第４章　動的ソルバーの構成

4.1 はじめに

本章では、動的解析システムの中の動的ソルバーがどのような数値解析手法を用いているか、また、どのような手順で行われているかについて述べる。次に、その流れの中で重要な部分について、プログラムコードを用いて詳しく説明する。本章を読むことによって、SPACE では、どうような流れで動的解析を行っているかが理解できよう。

SPACE における動的解析は次の 2 つに分類されている。

１）固有値問題（固有振動数、固有振動モードを求める）
２）振動解析

さらに、振動解析は、解析モデルによって、次の 4 つに分類されている。

１）線形振動解析
２）幾何学的非線形振動解析
３）弾塑性振動解析
４）幾何学的非線形性を考慮した弾塑性振動解析

上の２つについては、解析プログラムは全く別のモジュールとして作られており、それぞれについて数値解析の流れを説明する。また、下の４つについては、解析プログラムの中で、区別して説明する。

4.2 固有値問題の解析フロー

空間構造物の振動に関する固有値問題では、以下のようにせん断型モデルと異なる特徴が存在する。この相違を良く認識し、計算手法を選択すべきである。

１．質量マトリックス[M]にゼロ対角項が存在する場合がある。
２．重複固有値を持つ場合が多い。
３．多数の近接固有値が存在する。
４．高次モードも比較的固有周期が長い。

以上を考慮して、本ソルバーで用いる手法として、せん断型部材モデルで構成されるせん断系では、相似変換法の一種であるヤコビ法を、スペースフレームでは、サブスペース法を用いる。

固有値問題を表す方程式は、以下のように表される。

$$[K]\{\varphi\} = \omega^2 [M]\{\varphi\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots(4.1)$$

ここで、$[M]$と$[K]$が共に対称マトリクスであることから、固有値ωは実数であることが保証され、しかも、$[K]$が正定値であれば固有値は全て正の値を持つことになる。この条件によって構造物は固有の振動数を持ち、逆に正定値でなければ、ゼロもしくは負の固有値をひとつ以上持つことになる。そのとき、構造物の変位が急激に大きくなる状態、いわゆる動的に不安定な状態を示すことになる。

ヤコビ法の利点は、理論の単純さと安定性の良さにある。全ての対称マトリクスに対して適用でき、負の固有値やゼロの固有値を含んでいても計算が可能である。また、固有値が全て求まることも特徴のひとつである。ヤコビ法の基本的な考え方は、マトリクスの非対角成分がゼロに変るような回転行列P_iを作成し、３重行列積を行って非対角成分をゼロとすることである。次に、順次その位置をずらし、同じ計算を実行する。無論、一度ゼロにした非対角項も、他の位置に関する再度の計算で値が入るが、その値は小さく、全体として非対角項がゼロに接近していくことになる。ゼロにする非対角項の選択法として、行か列を一度に実行する周期的ヤコビ法や閾値を使って効率的にゼロ化していく閾値ヤコビ法などが開発されている。しかしながら、同方法はマトリクス乗算が多く含まれ、しかも固有値全体を一度に求めるため、かなり時間がかかり、大次元行列の固有値問題には不向きと言える。ただし、他の手法と組み合わせて効率良く解く手法が開発され、大次元行列固有値問題の解析手法の一部として、広く使われている。SPACE では、このヤコビ法がせん断系モデルに適用され、全ての振動数及び振動モードを求める手法として用いられる。

解析が大規模であると、マトリクス$[M]$と$[K]$は大次元となり、固有値問題の解法も効率のよい手法が求められる。サブスペース(Subspace iteration)

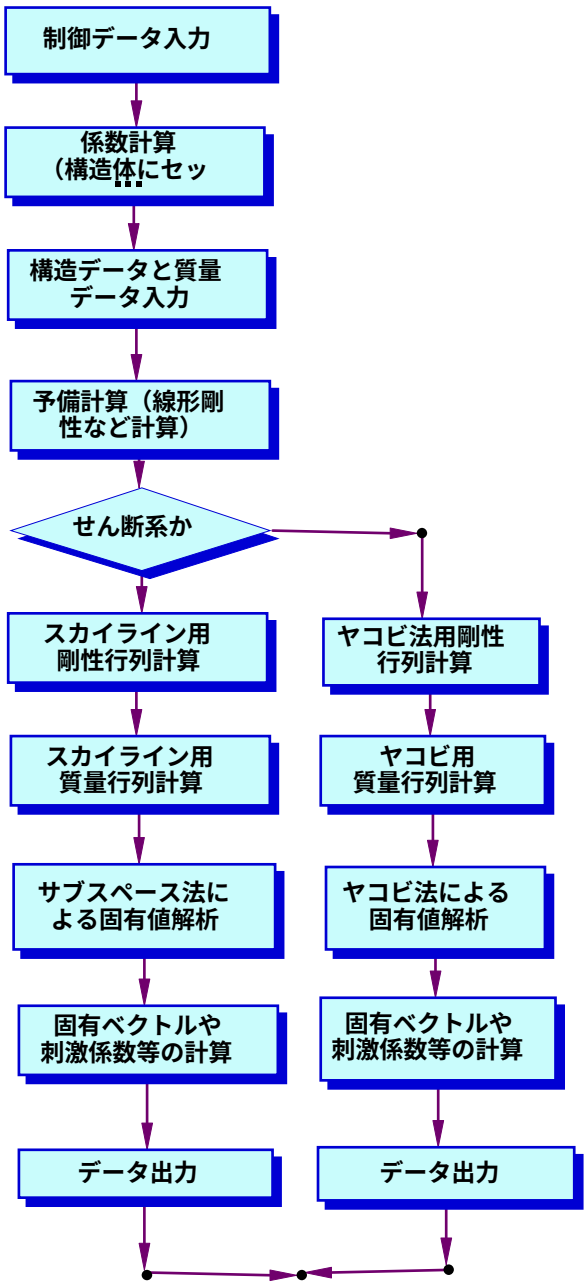

図 4-1 固有値問題解析フロー

法がこの種の問題に最も効果的であり、SPACE では、この大規模なスペースフレームの固有値問題を解く手法として用いられる。サブスペース法は大規模なバンド状の系の解析に適しており、逆反復法、サブスペースでの一般化ヤコビ法、また、スツルム列による固有値の検定などを併用する。同法は低次 P 個の固有値と対応する固有ベクトルを求めることになり、計算可能な固有値の数は、サブスペースの大きさに関連するため注意が必要で、例えば、サブスペースの自由度は、計算固有値に 5 程度足した数あるいは計算固有値の 2 倍程度が必要とされる。さらにサブスペースの自由度は質量行列におけるゼロ以外の自由度の数を越えてはならないとされていることからも、系全体の自由度と求める固有値の数が近いときは注意が必要である。詳しくは、リファレンスマニュアルを参照されたい。

