4.4.3 反復解法

次に、処理３の「反復解法」について述べる。非線形の振動方程式は以下のように与えられている。

$$\begin{aligned}&\left[[M]+\mu_1[C]+\mu_2[K]\right]\{\ddot{y}_{n+1}\}\\&\quad=-[M][I]\{\ddot{u}_g\}+\{P_S\}-\{Q(y_n)\}-[C]\{a\}-[K]\{\bar{b}\}\\&\qquad-\{\bar{f}_d\}-[K_T(y_n)]\{\Delta y_{n+1}\}+[K]\{y_{n+1}\}\end{aligned}\quad\cdots\cdots\cdots\cdots\cdots\cdots(4.7)$$

反復解法の左辺は既に LDU 分解されており、ここでは、主に右辺の定数ベクトルを作成することが仕事となる。右辺項は、以下のように、２つに分けられており、一つは、増分後の加速度、速度、変位に関連しない項$\{g\}$、他の一つは、関連する項$\{G\}$である（前章の 3.7.3 節の式(3.54)を参照)。

$$\left.\begin{aligned}\{G\}&=-\{\bar{f}_d\}-[K_T(y_n)]\{\Delta y_{n+1}\}+[K]\{y_{n+1}\}\\\{g\}&=-[M][I]\{\ddot{u}_g\}+\{P_S\}-\{Q(y_n)\}\\&\quad-[\bar{C}]\{a\}-[K]\{\bar{b}\}\end{aligned}\right\}\cdots\cdots\cdots\cdots\cdots\cdots(4.8)$$

ここで、
$[\bar{C}]=[C]+[C_0]$
である。
$[C_0]$はMaxwell モデルの線形減衰項
$[C]$はその他の減衰項

この増分加速度に関連する項$\{G\}$は、反復計算を行う過程で加速度を予測し、ニューマークβ法で変位と速度を求め、その値を用いて反復ループの中で常に再計算を行って求められる。以下に、プログラムの骨組みを示す。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★        非線形解析
c
c―――――――――――――――――――――――――――――――――――――――
 9981 continue
      if(Iteration_method .eq. 2) goto 9982
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(ld_point )                                    ! 3.1
c――――――――――――――――――――――――――――――★Work(a,b)ベクトルのセット(ok)
      call Set_a_b_vec()                                           ! 3.2
c――――――――――――――――――――――――――――――★部材節点力のセット(ok)
      call Get_pointforce_ld(ld_point,Member,n_member)              ! 3.3
c――――――――――――――――――――――――――――――★地震加速度セット(ok)
      acc1=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)       ! 3.4
      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()                                                           ! 3.5
c――――――――――――――――――――――★整合質量に関する慣性項
      call Add_earth2_ld()                                                           ! 3.6
c―――――――――――――――――――――――――――★節点荷重のセット(ok)
      p1=Get_Ps(T,1,fdd_point,Dynamic_load)                                          ! 3.7
      p2=Get_Ps(T,2,fdd_point,Dynamic_load)
      p3=Get_Ps(T,3,fdd_point,Dynamic_load)
      call Add_point_ld()
c―――――――――――――――――――――――――――★線形減衰項計算(ok)
c――――――――――――――――――――――★集中質量(ok)
      call Add_damp1_ld_ex()                                                         ! 3.8
c――――――――――――――――――――――★整合質量(ok)
      call Add_damp2_ld_ex()                                                         ! 3.9
c――――――――――――――――――――――★部材減衰(ok)
      call Add_damp3_ld_ex()                                                         ! 3.10
c―――――――――――――――――――――――――――★線形剛性項計算(ok)
c                                                     レーリー減衰も含む
      call Add_stiff1_ld ()                                                          ! 3.11
c―――――――――――――――――――――――――――★Δｔ秒後の変位と速度を予測(ok)
      n_err_roop = 0
 9991 continue
      call Estimate_disp_vel()                                                       ! 3.12
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         反復計算開始
c
c―――――――――――――――――――――――――――――――――――――――
      do iroop=1,n_roop                                                              ! 3.13
c―――――――――――――――――――――――――――★反復に関連する右辺ベクトルのゼロセット(ok)
      call Clear_vec(ld_point_repeat)                                                ! 3.14
c―――――――――――――――――――――――――――★線形剛性に関するベクトル(ok)
      call Add_stiff2_ld()                                                           ! 3.15
c―――――――――――――――――――――――――――★接線剛性に関する増分ベクトル(ok)
      call Add_tan_stiff_ld()                                                        ! 3.16
c―――――――――――――――――――――――――――★Maxwell 型モデルの計算(ok)
      call Add_fdd_ld()                                                              ! 3.17
c―――――――――――――――――――――――――――★右辺２項の和を取る(ok)
      call add_vec()                                                                 ! 3.18
c―――――――――――――――――――――――――――★線形方程式を解く(ok)
      call solve_sky()                                                               ! 3.19
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()                                                            ! 3.20
c―――――――――――――――――――――――――――★収束したかチェック(ok)
      if(ICheck_error().eq. 0) goto 9980                                             ! 3.21
c―――――――――――――――――――――――――――★次の増分値を予測(ok)
      call Estimate_disp_vel()                                                       ! 3.22
      end do
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         収束しなかった時の後処理
c
c―――――――――――――――――――――――――――――――――――――――
      write(76,*) '  収束エラー発生、陰解法処理開始'
```

