4.9 ニューマークβ法

本節からは、SPACE における動的ソルバーの細部について解説する。解析順序と多少異なって説明するので、前の節を参照して、どこの部分の説明であるか理解しながら読まれたい。

まず、ニューマークβ法を利用して、Δｔ秒後の変位と速度を予測するサブルーチン、また得られた加速度より変位と速度を求めるサブルーチンについて説明する。両者共にプログラムとしては非常に簡単であり、理解し易いと思う。なお、ニューマークβ法に関連する係数は既に計算され、構造体 Newmark_P に保存されている。Δｔ秒後の変位と速度を予測するサブルーチン Estimate_disp_vel()の全文は、CD 内の付録に掲載されているのでそちらを見られたい。

それでは、このプログラムの構造を見てみよう。このプログラムに 2 つに分かれており、一つは反復計算に入る前の初期値である変位と速度を予測するものであり、他は反復計算に入ってから次のステップの変位と速度を予測するものである。以下に、当該のプログラムの構造を示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Estimate_disp_vel
C     ―――――――――――――――――――――――――――――――――
C     ●    変位、速度の予測(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Estimate_disp_vel( )
C     ―――――――――――――――――――――――――――――――――
      if(nx.eq.0) then                                              ! 1
c――――――――――――――――――――――――――――――――★最初の予測
       a=Newmark_P.dt                !  Δt                          ! 2
       b=Newmark_P.dt5               !  0.5Δt2
       c=Newmark_P.bdt               !  βΔt2
       d=Newmark_P.ddt               !  δΔt
      do i=1,n_unknown                                              ! 3
c――――――――――――――――――――――――――――――――★ニューマークβ法予測
      est_ddisp_point(i)=a*past_vel_point(i)+b*past_acc_point(i)+
     *                   c*past_dacc_point(i)                       ! 4
      est_disp_point(i) =past_disp_point(i)+est_ddisp_point(i)      ! 5
      est_vel_point(i)  =past_vel_point(i)+a*past_acc_point(i)
     *                   +      d*past_dacc_point(i)               ! 6
      result_acc_point(i)= past_acc_point(i)+past_dacc_point(I)     ! 7
      enddo

      else                                                          ! 8
      gumma=Newmark_P.gumma                                         ! 9
      gummax=1.- gumma
c――――――――――――――――――――――――――――――――★2 回目以降の予測
       do i=1,n_unknown                                            ! 10
c――――――――――――――――――――――――――――――――★計算結果使用
      est_disp_point(i) =result_disp_point(i)*gumma +
     *                   gummax*est_disp_point(i)                   ! 11
```

```
 est_vel_point(i)  =result_vel_point(i)*gumma +
*                   gummax*est_vel_point(i)                          ! 12
 est_ddisp_point(i)=(est_disp_point(i)-past_disp_point(i))            ! 13
 end do
 endif
 return
 end
```

上記サブルーチンについて説明する。

１．パラメータ nx がゼロの場合は最初の予測を、ゼロ以外は反復過程における次回の予測値を求める処理へ移る。

２．ここでは、ニューマークβ法で使用する係数を構造体から変数にセットし直す。係数の意味はコメントに示した。

３．未知変数の数分、以下の処理を繰り返す。予測式は以下のようであり、

$$\Delta y_{n+1} = \dot{y}_n \Delta t + \left\{\left(0.5+\beta\right)\ddot{y}_n - \beta\ddot{y}_{n-1}\right\}\Delta t^2$$
$$y_{n+1} = y_n + \Delta y_{n+1}$$
$$\dot{y}_{n+1} = \dot{y}_n + \left\{\left(1+\delta\right)\ddot{y}_n - \delta\ddot{y}_{n-1}\right\}\Delta t$$
$$\ddot{y}_{n+1} = \ddot{y}_n + \Delta\ddot{y}_{n+1}$$

係数はサブルーチン set_newmark()で以下のように計算されている。ここで、dt は増分時間を表し、beta はβを、delta はδを表す。