本節では、図 4-1 に示すフローチャートを用いて、固有値問題に関する数値解析手法を概観しよう。解析は、大きく 2 つに分けられる。ひとつは、データ入力と予備計算であり、他は固有値解析の主要部分である。最初は、SPACE で設定した解析用パラメータをファイルから読み込み、構造体にセットすることから始まる。

次に、構造データと質量に関するデータをファイルより入力し、所定の構造体や配列にセットする。データを全て入力した後、部材長さや線形の剛性行列を計算する予備計算を行う。

次に固有値問題を解く解析手法によって、処理は 2 つに分岐する。ひとつは、全体剛性行列としてスカイライン行列を用いたサブスペース法による方法と、他は正方行列を用いたヤコビ法による方法である。固有値問題を解く手法と剛性行列や質量行列の保存方法は異なるが、解析全体の流れは同じである。ヤコビ法はせん断系モデルの解析に用いられ、全未知数と同じ固有値と固有ベクトルを得ることができる。一方、サブスペース法は、解析手法による制限があり、全自由度に対する固有値を求めることはできない。ただし、ヤコビ法は、自由度が大きくなると非常に計算時間がかかる欠点を有する。

最初に、全体剛性行列を求め、次に、質量行列を求める。この二つの行列を元に、サブスペース法もしくはヤコビ法によって、所定の固有値と固有ベクトルを求める。最後に、刺激係数などを求め、ファイルに出力する。ここで出力されたデータは、次節で説明する動的解析で使用する。動的解析においてレーリー減衰を使用する場合、この固有値問題を解いて得た振動数、振動モード、刺激係数などが収納されているファイルを用意する必要がある。

4.3 振動解析の計算フロー

スペースフレームに対する非線形振動方程式の数値解析手法は第３章で述べた。以降では、この解析手法をプログラミングコードとして展開するわけであるが、ここでは、まず、フローチャートを用いて解析の流れを理解することにしよう。

図 4-2 は、SPACE システムの動的解析部分の概略フローである。解析の流れは、一般の動的解析手法と何らかわることはない。しかし、このシステムは解析結果をリアルタイムでグラフィック表示することが特徴であるため、その部分の手当てを行う必要がある。グラフィック並びに GUI はマルチウインドウシステムを用いており、言語は C++を利用している。そのため、計算過程を C++のルーチンにデータ転送する必要があり、本システムでは、最も単純な方法でデータを交換している。その方法とは、サブルーチンの引数で受け渡すことである。これを実現するために、計算途中で計算プログラムから一旦抜け出し、C++のグラフィックルーチンにデータを受け渡さなければならない。これらを実現する技術的テクニックは、第８章で詳細に述べることにする。

最初に、SPACE システムの動的解析部分の概略フローを見てみよう。まず、動的解析システムが立ち上がると、GUI 用の解析モニターに起動がかかる。このモニターは、ひとつのウインドウを表示する前に、システムコントロール用ファイル、構造用データファイルなどをオープンし、その内容を読み込む。次に、ユーザーは、コントロールパネルを表示し、さらには、解析結果をリアルタイムで観測するための設定を行う。これらは、モニターの機能に含まれ、ファイルからのデータ入力以外は全て C++で書かれている。次に、モニターから解析種別（線形解析、固有値問題、幾何学的非線形解析など）が選択され、動的ソルバーが呼び出されることになる。動的ソルバーはマルチスレッドで起動されるため、モニターはフリーズすることなく、ユーザーとの情報交換が円滑に行われることになる。また、解析経過をグラフィック表示するため、動的ソルバーは、定期的に処理を休止し、一旦モニターに制御を移すことで引数を介してモニターに情報を転送する。

動的ソルバーに関する計算フローを見てみよう。解析は、

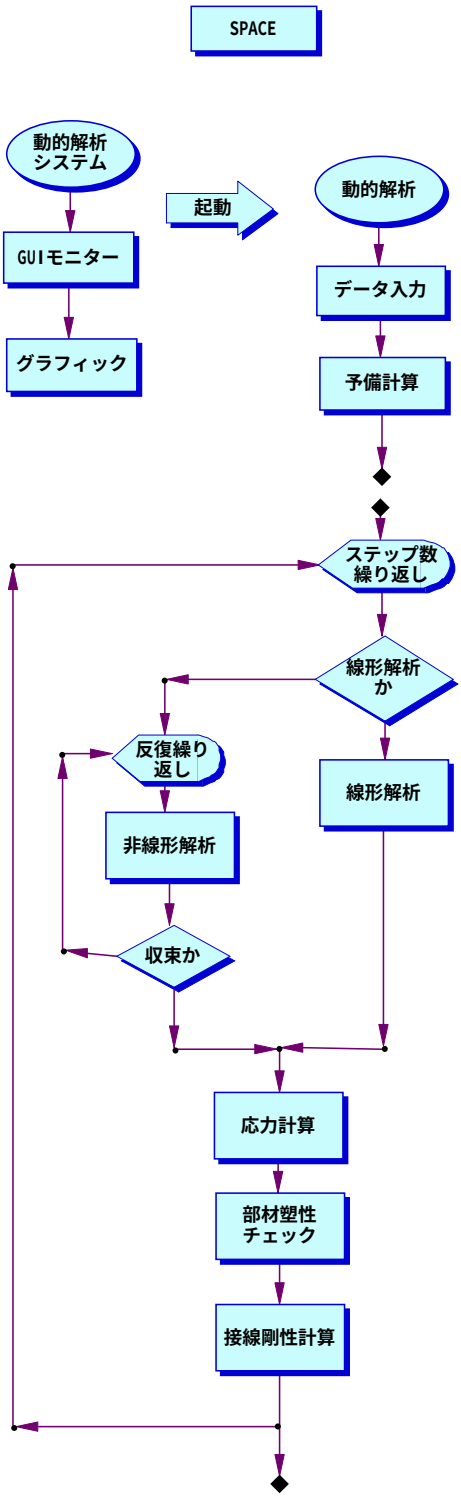

図 4-2 振動解析の計算フロー

大きく２つに分けられる。ひとつは、データ入力と予備計算であり、他は動的解析の主部分である。データ入力と予備計算の部分では、まず、解析制御用パラメータを入力し、構造体にセットする。次に、構造用データ、荷重データ、質量データ、地震波データ、減衰データなどを該当するファイルより入力し、解析に必要となるデータを所定の構造体や配列に収納する。

予備計算では、部材の長さ計算や部材の線形剛性計算、座標変換行列の作成、あるいは節点変位、速度、加速度などの初期値をセットする。さらに、ユーザーの指示に従って、計算結果を出力するファイルをオープンする。これで、データ入力と予備計算を終了し、一旦サブルーチンを抜ける。その後 C++で書かれた動的ソルバー管理システムに戻る。

