) ld_point(i1)=ld_point(i1) + Member(i).force(j)
      end do
      endif

      endif
      end do
      endif
      return
      end
```

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_earth1_ld
C       ――――――――――――――――――――――――――――――――――――
C     ●   集中質量行列に関する地震荷重(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Add_earth1_ld(n_istep,acc1,acc2,acc3,ld_point,
     *           Point,n_point,am_point,rot_local,Parameter_C)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension am_point(2,*)
      dimension rot_local(3,3,*)
      record / point_s     / Point
      record / parameter_s / Parameter_C
      dimension Point(*)
      dimension amk(6),amkk(3)
      real*8 ld_point(*)
c――――――――――――――――――――――――――――――――――――――――――――
c     n_istep                 :integer  第一ステップか第二ステップか
c     acc1,acc2,acc3          :real*8   ｘ、ｙ、ｚ方向の地震荷重
c     ld_point(*)             :real*8   右辺荷重項
c     Point                   :structure
c     n_point                 :integer  節点数
c     am_point(2,n_point)     :real*8   節点集中質量
c     rot_local(3,3, *)       :real*8   全体座標系から局所座標への回転行列
c     Parameter_C             :structure
c――――――――――――――――――――――――――――――――――――――――――――
      if(n_istep.eq.1) then
      ik=1
      else
      ik=2
      endif
c――――――――――――――――――――――――――――――――――――――――――――★座標変換あり
      if(Parameter_C.n_local_coord.ne.0
     *    .or. Parameter_C.n_gouyuka .ne. 0) then
      do i=1,n_point
      am=am_point(ik,i)
      amk(1)=am * acc1
      amk(2)=am * acc2
      amk(3)=am * acc3
c――――――――――――――――――――――――――――――――――――――――――――★局所座標系あり
      if(Point(i).local_coord.ne.0) then
      ij=Point(i).local_coord
```

```
      call RotateTLs_v(2,amk,rot_local(1,1,ij),amkk)
      do  j=1, 3
      irest=Point(i).irest(j)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amkk(j)
      end do
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      elseif(Point(i).n_group_gouyuka.ne.0) then
      amk(6)=0.
      call Set_gtrans_f(2,amk, Point(i).coord_local(1),
     *                         Point(i).coord_local(2) )
      do  j=1, 3
      irest=Point(i).irest(j)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amk(j)
      end do
      irest=Point(i).irest(6)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amk(6)
c―――――――――――――――――――――――――――――――――★座標変換なし
      else
      do  j=1,3
      irest =Point(i).irest(j)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amk(j )
      end do
      endif
      end do

c―――――――――――――――――――――――――――――――――★座標変換なし
      else
      do i=1,n_point
      am=am_point(ik,i)
      amk(1)=am * acc1
      amk(2)=am * acc2
      amk(3)=am * acc3
      do  j=1, 3
      irest=Point(i).irest(j)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amk(j)
      end do
      end do

      end if
      return
      end
```

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_earth2_ld
C       ―――――――――――――――――――――――――――――――――――
C     ●   整合質量質量行列に関する地震荷重(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Add_earth2_ld(Point,acc1,acc2,acc3,ld_point,
     *    Member,n_member,am_member,rot_local,Parameter_C,Dynamic_load)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension am_member(12,12,*)
      dimension rot_local(3,3,*)
```

```
      record / member_s     / Member
      record / parameter_s / Parameter_C
      record / dynamic_load_s/ Dynamic_load
      record / point_s     / Point
      dimension Point(*)
      dimension Member(*)
      dimension amk(12),amkk(6),accx(3)
      real*8 ld_point(*)
c-----------------------------------------------------------------
c     n_istep                 :integer  第一ステップか第二ステップか
c     acc1,acc2,acc3          :real*8   ｘ、ｙ、ｚ方向の地震荷重
c     ld_point(*)             :real*8   右辺荷重項
c     Point                   :structure
c     n_point                 :integer  節点数
c     am_member               :real*8   節点整合質量
c     rot_local(3,3,*)        :real*8   全体座標系から局所座標への回転行列
c     Parameter_C             :structure
c-----------------------------------------------------------------
      if(Dynamic_load.load_mass .eq.0 ) return
      if(Parameter_C.n_local_coord.eq.0
     *    .and. Parameter_C.n_gouyuka .eq. 0) then
c-------------------------------★座標変換なし
      do i=1,n_member
      do j=1,12
      amk(j)=acc1*am_member(j,1,i)+acc1*am_member(j,7,i)
     *      +acc2*am_member(j,2,i)+acc2*am_member(j,8,i)
     *      +acc3*am_member(j,3,i)+acc3*am_member(j,9,i)
      enddo
      do  j=1, 12
      i1=Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amk(j)
      end do
      end do
c-------------------------------★座標変換あり
      else
      accx(1)=acc1
      accx(2)=acc2
      accx(3)=acc3
      do i=1,n_member
      n_henkann=0
      do k=1,2
      i1= Member(i).nm_point(k)
      if(Member(i).nm_local_coord(k).ne.0 .or.
     *   Point(i1).nm_group_gouyuka .ne. 0) n_henkann=1
      enddo
c-------------------------------★座標変換なし
      if(n_henkann .eq. 0) then
      do j=1,12
      amk(j)=acc1*am_member(j,1,i)+acc1*am_member(j,7,i)
     *      +acc2*am_member(j,2,i)+acc2*am_member(j,8,i)
     *      +acc3*am_member(j,3,i)+acc3*am_member(j,9,i)
      enddo
      do  j=1, 12
      i1=Member(i).irest(j)
```

```
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amk(j)
      end do

      else
c―――――――――――――――――――――――――――――――――★局所座標変換あり
      do j=1,12
      amk(j)=0.
      amkk(j)=0.
      enddo
      do j=1,3
      if(accx(j).ne.0.) then
      amkk(j)=accx(j)
      amkk(j+6)=accx(j)
      do k=1,2
      i1= Member(i).nm_point(k)
      if(Member(i).nm_local_coord(k).ne.0 )then
      ij= Member(i).nm_local_coord(k)
      k1=(k-1)*6+1
      call RotateTLs_vm(2,amk(k1),rot_local(1,1,ij))
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      if(Point(i1).nm_group_gouyuka.ne.0) then
      k1=(k-1)*6+1
      call Set_gtrans_f(2,amkk(k1), Point(i1).coord_local(1), Point(i1).coord_local(2) )
      endif
      enddo
c―――――――――――――――――――――――――――――――――★質量行列と掛け算
      do k=1,12
      sum=0.
      do kk=1,12
      sum=sum+ am_member(k,kk,i)*amkk(kk)
      enddo
      amk(k)=amk(k)+sum
      enddo
      endif
      enddo
c―――――――――――――――――――――――――――――――――★荷重項へセット
      do  j=1, 12
      i1=Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amk(j)
      end do
      endif
      enddo

      end if
      return
      end
```

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_damp1_ld_ex
C       ―――――――――――――――――――――――――――――――――――
C     ●   減衰・集中質量による減衰項のセット(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Add_damp1_ld_ex(nx,ld_point,Point,n_point,
```

```
     *       a_vector,am_point,Newmark_P,Parameter_C,rot_local)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s   / Newmark_P
      record / parameter_s /Parameter_C
      record / point_s     /Point
      dimension Point(*)
      real*8 ld_point(*),am_point(2,*),rot_local(3,3,*)
      dimension u(6),amm(3),uu(6)
      dimension a_vector(*)
c-----------------------------------------------------------------
c     nx                      :integer  第一ステップか第二ステップか
c     ld_point(*)             :real*8   右辺荷重項
c     Point                   :structure
c     n_point                 :integer  節点数
c     a_vector                :real*8   a vactor
c     am_point(2,n_point)     :real*8   節点集中質量
c     Newmark_P               :structure
c     Parameter_C             :structure
c     rot_local(3,3,*)        :real*8   全体座標系から局所座標への回転行列
c-----------------------------------------------------------------
      if(nx.eq.1) then
      a= Newmark_P.alf1_1
      ik=1
      else
      a= Newmark_P.alf2_1
      ik=2
      endif
c-------------------------------★座標変換あり
      if(Parameter_C.n_local_coord.ne.0
     *    .or. Parameter_C.n_gouyuka .ne. 0) then
      do i=1,n_point
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      if(am.ne.0.) then
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      u(j)=a*a_vector(irest)
      else
      u(j)=0.0
      endif
      end do
c-------------------------------★局所座標系あり
      if(Point(i).local_coord.ne.0) then
      ij=Point(i).local_coord
c-------------------------------★局所座標変換は変換前の質量剛列と同一
      do j=1,3
      u(j)=amm(j)*u(j)
      enddo
       do  j=1, 3
      i1=Point(i).irest(j)
```

```
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - u(j)
      end do
c―――――――――――――――――――――――――――――――――★剛床座標系あり
      elseif(Point(i).n_group_gouyuka.ne.0) then
      j=6
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      u(j)=a*a_vector(irest)
      else
      u(j)=0.0
      endif
      call Set_gtrans_u(1,u, Point(i).coord_local(1),
     *                       Point(i).coord_local(2) )
      do j=1,3
      uu(j)=amm(j)*u(j)
      enddo
      uu(6)=0.
      call Set_gtrans_f(2,uu, Point(i).coord_local(1),
     *                        Point(i).coord_local(2) )
       do  j=1, 3
      i1=Point(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - uu(j)
      end do
      i1=Point(i).irest(6)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - uu(6)

c―――――――――――――――――――――――――――――――――★座標変換なし
      else
      do  j=1, 3
      i1=Point(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amm(j)*u(j)
      end do
      endif
      endif
      end do
c―――――――――――――――――――――――――――――――――★座標変換なし
      else
      do i=1,n_point
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      if(am.ne.0.) then
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.ne.0) then
      u(j)=a*a_vector(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1, 3
      i1=Point(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amm(j)*u(j)
```

```
      end do
      endif
      end do
c-----------------------------------------------------------------------
      endif
      return
      end
```

```
C     ------------------------------------------
C     ●   SUBROUTINE /Add_damp3_ld_ex
C       ----------------------------------------
C     ●   部材減衰による減衰項のセット(ok)
C     ------------------------------------------
      subroutine Add_damp3_ld_ex(nx,ld_point,Member,n_member,ac_member,
     *       a_vector,Element,rot_memb,E_model6_real,Newmark_P,n_damp,Point)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s    /Newmark_P
      record / member_s     /Member
      record / point_s      /Point
      record / element_s    / Element
      record /E_model6_real_s / E_model6_real
      dimension Member(*),Element(*),Point(*)
      real*8 ld_point(*),ac_nonlinear(12,12),bk(12,12)
      dimension rot_memb(3,3,2,*),ac_member(12,12,*)
      dimension u(12),aly(2),alx(2)
      dimension a_vector(*)
      dimension E_model6_real(*)
c-----------------------------------------------------------------------
c     nx                      :integer  第一ステップか第二ステップか
c     ld_point(*)             :real*8   右辺荷重項
c     Point                   :structure
c     n_point                 :integer  節点数
c     a_vector                :real*8   a ベクトル
c     ac_point(2,n_point)     :real*8  部材減衰行列（釣合系）
c     Newmark_P               :structure
c     Parameter_C             :structure
c     n_damp                  :integer  部材減衰の部材がありか
c-----------------------------------------------------------------------
      if(n_damp.eq.0) return
c-----------------------------------------------------------------------
      do i=1,n_member
      ij = Member(i).nm_damp
      if(ij .ne. 0) then
      do j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)=  a_vector(irest)
      else
      u(j)=0.0
      endif
      end do
```

```
c――――――――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル
      if(Member(i).element_type.eq.6) then
      ien= Member(i).n_model_type
      ie = Member(i).nm_element
      ii=Element(ie).nm_type
      call Cal_nonlin_maxwelldamp(E_model6_real(ien),
     *                         bk,ii)
      call Rotate_K(bk,rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),ac_nonlinear)
c――――――――――――――――――――――――――――――――――――――★剛床変換チェック
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       n_gouyuka=1
        aly(k)= Point(i1).coord_local(1)
        alx(k)= Point(i1).coord_local(2)
       else
        aly(k)=0.
        alx(k)=0.
       endif
      enddo
      if(n_gouyuka.ne.0) call Set_gtrans_k(ac_nonlinear(1,1,i),aly(1),alx(1), aly(2),alx(2) )

c――――――――――――――――――――――――――――――――――――――★その他の減衰部材
      else
      do j=1,12
      do k=1,12
      ac_nonlinear(j,k)=ac_member(j,k,ij)
      enddo
      enddo
      endif
c――――――――――――――――――――――――――――――――
      do  j=1,12
      irest=Member(i).irest(j)
      if(irest.gt.0) then
      sum=0.
      do k=1,12
      sum=sum+ac_nonlinear(j,k)*u(k)
      enddo
      ld_point(irest)=ld_point(irest) - sum
      endif
      end do
      endif
      enddo
c――――――――――――――――――――――――――――――――――――――――――――――――――――

      return
      end
```

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_fdd_ld
C       ――――――――――――――――――――――――――――――――――
```

```
C     ●   Maxwell 減衰項に関するベクトルを加える。(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Add_fdd_ld(ld_point_repeat,E_model6_real,Element,
     *       Member,n_member,est_vel_point,rot_memb,Point)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      record / point_s    / Point
      record / e_model6_real_s / E_model6_real
      record /element_s   / Element
      dimension Member(*),E_model6_real(*),Element(*)
      dimension Point(*)
      real*8 ld_point_repeat(*),est_vel_point(*)
      dimension rot_memb(3,3,2,*)
      dimension av(12),ud(12),vpp(12)
c―――――――――――――――――――――――――――――――――――――――
c     ld_point_repeat(*)    : real*8   線形右辺項ベクトル
c     Member                :structure
c     n_member              :integer   部材数
c     est_vel_point         :real*8    予測節点速度
c―――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      ie = Member(i).nm_element
c――――――――――――――――――――――――――――――――★部材減衰を持っているか
      if( Element(ie).nm_damp .ne. 0) then
      ien= Member(i).n_model_type
c      write(76,'(a,3i4)') ' mem:',i,Member(i).nm_point(1),
c     *                    Member(i).nm_point(2)
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      ud(j)=est_vel_point(irest)
      else
      ud(j)=0.
      endif
      enddo
c――――――――――――――――――――★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,ud(k1),Point(i1).coord_local(1),
     *                                  Point(i1).coord_local(2))
       endif
      enddo
c――――――――――――――――――――★座標変換
c     write(76,'(6e12.4,3x,6e12.4)') ( ud(j),j=1,12)
      call RotateL_v(1,ud,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c     write(76,'(6e12.4,3x,6e12.4)') ( vpp(j),j=1,12)
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL 要素
      if(iett.eq.0)then
```

```
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
c――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル

      goto 100
   12 continue
c――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル

      goto 100
   13 continue
c――――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル

      goto 100
   14 continue
c――――――――――――――――――――――――――――――――★Model_No.4 3 次元ケーブル弾塑性モデル

      goto 100
   15 continue
c――――――――――――――――――――――――――――――――★Model_No.5 3 次元免振モデル

      goto 100
   16 continue
c――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル
      ii=Element(ie).nm_type
      call Cal_force_maxwelldamp(vpp,E_model6_real(ien),av,ii,i)
      goto 100
   17 continue
      ・
      ・
  100 continue
c――――――――――――――――――――――――――――――――★右変更への追加
      call RotateL_v(2,av,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c――――――――――――――――――――――――★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
        k1=6*(k-1)+1
        call Set_gtrans_f(2,vpp(k1),
     *     Point(i1).coord_local(1),Point(i1).coord_local(2))
        endif
      enddo
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      ld_point_repeat(irest)=ld_point_repeat(irest) - vpp(j)
      end if
      end do
      endif
      end do
      return
      end
```

9.5.10.4 動的固有問題解析への組み込み

動的固有問題解析においても、座標変換を行なわなければならない。固有問題解析用の主サブルーチン submain_dynamic_b()で、まず、変更しなければならないサブルーチンを示す。