```
n_iterate = 9999
9982 continue
```

反復計算を行っている部分の処理内容を、以下で説明する。

3.1：最初に、釣合式の右辺項ベクトル$\{g\}$を配列 ld_point に求める。まず、この配列をゼロセットする。

3.2：ワーク用として以下に示す$\{a\}$ベクトルと$\{b\}$ベクトルを計算する。

$$\{a\} = \{\dot{y}_n\} + \Delta t(1-\delta)\{\ddot{y}_n\}$$
$$\{b\} = \Delta t\{\dot{y}_n\} + \Delta t^2(0.5-\beta)\{\ddot{y}_n\}$$

3.3：最初に、増分後の加速度、速度、変位に関連しない右辺項を求める。まず、部材節点力$\{Q(y_n)\}$を右辺項ベクトル ld_point に足しこむ。

3.4：時刻 T の x 方向、y方向、z方向の地震加速度$\{\ddot{u}_g\}$を求める。

3.5：集中質量に関する慣性項$[M][I]\{\ddot{u}_g\}$を計算し、右辺項ベクトル ld_point に足し込む。

3.6：部材分布質量に関する慣性項$[M][I]\{\ddot{u}_g\}$を計算し、右辺項ベクトル ld_point に足し込む。

3.7：x 方向、y方向、z方向の静的荷重$\{P_S\}$を求め、右辺項ベクトル ld_point に足し込む。

3.8：レーリー減衰の集中質量に関する減衰項$[C]\{a\}$を求め、右辺項ベクトル ld_point に足し込む。

3.9：レーリー減衰の部材分布質量に関する減衰項$[C]\{a\}$を求め、右辺項ベクトル ld_point に足し込む。

3.10：部材減衰に関する減衰項$[C]\{a\}$を求め、右辺項ベクトル ld_point に足し込む。Maxwell モデルの線形減衰項を含む。

3.11：線形の剛性項$[K]\{\bar{b}\}$を計算し、右辺項ベクトル ld_point に組み込む。同時に、レーリー減衰の係数が構造体 Newmark_P にセットされており、これを利用して$[C]\{a\}$の剛性に関する項を右辺項ベクトル ld_point に足し込む。

3.12：反復計算のために、Δt 秒後の加速度、速度、変位を予測する。

3.13：反復計算を開始し、増分後の加速度、速度、変位に関連する右辺項を求める。

3.14：これ以降の処理は、反復計算のための右辺項ベクトル$\{G\}$を配列 ld_point_repeat に足しこむ。まず、この配列をゼロクリアする。

3.15：線形剛性に関する右辺項ベクトル$[K]\{y_{n+1}\}$を計算し、ld_point_repeat に足し込む。

増分加速度に関連しない項$\{g\}$は、次のプログラム番号で求められる。

$\{g\}=$

$-[M][I]\{\ddot{u}_g\}$ (3.5),(3.6)

$+\{P_S\}$ (3.7)

$-\{Q(y_n)\}$ (3.3)

$-[\bar{C}]\{a\}$ (3.8),(3.9)

$-[K]\{\bar{b}\}$ (3.10) ,(3.11) (3.11)

増分加速度に関連する項$\{G\}$は、次のプログラム番号で求められる。

$\{G\}=$

$-\{\bar{f}_d\}$ (3.17)

$-[K_T(y_n)]\{\Delta y_{n+1}\}$ (3.16)

$+[K]\{y_{n+1}\}$ (3.15)

3.16：接線剛性に関する右辺項ベクトル$\left[K_T(y_n)\right]\{\Delta y_{n+1}\}$を計算し、ld_point_repeat に足し込む。

3.17：Maxwell モデルによるベクトル$\left\{\bar{f}_d\right\}$を計算し、ld_point_repeat に足し込む。

3.18：右辺定数ベクトル$\{g\}$：ld_point と反復計算用ベクトル$\{G\}$：ld_point_repeat の和を取り、その結果を ld_point_repeat にセットする。

3.19：得られた右辺項 ld_point_repeat を用いて、方程式を解くサブルーチン solve_sky()より解を得る。

3.20：得られた加速度より、Newmarkβ法の基本式より、速度と変位を求める。

3.21：反復解が収束したか否かをチェックする。収束した場合は次のステップへ進む。

3.22：収束していない場合は、加速度、速度、変位を予測し直す。もし、指定したループ数で収束しない場合は、陰解法に制御が移ることになる。

以上が、処理３の「反復解法」である。さらに、理解を深めるために主要なサブルーチンを以下に示す。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Set_a_b_vec
C       ―――――――――――――――――――――――――――――――――
C     ●    Work ベクトルのセット
C     ―――――――――――――――――――――――――――――――――
      subroutine Set_a_b_vec(n_unknown,a_vector,b_vector,Newmark_P,
     *                              past_vel_point, past_acc_point)
C     ―――――――――――――――――――――――――――――――――
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension past_vel_point(*),past_acc_point(*)
      dimension a_vector(*),b_vector(*)
      record / newmark_s  / Newmark_P
      ddt_1= Newmark_P.ddt_1                    ! (1-δ)Δt
      dt   = Newmark_P.dt                       ! Δt
      bdt_5= Newmark_P.bdt_5                    ! (0.5-β) Δt2
      do i=1,n_unknown
      a_vector(i)=past_vel_point(i) + ddt_1*past_acc_point(i)
      b_vector(i)=dt*past_vel_point(i) + bdt_5*past_acc_point(i)
      enddo
      return
      end
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)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      dimension Member(*)
      real*8 ld_point(*)
C     ―――――――――――――――――――――――――――――――――
c           ld_point      :real*8 右辺項
c           Member        :structure
c           n_member      :integer 部材数
C     ―――――――――――――――――――――――――――――――――
      do i=1,n_member
      do j=1,12
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
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - Member(i).force(j)
      endif
      end do
      end do
      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(3),amm(3),uu(3)
      dimension a_vector(*)
c―――――――――――――――――――――――――――――――――――――――――――――――
c     nx                      :integer  第一段階か第二段階か
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-----------------------------------------------------------
      if(Parameter_C.n_local_coord.ne.0) then
      do i=1,n_point
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      if(am.eq.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
      ij=Point(i).local_coord
      if(ij.eq.0) then
      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
      else
      do j=1,3
      uu(j)=amm(j)*u(j)
      enddo
      call RotateTL_amm(uu,rot_local(1,1,ij),u)
       do  j=1, 3
      i1=Point(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - u(j)
      end do
      endif
      endif
      end do
c-----------------------------------------------------------
      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 /RotateTL_amm
C       -------------------------------------------------
C     ●   質量行列の座標変換
C     ---------------------------------------------------
      subroutine RotateTL_amm(am,ri,amm)
      implicit real*8(A-H,O-Z)
      dimension am(3),ri(3,3),amm(3)
c-----------------------------------------------------------------
      do i=1,3
      sum=0.0
      do j=1,3
      sum=sum+ri(j,i)*am(j)
      enddo
      amm(i)=sum
      enddo
      return
      end
C     ---------------------------------------------------
C     ●   SUBROUTINE /Add_stiff1_ld
C       -------------------------------------------------
C     ●   線形剛性項に関するベクトルを加える。その１(ok)
C     ---------------------------------------------------
      subroutine Add_stiff1_ld(n_istep,ld_point,Member,
     *       n_member,past_disp_point,past_vel_point,past_acc_point,
     *       ak_linear,Newmark_P)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      record / newmark_s  / Newmark_P
      dimension Member(*)
      real*8 ld_point(*),past_disp_point(*)
      dimension past_vel_point(*),past_acc_point(*)
      dimension u(12),ak_linear(12,12,*)
c-----------------------------------------------------------------
c     n_istep              : integer   第一段階か第二段階か
c     ld_point(*)          : real*8    線形右辺項ベクトル
c     Member               : structure
c     n_member             : integer   部材数
c     past_disp_point      : real*8    Δｔ秒前の節点変位
c     past_vel_point       : real*8    Δt 秒前の節点速度
c     past_acc_point       : real*8    Δt 秒前の節点加速度
c     ak_linear(*)         : real*8    線形剛性行列
```

```
c     Newmark_P             : structure
c-----------------------------------------------------------------
      if(n_istep.eq.1) then
c        剛性                  減衰
      a=Newmark_P.dt    + Newmark_P.alf1_2
      b=Newmark_P.bdt_5 + Newmark_P.alf1_2*Newmark_P.ddt_1
      else
      a=Newmark_P.dt    + Newmark_P.alf2_2
      b=Newmark_P.bdt_5 + Newmark_P.alf2_2*Newmark_P.ddt_1
      endif
      do i=1,n_member
      do j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      u(j)=past_disp_point(irest)+a*past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      sum=0.0
      do  k=1,12
      sum=sum+ak_linear(j,k,i)*u(k)
      end do
      ld_point(irest)=ld_point(irest) - sum
      end if
      end do
      end do