動的解析管理システムから振動解析を行うために、再び FORTRAN で書かれた動的ソルバーに戻る。管理ルーチンにおいてリアルタイムで図形表示を行う場合、数ステップ毎に、解析プログラムから管理ルーチンに戻ることになる。このステップ数はユーザーが設定することになる。また、SPACE における動的解析では、擬似的静的解析と動的応答解析の２つの処理を続けて行う。以後は、擬似的静的解析を第 1 段階振動解析、動的応答解析を第２段階振動解析と呼ぶことにする。

動的解析システムでは、動的解析管理システムに制御が戻ったとき、図形が再描画される。そのため設定するスキップ数が大きいと図形はコマ落とし状態となる。逆に、スキップ数を小さくすると図形は滑らかに表示されるが、それだけ解析に必要な時間がかかることになる。

振動解析に入った直後、つまり、第１段階振動解析と第２段階振動解析が始まった直後に、振動方程式の左辺係数項を作成し、その後、そのスカイライン行列を LDU 分解する。分解を２回行う理由は、擬似的静的荷重時と動的応答解析時（地震時）とで、質量を変化させることができるようにするためである。この LDU 分解を終えた後、処理は線形解析と非線形解析に分かれ、振動方程式が数値解析されることになる。

線形解析では、釣合式右辺項を計算し、方程式を解いて加速度ベクトルを求める。求めた後は、ニューマークβ法に基づき速度と変位を計算し、次の処理に移る。一方、非線形解析では、非線形の振動方程式を、反復計算を用いて解くことになる。まず、Δｔ秒後の加速度を予測する。次に、Δｔ秒後の加速度（未知ベクトル）に関連する右辺項ベクトルと関連しない右辺項ベクトルを分けて各々求める。未知ベクトルに関連していない右辺項を計算し、その後、反復計算ルーチンに入る。反復ルーチン内に入ると、予測した加速度ベクトルと変位ベクトル、速度ベクトルを用いて、未知ベクトルに関連する項を求める。２つの右辺項を加えた後、連立方程式を解き、未知ベクトルを求める。求めた未知ベクトルと予測した加速度とを比較し、収束したかどうかチェックする。収束した場合は、ニューマークβ法に基づき速度と変位を計算し、次の処理に

進む。収束しない場合は、誤差ベクトルを求めると共に、次の未知加速度を予測し、反復計算の最初の部分に戻り、同様の解析を続ける。

SPACE では、反復計算を行う場合、最大反復数を指定できる。この指定した最大反復数を超えてしまった場合、解析手法は自動的に陰解法へと変更になる。陰解法では、釣合式左辺項の係数ベクトルは、反復解法とは異なるため、再計算し、LDU 分解を行う必要がある。未知加速度を求めた後、SPACE では、一回で反復解法に戻るため、再度線形の左辺係数行列を求めた後、LDU 分解を行い、次ステップの反復計算に備える。

未知加速度が求まった後、部材の材端応力を求める。次に、変位、速度、加速度、部材応力などを所定のファイルに出力した後、部材の弾塑性チェックを行う。最後に部材の接線剛性、つまり幾何学的非線形性と塑性状態を考慮した接線剛性を求め、次のステップに移ることになる。

振動解析が全て終わった後、あるいは、ユーザーが途中で計算終了を指定したり、データにエラーがあって途中終了したりした場合、計算途中で動的に確保した構造体や配列のメモリー領域を解放する必要がある。解放し忘れるとメモリーリークが生じることになり、ウインドウシステムを停止しない限り、そのメモリーを使用できないことになる。逆に、動的確保が行われていない動的領域の解放処理を行うと、重大なシステムエラーが発生する。これらを避けるために SPACE では、動的領域の管理を行っている。これについては後節で説明する。

以上で、固有値解析と振動解析に関する主要な計算の流れを説明した。数値計算の概略が理解できただろうか。次節からは、具体的なプログラムコードを用いて詳細に説明を行う。これにより部分的に理解すると同時に、再度この解析フローを確認することで、SPACE の動的ソルバーをより深く理解できよう。

4.4 解析手法とプログラミング

4.4.1 動的解法

前節では動的ソルバーの概要について述べた。本節では、実際のプログラムコードを具体的に検証しながら、解析の流れを追跡する。ただし、各サブルーチンの中身については後章で述べる。この部分を検討することで、さらに理解を深めていくことになろう。

まず、反復解法部分を見てみよう。サブルーチン submain_dynamic_a() の該当するコードを以下に示すので、ざっと目を通されたい。また、このサブルーチン全体は、付録として添付されているので、それを参照されたい。変数、配列、構造体などの意味は、付録のインクルードファイル、あるいは、このサブルーチンの最初の部分にある配列定義に記述さ