```
Rotate_stiffness()
Build_sky_mm_E
Rotate_mass()
Out_Eigen
```

ただし、Rotate_stiffness()、Rotate_mass()は既に変更を行っている。

まず、主サブルーチンで挿入するサブルーチンは、以下のようである。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★        構造・荷重データを入力し、データの設定を行う
c
c―――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――★基本構造データを入力(ok)
c      open (damp_out,FILE='EOUTPUT')
      nfix=5
      nfi=1
      call infile(nfi,nfix,ierr)
      if(ierr.ne.0) then
      ierr_dat =10
      return
      endif
c                          write(damp_out,*) ' Get_structure in Ok'
      call Get_structure(Point,Member,Element,Parameter_C,
     *                   Model_type,ierr,
     *                   S_comp_model,E_Fiber_work)
      close(nfix)
c                            write(damp_out,*) ' Get_structure Ok'
      if(ierr .ne. 0) return
c―――――――――――――――――――――――――――――――――――――――
c
c    ★        剛床の設定
c
c―――――――――――――――――――――――――――――――――――――――
      call Set_gouyuka(Parameter_C,Point)


      if(Eigen_d.load_mass .ne. 0.or. Parameter_C.n_gouyuka.ne.0) then
      load_mass_x=1
      else
      load_mass_x=0
      endif
      call Build_sky_mm_E(n_step,gskymm,
     *          Point,n_point,am_point,rot_local,
```

```
     *           n_local_coord,max_h_sky,load_mass_x)
c       write(damp_out,*) ' Build_sky_mm ok'
c     write(76,'(12e12.4)')(gskymm(j),j=1,n_unknown)

c―――――――――――――――――――――――――――――――――――――――――
c
c    ★          サブスペース法の計算用データをセットする
c
c―――――――――――――――――――――――――――――――――――――――――
      nroot = Eigen_d.n_modes
      nc = min(2*nroot,nroot+8)
      nnc = nc*(nc+1)/2
      nnm=Parameter_C.n_unknown+1
      IFSS=0  !Ｓｔｕｒｍ列による検定のためのフラグ 0 : 検定しない
      IFPR=0  !反復の間にプリントするためのフラグ 0 : プリントしない
      if(Eigen_d.load_mass .ne. 0 .or. Parameter_C.n_gouyuka .ne.0 ) then
      nm_skyline=n_skyline
      load_mass=1
      else
      nm_skyline=n_unknown
      load_mass=0
      endif
```

次に、刺激係数を求める式について検討する。第ｉ次モードの刺激係数は、次式で与えられる。

$$\beta_{xi}=\frac{\varphi_i^T MI_x}{\varphi_i^T M\varphi_i};\quad \beta_{yi}=\frac{\varphi_i^T MI_y}{\varphi_i^T M\varphi_i};\quad \beta_{zi}=\frac{\varphi_i^T MI_z}{\varphi_i^T M\varphi_i}$$

上式は全体座標系で表されたものであり、釣合座標系で評価された振動モードや質量行列ではない。ここで、式(9.3)を用いて、上式を釣合座標系に変換する。ただし、以下の式は、各節点について評価する。まず、刺激係数の分母であるが、

$$\varphi_i^T M\varphi_i=\phi_i^T R_u^T MR_u\phi_i$$

となり、釣合座標系で表した質量行列をｍとすると、

$$m=R_u^T MR_u$$

であり、したがって、

$$\varphi_i^T M\varphi_i=\phi_i^T m\phi_i$$

となり、刺激係数の分母は、全体座標系で評価しても、釣合座標系で評価しても同じ値となる。ここで、φ_i は全体座標系の第ｉ次振動モードであり、ϕ_i は釣合座標系におけるそれである。

次に、刺激係数における各分子を検討しよう。まず、x 方向の刺激係

数の分子を座標変換する。

$$\varphi_i^T M I_x = \phi_i^T R_u^T M R_u R_u^{-1} I_x$$

上式で、全体座標系から釣合座標系で評価した質量行列を用いると

$$\varphi_i^T M I_x = \phi_i^T m R_u^{-1} I_x$$

となる。ここで、式(9.6)を用いると、

$$\varphi_i^T M I_x = \phi_i^T m R_f^T I_x$$

であり、この式は、節点で評価した後、和を取って求めても良い。釣合座標系で計算した $\phi_i^T m$ を任意の節点で取り出したものを w_i とすると、

$$\varphi_i^T M I_x = \sum w_i R_f^T I_x$$

となり、これを書き下すと

$$w_i R_f^T I_x = \begin{pmatrix} w_{xi} & w_{yi} & w_{zi} & w_{\theta xi} & w_{\theta yi} & w_{\theta yi} \end{pmatrix} \begin{bmatrix} 1 & & & & & L_y \\ & 1 & & & & -L_x \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$= w_{xi}$$

となる。他のｙ方向、ｚ方向の刺激係数分子も同様となる。ただし、節点毎に上記の分子を計算して和を取ると、他の節点の自由度と同一視する変位は、2 重に評価することになる。そこで、釣合座標系で使用された節点自由度のみ評価するするように **Copy_restraint_point()**サブルーチンを用いて同一視する自由度を取り除く。

次に、変更すべき 2 つのサブルーチンを示すことにする。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Build_sky_mm_E
C       ―――――――――――――――――――――――――――――――――
C     ●     集中質量行列のスカイライン行列への組み込み（固有問題用）
C     ―――――――――――――――――――――――――――――――――
      subroutine Build_sky_mm_E(n_istep,gskym,
     *           Point,n_point, am_point , rot_local,
     *           n_local_coord ,max_h_sky,load_mass)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension am_point(2,*),gskym(*)
```

```
      dimension max_h_sky(0:*),rot_local(3,3,*)
      record / point_s     / Point
      dimension Point(*)
      dimension amm(6,6)
c-----------------------------------------------------------------
c     gskym(n_skyline)          :real*8    スカイライン行列
c     Point                     :structure
c     n_point                   :integer   節点数
c     am_point(2,n_point)       :real*8   節点集中質量
c     rot_local(3,3, *)         :real*8   全体座標系から局所座標への回転行列
c     n_local_coord             :integer    局所座標系を用いている節点数
c     Newmark_P                 :structure
c     max_h_sky(n_unknown+1)    :integer     スカイライン行列の各列の高さ
c     load_mass                 :integer    部材分布質量の使用
c
c     集中系では、座標変換は必要でない。
c
c-----------------------------------------------------------------
      if(n_istep.eq.1) then
      ik=1
      else
      ik=2
      endif

      if(load_mass.eq.0) then
c-------------------------★集中質量系の行列への組み込み
      do i=1,n_point
      am=am_point(ik,i)
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      gskym(irest)=gskym(irest)+am
      endif
      enddo
      enddo
      else
c-------------------------★整合質量系の行列への組み込み
      do i=1,n_point
      am=am_point(ik,i)
c------------------------- 剛床座標変換
      if(Point(i).n_group_gouyuka.ne.0) then
      call Set_gtrans_m(am,amm, Point(i).coord_local(1),
     *                          Point(i).coord_local(2))
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      i3=max_h_sky(irest)
      gskym(i3)=gskym(i3)+am
      endif
      enddo
      irest = Point(i).irest(6)
      if(irest.gt.0) then
      i3=max_h_sky(irest)
      gskym(i3)=gskym(i3)+amm(6,6)
```

```
      endif

      i1= Point(i).irest(1)
      if(i1.gt.0) then
      i2= Point(i).irest(6)
       if(i2.gt.0.and.i2.le.i1) then
       i3=max_h_sky(i1)-(i1-i2)
       gskym(i3)=gskym(i3)+amm(1,6)
       elseif(i2.gt.0.and.i1.le.i2) then
       i3=max_h_sky(i2)-(i2-i1)
       gskym(i3)=gskym(i3)+amm(1,6)
      endif
     endif
      i1= Point(i).irest(2)
      if(i1.gt.0) then
      i2= Point(i).irest(6)
      if(i2.gt.0.and.i2.le.i1) then
      i3=max_h_sky(i1)-(i1-i2)
      gskym(i3)=gskym(i3)+amm(2,6)
      elseif(i2.gt.0.and.i1.le.i2) then
      i3=max_h_sky(i2)-(i2-i1)
      gskym(i3)=gskym(i3)+amm(2,6)
       endif
       endif
     else
     do j=1,3
     irest = Point(i).irest(j)
     if(irest.gt.0) then
     i3=max_h_sky(irest)
     gskym(i3)=gskym(i3)+am
     endif
     enddo
     endif

     end do
     endif
     return
     end
```

```
C    ―――――――――――――――――――――――――――――――――
C    ●   SUBROUTINE /Copy_restraint_point
C    ―――――――――――――――――――――――――――――――――
C    ●    刺激係数を計算するための節点拘束表の設定
C    ―――――――――――――――――――――――――――――――――
     subroutine Copy_restraint_point(Parameter_C,Point,Control,
    *                                Point_rest)
     implicit real*8(A-H,O-Z)
     include "submain.h"
     record /control_s     / Control
     record / parameter_s  / Parameter_C
     record / point_s      / Point
     dimension Point(*)
     integer Point_rest(6,*)
```

```
C     ―――――――――――――――――――――――――――――――――――
      do i=1,Parameter_C.n_point
      do j=1,6
      Point_rest(j,i)=0
      enddo
      enddo
      do ii=1,Parameter_C.n_unknown
      do i=1,Parameter_C.n_point
      do j=1,6
      if(ii.eq.Point(i).irest(j)) then
      Point_rest(j,i)=1
      goto 100
      endif
      enddo
      enddo
  100 continue
      enddo
      write(76,'(//a,2i4)') ' 刺激係数用拘束表 ',Parameter_C.n_point,
     *Parameter_C.n_unknown
      do i=1,Parameter_C.n_point
      write(76,'(7i8)') i,(Point_rest(j,i),j=1,6)
      enddo
      return
      end
```

```
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,Point_rest)
      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(6),yy(6),Shigeki(3),tdisp(6)
      integer Point_rest(6,*)
C     ―――――――――――――――――――――――――――――――――――
      n_point=Parameter_C.n_point
      n_local_coord = Parameter_C.n_local_coord
      n_mode = Eigen_d.n_modes
      load_mass=Eigen_d.load_mass
      n_gouyuka = Parameter_C.n_gouyuka
      if(n_gouyuka .ne. 0) load_mass=1
```

```
      pi = 3.14159265D0
c―――――――――――――――――――――――――★刺激係数＊振動モード
      do i=1,n_point
      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
      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
       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))

   10 enddo
c―――――――――――――――――――――――――★レーリー減衰等に関する係数計算
      A0 = 0.0D0
      A1 = 0.0D0
      n_damp_1=Newmark_P.n_damp_1
      n_damp_2=Newmark_P.n_damp_2
      n_damp_type=Newmark_P.n_damp_type
      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
      if (n_damp_type.eq.1 ) then
      A0 = 2.0*h1*Omega(n_damp_1)
      elseif (n_damp_type.EQ.2 ) then
      A1 = 2.0*h1/Omega(n_damp_1)
      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
  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      write(76,535) n_mode
c――――――――――――――――――――――――――★刺激係数、固有周期等計算
      do 100 imode=1,n_mode
      do i=1,n_unknown
      disp(i) = Eigen_Vector(i,imode)
      enddo
c――――――――――――――――――――――――――★刺激係数のゼロセット
      do j=1,3
      Shigeki(j)=0.
      enddo
      call Multm(load_mass,gskymm,disp,w,max_h_sky,n_unknown)
c――――――――――――――――――――――――――★刺激係数の分母計算
      t_dmd = 0.0
      do i = 1, n_unknown
      t_dmd = t_dmd + disp(i)*w(i)
      enddo
c     write(76,'(a,e16.5)') ' 刺激係数分母',t_dmd
c――――――――――――――――――――――――――★刺激係数
      if(n_local_coord.eq.0.and.n_gouyuka.eq.0) then
         do i= 1, n_point
         do j=1,3
         irest =Point(i).irest(j)
         if(irest.gt.0.and.Point_rest(j,i).ne.0)
     *               Shigeki(j) = w(irest) + Shigeki(j)
         enddo
        enddo
      else
c――――――――――――――――――――――――――★座標変換
        do i= 1, n_point
         local_coord=Point(i).local_coord
         if(local_coord.eq.0.and.Point(i).n_group_gouyuka.eq.0) then
           do j=1,3
           irest =Point(i).irest(j)
           if(irest.gt.0.and.Point_rest(j,i).ne.0)
     *              Shigeki(j) = w(irest) + Shigeki(j)
           enddo
         elseif(Point(i).n_group_gouyuka.ne.0) then
c――――――――――――――――――――――――――★剛床変換
           do j=1,3
           yy(j)=0.
           irest =Point(i).irest(j)
           if(irest.gt.0.and.Point_rest(j,i).ne.0)  yy(j) = w(irest)
           enddo
c          j=6
c          irest =Point(i).irest(j)
```

```
c                                        yy(j)=0.
c          if(irest.gt.0)  yy(j) = w(irest)
c     write(76,'(a,i4,4f12.4)') ' yy2',i,(yy(j),j=1,3),yy(6)
c          call Set_gtrans_f(1,yy, Point(i).coord_local(1),
c     *                      Point(i).coord_local(2) )

           do j= 1, 3
           Shigeki(j) = yy(j) + Shigeki(j)
           enddo
         else
c―――――――――――――――――――――――――★局所座標変換
           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
           if(Point_rest(j,i).ne.0) Shigeki(j) = yy(j) + Shigeki(j)
           enddo
         endif
        enddo

      endif
c     write(76,'(a,3e16.5)') ' 刺激係数分子',(Shigeki(j),j=1,3)
c―――――――――――――――――――――――――★刺激係数の計算
      if(t_dmd.ne.0.)then
      do j=1,3
      Shigeki(j)=Shigeki(j)/t_dmd
      enddo
      endif
c―――――――――――――――――――――――――★刺激係数のチェック
      if(n_local_coord.eq.0 .and. n_gouyuka.eq.0) then
         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
c―――――――――――――――――――――――――★座標変換
      else
        do i= 1, n_point
         local_coord=Point(i).local_coord
         if(local_coord.eq.0.and.Point(i).n_group_gouyuka.eq.0) then
           do j=1,3
            yy(j)=0.
           irest =Point(i).irest(j)
           if(irest.gt.0) yy(j) = disp(irest)
```

```
          enddo
        elseif(Point(i).n_group_gouyuka .ne.0) then
c―――――――――――――――――――――――――★剛床変換
          do j=1,3
          yy(j)=0.
          irest =Point(i).irest(j)
          if(irest.gt.0)  yy(j) = disp(irest)
          enddo
          j=6
          irest =Point(i).irest(j)
           yy(j)=0.
          if(irest.gt.0)  yy(j) =  disp(irest)
           call Set_gtrans_u(1,yy, Point(i).coord_local(1),
     *                       Point(i).coord_local(2) )
        else
c―――――――――――――――――――――――――★局所座標変換
          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
      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)
      endif

      if ( ifl1.EQ.1 ) then
c―――――――――――――――――――――――――――――★振動モードの出力
```

```
      write(iflz1,1000) imode
 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
c      write(76,1000) imode
       do i=1,n_unknown
       disp(i) = Eigen_Vector(i,imode)
       enddo