      return
      end
C     ------------------------------------------------------
C     ●   SUBROUTINE /Add_stiff2_ld
C       ----------------------------------------------------
C     ●   線形剛性項に関するベクトルを加える。その２(ok)
C     ------------------------------------------------------
      subroutine Add_stiff2_ld(ld_point_repeat,
     *       Member,n_member,est_disp_point,ak_linear)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s  /Newmark_P
      record / member_s   / Member
      dimension Member(*)
      real*8 ld_point_repeat(*),est_disp_point(*)
      dimension u(12),ak_linear(12,12,*)
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_disp_point        :real*8    予測節点変位
c     ak_linear(*)          :real*8    線形剛性行列
c-----------------------------------------------------------------
```

```
      do i=1,n_member
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)= est_disp_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      sum=0.0
      do  k=1,12
      sum=sum+ak_linear(j,k,i)*u(k)
      end do
      ld_point_repeat(irest)=ld_point_repeat(irest)+sum
      end if
      end do
      end do
      return
      end
C     ――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_stiff2x_ld
C       ――――――――――――――――――――――――――――――――――
C     ●   線形剛性項に関するベクトルを加える。その２(ok)
C     ――――――――――――――――――――――――――――――――――
      subroutine Add_stiff2x_ld(ld_point_repeat,
     *       Member,n_member,result_acc_point,ak_linear,Newmark_P)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s  /Newmark_P
      record / member_s   / Member
      dimension Member(*)
      real*8 ld_point_repeat(*),result_acc_point(*)
      dimension u(12),ak_linear(12,12,*)
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     result_acc_point      :real*8    予測節点加速度
c     ak_linear(*)          :real*8    線形剛性行列
c――――――――――――――――――――――――――――――――――――――――――――
      a =  Newmark_P.bdt
      do i=1,n_member
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)= a * result_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
```

```
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      sum=0.0
      do  k=1,12
      sum=sum+ak_linear(j,k,i)*u(k)
      end do
      ld_point_repeat(irest)=ld_point_repeat(irest)+sum
      end if
      end do
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_tan_stiff_ld
C       ―――――――――――――――――――――――――――――――――
C     ●   接線剛性項に関するベクトルを加える。その２(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Add_tan_stiff_ld(ld_point_repeat,
     *       Member,n_member,est_ddisp_point,ak_nonlinear)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      dimension Member(*)
      real*8 ld_point_repeat(*)
      dimension u(12),ak_nonlinear(12,12,*)
      dimension est_ddisp_point(*)
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_ddisp_point       :real*8    予測増分変位
c     ak_nonlinear(*)       :real*8    接線剛性行列（釣合系）
c―――――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
      ipp=0
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)=est_ddisp_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      sum=0.0
      do  k=1,12
      sum=sum+ak_nonlinear(j,k,i)*u(k)
      end do
      ld_point_repeat(irest)=ld_point_repeat(irest) - sum
      end if
      end do
      end do
```

```
      return
      end
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_tan_stiff2_ld
C       ―――――――――――――――――――――――――――――――――
C     ●   接線剛性項に関するベクトルを加える。その３(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Add_tan_stiff2_ld(ld_point_repeat,
     *       Member,n_member,result_acc_point,ak_nonlinear,Newmark_P)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s  /Newmark_P
      record / member_s   / Member
      dimension Member(*)
      real*8 ld_point_repeat(*)
      dimension u(12),ak_nonlinear(12,12,*)
      dimension result_acc_point(*)
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     result_acc_point      :real*8    予測加速度
c     ak_nonlinear(*)       :real*8    接線剛性行列（釣合系）
c―――――――――――――――――――――――――――――――――――――――――――――
      a =  Newmark_P.bdt
      do i=1,n_member
      ipp=0
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)=a * result_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      sum=0.0
      do  k=1,12
      sum=sum+ak_nonlinear(j,k,i)*u(k)
      end do
      ld_point_repeat(irest)=ld_point_repeat(irest) - sum
      end if
      end do
      end do
      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)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s   / Member
      record / e_model6_real_s / E_model6_real
      record /element_s / Element
      dimension Member(*),E_model6_real(*),Element(*)
      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
      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
      call RotateL_v(1,ud,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
      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
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
 9999 continue
c――――――――――――――――――――――――――――――――★Model_No.DLL
c      call Cal_lin_damp_dll(mem,
c     *   work1_elememt,work2_elememt,work1_member,work2_member)
  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)
```

```
      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
C     ―――――――――――――――――――――――――――――――――――
C     ●   FUNCTION /Check_error
C       ―――――――――――――――――――――――――――――――――――
C     ●   収束判定を行う関数(ok)
C     ―――――――――――――――――――――――――――――――――――
      integer function ICheck_error(ikai,n_point,Point,n_unknown,
     *               result_disp_point,est_disp_point, Newmark_P)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s     / Newmark_P
      record /point_s        / Point
      dimension result_disp_point(*),est_disp_point(*)
      dimension Point(*)
C     ―――――――――――――――――――――――――――――――――――
c          Check_error         :integer    0 ; 収束  1: 収束せず
c          n_unknown           :integer  未知数
c          result_disp_point   :real*8   計算結果節点変位
c          est_disp_point      :real*8   予測節点変位
c          Newmark_P           :structure
c          Newmark_P.eps_v     :real*8   計算誤差閾値
C     ―――――――――――――――――――――――――――――――――――
      ICheck_error=1
      Error_dat=0.
      dmax=0.
      do i=1,n_point
      do j=1,6
      ires= Point(i).irest(j)
      if(ires.ne.0) then
      er=dabs(result_disp_point(ires)-est_disp_point(ires))
      if(er.gt.dmax) then
      dmax=er
      iim=i
      jjm=j
      endif
      Error_dat=Error_dat+er
      end if
      end do
      end do
      Error_dat=Error_dat/n_unknown
      write(76,'(a,i4,a,2e12.4,a,2i4,a,e12.4)') ' Iteration No.：',ikai,
     *               '   Err. Norm:',Error_dat,Newmark_P.eps_v,
     *               '   Max. Position：',iim,jjm,
     *               '   Disp. :',dmax
      if(Error_dat .le. Newmark_P.eps_v) ICheck_error=0
```

```
      return
      end
```