れている。また、ほとんどの変数の意味は、プログラムの中にコメントとして示されているので、その箇所を参照されたい。

```
c——————————————————————————————————————
c
c    ★         動的解析開始
c
c——————————————————————————————————————
c——————————————————————————————————————
      i_print = 0
 9990 continue
c——————————————————————————————————————
c              解析手法を途中で変更
c——————————————————————————————————————
      call Reset_control(Control,i_calnum)
c——————————————————————————————————————
      n_iterate = 0                         !  反復回数ゼロセット
      n_member=Parameter_C.n_member
      n_point =Parameter_C.n_point
      n_unknown=Parameter_C.n_unknown
      n_local_coord=Parameter_C.n_local_coord
      if(numb_method.eq.1) then
      N_implicit_method = -1 !陰解法に設定
      Iteration_method = 2
      Endif
      write(damp_out,'(///a,3i4)') ' Dynamic analysis start :',
     *                 ns_step,n_step,Dynamic_load.load_dynamic(1)
      do 9999 istep=ns_step,ns_step+n_step - 1
      T=Newmark_P.dt*istep
      i_print=mod(istep,Control.interval_out)
      stimes =secnds(stimes)
      write(damp_out,'(/a,i6,a,f10.3,a,f12.4,a,i4,a,f12.4)')
     *     ' Step No,:',istep,
     *     '    Time：',T,
     *     'sec.    Max. Disp.：',d_max_v,
     *     '    Node Number：',id_max_v,
     *     ' times (s):',stimes
      stimes=secnds(0.0)
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
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)
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)
      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？
      if(iexit.ne.0) then
      ierr_dat =100
      write(damp_out,*) ' decomp_sky err',iexit
      return
      endif
c      write(damp_out,*) ' decomp_sky ok',iexit
      endif
c——————————————————————————————————————
c
c    ★         動的解析の開始
c
c——————————————————————————————————————
      if(Control.type_analysis .ne. 7) goto 9981
c——————————————————————————————————————
c
```

```
c    ★         線形解析
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－－－－－－－－－－－－－－－－－－－－－－－－－－－★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)') ' 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)
      call Add_damp1_ld(n_istep,ld_point,Point,n_point,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_point,Newmark_P,Parameter_C,rot_local)
c      write(damp_out,*) ' Add_damp1_ld ok'
c－－－－－－－－－－－－－－－－－－－－－－★整合質量(ok)
      call Add_damp2_ld(n_istep,ld_point,Member,n_member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,Dynamic_load.load_mass)
c       write(damp_out,*) ' Add_damp2_ld ok'
c－－－－－－－－－－－－－－－－－－－－－－★部材減衰(ok)
       call Add_damp3_ld(n_istep,ld_point,Member,n_member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ac_member,Newmark_P,Model_type.n_m_damp)
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－－－－－－－－－－－－－－－－－－－－－－－－－－－★Δｔ秒後の変位と速度を予測(ok)
      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-----------------------------------★Maxwell 型モデルの計算(ok)
      call Add_fdd_ld(ld_point,E_model6_real,Element,
     *       Member,n_member,est_vel_point,rot_memb)
c      write(damp_out,*) ' Add_fdd_ld ok'
c-----------------------------------★線形方程式を解く(ok)
      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,result_acc_point)
c      write(damp_out,*) ' solve_sky ok'
c-----------------------------------★加速度よりβ法に基づき変位と速度を計算(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'
      goto 9980
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(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-----------------------------------★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)
      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)
      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)
      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)
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――――――――――――――――――――――――――――――――――★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-----------------------------------★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-----------------------------------★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)
      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(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
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
      Iteration_method = 2
      N_implicit_method = 1
      nm_iterate = nm_iterate+1  !陰解法使用回数の計算
 9982 continue
c       write(76,'(a,10f16.5)') ' 陽解法'
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――――――――――――――――――――――――――――――――――――――――
      n_istep=1
      if(istep.ge.Newmark_P.n2_step) n_istep=2
c――――――――――――――――――――――――――★スカイライン行列のゼロセット
      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
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
      call Rotate_mass(n_istep,Element,Member,n_member,am_member,
     *                 rot_memb,Dynamic_load.load_mass)
c      write(damp_out,*) ' Rotate_mass ok'
c――――――――――――――――――――――――――★整合質量系の足し込み
      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(gskym,Member,Element,n_member,rot_memb,
     *               ac_member,Newmark_P, max_h_sky,E_model6_real)
c       write(damp_out,*) ' Build_sky_c ok'
      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(n_istep,gskym,n_skyline, Member,n_member,
     *              Ak_linear,Ak_nonlinear, Newmark_P, max_h_sky)
c       write(damp_out,*) ' Build_sky_k ** ok'
c――――――――――――――――――――――――――――――――★行列の LDU 分解
      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 ok',
```

```
c―――――――――――――――――――――――――――――――――★分解成功か？
      if(iexit.ne.0) then
      ierr_dat = 100
      write(damp_out,*) ' decomp_sky err',iexit
      return
      endif
c      write(damp_out,*) ' decomp_sky ok',iexit
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(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
       call Get_pointforce_ld(ld_point,Member,n_member)
c             write(76,'(a,10f16.5)') 'lp_point_1',(ld_point(j),j=1,n_unknown)
c―――――――――――――――――――――――――――――――――★地震加速度セット
      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(76,'(a,10f16.5)') 'lp_point_2'
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
      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)
      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)
      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（Maxwell 型モデル）
      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)
c                          write(76,'(a,10f16.5)') 'lp_point_4'
c--------------------------------★線形剛性によるレーリー減衰
      call Add_stiff1_ld_ex(n_istep,ld_point,Member,n_member,
     *       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
      call Add_tan_stiff_ld(ld_point,
     *       Member,n_member,b_vector,ak_nonlinear)
c     write(damp_out,*) ' Add_tan_stiff_ld ok'
c--------------------------------★Maxwell 型モデルの右辺項 fd の計算(ok)
      call Add_fdd_ld_ex(ld_point,E_model6_real,Element,
     *       Member,n_member,est_vel_point,rot_memb)
c     write(damp_out,*) ' Add_fdd_ld ok'
c--------------------------------★線形方程式を解く(ok)
      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,result_acc_point)
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-----------------------------------------------------------
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)
      n_skyline=Parameter_C.n_skyline
      call Set_sky_zero(gskym,n_skyline)
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―――――――――――――――――――――――――★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―――――――――――――――――――――――――★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)
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―――――――――――――――――――――――――★部材減衰系の足し込み(ok)
      if(Parameter_C.nc_member .ne. 0) then
       call Build_sky_c(gskym,Member,n_member,
     *                ac_member ,Newmark_P, max_h_sky)
      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―――――――――――――――――――――――――――――――★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―――――――――――――――――――――――――――――――★分解成功か？
      if(iexit.ne.0) then
      ierr_dat =100
      write(damp_out,*) ' decomp_sky err',iexit
      return
      endif
c      write(damp_out,*) ' decomp_sky ok',iexit
c―――――――――――――――――――――――――――――――★
      endif
      endif
c
c―――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        計算終了・後処理開始
c
c―――――――――――――――――――――――――――――――――――――――――――――――
```

上記が Newmark β法による数値解析部分を実際のプログラムから取り出したものである。プログラムが長すぎて、多分、良く理解できなかったのではないだろうか。最初は、コードの細部を無視して、解析手法の構造を理解することが大切である。そこで、この部分の概略フローを図 4-3 に示すことにしよう。

このフローは 4 つの処理より成り立っていることが分かる。第 1 は、

判定処理の下にある「反復解法の係数行列作成と分解」である。これは、振動方程式(4.2)の左辺項を計算し、LDU 分解する部分である。SPACE では、2 つの段階で動的解析することから、両者の最初の部分で、この係数行列作成と分解が行われることになる。なお、第 1 段階は、静的荷重に対する擬似的な応力解析に相当し、ここでは、構造物が振動しないように減衰定数を 100%にして計算を行う。第 2 段階は、通常の地震荷重（風荷重）を受ける動的応答解析である。

第 2 は、Newmark β 法を用いた線形解析の処理部分である。解析種別によって、線形解析が選択されると第 2 の処理部分上の判定処理によって、線形解析が実行される。第 3 は、Newmark β 法を用いた非線形解析の処理部分である。ここでの処理は、反復解法を用いているため、ループ処理となっており、計算が収束したか否かの判定処理が必要となる。反復収束が満たされると、次の処理へと移っていくことになるが、ユーザーが設定した最大反復回数の収束計算を実行しても、収束しない場合は、第 4 の処理へと進んでいく。