       if(n_local_coord.eq.0.and. n_gouyuka.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
c――――――――――――――――――――――――★座標変換
       else
         do i= 1, n_point
          local_coord=Point(i).local_coord
          if(local_coord.eq.0.and.Point(i).n_group_gouyuka .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)
          elseif(Point(i).n_group_gouyuka.ne.0) then
c――――――――――――――――――――――――★剛床変換
            do j=1,6
            tdisp(j)=0.
            irest =Point(i).irest(j)
            if(irest.gt.0)  tdisp(j) = disp(irest)
            enddo
             call Set_gtrans_u(1,tdisp, Point(i).coord_local(1),
     *                        Point(i).coord_local(2) )
                              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'
      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
      return
      end
```

9.6 塑性変形倍率

9.6.1 はじめに

SPACE（Ver.3.00）では、立体骨組みの静的解析や動的解析を行っている。その際、妥当な耐震性能評価が重要となる。そこで、3 次元構造物の地震応答解析を行い、各部材の塑性変形倍率及び累積塑性変形倍率、すなわち部材に吸収された塑性ひずみエネルギーを評価する。本節で、SPACE で実行している部材レベルの塑性変形倍率及び累積変形倍率の計算手法を定義し、処理方法を解説する。

9.6.2 塑性変形率

断面内の微小部分における塑性変形率は、１軸応力状態で評価すると、

$$\mu = \frac{\varepsilon_x}{\varepsilon_y} \qquad \cdots\cdots\cdots\cdots(9.16)$$

で表される。ここで、ε_x は軸方向ひずみであり、ε_y は降伏点ひずみを表す。しかし、断面全体で塑性変形率を評価することは、かなり困難を伴う。

まず、単純曲げの状態について考えよう。この場合、次式のように塑性ヒンジが生じたときの曲率 ϕ_P を用いて評価する。

$$\mu = \frac{\phi}{\phi_P} \qquad \cdots\cdots\cdots\cdots(9.17)$$

線形部分の曲げモーメントと曲率の関係は、

$$M = EI\phi$$

で表されることから、曲率 ϕ_P は次式で与えられる。

$$\phi_P = \frac{M_P}{EI} \qquad \cdots\cdots\cdots\cdots(9.18)$$

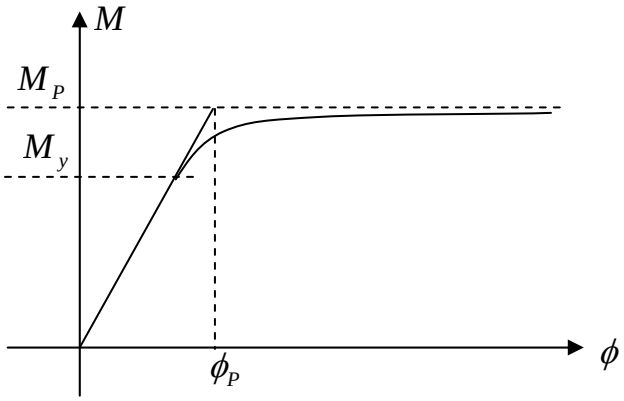


図 9-3　曲げモーメントと曲率の関係

次に、軸力が存在する場合で、しかも 2 軸の曲げモーメントが発生している場合の塑性変形率を検討しよう。SPACE では、塑性論アナロジーモデルの塑性関数として、次の 3 つの関数が用いられている。

$$\left.\begin{aligned} f &= \left(\frac{N}{N_p}\right)^2 + \sqrt{\left(\frac{M_y}{M_{yp}}\right)^2 + \left(\frac{M_z}{M_{zp}}\right)^2} - 1 \\ f &= \left(\frac{N}{N_p}\right)^2 + \left(\frac{M_y}{M_{yp}}\right)^2 + \left(\frac{M_z}{M_{zp}}\right)^2 - 1 \\ f &= \left|\frac{N}{N_p}\right| + \sqrt{\left(\frac{M_y}{M_{yp}}\right)^2 + \left(\frac{M_z}{M_{zp}}\right)^2} - 1 \end{aligned}\right\} \cdots\cdots\cdots\cdots(9.19)$$

上記の３つの関数を応用して、塑性変形倍率 μ を評価する関数を次のように定義する。

$$\left.\begin{aligned} \mu &= \left(\frac{\varepsilon_x}{\varepsilon_y}\right)^2 + \sqrt{\left(\frac{\phi_y}{\phi_{yp}}\right)^2 + \left(\frac{\phi_z}{\phi_{zp}}\right)^2} \\ \mu &= \sqrt{\left(\frac{\varepsilon_x}{\varepsilon_y}\right)^2 + \left(\frac{\phi_y}{\phi_{yp}}\right)^2 + \left(\frac{\phi_z}{\phi_{zp}}\right)^2} \\ \mu &= \left|\frac{\varepsilon_x}{\varepsilon_y}\right| + \sqrt{\left(\frac{\phi_y}{\phi_{yp}}\right)^2 + \left(\frac{\phi_z}{\phi_{zp}}\right)^2} \end{aligned}\right\} \cdots\cdots\cdots\cdots(9.20)$$

ここで、ϕ_{yp} と ϕ_{zp} は、塑性ヒンジが生じたときの各々ｙ軸とｚ軸に関する曲率を表す。

塑性関数が任意断面について、精度良く塑性状態を表すことが可能で

あると、式(9.20)で塑性変形率を精度良く表すことができる。なお、ϕ_{yp} と ϕ_{zp} は、次式で求める。

$$\varepsilon_P = \frac{N_P}{EA};\quad \phi_{yP} = \frac{M_{yP}}{EI_y};\quad \phi_{zP} = \frac{M_{zP}}{EI_z} \qquad \cdots\cdots\cdots\cdots(9.21)$$

9.6.3 累積塑性変形率

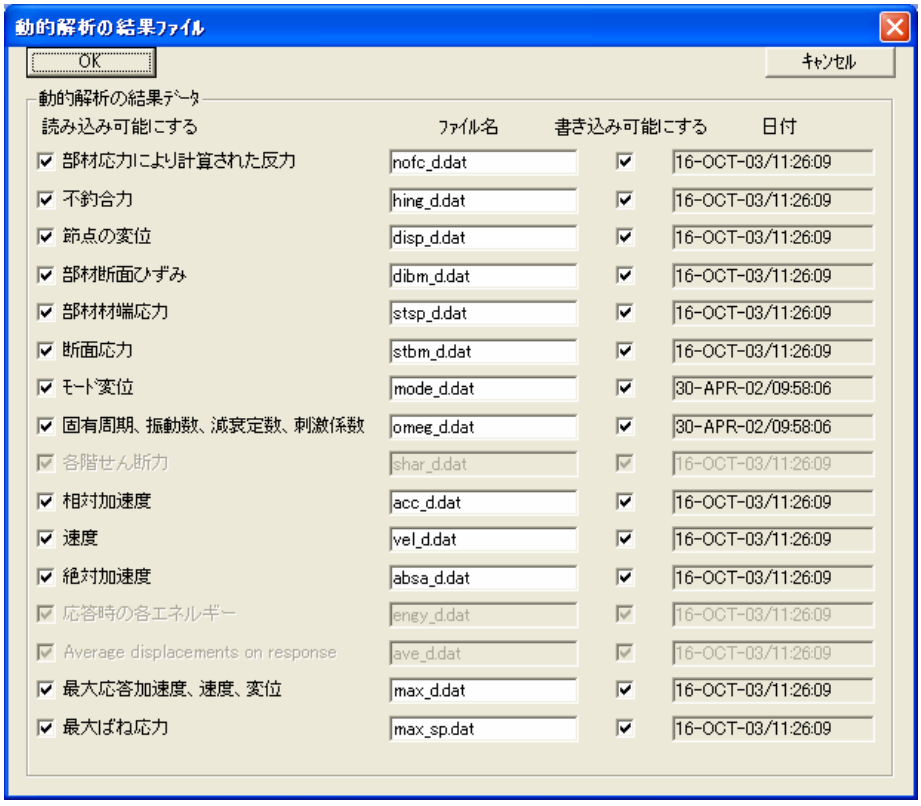
累積塑性変形率は、塑性変形率を利用して、以下のように定義する。

$$\eta = \sum(\mu_{\max} - 1) \qquad \cdots\cdots\cdots\cdots(9.22)$$

ここで、η は累積塑性変形率であり、$\mu_{\max}$ は各振動振幅時における最大塑性変形率を表す。ただし、値が 1 以下の場合、つまり、弾性状態の場合は参入しないものとする。また、累積塑性回数とは、式(9.22)で、η を数えた回数とする。

9.6.4 動的解析におけるひずみの出力

動的解析で、次のサブルーチン Out_Strain()を用いて、部材断面のひずみを出力する。出力するファイルは、以下に示す SPACE のダイアログ



断面内のひずみを出力するファイル

を用いて行う。上記ダイアログの中で、「部材断面ひずみ」が新たに設けられたファイルであり、ここで入出力許可並びにファイル名を設定する。このファイルに与えられたキィワードは、「dibm_d」である。

動的解析における主サブルーチンでは、以下のようにこのサブルーチンをコールする。

```
c------------------------------------------------------------
c
c    ★        解析結果を出力するファイル群をオープンする(ok)
c
c------------------------------------------------------------
      call flcheck(ifl,ifly,iflz,ierr,Control.type_analysis)
       if(ierr.ne.0) then
      ierr_dat =14
       endif
       write(damp_out,*) ' Out_Strain ok' ,n_member
c-----------------------------------★出力指定した断面がファイバー要素
c-----------------------------------★かどうかチェック(ok)
      call out_section_check(Member,Element,
     *               Parameter_C.n_member,No_section,Out_section,
     *               ifl,iflz,i_print,S_comp_model,E_modelx,
     *               M_modelx,E_fiber_work,M_fiber_work)
c       write(damp_out,*) ' out_section_check ok' ,n_member
c-----------------------------------★断面ひずみの個数出力
      call Out_Strain(0,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,
     *               Parameter_C.n_member,ifl,iflz,i_print,
     *               S_comp_model, E_modelx, M_modelx,
     *               E_fiber_work, M_fiber_work)
c       write(damp_out,*) ' Out_Strain ok' ,n_member
       .
       .
c-----------------------------------★断面ファイバー応力を出力
      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,
     *               S_comp_model, E_modelx, M_modelx,
     *               E_fiber_work, M_fiber_work)
c      write(damp_out,*) ' Out_stress ok'
c-----------------------------------★断面ひずみを出力
      call Out_Strain(1,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,
     *              S_comp_model, E_modelx, M_modelx,
     *              E_fiber_work, M_fiber_work)
      c      write(damp_out,*) ' Out_stress ok'
      c――――――――――――――――――――――――――――――――――★応力等の最大値セット
            call Get_max_stress(Member,n_member,Max_stress)
```

最初のサブルーチンコールでは、初期設定を行い、次のサブルーチンコールで部材のひずみがファイルに出力される。次に、このサブルーチンの内容を示そう。

```
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Out_Strain
C       ――――――――――――――――――――――――――――――――――
C     ●   部材の断面ひずみを出力(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine Out_Strain(nx_han,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,
     *            S_comp_model, E_modelx, M_modelx,
     *            E_fiber_work, M_fiber_work)

      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.h"
      include "New_submain.h"
c――――――――――――――――――――――――――――――――――★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
     record / Member_s       / Member
     record / Element_s      / Element
      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)
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work(*)
      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),vfx
      dimension vff(3)
      write(76,'(a,3i4)') ' out_strain',nx_han,ifl(4),n_member
c     i_print         :integer 出力制御変数 0:ファイル出力あり
c-----------------------------★全部材の断面個数検索、出力
      if(nx_han.eq.0.and.ifl(4).ne.0) then
      do i=1,n_member
      vfx=0.
c-------------------------------★断面ひずみの個数
      ie = Member(i).nm_element
      imm= Element(ie).n_element              ! 要素番号
      immm= Member(i).n_model_type              ! モデルタイプ別番号
      iet  = Member(i).element_type
      ax=(Element(ie).A*Element(ie).E)
      if(ax.ne.0.)then
      vff(1)=Element(ie).ANP/ax
      else
      vff(1)=0.
      endif
      ax=(Element(ie).RIy*Element(ie).E)
      if(ax.ne.0.)then
      vff(2)=Element(ie).AMPY/ax
      else
      vff(2)=0.
      endif
      ax=(Element(ie).RIz*Element(ie).E)
```

```
      if(ax.ne.0.)then
      vff(3)=Element(ie).AMPZ/ax
      else
      vff(3)=0.
      endif
c      write(76,'(i4,5f12.4)') i,vfx,(vff(j),j=1,3)
      if(iet.eq.11) then
      vfx=2.1
      elseif(iet.eq.12) then
      vfx=3.1
      elseif(iet.eq.15) then
      vfx=2.1
      elseif(iet.eq.13) then
      vfx=2.1
      elseif(iet.eq.14) then
      vfx=3.1
      elseif(iet.eq.16) then
      vfx=2.1
      elseif(iet.eq.17) then
      vfx=1.1
      elseif(iet.eq.18.or.iet.eq.19) then
      vfx=2.1
      elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
      i_comp= iet - 50                                                    ! 1
      n_out_stress  = S_comp_model(i_comp).n_out_stress
      do m=1, n_out_stress
      kk = S_comp_model(i_comp).n_out_stress_x(m)       ! 出力エレメント番号      3
      if(kk.ne.0) then
      vfx=vfx+1.
      endif
      enddo
      vfx=vfx+0.2
      endif
      n_sec=vfx
      if(n_sec.ne.0) then
      do j=1,3
      if(vff(j).ne.0.) vff(j)=1./vff(j)
      vf(j)=vff(j)
      enddo
      else
      do j=1,3
      vf(j)=0.
      enddo
      endif
      write(iflz(4))vfx,(vf(j),j=1,3)
      enddo
c―――――――――――――――――――――――――――――★断面ひずみ
      else
      if(i_print.ne.0) return
      if(ifl(4).eq.0) return
      do i=1,n_member
c―――――――――――――――――――――――――――――★断面ひずみ
      ie = Member(i).nm_element
      imm= Element(ie).n_element              ! 要素番号
```

```
      immm= Member(i).n_model_type              ! モデルタイプ別番号
      iet  = Member(i).element_type
      if(iet.eq.11) then
c―――――――――――――――――――――――★モデル１１
      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(4))(vf(k),k=1,3)
      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(4))(vf(k),k=1,3)

      elseif(iet.eq.12) then
c―――――――――――――――――――――――★モデル１２
      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(4))(vf(k),k=1,3)
      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(4))(vf(k),k=1,3)
      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(4))(vf(k),k=1,3)

      elseif(iet.eq.15) then
c―――――――――――――――――――――――★モデル１５

      vf(1) = M_model15(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model15(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model15(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model15(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model15(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model15(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif(iet.eq.13) then
c―――――――――――――――――――――――★モデル２１
      vf(1) = M_model21(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model21(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model21(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model21(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model21(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model21(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif(iet.eq.14) then
c―――――――――――――――――――――――★モデル２２
      vf(1) = M_model22(immm).d_epsi_x_1          !  軸方向歪
```

```
      vf(2) = M_model22(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model22(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model22(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model22(immm).d_epsi_x_c          !  軸方向歪
      vf(2) = M_model22(immm).d_epsi_y_c          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_c          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif(iet.eq.16) then
c-----------------------★モデル３１
      vf(1) = M_model31(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model31(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model31(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model31(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model31(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model31(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif(iet.eq.17) then
c-----------------------★モデル３２
      vf(1) = M_model32(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model32(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)
      vf(1) = M_model32(immm).d_epsi_x_c          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_c          !  ｙ軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_c          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif(iet.eq.18.or.iet.eq.19) then
c-----------------------★モデル１３
      vf(1) = M_model13(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model13(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model13(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
      write(iflz(4))(vf(k),k=1,3)

      elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
c-----------------------★新規モデル 51-70
      i_comp= iet - 50                                                  ! 1
      n_out_stress  = S_comp_model(i_comp).n_out_stress
      nmmx= Member(i).n_model_type -1                ! M_Fiber_work の開始番号   2
      nmx = Element(ie).n_section(1) -1              ! E_Fiber_work の開始番号
      do m=1, n_out_stress
      kk = S_comp_model(i_comp).n_out_stress_x(m)      ! 出力エレメント番号       3
      if(kk.ne.0) then
```