4.4.4 陰解法

次に、処理４の「陰解法」について述べる。ここでの処理は、２つに分かれている。最初は、陰解法についての処理であり、その後、反復解法に戻るために､反復解法に必要な左辺項の係数行列を求め､その後 LDU 分解する処理が続くことになる。陰解法での処理は、左辺項の係数行列を計算し、LDU 分解することと、右辺項ベクトルを計算し、方程式の解を得ることである。無論、左辺項の係数行列は、非線形の接線剛性を含む。以下に、プログラムの骨組みを示すので、処理の流れを理解して頂きたい。

陰解法の振動方程式(3.23)にMaxwell モデルの節点力ベクトルを加えると、方程式は

$$\begin{aligned}&\left[\left[M\right]+\mu_1\left[\bar{C}\right]+\mu_2\left[K_T(y_n)\right]\right]\{\ddot{y}_{n+1}\}\\&\quad=-\left[M\right]\left[I\right]\{\ddot{u}_g\}+\{P_S\}-\{Q(y_n)\}\\&\qquad-\{f_d\}-\left[C\right]\{a\}-\left[K_T(y_n)\right]\{b\}\end{aligned}\qquad\cdots\cdots(4.9)$$

となり、上式を解くことになる。解析フローの骨組みを以下に示す。

```
c-----------------------------------------------------------------
c
c    ★         動的解析（陰解法）
c
c-----------------------------------------------------------------
c-----------------------------------------------------------------
c
c    ★         係数行列の計算と分解
c
c-----------------------------------------------------------------
c      if(istep.eq.1.or.istep.eq.Newmark_P.n2_step) then
        n_istep=1
      if(istep.ge.Newmark_P.n2_step) n_istep=2
c      endif
c-------------------------★スカイライン行列のゼロセット
      call Set_sky_zero(gskym)                                  ! 4.1
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()                                       ! 4.2
c-------------------------★部材の整合質量系の行列への足し込み
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()                                    ! 4.3
c---------------------★整合質量の釣合座標系への変換
      call Rotate_mass()                                        ! 4.4
```

```
c――――――――――――――――――――――――★整合質量系の足し込み
      call Build_sky_m()                                              ! 4.5
      endif
c――――――――――――――――――――――――――★部材非線形減衰の足し込み(含む Maxwell 型)
      if(Parameter_C.nc_member .ne. 0) then
      call Build_sky_c_ex()                                           ! 4.6
      endif
c――――――――――――――――――――――――――★線形剛性によるレーリー減衰
c――――――――――――――――――――――――――★接線剛性の足し込み
      call Build_sky_kk()                                             ! 4.7
c――――――――――――――――――――――――――――――――★行列の LDU 分解
      call decomp_sky()                                               ! 4.8
c――――――――――――――――――――――――――――――――★分解成功か？
      if(iexit.ne.0) then                                             ! 4.9
      write(damp_out,*) ' decomp_sky err',iexit
      return                                                          ! 4.10
      endif
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        右辺項計算
c
c――――――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――★右辺の定数ベクトルゼロセット(ok)
      call Clear_vec(ld_point )                                       ! 4.11
c――――――――――――――――――――――――――――――――★Work(a,b)ベクトルのセット(ok)
      call Set_a_b_vec()                                              ! 4.12
c――――――――――――――――――――――――――――――――★部材節点力のセット
      call Get_pointforce_ld()                                        ! 4.13
c――――――――――――――――――――――――――――――――★地震加速度セット
      acc1=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)         ! 4.14
      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――――――――――――――――――――――――――★集中質量に関する慣性項
      call Add_earth1_ld()                                            ! 4.15
c――――――――――――――――――――――――――★整合質量に関する慣性項
      call Add_earth2_ld()                                            ! 4.16
c――――――――――――――――――――――――――――――――★節点荷重のセット
      p1=Get_Ps(T,1,fdd_point,Dynamic_load)                           ! 4.17
      p2=Get_Ps(T,2,fdd_point,Dynamic_load)
      p3=Get_Ps(T,3,fdd_point,Dynamic_load)
      call Add_point_ld()
c――――――――――――――――――――――――――――――――★線形減衰項計算(ok)
c――――――――――――――――――――――――――★集中質量(ok)
      call Add_damp1_ld_ex()                                          ! 4.18
c――――――――――――――――――――――――――★整合質量(ok)
      call Add_damp2_ld_ex()                                          ! 4.19
c――――――――――――――――――――――――――★部材減衰（Maxwell 型モデル）
      call Add_damp3_ld_ex()                                          ! 4.20
c――――――――――――――――――――――――――★線形剛性によるレーリー減衰
      call Add_stiff1_ld_ex()                                         ! 4.21
c――――――――――――――――――――――――――――――――★接線剛性に関する増分ベクトル
      call Add_tan_stiff_ld()                                         ! 4.22
c――――――――――――――――――――――――――――――――★Maxwell 型モデルの右辺項 fd の計算(ok)
      call Add_fdd_ld_ex()                                            ! 4.23
```

```
c――――――――――――――――――――――――――――――――★線形方程式を解く(ok)
      call solve_sky()                                                  ! 4.24
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()                                               ! 4.25
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★         陰解法の終了チェック
c
c――――――――――――――――――――――――――――――――――――――――――――――
      if(N_implicit_method.ne.-1) then  !全て陰解法で解析する
      N_implicit_method = N_implicit_method - 1
      if(N_implicit_method.eq.0) then
      Iteration_method  = 1  !反復法に戻す
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★         反復解法のために係数行列を作成し直す
c
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
      endif
c――――――――――――――――――――――――――――――――★スカイライン行列のゼロセット(ok)
      call Set_sky_zero(gskym)                                          ! 4.26
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()                                               ! 4.27
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()                                            ! 4.28
c――――――――――――――――――――――――――――――――★整合質量の釣合座標系への変換(ok*)
      call Rotate_mass()                                                ! 4.29
c――――――――――――――――――――――――――――――――★整合質量系の足し込み(ok*)
      call Build_sky_m()                                                ! 4.30
      endif
c――――――――――――――――――――――――――――――――★部材減衰系の足し込み(ok)
      if(Parameter_C.nc_member .ne. 0) then
      call Build_sky_c()                                                ! 4.31
      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()                                                ! 4.32
c――――――――――――――――――――――――――――――――――――――――★行列の LDU 分解(ok)
      call decomp_sky()                                                 ! 4.33
c――――――――――――――――――――――――――――――――――――――――★分解成功か？
      if(iexit.ne.0) then                                               ! 4.34
      ierr_dat =100
      return
      endif
c――――――――――――――――――――――――――――――――――――――――★
```

```
      endif
      endif
c-------------------------------------------------------------
c
c    ★          計算終了・後処理開始
c
c-------------------------------------------------------------
```