第 4 の処理は、陰解法を行う部分である。陰解法は、非線形振動方程式の左辺、及び右辺を計算し、方程式を解いて未知加速度ベクトルを求めることになる。さらに、ここでは、左辺の剛性行列は、反復解法の剛性行列の同じ記憶領域を使用しているため、次の反復解法のために、左辺の係数行列を計算し、LDU 分解を行うサブルーチンが追加されている。

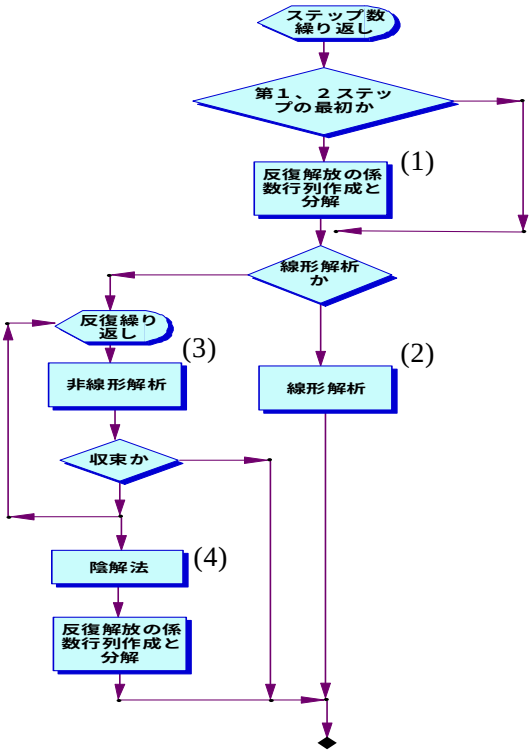

図 4-3 反復計算フロー

以上の 4 つの処理を、さらに詳細に見ていこう。先に示したプログラムコードの細かい部分は省略し、サブルーチンの名前のみを以下に示す。このサブルーチンは太文字で示し、引数等は省略する。

最初に、第 1 の処理について詳細に分析する。Newmark β 法を用いた非線形振動方程式は、

$$\begin{aligned}&\left[[M]+\mu_1[\bar{C}]+\mu_2[K]\right]\{\ddot{y}_{n+1}\}\\&\quad=-[M][I]\{\ddot{u}_g\}+\{P_S\}-\{Q(y_n)\}-[\bar{C}]\{a\}-[K]\{\bar{b}\}\\&\qquad-\{\bar{f}_d\}-[K_T(y_n)]\{y_{n+1}\}+[K]\{\Delta y_{n+1}\}\end{aligned}\qquad\cdots\cdots\cdots\cdots\cdots\cdots(4.2)$$

で与えられている。この方程式の左辺は、質量行列$[M]$、線形剛性行列$[K]$、減衰行列$[C]$と係数μ_1、μ_2とで構成されている。係数μ_1、μ_2は既に計算され、構造体 Newmark_P にセットされている。左辺の係数行列は、

スカイライン行列 gskym として作成し、LDU 分解する。本システム SPACE では、質量項は、節点集中質量と部材分布質量の２種類が用意されている。部材分布質量の有無は、構造体 Dynamic_load の成分 load_mass にセットされている。また、減衰項は、レーリー減衰と部材減衰が用意されており、レーリー減衰は、各部材の剛性行列と質量量列から直接計算し、スカイライン行列 gskym に組み込まれる。部材減衰の有無は、構造体成分 Parameter_C.nc_member にセットされている。

以上の処理を実際のプログラムコードを用いて検証しよう。以下に、全体の流れに関係のない部分やサブルーチンの引数を取り除いたプログラムの骨組みを示す。処理の内容と流れを理解されたい。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★        動的解析開始
c
c―――――――――――――――――――――――――――――――――――――――
      do 9999 istep=ns_step,ns_step+n_step - 1                          ! 1.1
      T=Newmark_P.dt*istep
      i_print=mod(istep,Control.interval_out)
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  No.1
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                 ! 1.2
      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)
      call Set_sky_zero()                                               ! 1.3
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()                                               ! 1.4
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                            ! 1.5
      call Cal_mass_linear()
c――――――――――――――――――――――――――――――★整合質量の釣合座標系への変換(ok)
      call Rotate_mass()                                                ! 1.6
c――――――――――――――――――――――――――――――★整合質量系の足し込み(ok*)
      call Build_sky_m()                                                ! 1.7
c――――――――――――――――――――――――――――――★部材減衰系の足し込み(ok)
      if(Parameter_C.nc_member .ne. 0) then
       call Build_sky_c()                                               ! 1.8
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()                                                ! 1.9
```

```
c―――――――――――――――――――――――――――――――★行列の LDU 分解(ok)
      call decomp_sky()                                          ! 1.10
c     write(damp_out,'(a,12f12.3)')' gskym *'
c―――――――――――――――――――――――――――――――――★分解成功か？
      if(iexit.ne.0) then                                        ! 1.11
      ierr_dat =100
      write(damp_out,*) ' decomp_sky err',iexit
      return
      endif
c      write(damp_out,*) ' decomp_sky ok',iexit
      endif
```

プログラム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。ここでは、次式で示す係数行列を求め、LDU 分解する。

$$[F]=\left[[M]+\mu_1[\bar{C}]+\mu_2[K]\right] \qquad \cdots\cdots\cdots\cdots\cdots\cdots(4.3)$$

1.1：動的解析を実行するための外側のループであり、ここで示されている回数を処理した後、一旦、モニターに処理が戻ることになる。

1.2：第１段階と第２段階の最初かどうか判定処理を行う。最初であれば振動方程式の左辺項の計算と LDU 分解を行う。

1.3：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を、全てこのスカイライン行列に足し込むことになる。

1.4：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に加える。

1.5：次に部材分布質量があるか否かの判定処理があり、その結果、部材分布質量がある場合は、1.6、1.7 の処理を行い、部材座標系で整合質量行列を求める。ここでは、まず部材座標系における整合質量行列を求める。

1.6：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。

1.7：座標変換された整合質量行列を係数行列 gskym に足し込む。

1.8：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。ここには、Maxwell 型の線形減衰行列も含まれており、この減衰行列を係数行列 gskym に足し込んでいる。

1.9：線形の剛性行列を係数行列 gskym に足し込む。同時に、レーリー

式(4.3)は、次に示すプログラム番号で求められる。

$[F]=$
$[M]$ (1.4),(1.7)
$+\mu_1[\bar{C}]$ (1.4),(1.7), (1.8) ,(1.9)
$+\mu_2[K]$ (1.9)

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。

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

1.11：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、処理１の「反復解法の係数行列作成と分解」である。さらに、上記の骨組みの中で、主要なサブルーチンを以下に示す。コメントを参照してその内容を理解されたい。