```
        immmx= M_Fiber_work(nmmx+kk).nm_section          ! 内部エレメント番号
        vf(1) = M_modelx(immmx).d_epsi_x                 !  軸方向歪
        vf(2) = M_modelx(immmx).d_epsi_y                 !  ｙ軸に関する曲げ歪
        vf(3) = M_modelx(immmx).d_epsi_z                 !  ｚ軸に関する曲げ歪
        write(iflz(4))(vf(k),k=1,3)                      ! ひずみ              ! 9
        endif
        enddo
        endif
c――――――――――――――――――――――――――――――★断面ひずみ出力終わり
        enddo
       endif
       return
       end
```

断面のひずみを出力するファイルの仕様について説明する。断面内のひずみは、全ての部材について出力しているわけではない。部材モデルの弾塑性状態を評価する特殊な位置で、軸方向ひずみ、ｙ軸の曲げひずみ、ｚ軸の曲げひずみを単精度で出力する。さらに、他のデータと同様に、ユーザーが設定した時間ごとに、結果が出力される。したがって、解析増分ごとにひずみが出力されないので注意されたい。

このファイルの最初は、全部材について次の情報が出力される。

1. 当該部材のひずみ出力位置数（ただし、実数で出力）
2. 式(9.21)で計算される塑性ひずみと塑性ヒンジが生じたときの各々ｙ軸とｚ軸に関する曲率の逆数

上記の２．の値は、塑性率を計算するために利用する。1．の値は、次に示すひずみを入力する際用いることになる。ひずみは、次のように部材モデルの弾塑性状態を評価する特殊な位置で、軸方向ひずみ、ｙ軸の曲げひずみ、ｚ軸の曲げひずみを単精度で出力する。

1. 軸方向ひずみ
2. ｙ軸の曲げひずみ
3. ｚ軸の曲げひずみ

9.6.5 ひずみの入力と塑性率計算

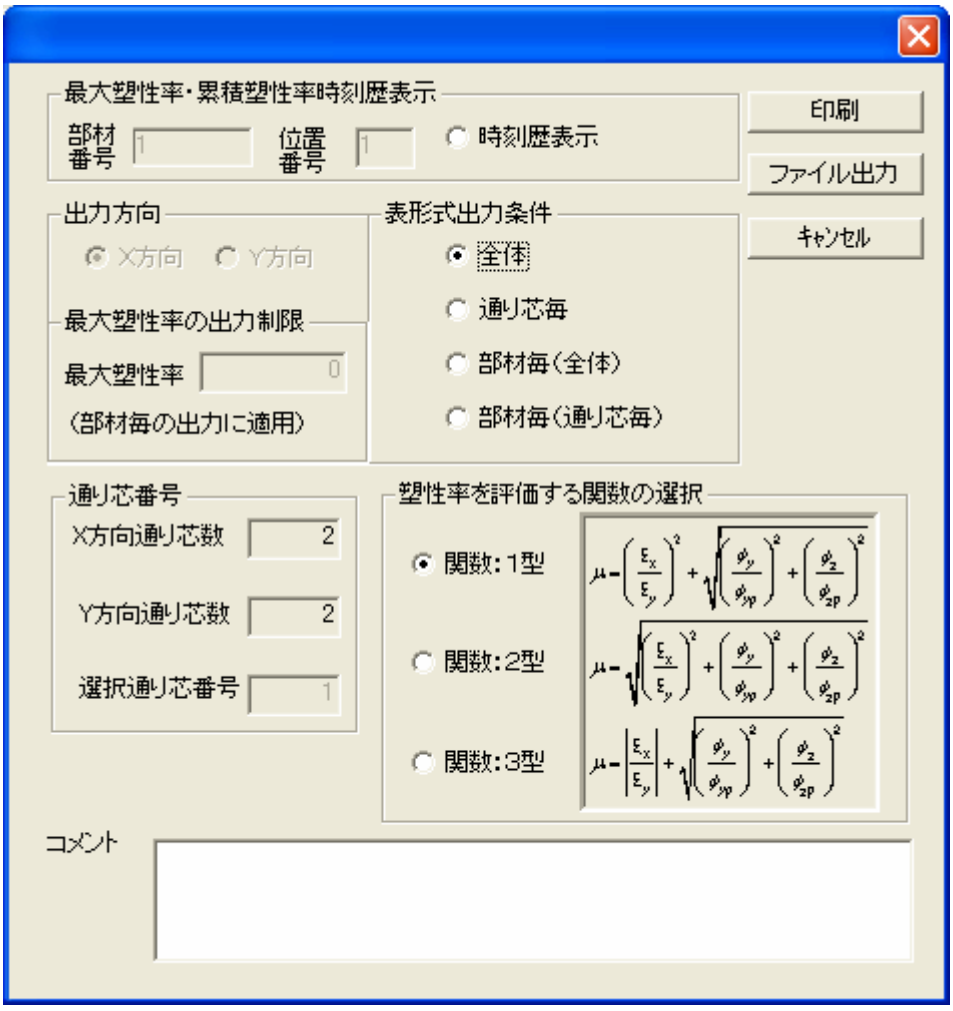
部材ひずみの最大塑性率などの処理は、プレゼンターで行う。最初に、プレゼンターで、各種の処理を行うためのデータ設定を行うダイアログを示す。

ここでは、次に示す５種の最大塑性率などの処理を行う。

1．全フレームについて、最大塑性率、最大累積塑性率など求める。
2．指定したフレーム毎に、最大塑性率、最大累積塑性率など求める。

3．全フレームについて、部材毎に最大塑性率や最大累積塑性率を求める。
4．指定したフレームの全部材について、最大塑性率や最大累積塑性率を求める。
5．指定した部材の塑性率、累積塑性率などを時刻歴で、グラフ表示する。

プレゼンターで使用するダイアログを次に示す。



上記のダイアログでデータを設定した後、動的解析で求めた各時刻におけるひずみを読むことになる。処理プログラムは C++で記述されており、関数 OnfilePrprint()において、書類として出力される 5 つの処理が行われる。この処理の選択は、ユーザーがダイアログで設定するが、プログラムでは、変数 m_radio_jyouken の値を用いて行われる。

```
void CDlg_sosei::OnFilePrPrint()
{
      UpdateData(TRUE);
//-------------------------------------------------------------
//     kai      : 階数
//     ifrm     : フレーム最大数
//     ix       : Ｘ方向フレーム数
//     iy       : Ｙ方向フレーム数
//     kn       : 階名
//     sn       : 層名
//     xn       : Ｘ方向フレーム名
//     yn       : Ｙ方向フレーム名
//     zh       : 階高
//     IERR     : エラー番号  0 : 無   1 : リードプロテクト   2 : ファイルなし
//-------------------------------------------------------------
// info.dat 基本データ読み込み
//        ikn = new char[5*ikai]; //階名
//        isn = new char[5*I_so]; //層名
//        ixn = new char[5*ifrm]; //Ｘフレーム名
//        iyn = new char[5*ifrm]; //Ｙフレーム名
//        izh = new float[ikai];  //階高
switch(m_radio_jyouken){
      case 0://全体（表）

      break;
      case 1://通り芯毎（表）
         if(m_radio_tori == 0){
                  if(m_edit_tori_set > iix || m_edit_tori_set < 1){
      MessageBox("x 方向通り芯番号入力エラー");
      return;
                  }
         }else{
         if(m_edit_tori_set > iiy || m_edit_tori_set < 1){
      MessageBox("y 方向通り芯番号入力エラー");
      return;
                  }
         }
      break;
      case 2://部材毎：全体（表）
      break;
      case 3://部材毎：通り芯毎（表）
         if(m_radio_tori == 0){
                  if(m_edit_tori_set > iix || m_edit_tori_set < 1){
      MessageBox("x 方向通り芯番号入力エラー");
      return;
                  }
         }else{
         if(m_edit_tori_set > iiy || m_edit_tori_set < 1){
      MessageBox("y 方向通り芯番号入力エラー");
      return;
                  }
         }
      break;
      case 4://部材毎：時刻歴（グラフ）
```

```
        break;
        default:
         MessageBox("デフォルト");
}
        int ihan=0;
        int nfl=39;// 部材ひずみファイル番号
         int n_type = m_radio_type;//関数型（in）
         int n_member = mnbsb;//部材数
         float dt =    F_nstep_k;//増分時間
         int n_time =   F_all_step;//増分分割数
         int n_radio_tori = m_radio_tori + 1;//通り芯方向 0:x 方向  1:y 方向
         int n_edit_tori_set = m_edit_tori_set;//通り芯番号
         int n_sosu =I_so;
         int m_radio_J = m_radio_jyouken;
         float m_edit_s_soseix = m_edit_s_sosei;
         int n_mem =  m_edit_n_member;
         int n_ichi=  m_edit_nx_member;

         IP_strain = new int[10*n_member];
       nx_member = new int[n_member];
       P_strain = new double[35*n_member];
       Sec_epsy = new double[3*n_member];
        if(m_radio_jyouken < 4){
      CAL_P_STRAIN(&ihan,&nfl,&n_type,&n_member,P_strain,
            IP_strain,nx_member,Sec_epsy,&dt,&n_time);
            if(ihan == 0 ){
            }else{
        MessageBox("最大塑性率は入力失敗");
        return;
        }}
        if(m_radio_jyouken < 2){
       S_hyou = new float[3*6*n_sosu];
       IS_hyou = new int[3*13*n_sosu];
        }
        if(m_radio_jyouken > 1 && m_radio_jyouken < 4){
       S_hyou = new float[3*n_member];
       IS_hyou = new int[6*n_member];
        }
        if(m_radio_jyouken == 4){
       S_hyou = new float[5*n_time];
       IS_hyou = new int[n_time];
        }
        int i_out=0;
        char path_name;
        int istrlen;
switch(m_radio_jyouken){
        case 0://全体（表）
      SET_HYO_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
            F_iso,P_strain,IP_strain,
            nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&n_sosu);
        I_print = 13;//最大塑性率
        break;
        case 1://通り芯毎（表）
```

```
        I_print = 13;//最大塑性率
    SET_HYO_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
          F_iso,P_strain,IP_strain,
           nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&n_sosu);
        break;
        case 2://部材毎：全体（表）
        I_print = 14;//最大塑性率
    SET_MEMBER_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
          F_iso,P_strain,IP_strain,
           nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&m_edit_s_soseix);
        break;
        case 3://部材毎：通り芯毎（表）
        I_print = 14;//最大塑性率
    SET_MEMBER_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
          F_iso,P_strain,IP_strain,
           nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&m_edit_s_soseix);
        break;
        case 4://部材時刻歴（グラフ）
        I_print = 15;//最大塑性率
    CAL_MEMBER_P_STRAIN(&i_out,&path_name,&istrlen,&n_mem,&n_ichi,&ihan,
         &nfl,&n_type,&n_member,S_hyou,nx_member,Sec_epsy,&n_time,&dt);
        break;
}
        CDialog::OnOK();
}
```

上記プログラムでは、最初に、ダイアログで入力したデータに矛盾がないかチェックした後、部材断面のひずみを入力し、各部材の最大塑性率を求めるサブルーチン Cal_P_Strain()をコールする。このサブルーチンは、処理１から４の場合使用され、塑性率などの時刻歴表示の処理５ではコールしない。

次に、これらの処理に必要となるワーク領域を動的に確保する。その後、５つの処理に分類して、書類として出力する表や図形用データを求める。これらの処理は次の３つのサブルーチンによって行われる。

1. SET_HYO_STRAIN（処理１、処理２）
2. SET_MEMBER_STRAIN（処理３、処理４）
3. CAL_MEMBER_P_STRAIN（処理５）

最初に、サブルーチン Cal_P_Strain()を検討しよう。このサブルーチンでは、まず、全部材について、ひずみの出力個数と塑性ひずみと塑性ヒンジが生じたときの各々ｙ軸と ｚ 軸に関する曲率の逆数を入力し、配列 nx_member と Sec_epsy にセットする。次に、処理結果を設定する配列 P_strain と IP_strain をゼロセットする。これ以降、出力時刻毎

にひずみを入力した後、サブルーチン check_strain()を用いて、最大塑性率などをチェックする。

サブルーチン check_strain()では、ひずみを各塑性ひずみで割り、軸方向、２つの曲げに関する塑性率を計算する。この値を元にユーザーが選択した計算式を用いて、断面塑性率を求める。次からは、最大塑性率、累積塑性率、累積損傷回数を計算する。

それでは、この２つのサブルーチンを示そう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE Cal_P_Strain
C       ―――――――――――――――――――――――――――――――――
C     ●   各部材の塑性率と最大累積塑性率の計算
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_P_Strain(ihan,nfl,n_type,n_member,P_strain,
     *                        IP_strain,nx_member,Sec_epsy,dt,n_time)
      implicit real*8(A-H,O-Z)
      common /sf01/fnfile,jdfile,kdfile,iidat,title,lengf,ltitle,timex
      CHARACTER fnfile(60)*50,title*50,timex*20(60)
      integer*4 jdfile(60),kdfile(60),lengf(60)
      dimension P_strain(7,5,*), IP_strain(2,5,*), Sec_epsy(3,*),
     *          nx_member(*)
      real*4  eps(3),vf,dt
c―――――――――――――――――――――――――――――★
c     nfx:              ひずみ入力ファイル番号
c     n_time:           時間個数
c     dt:               増分時間
c     n_type:           塑性率を計算する型
c     n_member:         部材数
c     P_strain:         1：最大塑性率　2：時刻　3:最大累積塑性率　4:増分前の塑性率
c     IP_strain:        1：累積回数　　2：回数のためのワーク領域
c     nx_member:        部材内塑性計算を行う個数
c     Sec_epsy:         塑性時ひずみの逆数
c―――――――――――――――――――――――――――――★断面個数入力
      ihan=1
      nfx=98
      open(unit=nfx,file=fnfile(NFL),FORM='UNFORMATTED',
     *    status='old',err=199)
c      write(77,'(a,2i4,f12.4)')fnfile(NFL),n_member,n_time,dt
      do i=1, n_member
      read(nfx,end=1000) vf,(eps(j),j=1,3)
c     write(77,'(i4,4f12.4)') i,vf,(eps(j),j=1,3)
      nx_member(i)=vf
      do j=1,3
      Sec_epsy(j,i)=eps(j)
      enddo
      do k=1,5
       do j=1,7
       P_strain(j,k,i)=0.
       enddo
       do j=1,2
```

```
        IP_strain(j,k,i)=0.
       enddo
      enddo
     enddo
c―――――――――――――――――――――――――――――★断面個数入力
      tt=0.
      do nt=1, n_time
      tt=dt*nt
        do i=1,n_member
        n_sec= nx_member(i)
        if(n_sec.ne.0) then
         do j=1,n_sec
           read(nfx,end=1001) (eps(k),k=1,3)
c     write(77,'(3i4,4f12.4)') nt,i,j,(eps(k),k=1,3)
          call check_strain(n_type ,tt,eps, Sec_epsy(1,i),
     *      P_strain(1,j,i),IP_strain(1,j,i),i)
         enddo
        endif
       enddo
      enddo
 1000 continue
      close(nfx)
c     write(77,'(a)')' shuuryou '
      ihan=0
      return
 1001 continue
      close(nfx)
c     write(77,'(a)')' データ不足 '
      ihan=0
      return
 199  continue
      return
      end
```

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / check_strain
C       ―――――――――――――――――――――――――――――――――
C     ●   塑性率と最大累積塑性率の計算
C     ―――――――――――――――――――――――――――――――――
      subroutine check_strain(n_type,tt,eps,Sec_epsy,P_strain,
     *    IP_strain,nm)
      implicit real*8(A-H,O-Z)
      dimension P_strain(7),Sec_epsy(3),IP_strain(2),ss(3)
      real*4 eps(3)
      do i=1,3
      ss(i)=eps(i)*Sec_epsy(i)
      P_strain(i+4)=eps(i)
      enddo
      goto(11,12,13), n_type
11    continue
      P_strainx= ss(1)**2+dsqrt(ss(2)**2 + ss(3)**2)
      goto 100
12    continue
      P_strainx= dsqrt(ss(1)**2+ ss(2)**2 + ss(3)**2)
```

```
      goto 100
13    continue
      P_strainx= dabs(ss(1))+dsqrt(ss(2)**2+ ss(3)**2)
      goto 100
100   continue
c―――――――――――――――――――――――――――――――★最大塑性率
      if(P_strainx.gt. P_strain(1))then
      P_strain(1) = P_strainx
      P_strain(2) = tt
      endif
c―――――――――――――――――――――――――――――――★累積塑性率
      if(P_strainx .ge. P_strain(4)) then
c―――――――――――――――――――――――――――――――★ひずみ増加
       if(P_strain(4).gt.1. .and. P_strainx .gt.1.) then
        P_strain(3)= P_strain(3)+ P_strainx - P_strain(4)
       if(IP_strain(2).le.0) IP_strain(2)=0
        IP_strain(2)= IP_strain(2)+1
       elseif(P_strainx .gt.1.) then
        P_strain(3)= P_strain(3)+ P_strainx - 1.
       if(IP_strain(2).le.0) IP_strain(2)=0
        IP_strain(2)= IP_strain(2)+1
       endif
c―――――――――――――――――――――――――――――――★ひずみ減少
      else
      if(P_strain(4).gt.1.) then
      if(IP_strain(2).gt.0.) then
c―――――――――――――――――――――――――――――――★累積損傷回数チェック
        IP_strain(1)= IP_strain(1)+1
        IP_strain(2)=-1
        else
        IP_strain(2)= IP_strain(2)-1
        endif
       endif