```
Newmark_P.dt        =  dt                          a
Newmark_P.ddt       =  delta*dt                    d
Newmark_P.bdt       =  beta*dt*dt                  c
Newmark_P.dt5       =  0.5*dt*dt                   b
```

４．ここでは、予測増分変位 Δy_{n+1} を以下の式に変更して求めている。ただし、配列 past_acc_point は $\ddot{y}_n - \ddot{y}_{n-1}$ を表す。

$$\Delta y_{n+1} = \dot{y}_n \Delta t + 0.5\ddot{y}_n \Delta t^2 + \beta(\ddot{y}_n - \ddot{y}_{n-1})\Delta t^2$$

５．予測変位は、４．で求めた増分変位 Δy_{n+1} に過去の変位を足せば良い。ここで言う過去の変位とは１ステップ前の変位を意味する。

６．予測増分速度 $\dot{y}_{n+1}$ は、４．と同様に次式に変更して用いている。

$$\dot{y}_{n+1} = \dot{y}_n + \Delta t\ddot{y}_n + \delta\Delta t\left(\ddot{y}_n - \ddot{y}_{n-1}\right)$$

７．予測加速度 $\ddot{y}_{n+1}$ は、過去の加速度に１ステップ前の加速度の変化をそのまま加えて使用する。

$$\ddot{y}_{n+1} = \ddot{y}_n + \frac{\ddot{y}_n - \ddot{y}_{n-1}}{\Delta t}\Delta t = \ddot{y}_n + \left(\ddot{y}_n - \ddot{y}_{n-1}\right)$$

これで、Δｔ秒後の変位、速度、加速度を予測し、これを釣合式の右辺項を計算する際使用することになる。

８．ここからは、反復時における次回の各予測値を求める。

９．ユーザーが設定する収束用パラメータをセットする。ただし、通常は１とする。

10. 全未知数について次の処理を行う。

11. 予測変位と釣合式を解いて得た変位を用いて、次の反復計算のための変位を予測する。その割合は収束用パラメータによる。

12. 予測速度と釣合式を解いて得た速度を用いて、次の反復計算のための速度を予測する。その割合は収束用パラメータによる。

13. 予測変位と過去の変位から予測増分変位を求める。

これで、Δｔ秒後の変位と速度を予測するサブルーチンの説明は終わるが、プログラム自身は単純なので理解は容易であろう。

続いて、加速度からニューマークβ法を用いて、変位と速度を求めるサブプログラム Cal_disp_vel()について説明しよう。このサブプログラムの骨組みを取り出し、以下に示す。全文は付録を参照されたい。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_disp_vel
C       ―――――――――――――――――――――――――――――――
C     ●   加速度よりβ法に基づき変位と速度を計算(ok)
C     ―――――――――――――――――――――――――――――――
      subroutine Cal_disp_vel()
       a=Newmark_P.dt                                          ! 1
       b=Newmark_P.bdt_5
       c=Newmark_P.bdt
       d=Newmark_P.ddt_1
       e=Newmark_P.ddt
       do i=1,n_unknown                                        ! 2
       result_disp_point(i)=past_disp_point(i) + a*past_vel_point(i)
      *   +b*past_acc_point(i) + c*result_acc_point(i)         ! 3
       result_vel_point(i)=past_vel_point(i)+d*past_acc_point(i)
      *   +e*result_acc_point(i)                               ! 4
       enddo
       return
       end
```

1. ここでは、ニューマークβ法で使用する係数を構造体から変数にセットし直す。各係数はサブルーチン set_newmark()で以下のように計算されている。

```
Newmark_P.dt          =  dt                          a
Newmark_P.ddt         =  delta*dt                    e
```

```
      Newmark_P.bdt        =  beta*dt*dt                  c
      Newmark_P.ddt_1      =  (1. - delta)*dt             d
      Newmark_P.bdt_5      =  (0.5- beta )*dt*dt          b
```

2. 全未知数に対して変位と速度を求める。変位と速度は以下の式より求める。

$$\dot{y}_{n+1} = \dot{y}_n + \left\{(1-\delta)\ddot{y}_n + \delta\ddot{y}_{n+1}\right\}\Delta t$$
$$y_{n+1} = y_n + \dot{y}_n\Delta t + \left\{(0.5-\beta)\ddot{y}_n + \beta\ddot{y}_{n+1}\right\}\Delta t^2$$