プログラムの右側には番号が振られており、その番号にしたがって処理内容を説明する。

4.1：振動方程式左辺項の係数行列 gskym のゼロセットを行う。係数行列 gskym はスカイライン行列であり、以後の処理で求めた係数行列は全てこのスカイライン行列に足し込まれることになる。

4.2：最初に、節点集中質量による質量行列を足し込む。同時に、レーリー減衰の係数が構造体 Newmark_P にセットされており、これを利用して減衰行列を係数行列 gskym に足し込む。

4.3：部材分布質量があるか否かの判定処理があり、部材分布質量がある場合は、4.4、4.5 の処理を行う。ここでは部材座標系で整合質量行列を求める。

4.4：求めた整合質量行列を部材座標系から釣合座標系に座標変換する。

4.5：座標変換された整合質量行列を係数行列 gskym に足し込む。

4.6：部材減衰系の減衰行列を係数行列 gskym に足し込む。部材減衰の有無は、構造体 Parameter_C の成分 nc_member に設定されている。もし、部材減衰が存在する場合は、予備計算において釣合座標系で得られている。ここでは、Maxwell 型の部材減衰も含む。

4.7：接線剛性行列を係数行列 gskym に足し込む。同時に、レーリー減衰の係数が構造体 Newmark_P にセットされており、これを利用して線形剛性行列を係数行列 gskym に足し込む。線形の剛性行列は、予備計算で求められ、釣合座標系に変換されている。

4.8：得られた釣合式の係数行列（スカイライン行列）を LDU 分解する。

4.9：分解に成功したか否かを判定し、成功した場合は次のステップへ。

4.10：失敗した場合はエラー出力を行った後、モニターに戻る。

以後は、右辺項の計算を行い、方程式を解く。

4.11：釣合式の右辺項ベクトル ld_point をゼロセットする。

4.12：ワーク用として以下に示す $\{a\}$ ベクトルと $\{b\}$ ベクトルを計算する。

$$\{a\} = \{\dot{y}_n\} + \Delta t(1-\delta)\{\ddot{y}_n\}$$

$$\{b\} = \Delta t\{\dot{y}_n\} + \Delta t^2(0.5-\beta)\{\ddot{y}_n\}$$

陰解法の振動方程式の左辺係数行列は次のプログラム番号で行われる。

$[F]=$

$[M]$　(4.2),(4.5)

$+\mu_1[\bar{C}]$　(4.2),(4.5),(4.6),(4.7)

$\mu_2[K_T(y_n)]$　(4.7)

4.13：部材節点力 $\{Q(y_n)\}$ を右辺項ベクトル ld_point に足し込む。
4.14：時刻 T の x 方向、y 方向、z 方向の地震加速度 $\{\ddot{u}_g\}$ を求める。
4.15：集中質量に関する慣性項 $[M][I]\{\ddot{u}_g\}$ を計算し、右辺項ベクトル ld_point に足し込む。
4.16：部材分布質量に関する慣性項 $[M][I]\{\ddot{u}_g\}$ を計算し、右辺項ベクトル ld_point に足し込む。
4.17：x 方向、y 方向、z 方向の静的荷重 $\{P_S\}$ を求め、右辺項ベクトル ld_point に足し込む。
4.18：レーリー減衰の集中質量に関する減衰項 $[C]\{a\}$ を求め、右辺項ベクトル ld_point に足し込む。
4.19：レーリー減衰の整合質量に関する減衰項 $[C]\{a\}$ を求め、右辺項ベクトル ld_point に足し込む。
4.20：部材減衰（Maxwell 型を含む）に関する減衰項 $[C]\{a\}$ を求め、右辺項ベクトル ld_point に足し込む。
4.21：線形の剛性項より、$[C]\{a\}$ の剛性に関する項を右辺項ベクトル ld_point に足し込む。
4.22：接線剛性に関する増分ベクトル $[K_T(y_n)]\{b\}$ を計算し、右辺項ベクトル ld_point に足し込む。
4.23：Maxwell モデルによるベクトル $\{f_d\}$ を計算し、ld_point に足し込む。
4.24：得られた右辺項を用いて、方程式の解を得る。
4.25：得られた加速度を用いて Newmark β 法の基本式より、速度と変位を求める。