      endif
      if(nm.eq.18)then
c      write(77,'(a,3i4,4f12.5)') ' 累積 ',nm,IP_strain(1),IP_strain(2),
c     *  P_strain(4),P_strainx
      endif
      P_strain(4) = P_strainx
      return
      end
```

上記のサブルーチンで計算した各部材の最大塑性率、最大累積塑性率などを用いて、出力用の表をセットする。

サブルーチン Set_Hyo_strain()では、まず、表となる配列 s_hyou と is_hyou をゼロセットする。次に、全部材について、サブルーチン Cal_hyo_strain()を用いて塑性率などの最大値をチェックし、表内の適切な位置にセットする。ここで、当該部材内でひずみが出力されている

か、配列 nx_member を用いてチェックし、出力していない場合は、その部材をスキップする。サブルーチン check_floor()を用いて、当該部材のコード iso(i)から部材種別（はり、柱、ブレースなど）や層、通り芯番号などを取得する。この情報を元に得られた最大値を表にセットすることになる。最後は、得られた表を指定したファイルに出力する。

サブルーチン Cal_Hyo_strain()では、各層、部材種別毎に、塑性率などの最大値をチェックする。

ここで説明した３つのサブルーチンの内容を以下に示す。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / Set_Hyo_strain
C       ―――――――――――――――――――――――――――――――
C     ●   塑性率と最大累積塑性率の表作成
C     ―――――――――――――――――――――――――――――――
      subroutine Set_Hyo_strain(i_out,ifn,ilen,
     *                          m_radio_J,n_member,iso,P_strain,
     *                          IP_strain,nx_member,S_hyou,IS_hyou,
     *                           n_radio_tori,n_edit_tori_set,jjso)
      implicit real*8(A-H,O-Z)
      dimension P_strain(7,5,*), IP_strain(2,5,*),
     *          nx_member(*),iso(*)
      dimension IS_hyou(3,13,*)
      real*4 S_hyou(3,6,*)
      character FN*200,fnn*1(200)
      integer*1 ifn(200)
      integer*4 ilen
      equivalence (fnn,fn)

      do i=1,jjso
      do j=1,6
      do k=1,3
      s_hyou(k,j,i)=0.
      enddo
      enddo
      enddo
      do i=1,jjso
      do j=1,13
      do k=1,3
      is_hyou(k,j,i)=0
      enddo
      enddo
      enddo

      do i=1,n_member
      if(nx_member(i).ne.0) then
         call check_floor(iso(i),ix,iy,ibb,jso,itype)
      do j=1,nx_member(i)
c     write(77,'(7i4)') i,j,jso,itype,ix,iy,ibb
      call Cal_hyo_strain(i,j,m_radio_J,ix,iy,itype,
     *                    n_radio_tori,n_edit_tori_set,
```

```
     *                    P_strain(1,j,i),IP_strain(1,j,i),
     *                    IS_hyou(1,1,jso),S_hyou(1,1,jso))
      enddo
      endif
      enddo
c-----------------------------★ファイル出力
      if(i_out.eq.1) then
      fn=' '
      do i=1,ilen
      fnn(i)=char(ifn(i))
      enddo
      open(77,file=FN,err=810)

      write(77,'(a,3i4)') ' 通り芯番号',m_radio_J,
     *     n_radio_tori,n_edit_tori_set
      write(77,'(/a/a,a)') ' 最大塑性率',
     *'                柱                                        はり'
     *,'                                    ブレース'
      do i=jjso,1,-1
      write(77,'(i4,3(f10.3,a,f8.3,a,i5,i2,2x,2i3,3x))')
     *    i,((S_hyou(k,1,i),' (',
     *    S_hyou(k,2,i),' )',
     *    IS_hyou(k,8,i),IS_hyou(k,9,i),
     *    IS_hyou(k,1,i),IS_hyou(k,2,i)),k=1,3)
      enddo
      write(77,'(/a/a,a)') ' 最大累積塑性率',
     *'                柱                              はり',
     *'                     ブレース'
      do i=jjso,1,-1
      write(77,'(i4,3(f10.3,i5,i2,2x,2i3,3x))') i,((S_hyou(k,3,i),
     *    IS_hyou(k,10,i),IS_hyou(k,11,i),
     *    IS_hyou(k,3,i),IS_hyou(k,4,i)),k=1,3)
      enddo
      write(77,'(/a/a,a)') ' 最大累積回数',
     *'              柱                    はり',
     *'               ブレース'
      do i=jjso,1,-1
      write(77,'(i4,3(i4,i5,i2,2x,2i3,3x))') i,((IS_hyou(k,5,i),
     *    IS_hyou(k,12,i),IS_hyou(k,13,i),
     *    IS_hyou(k,6,i),IS_hyou(k,7,i)),k=1,3)
      enddo
      close(77)
      return
  810 continue
      write(77,'(a)') ' 入力ファイル名エラー'
      endif

      return
      end
```

```
C     ------------------------------------
C     ●   SUBROUTINE / Cal_Hyo_strain
C       ------------------------------------
```

```
C     ●　塑性率と最大累積塑性率の表作成
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_hyo_strain(i_mem,j_mem,m_radio_J,ix,iy,itype,
     *                    n_radio_tori,n_edit_tori_set,
     *                    P_strain,IP_strain,IS_hyou,S_hyou)
      implicit real*8(A-H,O-Z)
      dimension P_strain(7),IP_strain(2)
      dimension IS_hyou(3,13)
      real*4    S_hyou(3,6)
C     ―――――――――――――――――――――――――――――――――
c   i:1 はしら  2:はり 3:ブレース
c   S_hyou(i,1):最大塑性率
c   S_hyou(i,2):その時刻
c   S_hyou(i,3):最大累積塑性率
c   IS_hyou(i,1):最大塑性率発生 x 方向通り芯番号
c   IS_hyou(i,2):最大塑性率発生 y 方向通り芯番号
c   IS_hyou(i,3):最大累積塑性率発生 x 方向通り芯番号
c   IS_hyou(i,4):最大累積塑性率発生 y 方向通り芯番号
c   IS_hyou(i,5):最大累積塑性回数
c   IS_hyou(i,6):最大累積塑性回数発生 x 方向通り芯番号
c   IS_hyou(i,7):最大累積塑性回数発生 y 方向通り芯番号
c
c
      if(m_radio_J.eq.0)then
c     write(77,'(3i4,3f12.3,i4)') itype,ix,iy,P_strain(1),
c     *  P_strain(2),P_strain(3),IP_strain(1)
C     ―――――――――――――――――――――――全部材について
      goto(11,12,12,14),itype
C     ―――――――――――――――――柱
  11  continue
      ihashira=1
C     ―――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ―――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
c     *   S_hyou(ihashira,3)
```

```
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return
C     ――――――――――――――――――はり
  12  continue
      ihashira=2
C     ――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
c     *   S_hyou(ihashira,3)
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return

C     ――――――――――――――――――ブレース
  14  continue
      ihashira=3
```

```
C     ――――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ――――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
c     *   S_hyou(ihashira,3)
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return
C     ――――――――――――――――――――――――通り芯について
      else
       if(n_radio_tori .eq. 1 .and. ix .eq. n_edit_tori_set) goto 110
       if(n_radio_tori .eq. 2 .and. iy .eq. n_edit_tori_set) goto 110
      return
 110  continue
c     write(77,'(3i4,3f12.3,i4)') itype,ix,iy,P_strain(1),
c     *  P_strain(2),P_strain(3),IP_strain(1)
C     ――――――――――――――――――――――――全部材について
      goto(21,22,22,24),itype
C     ――――――――――――――――――柱
  21  continue
      ihashira=1
C     ――――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
```

```
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ――――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
c     *   S_hyou(ihashira,3)
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return
C     ――――――――――――――――――はり
   22 continue
      ihashira=2
C     ――――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ――――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
```

```
c     *   S_hyou(ihashira,3)
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return
C     ――――――――――――――――――ブレース
  24  continue
      ihashira=3
C     ――――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(ihashira,1))then
      S_hyou(ihashira,1)=P_strain(1)
      S_hyou(ihashira,2)=P_strain(2)
      IS_hyou(ihashira,1)=ix
      IS_hyou(ihashira,2)=iy
      S_hyou(ihashira,4)=P_strain(5)
      S_hyou(ihashira,5)=P_strain(6)
      S_hyou(ihashira,6)=P_strain(7)
      IS_hyou(ihashira,8)=i_mem
      IS_hyou(ihashira,9)=j_mem
c     write(77,'(a,i4,3f12.3)')' 塑性率',ihashira,
c     *   S_hyou(ihashira,1),S_hyou(ihashira,2)
      endif
C     ――――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(ihashira,3))then
      S_hyou(ihashira,3)=P_strain(3)
      IS_hyou(ihashira,3)=ix
      IS_hyou(ihashira,4)=iy
      IS_hyou(ihashira,10)=i_mem
      IS_hyou(ihashira,11)=j_mem
c     write(77,'(a,i4,3f12.3)')' 累積率',ihashira,
c     *   S_hyou(ihashira,3)
      endif
C     ――――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(ihashira,5))then
      iS_hyou(ihashira,5)=IP_strain(1)
      IS_hyou(ihashira,6)=ix
      IS_hyou(ihashira,7)=iy
      IS_hyou(ihashira,12)=i_mem
      IS_hyou(ihashira,13)=j_mem
c     write(77,'(a,2i4)')' 塑性率',ihashira,
c     *   iS_hyou(ihashira,5)
      endif
      return
      endif
      return
      end
```

```
C     ―――――――――――――――――――――――――――――――――
*         SUBROUTINE check_floor                          *
C     ―――――――――――――――――――――――――――――――――
      subroutine check_floor(iso,ix,iy,ibb,jso,itype)
      ix=0    !ｘ方向通り番号
      iy=0    !ｙ方向通り番号
      ibb=0   !部材内通り番号
      jso=0   !層
      itype= 0!部材型
      if(iabs(iso).lt.100000) return
      iiso = iso/10000000
      goto(9,10,11,12,13,14),iiso+4
   9  continue
c                  せん断
      itype= 6
      goto 100

  10  continue
c                  ブレース
      itype= 4
      goto 100
  11  continue
c                 柱
      itype= 1
      goto 100
  12  continue
c                  制振ダンパー
      itype= 5
      goto 100
  13  continue
c                  ｘ方向梁
      itype= 2
      goto 100
  14  continue
c                  ｙ方向梁
      itype= 3
      goto 100

 100  continue
      iso1=iabs(iso)
      iiso=iso1-(iso1/10000000)*10000000
      jso=iiso/100000
      iiso=iiso-jso*100000
      iy = iiso/1000
      iiso= iiso - iy*1000
      ix = iiso/10
      ibb= iiso - ix*10

      return
      end
```

次は、全フレーム、あるいはユーザーが指定したフレームについて、部材毎の最大塑性率、最大累積塑性率などを求めるサブルーチンを示す。ここでは、ダイアログを用いて、最大塑性率の最低値を指定することで、出力部材の数を少なくすることができる。処理内容は、上記のサブルーチンとほとんど同じであり、理解することは難しくないであろう。

該当するサブルーチン Set_member_strain()と Cal_member_strain()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / Set_member_strain
C       ―――――――――――――――――――――――――――――――
C     ●   塑性率と最大累積塑性率の表作成
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_member_strain(i_out,ifn,ilen,
     *                         m_radio_J,n_member,iso,P_strain,
     *                         IP_strain,nx_member,S_hyou,IS_hyou,
     *                          n_radio_tori,n_edit_tori_set,m_edit_s)
      implicit real*8(A-H,O-Z)
      dimension P_strain(7,5,*), IP_strain(2,5,*),
     *          nx_member(*),iso(*)
      dimension IS_hyou(6,*)
      real*4 S_hyou(3,*),m_edit_s
      character FN*200,fnn*1(200)
      integer*1 ifn(200)
      integer*4 ilen
      equivalence (fnn,fn)

      do i=1,n_member
      do j=1,3
      s_hyou(j,i)=0.
      enddo
      enddo
      do i=1,n_member
      do j=1,6
      is_hyou(j,i)=0
      enddo
      enddo
      write(77,'(a,3i4)') ' 通り芯番号',m_radio_J,
     *     n_radio_tori,n_edit_tori_set
      do i=1,n_member
      if(nx_member(i).ne.0) then
         call check_floor(iso(i),ix,iy,ibb,jso,itype)
       IS_hyou(6,i)=itype
       IS_hyou(3,i)=ibb
       if(m_radio_J .eq. 2) goto 110      ! 全部材出力
       if(n_radio_tori .eq. 1 .and. ix .eq. n_edit_tori_set) goto 110 ! 通り芯出力
       if(n_radio_tori .eq. 2 .and. iy .eq. n_edit_tori_set) goto 110
        nx_member(i)=0
      goto 112
  110 continue
      do j=1,nx_member(i)
```

```
c     write(77,'(7i4)') i,j,jso,itype,ix,iy,ibb
         call Cal_member_strain(nx,j,ix,iy,itype,
     *                      P_strain(1,j,i),IP_strain(1,j,i),
     *                      IS_hyou(1,i),S_hyou(1,i))
      enddo
       IS_hyou(4,i)=nx

      if(m_edit_s.gt.S_hyou(1,i)) nx_member(i)=0  ! 最大塑性率の制限チェック
 112  continue
      endif
      enddo
c-------------------------------★ファイル出力
      if(i_out.eq.1) then
      fn=' '
      do i=1,ilen
      fnn(i)=char(ifn(i))
      enddo
      open(77,file=FN,err=810)
      write(77,'(/a/)')'部材塑性率の表示'
      do j=1,4
      if(j.eq.1) write(77,'(/a/a,a)')'  柱',' 部材番号     ix    iy',
     *'      最大塑性率   発生時刻   累積塑性率   発生回数'
      if(j.eq.2) write(77,'(/a/a,a)')'  x方向はり',
     *'  部材番号     ix    iy',
     *'      最大塑性率   発生時刻   累積塑性率   発生回数'
      if(j.eq.3) write(77,'(/a/a,a)')'  y方向はり',
     *'  部材番号     ix    iy',
     *'      最大塑性率   発生時刻   累積塑性率   発生回数'
      if(j.eq.4) write(77,'(/a/a,a)')'  ブレース',
     *'  部材番号     ix    iy',
     *'      最大塑性率   発生時刻   累積塑性率   発生回数'
      do i=1,n_member
c     write(77,'(4i4)') i,j,IS_hyou(9,i),nx_member(i)
      if(nx_member(i).ne.0.and.j.eq.IS_hyou(6,i)) then
c     if(nx_member(i).ne.0) then

      write(77,'(2i4,3x,3i4,3x,3f10.3,i6)')
     *    i,IS_hyou(4,i),IS_hyou(1,i),IS_hyou(2,i),IS_hyou(3,i),
     *    S_hyou(1,i),S_hyou(2,i),S_hyou(3,i),IS_hyou(5,i)
      endif
      enddo
      enddo
      close(77)
      return
 810  continue
      write(77,'(a)') ' 入力ファイル名エラー'
      endif
      return
      end
```

```
C     ------------------------------------
C     ●   SUBROUTINE / Cal_member_strain
C       ------------------------------------
```

```
C     ●　塑性率と最大累積塑性率の表作成
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_member_strain(nxx,jx,ix,iy,itype,
     *                    P_strain,IP_strain,IS_hyou,S_hyou)

      implicit real*8(A-H,O-Z)
      dimension P_strain(7),IP_strain(2)
      dimension IS_hyou(6)
      real*4    S_hyou(3)
C     ―――――――――――――――――――――――――――――――――
c   i:1 はしら  2:はり 3:ブレース
c   S_hyou(i,1):最大塑性率
c   S_hyou(i,2):その時刻
c   S_hyou(i,3):最大累積塑性率
c   IS_hyou(i,1):最大塑性率発生 x 方向通り芯番号
c   IS_hyou(i,2):最大塑性率発生 y 方向通り芯番号
c   IS_hyou(i,3):最大累積塑性率発生 x 方向通り芯番号
c   IS_hyou(i,4):最大累積塑性率発生 y 方向通り芯番号
c   IS_hyou(i,5):最大累積塑性回数
c   IS_hyou(i,6):最大累積塑性回数発生 x 方向通り芯番号
c   IS_hyou(i,7):最大累積塑性回数発生 y 方向通り芯番号
c
c