3. ここでは、変位を求める。ただし求める式を以下のように変形する。

$$y_{n+1} = y_n + \dot{y}_n\Delta t + (0.5-\beta)\Delta t^2\ddot{y}_n + \beta\Delta t^2\ddot{y}_{n+1}$$

4. 次に速度を求めるが、次式に変更して使用する。

$$y_{n+1} = y_n + \dot{y}_n\Delta t + (0.5-\beta)\Delta t^2\ddot{y}_n + \beta\Delta t^2\ddot{y}_{n+1}$$

4.10 部材長さの計算と部材モデルの初期設定

本節は、予備計算の中で比較的単純な部分である部材長さと部材モデルの初期設定の説明を行う。前節で説明した予備計算の流れの中から、以下に該当する部分を抜き出して掲載する。ここでは、2 つのサブルーチンについて内容を説明しよう。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         動的解析のための予備計算を行う
c
c―――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――★部材長さ計算(ok)
      call Cal_member_length(Member,Point,Parameter_C)
c――――――――――――――――――――――――――――――★モデルの初期設定
      call Set_initial_data(Element,Member,Parameter_C,Newmark_P,
     *           E_model6_real,Model_type)
```

まず、部材長さを計算するサブプログラムの骨組みを以下に示す。全文は付録を参照されたい。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_member_length
C       ―――――――――――――――――――――――――――――――――
C     ●   部材の長さ（節点間距離）を計算する(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_member_length(Member,Point,Parameter_C)
      implicit real*8(A-H,O-Z)
      include "submain.h"
```

```
record / parameter_s  / Parameter_C
record / member_s     / Member
record / point_s      / Point
dimension Member(*),Point(*)
do i=1,Parameter_C.n_member                                    ! 1
al=0.                                                           ! 2
i1 = Member(i).nm_point(1)                                     ! 3
j1 = Member(i).nm_point(2)                                     ! 4
do j=1,3                                                        ! 5
al=al + (Point(i1).coord(j) - Point(j1).coord(j))**2           ! 6
end do
Member(i).alength = dsqrt(al)                                  ! 7
end do
return
end
```

1. 構造体の中にセットされている部材総数を用いて、全部材の長さを計算する。
2. 部材の長さを計算するワークエリア al をゼロセットする。
3. 構造体の中にセットされている部材の i 端の節点番号を取り出す。
4. 同様に、j 端の節点番号を取り出す。
5. 3 方向について計算を行う。
6. ｊ方向の部材長さの 2 乗を計算し、ワークエリア al に足しこむ
7. 各方向の長さの 2 乗の和の平方根を求め、構造体中の長さに関する成分 Member(i).alength にセットする。

これで、全部材の長さを計算したことになる。ただし、このプログラム内には、データの設定ミスに対する対処法は施されていない。例えば、部材両端の節点番号の間違い、あるいは、部材長さがゼロとなる場合など、今後の計算に重要な支障をきたすことが予想される。SPACE では、他の部分でこれらのデータの誤りに対して、チェックを行っている。

続いて、サブルーチン Set_initial_data()について説明する。このサブルーチンは、各部材モデルで何らかの初期設定を行う場所として確保しているルーチンであるが、現在では、Maxwell モデルのみが使用している。他のモデルは、線形剛性行列を求めるサブプログラムの中で行っており、これについては次節で説明する。このサブプログラムの骨組みを見てみよう。

```
C     ――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_initial_data
C       ――――――――――――――――――――――――――――――
C     ●   各モデルの初期設定(各部材モデルで初期設定が必要な場合はここを使用)
C     ――――――――――――――――――――――――――――――――
```

```
      subroutine Set_initial_data(Element,Member,Parameter_C,Newmark_P,
     *           E_model6_real,Model_type)
C     ―――――――――――――――――――――――――――――――――――――
      n_member = Parameter_C.N_member
      do i=1,n_member                                                    ! 1
      mem = i
      iet  = Member(i).element_type
      iett=(iet-1)/10
      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
      call Initset_maxwelldamp(Element(ie),Newmark_P,
     *           E_model6_real(ien),Model_type.n_m_filter)               ! 2
      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ル
      goto 100
  116 continue
c――――――――――――――――――――――――――――――――★Model_No.16 3 次元プレテンション動作モデル
      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
      endif
      goto 100
 9999 continue
  100 continue
      end do
      return
      end
```