ここからは、反復解法に戻るため、反復計算用左辺項の係数行列を計算し、LDU 分解を行う。

4.26：振動方程式左辺項の係数行列 gskym のゼロセットを行う。係数行列 gskym はスカイライン行列であり、以後の計算は全てこのスカイライン行列に足し込まれることになる。
4.27：最初に、節点集中質量による質量行列を足し込む。同時に、レーリー減衰の係数が構造体 Newmark_P にセットされており、これを利用して減衰行列を係数行列 gskym に足し込む。
4.28：部材分布質量があるか否かの判定処理があり、部材分布質量がある場合は、4.29、4.30 の処理を行う。ここでは部材座標系で整合質量行列を求める。
4.29：求めた整合質量行列を部材座標系から釣合座標系に座標変換する。
4.30：座標変換された整合質量行列を係数行列 gskym に足し込む。

陰解法の振動方程式の右辺定数ベクトルは次のプログラム番号で行われる。

$\{g\} =$

$-[M][I]\{\ddot{u}_g\}$ (4.15),(4.16)

$+\{P_S\}$ (4.17)

$-\{Q(y_n)\}$ (4,13)

$-\{f_d\}$ (4.23)

$-[C]\{a\}$ (4.18),(4.19),(4.20),(4.21)

$-[K_T(y_n)]\{b\}$ (4.22)

4.31：部材減衰系の減衰行列を係数行列 gskym に足し込む。部材減衰の有無は、構造体 Parameter_C の成分 nc_member に設定されている。もし、部材減衰が存在する場合は、予備計算において釣合座標系で得られている。

4.32：線形の剛性行列を係数行列 gskym に組み込む。同時に、レーリー減衰の係数が構造体 newmark_s にセットされており、これを利用して剛性行列を係数行列 gskym に足し込む。線形の剛性行列は、予備計算で求められ、釣合座標系に変換されている。

4.33：得られた釣合式の係数行列（スカイライン行列）を LDU 分解する。

4.34：分解に成功したか否かを判定し、成功した場合は次のステップへ、失敗した場合はエラー出力を行った後、モニターに戻る。

以上が Newmark β法で Δt 後の加速度を求め、その加速度より変位と速度を求める処理の全てである。処理の流れは理解できただろうか。なお、主要なサブルーチンについては、その内容を示したが、その他のサブルーチンは、付録を参照されたい。

4.5 応力計算、部材塑性チェック

反復計算が終了し、次の増分時間の反復計算に向かう前に、部材の応力計算、部材の塑性チェック、節点力の計算、接線剛性の計算、あるいは、各種データの出力などを行う必要がある。ここでは、反復計算の後処理として、応力計算等を行う処理の流れを見てみよう。前節と同様、SPACE の当該部分に関する実際のプログラムコードを示すことにする。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         計算終了・後処理開始
c
c―――――――――――――――――――――――――――――――――――――――
 9980 continue
      if(n_iterate .lt. iroop ) n_iterate = iroop
c――――――――――――――――――――――――――――――――★解析結果・増分変位をセット(ok)
      call Set_ddisp(n_unknown,est_ddisp_point,
     *     result_disp_point,past_disp_point)
c      write(damp_out,*) ' Set_disp_vel_acc ok'
c――――――――――――――――――――――――――――――――★変位、速度、加速度を出力
      vacc(1)=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(2)=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(3)=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
c      write(damp_out,'(a,4f10.3)') ' Get_Acc ok'
      call Out_disp_vel_acc(Point,n_point,Parameter_C,
     *     past_disp_point, past_vel_point, past_acc_point,
```

```
     *     rot_local,ifl,iflz,vacc,i_print)
c      write(damp_out,*) ' Out_disp_vel_acc ok'
c-----------------------------------★変位、速度、加速度の最大値セット(ok)
      call Get_max_disp(Max_disp,Parameter_C.n_point,Point,
     *      past_disp_point, past_vel_point, past_acc_point,
     *      d_max_v,id_max_v,vacc)
c       write(damp_out,*) ' Get_max_disp ok'
c-----------------------------------★部材両端の節点力計算（ok)
c      call  Set_pointforce(Member,n_member,ak_nonlinear,est_ddisp_point)
c     write(damp_out,*) ' Set_pointforce ok'
c-----------------------------------★部材応力を計算(ok)
      call 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      write(damp_out,*) ' Cal_stress ok'
c-------------------------------------------------------------
c
c    ★        部材の弾塑性状態をチェック
c
c-------------------------------------------------------------
c-----------------------------------★部材塑性をチェック
c-----------------------------------★部材両端の節点力計算（ok)
c-----------------------------------★ファイバー応力セット
      call Check_stress(Control,Control.type_analysis,
     *     ak_nonlinear,Member,n_member,Model_type,
     *     Element,past_disp_point,est_ddisp_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)
c       write(damp_out,*) ' Check_stress ok'
c-----------------------------------★部材応力を出力
      call Out_stress(Member,Element,E_model6_real,M_model11,M_model12,
     *                M_model13,M_model15,M_model21,M_model22,
     *                M_model31,M_model32,M_model33,
     *                n_member,ifl,iflz,i_print,Out_section)
c      write(damp_out,*) ' Out_stress ok'
c-----------------------------------★部材応力を出力
      call Out_Fiber(Member,Element,E_model11,M_model11,
     *               E_model12,M_model12,E_model13,M_model13,
     *               E_model15,M_model15,
     *               E_model21,M_model21,E_model22,M_model22,
     *               E_model31,M_model31,E_model32,M_model32,
     *               E_model33,M_model33,
     *               E_model_fiber,M_model_fiber,
```