C     ―――――――――――――塑性率
      if(P_strain(1).gt.S_hyou(1))then
      S_hyou(1)=P_strain(1)
      S_hyou(2)=P_strain(2)
      IS_hyou(1)=ix
      IS_hyou(2)=iy
      nxx=jx
c     write(77,'(a,2i4,3f12.3)')' 塑性率',jx,itype,
c     *   S_hyou(1),S_hyou(2)
      endif
C     ―――――――――――――累積塑性率
      if(P_strain(3).gt.S_hyou(3))then
      S_hyou(3)=P_strain(3)
c     write(77,'(a,2i4,3f12.3)')' 累積率',jx,itype,
c     *   S_hyou(3)
      endif
C     ―――――――――――――塑性回数
      if(IP_strain(1).gt.IS_hyou(5))then
      iS_hyou(5)=IP_strain(1)
c     write(77,'(a,3i4)')' 回数',jx,itype,
c     *   iS_hyou(5)
      endif

      return
      end
```

次は、部材の塑性率と累積塑性率の時刻履歴を求めるサブルーチンを示す。この部材は、ダイアログで指定することになる。塑性率などと同

様に、３つのひずみの時刻歴も配列 S_hyo にセットする。

サブルーチン Cal_member_P_strain()では、まず、ファイルから、全部材について、ひずみの出力個数と塑性ひずみと塑性ヒンジが生じたときの各々ｙ軸と ｚ 軸に関する曲率の逆数を入力し、配列 nx_member と Sec_epsy にセットする。次に、処理結果を設定する配列 S_hyo をゼロセットする。これ以降、出力時刻毎にひずみを入力した後、サブルーチン check_member_strain ()を用いて、最大塑性率、累世塑性率を計算し、時刻歴として配列 S_hyo にセットする。最後に、ユーザーが指定したファイルにデータを出力する。

ここで、上で説明した２つのサブルーチンを示す。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_member_P_strain
C       ―――――――――――――――――――――――――――――――
C     ●   任意部材の塑性率と最大累積塑性率の履歴
C     ―――――――――――――――――――――――――――――――
      subroutine Cal_member_P_strain(i_out,ifn,ilen,n_mem,n_ichi,ihan,
     *    nfl,n_type,n_member,S_hyo,nx_member,Sec_epsy,n_time,dt)

      implicit real*8(A-H,O-Z)
      common /sf01/fnfile,jdfile,kdfile,iidat,title,lengf,ltitle,timex
      CHARACTER fnfile(60)*50,title*50,timex*20(60)
      integer*4 jdfile(60),kdfile(60),lengf(60)
      dimension Sec_epsy(3,*),nx_member(*),ssx(3)
      real*4  eps(3),vf,S_hyo(5,*),dt
      character FN*200,fnn*1(200)
      integer*1 ifn(200)
      integer*4 ilen
      equivalence (fnn,fn)
c―――――――――――――――――――――――――――――★
c     n_mem:           計算したい部材
c     n_ichi:          部材の位置
c     nfx:             ひずみ入力ファイル番号
c     n_time:          時間個数
c     n_type:          塑性率を計算する型
c     n_member:        部材数
c     S_hyo:        1：最大塑性率　2:最大累積塑性率
c     nx_member:       部材内塑性計算を行う個数
c     Sec_epsy:        塑性時ひずみの逆数
c―――――――――――――――――――――――――――――★断面個数入力
      ihan=1
      nfx=98
      open(unit=nfx,file=fnfile(NFL),FORM='UNFORMATTED',
     *    status='old',err=199)
      ihan=2
      if(n_mem.le.0 .or. n_mem .gt. n_member) return
      do i=1, n_member
      read(nfx) vf,(eps(j),j=1,3)
      do j=1,3
```

```
      Sec_epsy(j,i)=eps(j)
      enddo
      nx_member(i)=vf
      enddo
      ihan=1
      if(nx_member(n_mem).lt. n_ichi) return
      do i=1, n_time
      do j=1,2
      S_hyo(j,i)=0.
      enddo
      enddo

      ihan=0
c-----------------------------★断面個数入力
      S_hyox=0.
      S_hyoy=0.
      do nt=1, n_time
        do i=1,n_member
        n_sec= nx_member(i)
        if(n_sec.ne.0) then
         do j=1,n_sec
           read(nfx,end=1000) (eps(k),k=1,3)
          if(n_mem .eq.i .and. n_ichi .eq. j) then
           call check_member_strain(n_type , eps, Sec_epsy(1,i),
     *                              S_hyo(1,nt), S_hyox,S_hyoy)
           S_hyox= S_hyo(1,nt)
           S_hyoy= S_hyo(2,nt)
          do k=1,3
          S_hyo(K+2,nt)=eps(k)*Sec_epsy(1,i)
          enddo
           endif
         enddo
        endif
       enddo
      enddo
 1000 continue
c-----------------------------★ファイル出力
      if(i_out.eq.1) then
      fn=' '
      do i=1,ilen
      fnn(i)=char(ifn(i))
      enddo
      open(77,file=FN,err=199)

      write(77,'(a,i4,a,i4,a,i4)')
     * ' 時刻歴出力:部材番号：',n_mem,'  位置：',
     * n_ichi,'  分割数：',n_time
      do j=1,3
      ssx(j)=0.
      if( Sec_epsy(j,n_mem).ne.0.) ssx(j)=1./Sec_epsy(j,n_mem)
      enddo
      write(77,'(/a,f12.8,a,f12.8,a,f12.8/)')
     * ' εy:',ssx(1),' Φyp:',ssx(2),' Φzp:',ssx(3)
      write(77,'(a,a)')
```

```
     *'        時刻       塑性率    累積塑性率',
     *'      εx         Φy         Φz'
      do i=1,n_time
      tt=dt*i
      write(77,'(f12.4,2f12.5,3f12.8)') tt,(S_hyo(j,i),j=1,5)
      enddo
      close(77)
      return
  199 continue
      write(77,'(a)') ' 入力ファイル名エラー'
      endif
      return
      end
```

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / check_strain
C       ―――――――――――――――――――――――――――――――――――
C     ●   塑性率と最大累積塑性率の計算
C     ―――――――――――――――――――――――――――――――――――
      subroutine check_member _strain(n_type,eps,Sec_epsy,
     *                                S_hyo, S_hyox,S_hyoy)
      implicit real*8(A-H,O-Z)
      dimension Sec_epsy(3), ss(3)
      real*4 eps(3),S_hyo(5)
      do i=1,3
      ss(i)=eps(i)*Sec_epsy(i)
      enddo
      goto(11,12,13), n_type
11    continue
      S_hyo(1) = ss(1)**2+dsqrt(ss(2)**2 + ss(3)**2)
      goto 100
12    continue
      S_hyo(1) = dsqrt(ss(1)**2+ ss(2)**2 + ss(3)**2)
      goto 100
13    continue
      S_hyo(1) = dabs(ss(1))+dsqrt(ss(2)**2+ ss(3)**2)
      goto 100
100   continue
      S_hyo(2)=S_hyoy
c―――――――――――――――――――――――――――――★累積塑性率計算
      if(S_hyo(1).gt. S_hyox) then
       if(S_hyox.gt.1. .and. S_hyo(1).gt.1.) then
        S_hyo(2)= S_hyo(2)+ S_hyo(1) - S_hyox
       elseif(S_hyo(1).gt.1.) then
        S_hyo(2)= S_hyo(2)+ S_hyo(1) - 1.
      endif
      endif
      return
      end
```

前節で説明した 5 つの処理が全て終了すると、CDialog::OnOK()関数がコールされる。

```
   SET_MEMBER_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
         F_iso,P_strain,IP_strain,
          nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&m_edit_s_soseix);
      break;
      case 3://部材毎：通り芯毎（表）
      I_print = 14;//最大塑性率
   SET_MEMBER_STRAIN(&i_out,&path_name,&istrlen,&m_radio_J,&n_member,
         F_iso,P_strain,IP_strain,
          nx_member,S_hyou,IS_hyou,&n_radio_tori,&n_edit_tori_set,&m_edit_s_soseix);
      break;
      case 4://部材時刻歴（グラフ）
      I_print = 15;//最大塑性率
   CAL_MEMBER_P_STRAIN(&i_out,&path_name,&istrlen,&n_mem,&n_ichi,&ihan,
        &nfl,&n_type,&n_member,S_hyou,nx_member,Sec_epsy,&n_time,&dt);
      break;
}
      CDialog::OnOK();
```

関数 Cdialog::OnOK()が実行されると、後で示す CSf40View::OnDraw ()が自動的にコールされることになる。関数 OnDraw()関数が実行され、最初に、関数 void CSf40View::OnDlgPrint()がコールされる。この関数によって、プリンター出力の準備が行われる。まず、この関数を見てみよう。この関数では、変数 I_print の値によって、異なるメッセージが発せられる。ひとつは ID_FILE_PRINT であり、他は ID_FILE_PRINT_PREVIEW である。前者は、直接プリンターに書類が出力される。後者は、プレビューで出力書類が画面に表示されることになる。

```
 void CSf40View::OnDlgPrint()
 {
     if(dlg_print.DoModal() == IDOK){
     if(I_print < 100){
 //AfxGetMainWnd()->PostMessage(WM_COMMAND,ID_FILE_PRINT);
     Flag_dialog = 1;
 //  if(I_print == 10 || I_print == 11|| I_print == 13|| I_print == 14|| I_print == 15){
     if(I_print == 10 || I_print == 11){
  AfxGetMainWnd()->PostMessage(WM_COMMAND,ID_FILE_PRINT);
 //I_print = 0;
     }
     else{
  AfxGetMainWnd()->PostMessage(WM_COMMAND,ID_FILE_PRINT_PREVIEW);
 //I_print = 0;
     }
     }
     }
     else{
```

```
    AfxGetMainWnd()->PostMessage(WM_CLOSE);
    }
}
```

メッセージは次のメッセージマップによって関数 OnPreparePrinting() が自動的に実行され、プリンターの準備が行われることになる。

```
BEGIN_MESSAGE_MAP(CSf40View, CView)
    //{{AFX_MSG_MAP(CSf40View)
    ON_COMMAND(ID_DLG_PRINT, OnDlgPrint)
    ON_COMMAND(ID_JISUU, OnJisuu)
    ON_WM_RBUTTONDOWN()
    ON_WM_LBUTTONDOWN()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONUP()
    ON_COMMAND(ID_BUTTON_PRINT, OnButtonPrint)
    ON_WM_DESTROY()
    ON_COMMAND(ID_BUTTON_PROPERTY, OnButtonProperty)
    ON_WM_LBUTTONDBLCLK()
    //}}AFX_MSG_MAP
    // 標準印刷コマンド
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
//  ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrintPreview)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

プリンターの準備が行われた後、具体的に図や表を出力するルーチンをコールすることになる。最大塑性率などを出力するルーチンは、ケース 13 から 15 でとなる。

```
void CSf40View::OnDraw(CDC* pDC)
{
    CSf40Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    HWND hwnd = this->GetSafeHwnd();

// TODO: この場所にネイティブ データ用の描画コードを追加します。
    if(Flag_dialog == 0){
    Flag_dialog = 1;
    OnDlgPrint();
    }
    if(I_print >= 100){
//  MessageBox("3");
    F_print_hen =0;
    disp_mem_persp(pDC);
    }
    else if(pDC->IsPrinting()){

    switch(I_print){
    case 1://固有値解析結果（表）
```

```
    print_koyu_hyo(pDC);
    break;
  case 2://応答最大層間変形角及び層間変位
    print_souhen_hyo(pDC);
    break;
  case 3://応答最大層せん断力及び層せん断力係数
    print_sousen_hyo(pDC);
    break;
  case 4://応答最大加速度・速度・変位
      print_kasoku_hyo(pDC);
      break;
  case 5://モード図
      if(iim == 1) print_mode_zu(pDC);
  //  else  pre4_disp_mem_persp(hwnd,ii);
      break;
  case 6://応答最大層間変形角・応答最大加速度・応答最大転倒モーメント・応答最大せん
断力のグラフ
    print_graph(pDC);
    break;
  case 7://Ｑ－δ図（静的）
    print_qd(pDC);
    break;
  case 8://制振装置応答最大相対（速度・変位）
    print_vd(pDC);
    break;
  case 9://解析波形とスペクトル
    print_fft_spc(pDC);
     break;
  case 10://応答最大加速度・速度・変位（図）
    disp_mem_persp_pr(hwnd,pDC);
    break;
  case 11://モード図（フレームモデル）
    disp_mem_persp_pr(hwnd,pDC);
    break;
  case 12://伏図・軸組図
    print_fj(pDC);
    break;
  case 13://最大塑性率・最大累積塑性率（全体、通り芯毎）
    print_soseiritu_hyo(pDC);
    break;
  case 14://全部材あるいは通り芯に関する部材毎の最大塑性率・最大累積塑性率
    print_buzai_soseiritu_hyo(pDC);
    break;
  case 15://任意部材の最大塑性率・最大累積塑性率の時刻歴図
    print_soseiritu_graph(pDC);
    break;
    default:
    break;
  }
  }
  }
```