１．　全部材について以下の処理を実行する。
２．　Maxwell 型の部材に関して初期設定行う。初期設定行うサブプログラム Initset_maxwelldamp()は、第 5.11.2 節を参照されたい。

4.11 各種データの初期設定

本節では、予備計算の中で行っている初期設定について説明する。初期設定に関するコール文を以下に示す。

```
c――――――――――――――――――――――――――――――――★部材両端中央応力、力のゼロセット(ok)
      call  Set_pointforce_zero(Member,Parameter_C.n_member)
c     write(damp_out,*) ' Set_pointforce_zero Ok'
c――――――――――――――――――――――――――――――――★部材応力のゼロセット(ok)
      call  Set_stress_zero(Member,Parameter_C.n_member)
c     write(damp_out,*) ' Set_stress_zero Ok'
c――――――――――――――――――――――――――――――――★ベクトルのゼロセット
c――――――――――――――――――――――――――――――――★増分前の変位、速度、加速度(ok)
      call Set_zero_v(past_disp_point, past_vel_point,
     *       past_acc_point,past_dacc_point,Parameter_C.n_unknown)
c     write(damp_out,*) ' Set_zero_v Ok'
c――――――――――――――――――――――――――――――――★最大変位等のゼロセット(ok)
```

```
      call Clear_max_disp(Max_disp,Parameter_C.n_point)
c     write(damp_out,*) ' Clear_max_disp Ok'
c―――――――――――――――――――――――――――――★最大、最小応力等のゼロセット(ok)
      call Clear_max_stress(Max_stress,Parameter_C.n_member)
c     write(damp_out,*) ' Clear_max_stress Ok'
c―――――――――――――――――――――――――――――★不釣合力のゼロセット(ok)
      call Set_zero_pointforce(fll_force_point,Parameter_C.n_point)
c
```

この初期設定プログラムは非常に単純で理解し易く、また、ほとんど同じ構造となっている。以下に、2つ代表的な初期設定サブルーチンを示す。

```
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_pointforce_zero
C       ――――――――――――――――――――――――――――――――――――
C     ●   部材節点力のゼロセット(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Set_pointforce_zero(Member,n_member)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      dimension Member(*)
C     ――――――――――――――――――――――――――――――――――――
c           Member              structure
c           n_member            integer   部材数
C     ――――――――――――――――――――――――――――――――――――
      do i=1,n_member
      do j=1,12
       Member(i).force(j)= 0.0
      enddo
      do j=1,18
       Member(i).stress(j)= 0.0
      enddo
       end do
       return
       end
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_stress_zero
C       ――――――――――――――――――――――――――――――――――――
C     ●   部材節点力のゼロセット(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Set_stress_zero(Member,n_member)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      dimension Member(*)
C     ――――――――――――――――――――――――――――――――――――
c           Member              structure
c           n_member            integer   部材数
C     ――――――――――――――――――――――――――――――――――――
      do i=1,n_member
      do j=1,18
```

```
        Member(i).stress(j)= 0.0
       enddo
       enddo
       return
       end
```

4.12 線形剛性と部材型減衰

本節からは、部材モデルの剛性などを作成するサブルーチンについて説明する。後章で説明するように部材モデルは、多種の部材モデルを組み込む必要性から、階層構造となっている。そのため、関連するサブルーチンは、プログラム自体に特殊な構造を有している。その仕組みを利用して、線形剛性、非線形剛性、部材減衰、弾塑性チェック、部材応力などが作成されており、このプログラム構造を理解することが大切となる。ただし、この構造は、各サブルーチンでほとんど同じであり、ひとつのサブルーチンで理解すれば、他のサブルーチンを理解することは難しくない。

ここでは、まず、線形の部材剛性行列を作成するサブルーチンを観察しよう。このサブルーチン Cal_stiff_linear()は、予備計算の中で以下のようにコールされている。ここで、全部材の線形剛性行列が計算される。また、次のサブルーチン Rotate_stiffness()でその部材剛性が、部材座標系から釣合座標系に変換されて、線形剛性を保存する配列 ak_linear にセットされる。この線形剛性の座標変換を行うサブルーチンについては、第２章で説明した。

```
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)
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)
c     write(damp_out,*) ' Rotate_stiffness Ok'
```