```
     *              n_member,ifl,iflz,i_print,Out_section)
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)
c       write(damp_out,*) ' Get_max_stress ok'
c-------------------------------------★Maxwell 型モデルの非線形性チェック(ok)
      call Check_Maxwell_stress(Member,n_member,
     *     Element,E_model6_real,Newmark_P)
c      write(damp_out,*) ' Check_Maxwell_stress ok'
c-----------------------------------★解析結果・変位、速度、加速度をセット(ok)
      call Set_disp_vel_acc(n_unknown,est_ddisp_point,
     *     result_disp_point, result_vel_point, result_acc_point,
     *     past_disp_point, past_vel_point,past_acc_point,
     *     past_dacc_point)                      ! past_dacc_point に過去の加速度をセットする
c       write(damp_out,*) ' Set_disp_vel_acc ok'
c-------------------------------★部材節点力（反力と荷重）の出力(ok)
      call Get_pointforce(fll_force_point,Member,n_member,
     *                    Point,n_point)
      call Out_pointforce(fll_force_point,Point,rot_local,
     *                   n_point,ifl(1),iflz(1),i_print)
c-------------------------★部材節点力の描画用データセット
      if(i_read_ndbalanceF .ne. 0  .and.
     *  istep .eq. ns_step+n_step - 1 ) then
      call Set_preset_nd(fll_force_point,Point,rot_local,n_point,
     *                  F_ndbalanceF)
      endif
c      write(damp_out,*) ' Out_pointforce ok'
c-----------------------------------★不釣合力の計算
c-------------------------★節点静的荷重項(ok)
      if(ifl(2).eq.1.and.i_print.eq.0) then
      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)
      call Add_pointforce(p1,p2,p3,fll_force_point,
     *                   fll_static_point,n_point)
c-------------------------★集中質量に関する地震項
      vacc(1)=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(2)=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(3)=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
      call Add_earth1_pointforce(n_istep,vacc,fll_force_point,
     *          Point,n_point,am_point,rot_local,Parameter_C)
c-------------------------★整合質量に関する地震項
      call Add_earth2_pointforce(vacc,fll_force_point,
     *    Member,n_member,am_member,rot_local,Parameter_C,Dynamic_load)
c-----------------------------------★慣性項と減衰項計算(ok)
c-------------------------★集中質量の慣性項と減衰項(ok)
      call Add_damp1_pointforce(n_istep,fll_force_point,Point,n_point,
     *       past_vel_point,past_acc_point,
     *       am_point,Newmark_P,Parameter_C,rot_local)
c-------------------------★整合質量の慣性項と減衰項(ok)
      call Add_damp2_pointforce(n_istep,fll_force_point,Member,n_member,
     *       past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,Dynamic_load.load_mass)
c-------------------------★部材減衰の減衰項(ok)
```

```
      call Add_damp3_pointforce(n_istep,fll_force_point,Member,n_member,
     *       past_vel_point,ac_member,Newmark_P,Model_type.n_m_damp)
c-----------------------------★線形剛性のレーリー減衰項
      call Add_stiff1_pointforce(n_istep,fll_force_point,Member,
     *       n_member,past_vel_point,ak_linear,Newmark_P)
c-------------------------------★不釣合力の出力(ok)
      call Out_pointforce(fll_force_point,Point,rot_local,
     *                   n_point,ifl(2),iflz(2),i_print)
      endif
c---------------------------------★終了か？（ステップと最大値チェック）
      if(istep .ge. Newmark_P.nn_step .or.
     *    d_max_v .gt.Control.collapse_maxdisp ) goto 9998
c-----------------------------------★接線剛性の計算(ok)
      call Get_nonlinear_stiff(Control.type_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,
     *     work1_elememt,work2_elememt, work1_member, work2_member )
c       write(damp_out,*) ' Get_nonlinear_stiff ok'
 9999 continue
c-------------------------------------------------------------
c
c    ★        時間増分更新
c
c-------------------------------------------------------------
      iend_code = 0
      ns_step=ns_step + n_step
c-------------------------------------------------------------
c
c    ★        描画用データのセット
c
c-------------------------------------------------------------
c-------------------------------★変位
      if(i_read_disp .ne. 0) then
c     write(damp_out,*) ' Set_preset_disp ok'
      call Set_preset_disp(1,n_point,past_disp_point,F_disp,Point,
     *                           rot_local,Parameter_C)
      endif
c-------------------------------★応力
      if(i_read_spring .ne. 0) then
      call Set_preset_spring(Member,Element,E_model6_real,
     *               M_model11,M_model12,M_model13,
     *               M_model15,M_model21,M_model22,
     *               n_member,F_fay,F_n_spring,F_my_spring,
```

```
     *              F_mz_spring,i_stat_spring)
c     write(damp_out,*) ' Set_preset_spring ok'
      endif
      return
c-------------------------------------------------------------
c
c    ★        応答解析終了処理
c
c-------------------------------------------------------------
```

前節と同様に、プログラムの骨組みを用いて、処理の流れを見てみよう。多少長いが、プログラムの構造は、非常に簡単なので理解し易い。

```
c-------------------------------------------------------------
c
c    ★        計算終了・後処理開始
c
c-------------------------------------------------------------
 9980 continue
      if(n_iterate .lt. iroop ) n_iterate = iroop
c-----------------------------------★解析結果・増分変位をセット(ok)
      call Set_ddisp()                                        ! 1
c-----------------------------------★変位、速度、加速度を出力
      vacc(1)=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)  ! 2
      vacc(2)=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(3)=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
      call Out_disp_vel_acc()
c-----------------------------------★変位、速度、加速度の最大値セット(ok)
      call Get_max_disp()                                     ! 3
c-----------------------------------★部材両端の節点力計算（ok）
c      call  Set_pointforce()                                 ! 4
c-----------------------------------★部材応力を計算(ok)
      call Cal_stress ()                                      ! 5
c-------------------------------------------------------------
c
c    ★        部材の弾塑性状態をチェック
c
c-------------------------------------------------------------
c-----------------------------------★部材塑性をチェック
c-----------------------------------★部材両端の節点力計算（ok）
c-----------------------------------★ファイバー応力セット
      call Check_stress()                                     ! 6
c-----------------------------------★部材応力を出力
      call Out_stress()                                       ! 7
c-----------------------------------★部材応力を出力
      call Out_Fiber()                                        ! 8
c-----------------------------------★応力等の最大値セット
      call Get_max_stress()                                   ! 9
c-----------------------------------★Maxwell 型モデルの非線形性チェック(ok)
      call Check_Maxwell_stress()                             ! 10
c-----------------------------------★解析結果・変位、速度、加速度をセット(ok)
      call Set_disp_vel_acc()    ! past_dacc_point に過去の加速度をセットする      11
c-----------------------------------★部材節点力（反力と荷重）の出力(ok)
      call Get_pointforce()                                   ! 12
```