```
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(*)
c―――――――――――――――――――――――――――――――――――――――――――――
c     n_istep                 :integer       1:第一段階解析 2:第二段階解析
c     gskym(n_skyline)        :real*8        スカイライン行列
c     Point                   :structur
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          [M] +μ1 [C] = (1 + μ1α1)[M]   μ1=Δtδ
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)
      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
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Build_sky_m
C       ―――――――――――――――――――――――――――――――――
C     ●    部材質量行列のスカイライン行列への組み込み(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Build_sky_m(n_istep,gskym,n_skyline, Member,n_member,
     *                am_member ,Newmark_P, max_h_sky,Element)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension am_member(12,12,*)
      dimension max_h_sky(0:*)
      record / newmark_s  / Newmark_P
      record / member_s   / Member
      record / element_s  / Element
      dimension Member(*),Element(*)
      dimension gskym(*)
c―――――――――――――――――――――――――――――――――――――――――――――
c      n_istep                  :integer     第一段階解析か段階解析か
c      gskym(n_skyline)         :real*8      スカイライン行列
c      n_skyline                :integer     スカイライン行列の大きさ
c      Member                   :structure
c      n_member                 :integer     部材数
c      am_member(12,12,n_member) :real*8      質量整合行列（釣合系―全体座標系）
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      Element                  :structure
c          [M] +μ1 [C] = (1 + μ1α1)[M]   μ1=Δtδ
c―――――――――――――――――――――――――――――――――――――――――――――
      if(n_istep.eq.1) then
      par=1.+ Newmark_P.ddt*Newmark_P.alf1_1  !レーリー減衰を含む
      ik=1
      else
      par=1.+ Newmark_P.ddt*Newmark_P.alf2_1  !レーリー減衰を含む
      ik=2
      endif
      do i=1,n_member
      nm=Member(i).nm_element
      if(Element(nm).am(ik) .ne. 0. ) then
      call Build_ak(gskym,
     *     Member(i).irest(1), max_h_sky,
     *     par,am_member(1,1,i) )
      endif
```

```
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Build_sky_c
C       ―――――――――――――――――――――――――――――――――――
C     ●    部材減衰行列のスカイライン行列への組み込み(ok)
C     ―――――――――――――――――――――――――――――――――――――
      subroutine Build_sky_c(gskym,Member,n_member,
     *                  ac_member ,Newmark_P, max_h_sky)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension ac_member(12,12,*),gskym(*)
      dimension max_h_sky(0:*)
      record / newmark_s  / Newmark_P
      record / member_s   / Member
      dimension Member(*)
c―――――――――――――――――――――――――――――――――――――――――――――――
c     n_istep                   :integer    第一段階解析か第二段階解析か
c     Member                    :structure
c     n_member                  :integer    部材数
c     Ac_member(12,12,nc_member) :real*8     部材減衰行列（釣合系）
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―――――――――――――――――――――――――――――――――――――――――――――――
      par=Newmark_P.ddt
      do i=1,n_member
      ij=Member(i).nm_damp
      if(ij.ne.0) then
      call Build_ak(gskym,
     *     Member(i).irest(1), max_h_sky,
     *     par,ac_member(1,1,ij) )
      end if
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Build_sky_k
C       ―――――――――――――――――――――――――――――――――――
C     ●    部材剛性行列のスカイライン行列への組み込み(ok)
C     ―――――――――――――――――――――――――――――――――――――
      subroutine Build_sky_k(n_istep,gskym,n_skyline, Member,n_member,
     *                  ak_linear ,Newmark_P, max_h_sky)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      dimension ak_linear(12,12,*),gskym(*)
      dimension max_h_sky(0:*)
      record / newmark_s  / Newmark_P
      record / member_s   / Member
      dimension Member(*)
c―――――――――――――――――――――――――――――――――――――――――――――――
c     n_istep                   :integer   第一段階解析か第二段階解析か
c     n_skyline                 :integer   スカイライン行列の大きさ
c     Member                    :structure
```

```
c     n_member                   :integer   部材数
c     Ak_linear(12,12,n_member)  :real*8    線形剛性行列（釣合系）
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――――――――――――――――――――――――――――――――――――――――
      if(n_istep.eq.1) then
      par=Newmark_P.bdt + Newmark_P.ddt*Newmark_P.alf1_2
      else
      par=Newmark_P.bdt + Newmark_P.ddt*Newmark_P.alf2_2
      endif
      do i=1,n_member
      call Build_ak(gskym,
     *     Member(i).irest(1), max_h_sky,
     *     par,ak_linear(1,1,i) )
      end do
      return
      end
C     ――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_lin_mass_M1
C       ――――――――――――――――――――――――――――――――――
C     ●   梁要素の整合質量行列計算(ok)
C     ――――――――――――――――――――――――――――――――――――
      subroutine Cal_lin_mass_M1(am_beam,Member,Element,am)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s     /Member
      record / element_s    /Element
      dimension am(12,12)
c――――――――――――――――――――――――――――――――――――――――
c     am_beam                  :real*8  単位長さ当たりの質量（tf/cm）
c     Member                   :structure
c     Element                  :structure
c     am                       :real*8   整合質量行列
c――――――――――――――――――――――――――――――――――――――――
      do i=1,12
      do j=1,12
      am(i,j)=0.
      enddo
      enddo
      rlength=Member.alength
      aria =Element.a
      riy=Element.RIy
      riz=Element.RIz
      rix=Element.RIx
      amkk=am_beam
      ram1=amkk*rlength
      ram2=13.0/35.0
      ram3=11.0/210.0*rlength
      ram4=9.0/70.0
      ram5=13.0/420.0*rlength
      ram6=6.0/(5.0*aria*rlength*rlength)
      ram7=10.0*aria*rlength
      ram8=rlength*rlength/105.0
      ram10=2.0/(15.0*aria)
```