サブルーチン Cal_stiff_linear()の全コードを以下に示す。

```
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)
C     ―――――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.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―――――――――――――――――――――――――――――――――――――――――――――――
c       モデル番号
c       Model_No.1 = 1              !  幾何学非線形型有限要素弾性モデル
c       Model_No.2 = 2              !  3 次元せん断弾塑性モデル
c       Model_No.3 = 3              !  3 次元軸力弾塑性モデル
c       Model_No.4 = 4              !  3 次元ケーブル弾塑性モデル
c       Model_No.5 = 5              !  3 次元免振モデル（MSS モデル）
c       Model_No.6 = 6              !  3 次元制震 Maxwell モデル
c       Model_No.7 = 7              !  3 次元弾塑性バネモデル(*)
c
c       Model_No.11 = 11            !  両端ファイバーモデル
c       Model_No.12 = 12            !  両端、中央ファイバーモデル
c       Model_No.15 = 15            !  ファイバーモデル
c       Model_No.21 = 13            !  両端 MS モデル
c       Model_No.22 = 14            !  両端、中央 MS モデル
c       Model_No.31 = 16            !  両端アナロジーモデル
c       Model_No.32 = 17            !  両端、中央アナロジーモデル
c       Model_No.51 = 51            !  3 次元プレテンション動作モデル
c       Model_No.1001 = 1001        !  DLL 有限要素弾塑性モデル
c    解析パラメータ
c      structure  / parameter_s/
c      integer    n_unknown        !  全自由度
c      integer    n_point          !  節点数
c      integer    n_element        !  要素数
c      integer    n_element_dll    !  DLL 用要素数
c      integer    n_member         !  部材数
c      integer    n_rot_axis       !  主軸回転部材数
c      integer    n_local_coord    !  局所座標系を使用する節点数
c      integer    n_boundary_p     !  境界節点数
c      integer    nc_member        !  部材減衰機構を有する部材数
c      integer    n_member_dll     !  DLL 用部材数
c      integer    n_free           !  節点当たり解析自由度数
```

```
c      integer    n_dim           !  解析次元数
c      integer    n_skyline       !  スカイライン行列の領域数
c      integer    n_sky_ave       !  平均バンド幅
c      end structure
c      record /parameter_s/ Parameter_C
c
c           Parameter_C          structure
c           Point                structure
c           Member               structure
c           ak_linear            real*8   線形剛性行列
c-----------------------------------------------------------------
      n_member = Parameter_C.N_member                                  ! 1
      do i=1,n_member
      mem = i
      iet  = Member(i).element_type                                    ! 2
      ie = Member(i).nm_element                                        ! 3
      iett=(iet-1)/10                                                  ! 4
      ien= Member(i).n_model_type                                      ! 5
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL要素        ! 6
      if(iett.eq.0)then                                                ! 7
      goto(11,12,13,14,15,16,17,18,19,20),iet                          ! 8
   11 continue
c-------------------------------★Model_No.1 通常の有限要素弾性モデル
      call Cal_lin_stiff_M1(Member(i),Element(ie),ak_linear(1,1,i))    ! 9
      if(Member(i).i_rigid_length.ne.0..or.
     *   Member(i).j_rigid_length.ne.0.)
     *    call Deal_Rigid_element(ak_linear(1,1,i),
     *              Member(i).i_rigid_length,Member(i).j_rigid_length)  ! 10
      goto 100
   12 continue
c-------------------------------★Model_No.2 3次元せん断弾塑性モデル
      call Cal_lin_stiff_M2(Member(i),Element(ie),ak_linear(1,1,i))    ! 11
      goto 100
   13 continue
c-------------------------------★Model_No.3 3次元軸力弾塑性モデル
      iee = Member(i).n_model_type
      call Cal_lin_stiff_M3(Member(i),Element(ie),
     *                  ak_linear(1,1,i))
      goto 100