次に、3 つの関数を示す。その内容は、図や表を出力するルーチンで

あり、簡単なので、理解するのは容易であろう。

```
//-------------------------------------------
//  最大塑性率・最大累積塑性率（全体、通り芯毎）
//-------------------------------------------
void CSf40View::print_soseiritu_hyo(CDC *pDC)
{
      CFont NewFont, *oldFont;
      CPen  NewPen,  *oldPen;
      CString m;
//     char buff[300];
      HWND hWnd = this->GetSafeHwnd();
    CRect rcFrame;
      ::GetClientRect(hWnd ,&rcFrame);
    int fontbase= (rectPage.right-rectPage.left)/58;
//                                        データ表示
      int iy=rectPage.top ;
      int ix=rectPage.left ;
      int F_ipy = fontbase*0.9;
      int x1= 12*fontbase;
      int icc=0;
      int iccc=0;
      int i,ic,jc;
      int xxx=1000;
      int n_k = 0;
      int nn_k = 0;
      int n_page=35;
      pDC->SetMapMode(MM_TWIPS);
      NewPen.CreatePen(PS_SOLID,20,RGB(0,0,0));
      oldFont= pr_font_style(pDC, 1);
      oldPen= pDC->SelectObject(&NewPen);
      for(ic=1;ic<=ipage;ic++){
      pDC->StartPage(); //  印刷ページ開始
// タイトル
      if(dlg_print.dlg_sosei.m_radio_jyouken == 0){
      pr_title(pDC, "最大塑性率・最大累積塑性率（全フレーム)");
      }else{
      pr_title(pDC, "最大塑性率・最大累積塑性率（通り芯)");
      }
// 条件
      pr_condition(pDC,dlg_print.dlg_sosei.m_radio_type,dlg_print.dlg_sosei.m_edit_sosei_hyo, 100);
      pr_font_style(pDC, 4);
      if(dlg_print.dlg_sosei.m_radio_jyouken == 0){
      m.Format(" 全フレーム、全部材の最大値を表示" );
      pDC->TextOut(8000-xxx, -(1780), m);
      }else{
      if(dlg_print.dlg_sosei.m_radio_tori == 0 ){
      m.Format("x 方向:            フレーム",
      dlg_print.dlg_sosei.m_edit_tori_set);
      pDC->TextOut(8000-xxx, -(1780), m);
      pDC->TextOut(9000-xxx,-(1780),_T(&ixn[(dlg_print.dlg_sosei.m_edit_tori_set-1)*5]));
      }else{
      m.Format("y 方向:          フレーム",dlg_print.dlg_sosei.m_edit_tori_set);
      pDC->TextOut(8000-xxx, -(1780), m);
```

```
        pDC->TextOut(9000-xxx,-(1780),_T(&iyn[(dlg_print.dlg_sosei.m_edit_tori_set-1)*5]));
        }
        }
        m.Format(" 部材：部材番号" );
        pDC->TextOut(8000-xxx, -(2080), m);
        m.Format(" 位置：１：i 端　２：j 端　３：部材中央" );
        pDC->TextOut(8000-xxx, -(2380), m);
    // 最大塑性率
        pr_font_style(pDC, 4);
        jc=0;
        pDC->TextOut(3300-xxx, -4560-300* jc, _T("最大塑性率"));
        jc=jc+1;
        int n_type = 0;
        pDC->TextOut(2480-xxx, -4640-300* jc, _T("   階"));
        pDC->TextOut(3160-xxx, -4640-300* jc, _T("      柱　(生起時刻)　部材 位置"));
        pDC->TextOut(5960-xxx, -4640-300* jc, _T("      はり(生起時刻)　部材 位置"));
        pDC->TextOut(8760-xxx,-4640-300*  jc, _T("  ブレース(生起時刻)　部材 位置"));
        pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(11560-xxx,-4560-300* jc);
        pDC->MoveTo(2480-xxx, -4860-300* jc);pDC->LineTo(11560-xxx, -4860-300* jc);
        pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(2480-xxx, -4860-300* jc);
        pDC->MoveTo(3160-xxx,-4560-300*jc);pDC->LineTo(3160-xxx, -4860-300* jc);
        pDC->MoveTo(5960-xxx, -4560-300* jc);pDC->LineTo(5960-xxx, -4860-300* jc);
        pDC->MoveTo(8760-xxx, -4560-300* jc);pDC->LineTo(8760-xxx, -4860-300* jc);
        pDC->MoveTo(11560-xxx,-4560-300* jc);pDC->LineTo(11560-xxx,-4860-300* jc);
        for(i=I_so;i>0;i--){
          n_k=(i-1)*18;
          nn_k=(i-1)*39;
          if(jc > n_page ) {
            pDC->EndPage();
           jc=-14;
            pDC->MoveTo(2480-xxx,-(4860+300*jc));pDC->LineTo(11560-xxx,-(4860+300* jc));
          }
          if(i == I_so){
          m.Format("      Rf");pDC->TextOut(2200-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%7df",i);pDC->TextOut(2200-xxx, -(4920+300*jc), m);
          }
          if(S_hyou[n_k+3] == 0.){
          m.Format("―");pDC->TextOut(4500-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%8.2f (%6.2f ) %6d %2d",S_hyou[n_k],S_hyou[n_k+3],IS_hyou[nn_k+21],IS_hyou[nn_k+24]);
          pDC->TextOut(3200-xxx, -(4920+300*jc), m);
          }
          if(S_hyou[n_k+4] == 0.){
          m.Format("―");pDC->TextOut(7300-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%8.2f (%6.2f ) %6d %2d" ,S_hyou[n_k+1],S_hyou[n_k+4],IS_hyou[nn_k+22],IS_hyou[nn_k+25]);
          pDC->TextOut(6000-xxx, -(4920+300*jc), m);
          }
          if(S_hyou[n_k+5] == 0.){
          m.Format("―");pDC->TextOut(10100-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%8.2f (%6.2f ) %6d %2d",S_hyou[n_k+2],S_hyou[n_k+5],IS_hyou[nn_k+23],IS_hyou[nn_k+26]);
          pDC->TextOut(8800-xxx, -(4920+300*jc), m);
```

```
        }
        pDC->MoveTo(2480-xxx,-(4860+300*(jc+1)));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
        pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(2480-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(3160-xxx, -(4860+300* jc   ));pDC->LineTo(3160-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(5960-xxx, -(4860+300* jc   ));pDC->LineTo(5960-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(8760-xxx, -(4860+300* jc   ));pDC->LineTo(8760-xxx, -(4860+300*(jc+1)));
       pDC->MoveTo(11560-xxx,-(4860+300* jc   ));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
        jc=jc+1;
        }
// 最大累積塑性率
        if(jc +3> n_page ) {
        pDC->EndPage();
        jc=-14;
        }
       jc=jc+2;
        pDC->TextOut(3300-xxx, -4560-300* jc, _T("最大累積塑性率"));
        jc=jc+1;
        pDC->TextOut(2480-xxx, -4640-300* jc, _T("   階"));
       pDC->TextOut(3160-xxx, -4640-300* jc, _T("      柱              部材 位置"));
        pDC->TextOut(5960-xxx, -4640-300* jc, _T("      はり            部材 位置"));
        pDC->TextOut(8760-xxx,-4640-300*  jc, _T("  ブレース            部材 位置"));
       pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(11560-xxx,-4560-300* jc);
       pDC->MoveTo(2480-xxx, -4860-300* jc);pDC->LineTo(11560-xxx, -4860-300* jc);
       pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(2480-xxx, -4860-300* jc);
       pDC->MoveTo(3160-xxx, -4560-300* jc);pDC->LineTo(3160-xxx, -4860-300* jc);
       pDC->MoveTo(5960-xxx, -4560-300* jc);pDC->LineTo(5960-xxx, -4860-300* jc);
       pDC->MoveTo(8760-xxx, -4560-300* jc);pDC->LineTo(8760-xxx, -4860-300* jc);
       pDC->MoveTo(11560-xxx,-4560-300* jc);pDC->LineTo(11560-xxx,-4860-300* jc);
       for(i=I_so;i>0;i--){
       n_k=(i-1)*18+6;
       nn_k=(i-1)*39;
       if(jc > n_page ) {
       pDC->EndPage();
       jc=-14;
       pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(11560-xxx,-(4860+300* jc   ));
       }
       if(i == I_so){
       m.Format("      Rf");pDC->TextOut(2200-xxx, -(4920+300*jc), m);
       }else{
       m.Format("%7df",i);pDC->TextOut(2200-xxx, -(4920+300*jc), m);
       }
       if(S_hyou[n_k-3] == 0.){
       m.Format("―");pDC->TextOut(4500-xxx, -(4920+300*jc), m);
       }else{
       m.Format("%8.2f           %6d %2d",S_hyou[n_k],IS_hyou[nn_k+27],IS_hyou[nn_k+30]);
       pDC->TextOut(3200-xxx, -(4920+300*jc), m);
       }
       if(S_hyou[n_k-2] == 0.){
       m.Format("―");pDC->TextOut(7300-xxx, -(4920+300*jc), m);
       }else{
       m.Format("%8.2f           %6d %2d" ,S_hyou[n_k+1],IS_hyou[nn_k+28],IS_hyou[nn_k+31]);
       pDC->TextOut(6000-xxx, -(4920+300*jc), m);
       }
       if(S_hyou[n_k-1] == 0.){
```

```
        m.Format("―");pDC->TextOut(10100-xxx, -(4920+300*jc), m);
        }else{
        m.Format("%8.2f          %6d %2d",S_hyou[n_k+2],IS_hyou[nn_k+29],IS_hyou[nn_k+32]);
        pDC->TextOut(8800-xxx, -(4920+300*jc), m);
        pDC->MoveTo(2480-xxx,-(4860+300*(jc+1)));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
        pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(2480-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(3160-xxx, -(4860+300* jc   ));pDC->LineTo(3160-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(5960-xxx, -(4860+300* jc   ));pDC->LineTo(5960-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(8760-xxx, -(4860+300* jc   ));pDC->LineTo(8760-xxx, -(4860+300*(jc+1)));
        pDC->MoveTo(11560-xxx,-(4860+300*jc));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
        jc=jc+1;
    //  累積回数
        if(jc +3> n_page ) {
        pDC->EndPage();
        jc=-14;
        }
        jc=jc+2;
        pDC->TextOut(3300-xxx, -4560-300* jc, _T("累積塑性回数"));
        jc=jc+1;
        pDC->TextOut(2480-xxx, -4640-300* jc, _T("   階"));
        pDC->TextOut(3160-xxx, -4640-300* jc, _T("      柱              部材 位置"));
        pDC->TextOut(5960-xxx, -4640-300* jc, _T("      はり            部材 位置"));
        pDC->TextOut(8760-xxx,-4640-300*  jc, _T("  ブレース          部材 位置"));
        pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(11560-xxx,-4560-300* jc);
        pDC->MoveTo(2480-xxx, -4860-300* jc);pDC->LineTo(11560-xxx, -4860-300* jc);
        pDC->MoveTo(2480-xxx, -4560-300* jc);pDC->LineTo(2480-xxx, -4860-300* jc);
        pDC->MoveTo(3160-xxx, -4560-300* jc);pDC->LineTo(3160-xxx, -4860-300* jc);
        pDC->MoveTo(5960-xxx, -4560-300* jc);pDC->LineTo(5960-xxx, -4860-300* jc);
        pDC->MoveTo(8760-xxx, -4560-300* jc);pDC->LineTo(8760-xxx, -4860-300* jc);
        pDC->MoveTo(11560-xxx,-4560-300* jc);pDC->LineTo(11560-xxx,-4860-300* jc);
        for(i=I_so;i>0;i--){
          nn_k=(i-1)*39;
          n_k=(i-1)*18+6;
          if(jc > n_page ) {
          pDC->EndPage();
          jc=-14;
          pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(11560-xxx,-(4860+300* jc   ));
          }
          if(i == I_so){
          m.Format("      Rf");pDC->TextOut(2200-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%7df",i);pDC->TextOut(2200-xxx, -(4920+300*jc), m);
          }
          if(S_hyou[n_k-3] == 0.){
          m.Format("―");pDC->TextOut(4500-xxx, -(4920+300*jc), m);
          }else{
         m.Format("%8d           %6d %2d",IS_hyou[nn_k+12],IS_hyou[nn_k+33],IS_hyou[nn_k+36]);
          pDC->TextOut(3200-xxx, -(4920+300*jc), m);
          }
          if(S_hyou[n_k-2] == 0.){
          m.Format("―");pDC->TextOut(7300-xxx, -(4920+300*jc), m);
          }else{
         m.Format("%8d          %6d %2d" ,IS_hyou[nn_k+13],IS_hyou[nn_k+34],IS_hyou[nn_k+37]);
          pDC->TextOut(6000-xxx, -(4920+300*jc), m);
```

```
          }
          if(S_hyou[n_k-1] == 0.){
          m.Format("―");pDC->TextOut(10100-xxx, -(4920+300*jc), m);
          }else{
          m.Format("%8d           %6d %2d",IS_hyou[nn_k+14],IS_hyou[nn_k+35],IS_hyou[nn_k+38]);
          pDC->TextOut(8800-xxx, -(4920+300*jc), m);
          }
          pDC->MoveTo(2480-xxx,-(4860+300*(jc+1)));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
          pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(2480-xxx, -(4860+300*(jc+1)));
          pDC->MoveTo(3160-xxx, -(4860+300* jc   ));pDC->LineTo(3160-xxx, -(4860+300*(jc+1)));
          pDC->MoveTo(5960-xxx, -(4860+300* jc   ));pDC->LineTo(5960-xxx, -(4860+300*(jc+1)));
          pDC->MoveTo(8760-xxx, -(4860+300* jc   ));pDC->LineTo(8760-xxx, -(4860+300*(jc+1)));
          pDC->MoveTo(11560-xxx,-(4860+300*jc));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
          jc=jc+1;
          }
          pDC->EndPage();                                  //  印刷ページ終了
          }
       pDC->EndDoc();                                  //  印刷ジョブページ終了
       pDC->SelectObject(oldFont);
       pDC->SelectObject(oldPen);
}
```

```
//-------------------------------------------
//  全部材あるいは通り芯に関する部材毎の最大塑性率・最大累積塑性率
//-------------------------------------------
void CSf40View::print_buzai_soseiritu_hyo(CDC *pDC)
{
       CFont NewFont, *oldFont;
       CPen  NewPen,  *oldPen;
       CString m;
       int i,j,ic,jc;
       int xxx=1000;
       int n_k = 0;
       int nn_k = 0;
       int n_page=35;
       int  n_out;
       pDC->SetMapMode(MM_TWIPS);
       fontbase= 50;
       NewPen.CreatePen(PS_SOLID,20,RGB(0,0,0));
       oldFont= pr_font_style(pDC, 1);
       oldPen       = pDC->SelectObject(&NewPen);
       char buff[300];
       pDC->StartPage();                //  印刷ページ開始
// タイトル
          if(dlg_print.dlg_sosei.m_radio_jyouken == 2){
          pr_title(pDC, "最大累積塑性率（全フレーム)");
          }else{
          pr_title(pDC, "最大累積塑性率（通り芯)");
          }
// 条件
          pr_condition(pDC, dlg_print.dlg_sosei.m_radio_type,
                   dlg_print.dlg_sosei.m_edit_sosei_hyo, 100);
          pr_font_style(pDC, 4);
```

```
          if(dlg_print.dlg_sosei.m_radio_jyouken == 2){
          m.Format(" 全フレーム、全部材の最大値を表示" );
             pDC->TextOut(8000-xxx, -(1780), m);
          }else{
           if(dlg_print.dlg_sosei.m_radio_tori == 0 ){
              m.Format("x 方向:            フレーム",
                   dlg_print.dlg_sosei.m_edit_tori_set);
              pDC->TextOut(8000-xxx, -(1780), m);
                   pDC->TextOut(9000-xxx,-(1780),_T(&ixn[(dlg_print.dlg_sosei.m_edit_tori_set-1)*5]));
            }else{
              m.Format("y 方向:            フレーム",
              dlg_print.dlg_sosei.m_edit_tori_set);
              pDC->TextOut(8000-xxx, -(1780), m);
              pDC->TextOut(9000-xxx, -(1780), _T(&iyn[(dlg_print.dlg_sosei.m_edit_tori_set-1)*5]));
            }
          }
             m.Format(" 部材：部材番号" );
             pDC->TextOut(8000-xxx, -(2080), m);
             m.Format(" 位置：１：i 端　２：j 端　３：部材中央" );
             pDC->TextOut(8000-xxx, -(2380), m);
             if(dlg_print.dlg_sosei.m_edit_s_sosei > 0.){
             m.Format(" 表示制限：最大塑性率　＞　%6.2f" ,dlg_print.dlg_sosei.m_edit_s_sosei);
             pDC->TextOut(8000-xxx, -(2780), m);
             }
  // 最大塑性率
          jc=0;
          for(j=0;j<4;j++){
          if(jc +3> n_page ) {
          pDC->EndPage();
          jc=-14;
          }
  // 出力個数チェック
          n_out=0;
          for(i=0;i<mnbsb;i++){
          nn_k=i*6;
          if(nx_member[i] != 0 && j+1 == IS_hyou[nn_k+5]) n_out++;
          }
          if(n_out != 0){
  // タイトル表示
          pr_font_style(pDC, 2);
          jc=jc+1;
          if(j == 0 ) {
          pDC->TextOut(2150-xxx, -4560-300* jc, _T("(1) 柱"));
          }
          if(j == 1 ) {
          jc=jc+1;
          pDC->TextOut(2150-xxx, -4560-300* jc, _T("(2) x 方向はり"));
          }
          if(j == 2 ) {
          jc=jc+1;
          pDC->TextOut(2150-xxx, -4560-300* jc, _T("(3) ｙ方向はり"));
          }
          if(j == 3) {
          jc=jc+1;
```

```
          pDC->TextOut(2150-xxx, -4560-300* jc, _T("(4) ブレース"));
          }
           pr_font_style(pDC, 4);
           jc=jc+1;
           int n_type = 0;
           pDC->TextOut(2650-xxx, -4640-300* jc, _T(" 部材 位置 x 通り y 通り"));