```
ram11=rlength*rlength/140.0
ram12=30.0*aria
am(1,1)=  ram1/3.0
am(1,7)=  ram1/6.0
am(2,2)=  ram1*(ram2+ram6*riz)
am(2,6)=  ram1*(ram3+riz/ram7)
am(2,8)=  ram1*(ram4-ram6*riz)
am(2,12)= ram1*(-ram5+riz/ram7)
am(3,3)=  ram1*(ram2+ram6*riy)
am(3,5)=  ram1*(-ram3-riy/ram7)
am(3,9)=  ram1*(ram4-ram6*riy)
am(3,11)= ram1*(ram5-riy/ram7)
am(4,4)=  ram1*rix/(3.0*aria)
am(4,10)= am(4,4)/2.0
am(5,5)=  ram1*(ram8+ram10*riy)
am(5,9)=  ram1*(-ram5+riy/ram7)
am(5,11)= ram1*(-ram11-riy/ram12)
am(6,6)=  ram1*(ram8+ram10*riz)
am(6,8)=  ram1*(ram5-riz/ram7)
am(6,12)= ram1*(-ram11-riz/ram12)
am(7,7)=  am(1,1)
am(8,8)=  am(2,2)
am(8,12)=-am(2,6)
am(9,9)=  am(3,3)
am(9,11)=-am(3,5)
am(10,10)=am(4,4)
am(11,11)=am(5,5)
am(12,12)=am(6,6)
do  i=1,11
do  j=i+1,12
am(j,i)=am(i,j)
enddo
enddo
return
end
```

プログラム構造が簡単なので、上記の内容は釣合式と比較することで容易に理解できよう。解析の骨組みが理解できたところで、もう一度実際のプログラムコードを見ると、より理解が深まることになろう。

4.4.2 線形解析

次に、処理 2 の「線形解析」に移ろう。線形解析の釣合式は、式(3.46)の非線形項を無視することで次のように得られる。

$$[M]\{\ddot{y}\}+[\bar{C}]\{\dot{y}\}+[K]\{y\}=-[M][I]\{\ddot{u}_g\}+\{P_S\}-\{\bar{f}_d\} \quad \cdots\cdots\cdots\cdots\cdots\cdots(4.4)$$

上式に Newmark β法を適用するため、式(3.6)を代入し、式(3.7)で係数をまとめると以下のように釣合式が得られる。

$$\left[M\right]\left\{\ddot{y}_{n+1}\right\}+\left[\bar{C}\right]\left\{\left\{a\right\}+\mu_1\left\{\ddot{y}_{n+1}\right\}\right\}+\left[K\right]\left\{\left\{\bar{b}\right\}+\mu_2\left\{\ddot{y}_{n+1}\right\}\right\}$$
$$=-\left[M\right]\left[I\right]\left\{\ddot{u}_g\right\}+\left\{P_S\right\}-\left\{\bar{f}_d\right\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots(4.5)$$

上式を整理すると、増分後の加速度を未知ベクトルとする釣合式が以下のように得られる。

$$\left[\left[M\right]+\mu_1\left[\bar{C}\right]+\mu_2\left[K\right]\right]\left\{\ddot{y}_{n+1}\right\}$$
$$=-\left[M\right]\left[I\right]\left\{\ddot{u}_g\right\}+\left\{P_S\right\}-\left\{\bar{f}_d\right\}-\left[\bar{C}\right]\left\{a\right\}-\left[K\right]\left\{\bar{b}\right\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots(4.6)$$

　線形解析部分の骨組みを以下に示す。釣合式の左辺係数行列の LDU 分解は処理１で得られている。ここでは、釣合式(4.6)の右辺項を計算し、方程式の解を求めることになる。

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         動的解析の開始
c
c―――――――――――――――――――――――――――――――――――――――
      if(Control.type_analysis .ne. 7) goto 9981
c―――――――――――――――――――――――――――――――――――――――
c
c    ★         線形解析
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 )                                           ! 2.1
c―――――――――――――――――――――――――――★地震加速度セット(ok)
      acc1=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)                       ! 2.2
      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()                                                          ! 2.3
c――――――――――――――――――――――★整合質量に関する慣性項
      call Add_earth2_ld()                                                          ! 2.4
c―――――――――――――――――――――――――――★節点荷重のセット(ok)
      p1=Get_Ps(T,1,fdd_point,Dynamic_load)                                         ! 2.5
      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()                                                           ! 2.6
c――――――――――――――――――――――★整合質量(ok)
      call Add_damp2_ld()                                                           ! 2.7
c――――――――――――――――――――――★部材減衰(ok)
      call Add_damp3_ld()                                                           ! 2.8
c―――――――――――――――――――――――――――★線形剛性項計算(ok)
c                                                          レーリー減衰も含む
      call Add_stiff1_ld()                                                          ! 2.9
c―――――――――――――――――――――――――――★Δｔ秒後の変位と速度を予測(ok)
```

```
      call Estimate_disp_vel()                                    ! 2.10
c-----------------------------★Maxwell 型モデルの計算(ok)
      call Add_fdd_ld()                                           ! 2.11
c-----------------------------★線形方程式を解く(ok)
      call solve_sky()                                            ! 2.12
c-----------------------------★加速度よりβ法に基づき変位と速度を計算(ok)
      call Cal_disp_vel()                                         ! 2.13
      goto 9980
```

プログラムの右側には番号が振ってあり、その処理内容について以下に示す。

2.1：釣合式の右辺項ベクトル ld_point をゼロセットする。
2.2：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。
2.3：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。
2.4：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。
2.5：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。
2.6：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。
2.7：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。
2.8：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。ここには、Maxwell 型の線形減衰項も含まれている。
2.9：線形の剛性項 $[K]\{\bar{b}\}$ を計算し、右辺項ベクトル ld_point 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 $[C]\{a\}$ の剛性に関する項を右辺項ベクトル ld_point に足し込む。
2.10：次のステップの Maxwell モデルで必要となるΔt 秒後の変位と速度を予測する。
2.11：Maxwell モデルによるベクトル $\{\bar{f}_d\}$ を計算し、ld_point に足し込む。
2.12：得られた右辺項とサブルーチン slove_sky()を用いて、方程式の解を得る。
2.13：求めた加速度を用いて Newmark β法の基本式より、速度と変位を計算する。

線形方程式(4.6)の右辺項は次のプログラム番号に従って求められる。

$-[M][I]\{\ddot{u}_g\}$	(2.3) (2.4)
$+\{P_S\}$	(2.5)
$-\{\bar{f}_d\}$	(2.11)
$-[\bar{C}]\{a\}$	(2.6),(2.7) (2.8),(2.9)
$-[K]\{\bar{b}\}$	(2.9)