   14 continue
c-------------------------------★Model_No.4 3次元ケーブル弾塑性モデル
      call Cal_lin_stiff_M4(Member(i),Element(ie),ak_linear(1,1,i))
      goto 100
   15 continue
c-------------------------------★Model_No.5 3次元免振モデル
      call Cal_lin_stiff_M5(Member(i),ak_linear(1,1,i),
     *      Model_type.cosin(1,1),Model_type.n_spring)
      goto 100
   16 continue
c-------------------------------★Model_No.6 3次元制震 Maxwell モデル
      call Cal_link_maxwelldamp(ak_linear(1,1,i))
      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_lin_stiff_M7(Member(i),Element(ie),ak_linear(1,1,i))
       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                                              ! 12
       goto(111,112,113,114,115,116,117,118,119,120),iet-10               ! 13
  111 continue
c――――――――――――――――――――――――――――――――★Model_No.11 両端ファイバーモデル
       call Cal_lin_stiff_M11(Model_type,Member(i),Element(ie),            ! 14
     *      ak_linear(1,1,i) ,E_model11, E_model_fiber,
     *      M_model11, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
       goto 100
  112 continue
c――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
       call Cal_lin_stiff_M12(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model12, E_model_fiber,
     *      M_model12, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
       goto 100
  113 continue
c――――――――――――――――――――――――――――――――★Model_No.13 両端 MS モデル
       call Cal_lin_stiff_M21(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model21, E_model_fiber,
     *      M_model21, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
       goto 100
  114 continue
c――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
       call Cal_lin_stiff_M22(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model22, E_model_fiber,
     *      M_model22, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
       goto 100
  115 continue
c――――――――――――――――――――――――――――――――★Model_No.15 幾何学非線形+弾塑性型有限要素モデル
       call Cal_lin_stiff_M15(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model15, E_model_fiber,
     *      M_model15, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
       goto 100
  116 continue
c――――――――――――――――――――――――――――――――★Model_No.16 両端アナロジーモデル
       call Cal_lin_stiff_M31(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model31, E_model_fiber,
     *      M_model31, M_model_fiber)
```

```
      goto 100
  117 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.17 両端、中央アナロジーモデル
      call Cal_lin_stiff_M32(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model32, E_model_fiber,
     *      M_model32, M_model_fiber)
      goto 100
  118 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.18 両端ピン、中央ファイバーモデル
      call Cal_lin_stiff_M13(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model13, E_model_fiber,
     *      M_model13, M_model_fiber,
     *      Bilinear_work,Trilinear_work,Concrete_work)
      goto 100
  119 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.19 両端ピン、中央アナロジーモデル
      call Cal_lin_stiff_M33(Model_type,Member(i),Element(ie),
     *      ak_linear(1,1,i) ,E_model33, E_model_fiber,
     *      M_model33, M_model_fiber)
      goto 100
  120 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.20
      goto 100
      endif
      goto 100
 9999 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.DLL
c      call Cal_lin_stiff_dll(mem,
c     *   work1_elememt,work2_elememt,work1_member,work2_member)
  100 continue
      end do
      return
      end
```