           pDC->TextOut(4950-xxx, -4640-300* jc, _T("最大塑性率　(生起時刻：秒)"));
           pDC->TextOut(8150-xxx, -4640-300* jc, _T("最大累積塑性率"));
           pDC->TextOut(10300-xxx,-4640-300* jc, _T("累積塑性回数"));
           pDC->MoveTo(2480-xxx, -4560-300* jc); pDC->LineTo(11560-xxx,-4560-300* jc);
           pDC->MoveTo(2480-xxx, -4860-300* jc); pDC->LineTo(11560-xxx, -4860-300* jc);
           pDC->MoveTo(2480-xxx, -4560-300* jc); pDC->LineTo(2480-xxx, -4860-300* jc);
           pDC->MoveTo(4620-xxx, -4560-300* jc); pDC->LineTo(4620-xxx, -4860-300* jc);
           pDC->MoveTo(7520-xxx, -4560-300* jc); pDC->LineTo(7520-xxx, -4860-300* jc);
           pDC->MoveTo(10040-xxx, -4560-300* jc);         pDC->LineTo(10040-xxx, -4860-300* jc);
           pDC->MoveTo(11560-xxx,-4560-300* jc); pDC->LineTo(11560-xxx,-4860-300* jc);
           for(i=0;i<mnbsb;i++){
           if(jc > n_page ) {
            pDC->EndPage();
           jc=-14;
           pDC->MoveTo(2480-xxx, -(4860+300* jc   ));     pDC->LineTo(11560-xxx,-(4860+300* jc   ));
           }
          n_k=i*3;
          nn_k=i*6;
           if(nx_member[i] != 0 && j+1 == IS_hyou[nn_k+5]){
           m.Format("%4d",i+1);pDC->TextOut(3500-xxx-12*fontbase, -(4920+300*jc), m);
           m.Format("%2d",IS_hyou[nn_k+3]);pDC->TextOut(3500-xxx-4*fontbase, -(4920+300*jc), m);
           m.Format("%3d",IS_hyou[nn_k]);pDC->TextOut(3500-xxx+2*fontbase, -(4920+300*jc), m);
           m.Format("%3d",IS_hyou[nn_k+1]);pDC->TextOut(3500-xxx+10*fontbase, -(4920+300*jc), m);
           m.Format("%8.2f",S_hyou[n_k]);pDC->TextOut(5500-xxx, -(4920+300*jc), m);
           m.Format("(%6.2f )",S_hyou[n_k+1]);pDC->TextOut(6320-xxx, -(4920+300*jc), m);
           m.Format("%8.2f",S_hyou[n_k+2]);pDC->TextOut(8780-xxx, -(4920+300*jc), m);
           m.Format("%8d",IS_hyou[nn_k+4]);pDC->TextOut(10240-xxx, -(4920+300*jc), m);
           pDC->MoveTo(2480-xxx, -(4860+300*(jc+1)));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
           pDC->MoveTo(2480-xxx, -(4860+300* jc   ));pDC->LineTo(2480-xxx, -(4860+300*(jc+1)));
           pDC->MoveTo(4620-xxx, -(4860+300* jc   ));pDC->LineTo(4620-xxx, -(4860+300*(jc+1)));
//         pDC->MoveTo(5120-xxx, -(4860+300* jc   ));pDC->LineTo(5120-xxx, -(4860+300*(jc+1)));
           pDC->MoveTo(7520-xxx, -(4860+300* jc   ));pDC->LineTo(7520-xxx, -(4860+300*(jc+1)));
           pDC->MoveTo(10040-xxx, -(4860+300* jc   ));pDC->LineTo(10040-xxx, -(4860+300*(jc+1)));
           pDC->MoveTo(11560-xxx,-(4860+300* jc   ));pDC->LineTo(11560-xxx,-(4860+300*(jc+1)));
          jc=jc+1;
           }}
           }}
       pDC->EndPage();
       pDC->EndDoc();                              //  印刷ジョブページ終了
       pDC->SetTextAlign(TA_LEFT);
       pDC->SelectObject(oldFont);
       pDC->SelectObject(oldPen);
}
```

```
//-------------------------------------------
//  任意部材の最大塑性率・最大累積塑性率の時刻歴図
//-------------------------------------------
```

```
void CSf40View::print_soseiritu_graph(CDC *pDC)
{
      CFont NewFont, *oldFont;
      CPen  NewPen,  *oldPen;
      CString m;
      int i,ic,jc;
      int xxx=1000;
      int ii_x;
       float ss_eps[3];
      pDC->SetMapMode(MM_TWIPS);
      NewPen.CreatePen(PS_SOLID,20,RGB(0,0,0));
      oldFont= pr_font_style(pDC, 1);
      oldPen       = pDC->SelectObject(&NewPen);
                    char buff[300];
//    float bairitu=2;
//    bairitu =  setpos(iww,iwh,rcFrame.right,rcFrame.bottom);
         pDC->StartPage();           //  印刷ページ開始
// タイトル
         pr_title(pDC, "部材の最大塑性率・最大累積塑性率の時刻歴図");
// 条件
         pr_condition(pDC, dlg_print.dlg_sosei.m_radio_type,
                  dlg_print.dlg_sosei.m_edit_sosei_hyo, 100);
         pr_font_style(pDC, 4);
         m.Format("部材番号:%8d  位置:%2d ",dlg_print.dlg_sosei.m_edit_n_member,
                  dlg_print.dlg_sosei.m_edit_nx_member);
            pDC->TextOut(8000-xxx, -(1480), m);
            m.Format(" (位置：１：i 端　２：j 端　３：部材中央)" );
            pDC->TextOut(8000-xxx, -(1780), m);
            for(int n_m=0;n_m<3;n_m++){
           ss_eps[n_m]=0.;
           if(Sec_epsy[n_m+3*(dlg_print.dlg_sosei.m_edit_n_member-1)] != 0.)
            ss_eps[n_m]=1./Sec_epsy[n_m+3*(dlg_print.dlg_sosei.m_edit_n_member-1)] ;
            }
            m.Format(" 塑性ひずみ εp: %8.6f",ss_eps[0]);
            pDC->TextOut(8000-xxx, -(2080), m);
            m.Format("          Φyp: %8.6f",ss_eps[1]);
            pDC->TextOut(8000-xxx, -(2380), m);
            m.Format("          Φzp: %8.6f",ss_eps[2]);
            pDC->TextOut(8000-xxx, -(2680), m);
//    rectPage.right=4500;
//    rectPage.left=500;
//    rectPage.top=1000;
//    rectPage.bottom=5000;
       F_pos_pr[1][1] = 4060-300* 25;
       F_pos_pr[1][0] = 4060;
       F_pos_pr[0][1] = 2480;
       F_pos_pr[0][0] = 11560;
      int ar_w = (F_pos_pr[0][0] - F_pos_pr[0][1]);
      int ar_h = (F_pos_pr[1][1] - F_pos_pr[1][0])/8.;
      fontbase= (ar_w)/50;
      char buffer[200];
      int x_axi;
      float y_axi;
      float load_data_max,ld_dt;
```

```
        float m_wave_max_v,m_wave_delt_v,ar_bi_x;
        int m_wave_time;
        int m_wave_max=F_all_step;
        m_wave_max_v=1.;
        float ar_sx;
        float ar_sy;
        jc=0;
        ar_sy = F_pos_pr[1][0] + ar_h*7.5  ;
//      図：
       for(int n_graph = 0 ;n_graph < 2; n_graph++){
      jc=jc+1;
 //図形中心設定
       ar_sx = F_pos_pr[0][1] - 5*fontbase ;
       ar_sy = ar_sy + ar_h*3. ;
       pDC->SelectObject (&NewPen_pr[0]);
       pDC->MoveTo ( ar_sx, ar_sy);
        pDC->LineTo ( ar_sx + ar_w, ar_sy);
        pDC->MoveTo ( ar_sx, ar_sy );
        pDC->LineTo ( ar_sx, ar_sy - 2*ar_h);
        pDC->SelectObject (&NewPen_pr[0]);
        pDC->MoveTo ( ar_sx,        ar_sy- 2*ar_h);
        pDC->LineTo ( ar_sx + ar_w, ar_sy- 2*ar_h);
        pDC->LineTo ( ar_sx + ar_w, ar_sy );
        pDC->LineTo ( ar_sx,        ar_sy );
        pDC->LineTo ( ar_sx,        ar_sy- 2*ar_h);

        int ist, jst;
        float ar_dtt = (float) ar_w /m_wave_max;
        float ar_bi  = (float)2*ar_h;
        int nn = 5;
// 図形軸表示
        SAIDAI_X(S_hyou,&m_wave_max,&nn,&n_graph,
             &m_wave_data_max,&load_data_max,&ld_dt);
        int iij =load_data_max/ld_dt+ 0.01;
        ar_bi = (float)ar_h*2 /load_data_max;
        for ( i=0;i <iij+1; i++)
        {
          jst = ar_sy - ld_dt*ar_bi * (i);
          y_axi= (i)*ld_dt;
          if(ld_dt >= 0.1 && ld_dt < 1.) sprintf(buffer,"%3.1f",y_axi);
          if(ld_dt >= 1 && ld_dt < 990.) sprintf(buffer,"%3.0f",y_axi);
          if(ld_dt >= 990 ) sprintf(buffer,"%5.0f",y_axi);
          if(ld_dt >= 990 ){
          pDC->TextOut(ar_sx-4*fontbase,jst+50,buffer);
          }else{
          pDC->TextOut(ar_sx-3*fontbase,jst+50,buffer);
          }
          pDC->MoveTo ( ar_sx, jst);
         pDC->LineTo ( ar_sx+ar_w, jst);
          }
// 図形 x 軸タイトル表示
         sprintf(buffer,"Time(sec.)");
         pDC->TextOut(ar_sx+ ar_w- 7.*fontbase,ar_sy-0.5*fontbase,buffer);
        if(F_all_time > 10.) {
```

```
          iij=F_all_time+0.01;
          ar_bi_x=ar_dtt/F_delt_cl;
        for ( i=0;i < iij; i++)
        {
            ii_x=((i+1)/5)*5;
            jst = ar_sx + ar_bi_x * (i+1);
            if(i+1 == ii_x ) {

             pDC->MoveTo ( jst, ar_sy+200);
             pDC->LineTo ( jst, ar_sy);
             pDC->MoveTo ( jst, ar_sy-2*ar_h-200);
             pDC->LineTo ( jst, ar_sy-2*ar_h);
            sprintf(buffer,"%2d",i+1);
            pDC->TextOut(jst-0.7*fontbase,ar_sy+200,buffer);
            }else{
           pDC->MoveTo ( jst, ar_sy+100);
           pDC->LineTo ( jst, ar_sy);
           pDC->MoveTo ( jst, ar_sy-2*ar_h-100);
           pDC->LineTo ( jst, ar_sy-2*ar_h);
        }
        }
        }else{
           iij=F_all_time*2+0.01;
           ar_bi_x=ar_dtt/F_delt_cl*0.5;
        for ( i=0;i <iij; i++)
        {
            ii_x=((i+1)/2)*2;
            jst = ar_sx + ar_bi_x * (i+1);
            if(i+1 == ii_x ) {
           pDC->MoveTo ( jst, ar_sy+200);
           pDC->LineTo ( jst, ar_sy);
           pDC->MoveTo ( jst, ar_sy-2*ar_h-200);
           pDC->LineTo ( jst, ar_sy-2*ar_h);
           sprintf(buffer,"%2d",(i+1)/2);
           pDC->TextOut(jst-0.7*fontbase,ar_sy+200,buffer);
            }else{
           pDC->MoveTo ( jst, ar_sy+100);
           pDC->LineTo ( jst, ar_sy);
           pDC->MoveTo ( jst, ar_sy-2*ar_h-100);
           pDC->LineTo ( jst, ar_sy-2*ar_h);
            }
        }}
  // 図形 y 軸タイトル表示
           if(n_graph == 0){
          sprintf(buffer,"塑性率");
           pDC->TextOut(ar_sx +ar_w*0.5- 3.*fontbase, ar_sy-0.5*fontbase,buffer);
           }
           if(n_graph == 1){
          sprintf(buffer,"累積塑性率");
           pDC->TextOut(ar_sx +ar_w*0.5- 4.*fontbase, ar_sy-0.5*fontbase,buffer);
           }
          int j;
          pDC->SelectObject (&NewPen_pr[3]);
           j=n_graph;
```

```
          int istt = ar_sx;
         int jstt = -S_hyou[j]*ar_bi+ ar_sy;
          pDC->MoveTo ( istt, jstt);
         for(  i = 1; i < m_wave_max; i++)
      {
          j=5*i+n_graph;
          ist = ar_sx + ar_dtt * (float)i;
          jst = -S_hyou[j]*ar_bi + ar_sy;
          pDC->LineTo ( ist, jst);
         }
           ar_sy = ar_sy - ar_h*0.5 ;
      }
     ar_sy = ar_sy - ar_h*0.5 ;
     for(n_graph = 2 ;n_graph < 5; n_graph++){
     jc=jc+1;
 //図形中心設定
       ar_sx = F_pos_pr[0][1] - 5*fontbase ;
       ar_sy = ar_sy + ar_h*2.5 ;
      pDC->SelectObject (&NewPen_pr[0]);
      pDC->MoveTo ( ar_sx, ar_sy);
     pDC->LineTo ( ar_sx + ar_w, ar_sy);
     pDC->MoveTo ( ar_sx, ar_sy + ar_h);
     pDC->LineTo ( ar_sx, ar_sy - ar_h);
      pDC->SelectObject (&NewPen_pr[0]);
      pDC->MoveTo ( ar_sx,        ar_sy- ar_h);
     pDC->LineTo ( ar_sx + ar_w, ar_sy- ar_h);
     pDC->LineTo ( ar_sx + ar_w, ar_sy + ar_h);
     pDC->LineTo ( ar_sx,        ar_sy + ar_h);
      pDC->LineTo ( ar_sx,        ar_sy- ar_h);
      int ist, jst;
      float ar_dtt = (float) ar_w /m_wave_max;
      float ar_bi  = (float)ar_h;
      int nn = 5;
// 図形軸表示
      SAIDAI_X(S_hyou,&m_wave_max,&nn,&n_graph,
           &m_wave_data_max,&load_data_max,&ld_dt);
       int iij =load_data_max/ld_dt+ 0.01;
      ar_bi = (float)ar_h /load_data_max;
       jst = ar_sy ;
      y_axi=0.;
       if(ld_dt >= 0.1 && ld_dt < 1.) sprintf(buffer,"%3.1f",y_axi);
       if(ld_dt >= 1 && ld_dt < 990.)  sprintf(buffer,"%3.0f",y_axi);
       if(ld_dt >= 990 ) sprintf(buffer,"%5.0f",y_axi);
        pDC->TextOut(ar_sx-3*fontbase,jst+50,buffer);
     for ( i=0;i <iij; i++)
     {
         jst = ar_sy - ld_dt*ar_bi * (i+1);
         ist = ar_sy + ld_dt*ar_bi * (i+1);
         y_axi= (i+1)*ld_dt;
         if(ld_dt >= 0.1 && ld_dt < 1.) sprintf(buffer,"%3.1f",y_axi);
         if(ld_dt >= 1 && ld_dt < 990.)  sprintf(buffer,"%3.0f",y_axi);
         if(ld_dt >= 990 ) sprintf(buffer,"%5.0f",y_axi);
         if(ld_dt >= 990 ){
         pDC->TextOut(ar_sx-4*fontbase,jst+50,buffer);
```

```
            }else{
            pDC->TextOut(ar_sx-3*fontbase,jst+50,buffer);
            }
             y_axi= -y_axi;
            if(ld_dt >= 0.1 && ld_dt < 1.) {
            sprintf(buffer,"%3.1f",y_axi);
            pDC->TextOut(ar_sx-3.5*fontbase,ist+50,buffer);
            }
            if(ld_dt >= 1){
            if(ld_dt >= 1 && ld_dt < 990.) {
            sprintf(buffer,"%3.0f",y_axi);
            pDC->TextOut(ar_sx-3*fontbase,ist+50,buffer);
            }
            if(ld_dt >= 990 ) {
            sprintf(buffer,"%5.0f",y_axi);
            pDC->TextOut(ar_sx-4*fontbase,ist+50,buffer);
            }}
           pDC->MoveTo ( ar_sx, jst);
           pDC->LineTo ( ar_sx+300, jst);
           pDC->MoveTo ( ar_sx, ist);
           pDC->LineTo ( ar_sx+300, ist);
           pDC->MoveTo ( ar_sx+ar_w-300, jst);
           pDC->LineTo ( ar_sx+ar_w, jst);
           pDC->MoveTo ( ar_sx+ar_w-300, ist);
           pDC->LineTo ( ar_sx+ar_w, ist);
         }
  // 図形軸タイトル表示
          sprintf(buffer,"Time(sec.)");
         pDC->TextOut(ar_sx+ ar_w- 7.*fontbase,ar_sy+ar_h-0.5*fontbase,buffer);
         if(F_all_time > 10.) {
         iij=F_all_time+0.01;
         ar_bi_x=ar_dtt/F_delt_cl;
        for ( i=0;i < iij; i++)
        {
        i_x=((i+1)/5)*5;
        jst = ar_sx + ar_bi_x * (i+1);
        if(i+1 == ii_x ) {
          pDC->MoveTo ( jst, ar_sy+200);
          pDC->LineTo ( jst, ar_sy-200);
          sprintf(buffer,"%2d",i+1);
          pDC->TextOut(jst-0.7*fontbase,ar_sy+200,buffer);
          }else{
           pDC->MoveTo ( jst, ar_sy+100);
           pDC->LineTo ( jst, ar_sy-100);
            }
       }
       }else{
           iij=F_all_time*2+0.01;
           ar_bi_x=ar_dtt/F_delt_cl*0.5;
       for ( i=0;i <iij; i++)
       {
           ii_x=((i+1)/2)*2;
           jst = ar_sx + ar_bi_x * (i+1);
           if(i+1 == ii_x ) {
```

```
            pDC->MoveTo ( jst, ar_sy+200);
            pDC->LineTo ( jst, ar_sy-200);
            sprintf(buffer,"%2d",(i+1)/2);
            pDC->TextOut(jst-0.7*fontbase,ar_sy+200,buffer);
             }else{
            pDC->MoveTo ( jst, ar_sy+100);
            pDC->LineTo ( jst, ar_sy-100);
             }
          }
          }
            if(n_graph == 2){
              sprintf(buffer,"εx/εp ：軸方向ひずみ");
             pDC->TextOut(ar_sx +ar_w*0.5- 9.*fontbase, ar_sy+ar_h-0.5*fontbase,buffer);
            }
            if(n_graph == 3){
              sprintf(buffer,"Φy/Φyp：曲げひずみ");
             pDC->TextOut(ar_sx +ar_w*0.5- 8.*fontbase, ar_sy+ar_h-0.5*fontbase,buffer);
            }
                     if(n_graph == 4){
           sprintf(buffer,"Φz/Φzp：曲げひずみ");
            pDC->TextOut(ar_sx +ar_w*0.5- 8.*fontbase, ar_sy+ar_h-0.5*fontbase,buffer);
            }
        int j;
    // 図形出力
            pDC->SelectObject (&NewPen_pr[3]);
            j=n_graph;
           int istt = ar_sx;
           int jstt = -S_hyou[j]*ar_bi+ ar_sy;
            pDC->MoveTo ( istt, jstt);
           for(  i = 1; i < m_wave_max; i++)
           {
            j=5*i+n_graph;
            ist = ar_sx + ar_dtt * (float)i;
            jst = -S_hyou[j]*ar_bi + ar_sy;
            pDC->LineTo ( ist, jst);
           }
        }  //  グラフ出力終了
        pDC->EndPage();                  //  印刷ページ終了
        pDC->EndDoc();                           //  印刷ジョブページ終了
        pDC->SetTextAlign(TA_LEFT);
        pDC->SelectObject(oldFont);
        pDC->SelectObject(oldPen);
    }
```