```
      call Out_pointforce()                                              ! 13
c-----------------------★部材節点力の描画用データセット
      if(i_read_ndbalanceF .ne. 0  .and. istep .eq. ns_step+n_step - 1 ) then
      call Set_preset_nd()                                               ! 14
      endif
c-------------------------------★不釣合力の計算
c-----------------------★節点静的荷重項(ok)
      if(ifl(2).eq.1.and.i_print.eq.0) then                              ! 15
      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)
      call Add_pointforce()
c-----------------------★集中質量に関する地震項
      vacc(1)=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)         ! 16
      vacc(2)=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      vacc(3)=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
      call Add_earth1_pointforce()
c-----------------------★整合質量に関する地震項
      call Add_earth2_pointforce()                                       ! 17
c-------------------------------★慣性項と減衰項計算(ok)
c-----------------------★集中質量の慣性項と減衰項(ok)
      call Add_damp1_pointforce()                                        ! 18
c-----------------------★整合質量の慣性項と減衰項(ok)
      call Add_damp2_pointforce()                                        ! 19
c-----------------------★部材減衰の減衰項(ok)
      call Add_damp3_pointforce()                                        ! 20
c-----------------------★線形剛性のレーリー減衰項
      call Add_stiff1_pointforce()                                       ! 21
c-----------------------★不釣合力の出力(ok)
      call Out_pointforce()                                              ! 22
      endif
c-------------------------------★終了か？（ステップと最大値チェック）
      if(istep .ge. Newmark_P.nn_step .or.                               ! 23
     *    d_max_v .gt.Control.collapse_maxdisp ) goto 9998
c-------------------------------★接線剛性の計算(ok)
      call Get_nonlinear_stiff()                                         ! 24
 9999 continue
c-------------------------------------------------------------
c
c    ★        時間増分更新
c
c-------------------------------------------------------------
      iend_code = 0
      ns_step=ns_step + n_step                                           ! 25
c-------------------------------------------------------------
c
c    ★        描画用データのセット
c
c-------------------------------------------------------------
c-----------------------★変位
      if(i_read_disp .ne. 0) then                                        ! 26
      call Set_preset_disp()
      endif
c-----------------------★応力
```

```
      if(i_read_spring .ne. 0) then                                    ! 27
      call Set_preset_spring()
      endif
      return
c-------------------------------------------------------------------
c
c    ★         応答解析終了処理
c
c-------------------------------------------------------------------
```

プログラムの右側には番号が振られており、その番号にしたがって処理内容を概説する。

1：反復計算で得た増分加速度、増分変位、増分速度等を所定の記憶域にセットする。
2：各節点の増分後加速度、速度、変位をファイルに出力する。絶対加速度を出力するために、地表加速度を再度取得している。
3：各節点の加速度、速度、変位の最大値をチェックし、所定の記憶域にセットする。
4：各部材の節点力を求めるルーチンであるが、都合で後の処理で行うことになり、ここでは計算しない。
5：各部材の両端の応力を求める。ここでは、各モデルの縮合された接線剛性より増分応力を計算し、実際の応力に加える。
6：各モデルの部材内ファイバーや、マルチスプリングなどの応力を計算し、塑性チェックを行う。ここで、各ファイバーエレメントなどの接線剛性を所定の記憶域にセットする。さらに、部材両端の節点力の計算も行う。
7：各部材の応力をファイルに出力する。
8：各モデルの断面内応力、ファイバー応力やひずみをファイルに出力する。
9：各部材の最大応力をチェックし、所定の記憶域にセットする。
10：Maxwell モデルの応力計算を行い、非線形性のチェックを行う。
11：変位、速度、加速度に増分変位、増分速度、増分加速度を加える。
12：部材節点力を節点力（不釣合節点力）にセットし、ファイルに出力する。各部材の節点力を節点で和を取ると、節点での力の釣合を満たしているためゼロとなる。また、荷重点または境界点では、それぞれ荷重（慣性力も含む）と反力に釣合うことになる。
13：部材節点力を、GUI を用いてリアルタイムで出力するかどうかチェックし、描画する場合はデータを所定の記憶領域にセットする。こ

の値は、引数を介して C++の描画ルーチンに渡される。

14：上記の節点力を画面表示する場合は、データをセットする。

15：以後、不釣合節点力の計算を行う。最初、静的荷重を取り出し、不釣合節点力に足し込む。

16：３方向の加速度を取り出す。集中質量に関する慣性項を不釣合節点力に足し込む。

17：部材分布質量に関する慣性項を不釣合節点力に足し込む。

18：レーリー減衰の集中質量に関する減衰項と慣性項を不釣合節点力に足し込む。

19：レーリー減衰の部材分布質量に関する減衰項と慣性項を不釣合節点力に足し込む。

20：部材減衰（Maxwell 型を含む）に関する減衰項を不釣合節点力に足し込む。

21：線形の剛性項より、レーリー減衰項を不釣合節点力に足し込む。

22：以上の処理でまとめた不釣合節点力をファイルに出力する。

23：解析が終了かどうかチェックする。終了の場合、文番号 9998 に飛び、後処理に移る。

24：各部材の接線剛性を求める。

ここで、増分計算ループが終了し、ループの始めである動的解析開始 1.1 に戻る。全ループの計算が終了すると次のステップに移る。

25：次回解析のため、解析ステップを更新する。

26：節点変位を、GUI を用いてリアルタイムで出力するかどうかチェックし、描画する場合はデータを所定の記憶領域にセットする。この値は引数を介して C++の描画ルーチンに渡される。

27：上記と同様に、部材応力を GUI を用いてリアルタイムで出力するかどうかチェックし、描画する場合はデータを所定の記憶領域にセットする。この値は引数を介して C++の描画ルーチンに渡される。サブルーチンを終了し、動的解析管理システムに戻る。