部材モデルの階層構造で、サブルーチン Cal_stiff_linear()で表した部分が最上位層であり、SPACE に組み込まれた部材モデルが直接表示されているのが分かる。この最上位層の構造は、非常に単純で、モデル番号 Member(i).element_type と呼ぶパラメータで分類し、該当する番号でそのモデルの剛性が計算されるようになっている。

部材モデルの次の層に話題を移す前に、このサブルーチンの説明を行うことにする。

１．構造体から部材数を取得し、以下の処理をその部材数分行う。
２．部材の要素のモデル番号 Member(i).element_type を取得する。
３．部材の要素番号 Member(i).nm_element を取得する。
４．部材モデル番号の2桁目の値を求める。この値を用いて、10個毎、部材モデルを分類する。

５．部材モデル別の通し番号 Member(i).n_model_type を取得する。
６．その部材モデルが DLL で定義したモデルである場合は、9999 に飛ぶ。ただし、現在は使用不可となっている。
７．部材モデル番号が１桁の場合、以下の処理を行う。
８．部材モデル番号にしたがって、処理を分類する。
９．最初に分類される部材モデルは有限要素モデルで、幾何学的非線形性を考慮した弾性部材である。ここでは、この部材モデルの線形剛性を求めるサブルーチン Cal_lin_stiff_M1()をコールする。
10. さらに、部材両端もしくはどちらかに剛域を有する場合は、サブルーチン Deal_Rigid_element()をコールし、剛性行列を変換する。上記の２つのサブルーチンについては次章で説明する。
11. 次の部材モデルは、3 次元のせん断弾塑性モデルであり、この部材の線形剛性を求めるために、サブルーチン Cal_lin_stiff_M2()をコールする。以下同様に分類され、各モデルに関する処理が行われる。なお、モデル番号 7,8,9,10 は、現在欠番となっており、将来ここに新たな部材モデルが設定される予定となっている。
12. 部材モデル番号が 11 番から 20 番までの処理を行う。
13. 部材モデル番号によって処理を分類する。
14. 静的縮合モデルである両端ファイバーモデルの線形剛性を、サブルーチン Cal_lin_stiff_M11()で求める。以下同様に分類され、各モデルに関する処理が行われる。

同様に、予備計算の中で、部材モデルの最上位層を利用しているサブルーチンとして、Cal_damp_linear()がある。このサブルーチンを示し、この層の構成を再確認しよう。

上記の Cal_stiff_linear()と異なる点は、プログラム右のコメント１の部分で示すように、その部材モデルが減衰項を有しているかどうかチェックしていることと、現在の部材モデルで減衰項を含むのは、コメント 2 で示すように Maxwell モデルのみであることである。他のモデルは全てダミーの処理をなっている。Maxwell モデルでは、サブルーチン Cal_lin_maxwelldamp()で減衰行列を評価している。減衰行列を保存する配列 ac_linear()は、部材の構造体 Member(i).nm_damp の連続番号で管理されている。そのモデルのデータを保存する構造体 E_model6_real は、モデル別通し番号を示す構造体の成分 Member(i).n_model_type で管理される。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_damp_linear
C       ―――――――――――――――――――――――――――――――――――
C     ●   線形減衰行列の計算(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Cal_damp_linear(Element,Member,Parameter_C,ac_linear,
     *           E_model6_real,
     *           work1_elememt,work2_elememt,work1_member,work2_member)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record /parameter_s  / Parameter_C
      record / member_s    / Member
      record / element_s   / Element
      record /E_model6_real_s / E_model6_real
      dimension Member(*),Element(*)
      dimension E_model6_real(*)
      dimension ac_linear(12,12,*)
c―――――――――――――――――――――――――――――――――――――――――――――
      n_member = Parameter_C.N_member
      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                                  ! 1
      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
      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
      ienn=Member(i).nm_damp
      ii=Element(ie).nm_type
      call Cal_lin_maxwelldamp(E_model6_real(ien),ac_linear(1,1,ienn),ii)          ! 2
      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ル
      goto 100
  116 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.16 3 次元プレテンション動作モデル
      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
      endif
      goto 100
 9999 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.DLL
c      call Cal_lin_damp_dll(mem,
c     *   work1_elememt,work2_elememt,work1_member,work2_member)

  100 continue
      endif
      end do
      return
      end
```

4.13 部材応力

本節では、部材の応力状態を求めるサブルーチン Cal_stress()について説明する。ここでも、前節と同様に部材モデルの最上位層を利用して、部材モデルが分類されている。まず、このサブルーチンの内容を示す。やはり、前節の２つのサブルーチンとほとんど同様の構造を持っていることが分かる。

```
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―――――――――――――――――――――――――――――――――――――――――――――――
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―――――――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
c――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12                                                    ! 1
      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)   ! 2
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp) ! 3
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 通常の有限要素弾性モデル
      call Cal_stress_M1(Member(i),Element(ie),                    ! 4
     *     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
      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))
      goto 100
   14 continue
c-------------------------------★Model_No.4 3 次元ケーブル弾塑性モデル
      call Cal_stress_M4(Member(i),Element(ie),vv)
      goto 100
   15 continue
c-------------------------------★Model_No.5 3 次元免振モデル
      call Cal_stress_M5(Member(i),Element(ie),
     *     ak_nonlinear(1,1,i),v,rot_memb(1,1,1,i),rot_memb(1,1,2,i) )
      goto 100
   16 continue
c-------------------------------★Model_No.6 3 次元制震 Maxwell モデル(ok)
       do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=past_vel_point(ires)
      else
      v(j)=0.
      endif
      enddo
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      ii=Element(ie).nm_type
      call Stress_maxwelldamp(vpp,vv,E_model6_real(ien),
     *             Element(ie),ii,Member(i),Model_type.n_m_filter)
      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_stress_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 )
      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
      call Cal_stress_M11( Model_type ,Member(i),Element(ie),                    ! 5
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  112 continue
c――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
       call Cal_stress_M12( Model_type ,Member(i),Element(ie),                   ! 6
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      do j=1,6                                                                  ! 7
      k=j+12
      Member(i).stress(k)=(Member(i).stress(j)+
     *                 Member(i).stress(j+6))*0.5
      enddo
      goto 100
  113 continue
c――――――――――――――――――――――――――――――――★Model_No.13 両端 MS モデル
      call Cal_stress_M21( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  114 continue
c――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
      call Cal_stress_M22( 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
  115 continue
c――――――――――――――――――――――――――――――――★Model_No.15 幾何学非線形+弾塑性型有限要素モデル
      call Cal_stress_M15( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  116 continue
c――――――――――――――――――――――――――――――――★Model_No.16
      call Cal_stress_M31( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  117 continue
```

```
c――――――――――――――――――――――――――――――――★Model_No.17
      call Cal_stress_M32( 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
  118 continue
c――――――――――――――――――――――――――――――――★Model_No.18
       call Cal_stress_M13( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  119 continue
c――――――――――――――――――――――――――――――――★Model_No.19
      call Cal_stress_M33( Model_type ,Member(i),Element(ie),
     *      ak_nonlinear(1,1,i) ,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i))
      goto 100
  120 continue
c――――――――――――――――――――――――――――――――★Model_No.20
      goto 100
      endif
      goto 100
 9999 continue
c――――――――――――――――――――――――――――――――★Model_No.DLL
c      call Cal_stress_dll(mem,Member(i),Element(ie),
c     *   work1_elememt,work2_elememt,work1_member,work2_member,
c     *     vv,ak)
  100 continue
      end do
      return
      end
```

ここでは、前節の線形剛性を求めるサブルーチンと異なった部分について説明しよう。

1. 各部材両端の変位を取り出す。部材の拘束条件 Member(i).irest(j) をチェックし、部材両端の増分前の変位と増分変位をセットする。
2. 増分変位 v を釣合座標系から部材座標系 vv に変換する。
3. 増分前の変位 vp を釣合座標系から部材座標系 vpp に変換する。
4. 部材モデル番号にしたがって処理を分類し、ここでは、有限要素弾性モデルの応力が求められる。
5. 部材モデル番号にしたがって両端ファイバーモデルの応力が求められる。
6. 部材モデル番号にしたがって両端・中央ファイバーモデルの応力が求められる。

7. ここでは、部材中央の応力は部材両端の応力の平均をセットしておく。ただし、部材弾塑性チェックで部材中央の応力を求め、ここにセットすることになる。

以下に示す２つのサブルーチンは、部材の応力を計算するサブルーチンであり、他の部材応力を計算するサブルーチンも、これらのコードとほぼ同じである。詳細は付録を参照されたい。

このプログラムコードに見られるように、SPACE では、部材両端の応力は縮合部材モデルも含めて接線剛性行列を用いて得た増分材端力と力の釣合から求めており、実際の部材内部の応力を計算して求めたものではない。これは、部材内部の応力、例えば両端のファイバーから積分して求めた合応力を両端の応力として設定すると、非線形性の強い領域で、隣接する部材のファイバーから得た合応力との誤差が生じるためである。曲げモーメントなどをそのまま図形表示すると、部材の端部節点で多少の不連続が生じ、ユーザーに誤解を与えてしまう恐れがある。そこで、SPACE ではこのような方法を取ることにしたものである。

```
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)
      implicit real*8(A-H,O-Z)
      include "submain.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――――――――――――――――――――――――――――――――――★線形応力の計算(ok)
      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
```

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_stress_M12（両端、中央ファイバーモデル）
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_M12(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
```