以上が、処理２の「線形解析」部分の骨組みである。構造が簡単なので、釣合式と比較することで容易に理解できよう。ここでまた、主要なサブルーチンを以下に示す。内容については、コメントなどを参照して理解されたい。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Clear_vec
C       ―――――――――――――――――――――――――――――――――
C     ●                          ベクトルをゼロクリアする(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine Clear_vec(n_unknown,vec)
      implicit real*8(A-H,O-Z)
      dimension vec(*)
C     ―――――――――――――――――――――――――――――――――――
c     n_unknown        :integer  未知数
c     vec              :real*8   ベクトル
C     ―――――――――――――――――――――――――――――――――――
      do i=1,n_unknown
      vec(i)=0.0
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   FUNCTION /Get_Acc
C       ―――――――――――――――――――――――――――――――――
C     ●    加速度データをセットする(ok)
C     ―――――――――――――――――――――――――――――――――――
      real*8 function Get_Acc(TT,nx,acc_earth,Dynamic_load,T0)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / dynamic_load_s  / Dynamic_load
      real*8 acc_earth(3,*)
C     ―――――――――――――――――――――――――――――――――――
c      structure / dynamic_load_s/
c      integer    load_point(3)         !  節点荷重ありか
c      real*8     dt_load_point(3)      !  節点荷重の増分時間
c      integer    n_load_point          !  節点荷重ファイルの最大個数
c      integer    load_dynamic(3)       !  地震荷重ありか
c      real*8     amp_load_dynamic(3)   !  地震荷重の大きさ
c      real*8     dt_load_dynamic(3)    !  地震荷重の増分時間
c      integer    n_load_dynamic        !  地震荷重ファイルの最大個数
c      integer    load_mass             !  整合質量ありか
C     ―――――――――――――――――――――――――――――――――――
      Get_Acc =0.
      if(Dynamic_load.load_dynamic(nx) .eq. 0) return
      dt = Dynamic_load.dt_load_dynamic(nx)
      if(dt.eq.0.) return
      T=TT-T0
      if(T.le.0.) return
      i= T/dt
      if(i.lt.1) then
      Acc = acc_earth(nx,1)*T/dt
```

```
      elseif(i.le.Dynamic_load.n_load_dynamic) then
      dtt = t - i*dt
      Acc = acc_earth(nx,i)+(acc_earth(nx,i+1) -
     *                     acc_earth(nx,i))*dtt/dt
      else
      nnx = Dynamic_load.n_load_dynamic
      Acc = acc_earth(nx,nnx)
      endif
      Get_Acc = Acc
      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(3 ),amkk(3)
      real*8 ld_point(*)
c――――――――――――――――――――――――――――――――――――――――
c     n_istep                 :integer  第１段階か第２段階か
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
      if(Parameter_C.n_local_coord.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
      ij=Point(i).local_coord
      if(ij.eq.0) then
      do  j=1,3
      irest =Point(i).irest(j)
      if(irest.gt.0) ld_point(irest)=ld_point(irest) - amk(j )
      end do
```

```
      else
      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
      endif
      end do
      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(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
      dimension Member(*)
      dimension amk(12),amkk(6)
      real*8 ld_point(*)
c―――――――――――――――――――――――――――――――――――――――――― c
c     n_istep                 :integer  第１段階か第２段階か
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(Dynamic_load.load_mass .eq.0 ) return
      do i=1,12
      amk(i)=0.0
      enddo
      if(Parameter_C.n_local_coord.ne.0) then
```

```
      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 k=1,2
      ij=Member(i).nm_local_coord(k)
      if(ij.eq.0) then
      do  j=1, 6
      j1=j+(k-1)*6
      i1=Member(i).irest(j1)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amk(j1)
      end do
      else
      call RotateTLss_v(2,amk(1+(k-1)*6),rot_local(1,1,ij),amkk)
      do  j=1, 6
      j1=j+(k-1)*6
      i1=Member(i).irest(j1)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - amkk(j)
      end do
      endif
      end do
      end do
      else
      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
      end if
      return
      end
C     ――――――――――――――――――――――――――――――――――
C     ●   FUNCTION /Get_Ps
C       ――――――――――――――――――――――――――――――――――
C     ●   時刻歴節点荷重を取り出す(ok)
C     ――――――――――――――――――――――――――――――――――
      real*8 function Get_Ps(T,nx,fdd_point,Dynamic_load)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / dynamic_load_s  / Dynamic_load
      real*8 fdd_point(3,*)
C     ――――――――――――――――――――――――――――――――――
c      structure / dynamic_load_s/
c      integer    load_point(3)         !  節点荷重ありか
c      real*8     dt_load_point(3)      !  節点荷重の増分時間
c      integer    n_load_point          !  節点荷重ファイルの最大個数
```

```
c
c      T            :real*8   時刻
c      nx           :integer  節点荷重番号
c      fdd_point    :real*8   時刻歴節点荷重
c      Dynamic_load :structure
C     ―――――――――――――――――――――――――――――――――
      Get_Ps =0.
      if(Dynamic_load.load_point(nx) .eq. 0) return
      dt = Dynamic_load.dt_load_point(nx)
      if(dt.eq.0.) return
      i= T/dt
      if(i.lt.1) then
      Acc = fdd_point(nx,1)*T/dt
      elseif(i.le.Dynamic_load.n_load_point) then
      dtt = t - i*dt
      Acc = fdd_point(nx,i)+(fdd_point(nx,i+1) -
     *                       fdd_point(nx,i))*dtt/dt
      else
      nnx = Dynamic_load.n_load_point
      Acc = fdd_point(nx,nnx)
      endif
      Get_Ps = Acc
      return
      end
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_point_ld
C       ―――――――――――――――――――――――――――――――――
C     ●   節点荷重のセット(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Add_point_ld(p1,p2,p3,ld_point,n_unknown,
     *           Dynamic_load,fld_static)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / dynamic_load_s / Dynamic_load
      real*8 fld_static(3,*),ld_point(*)
c―――――――――――――――――――――――――――――――――――――――――――――――
c           p1,p2,p3        real*8  : 荷重係数
c           fld_point       real*8  : 右辺定数項
c           ld_static       real*8  : 釣合系における節点荷重
c           Dynamic_load            : structure
c―――――――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_unknown
      ld_point(i)=ld_point(i) + p1*fld_static(1,i)
     *                        + p2*fld_static(2,i)
     *                        + p3*fld_static(3,i)
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Add_damp1_ld
C       ―――――――――――――――――――――――――――――――――
C     ●   減衰・集中質量による減衰項のセット(ok)
C     ―――――――――――――――――――――――――――――――――
      subroutine Add_damp1_ld(nx,ld_point,Point,n_point,
```

```
     *       past_disp_point,past_vel_point,past_acc_point,
     *       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 past_disp_point(*),past_vel_point(*),past_acc_point(*)
c-----------------------------------------------------------------
c     nx                      :integer  第１段階か第２段階か
c     ld_point(*)             :real*8   右辺荷重項
c     Point                   :structure
c     n_point                 :integer  節点数
c     past_disp_point         :real*8   変位
c     past_vel_point          :real*8   速度
c     past_acc_point          :real*8   加速度
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
      b= Newmark_P.alf1_1*Newmark_P.ddt_1
      ik=1
      else
      a= Newmark_P.alf2_1
      b= Newmark_P.alf2_1*Newmark_P.ddt_1
      ik=2
      endif
c-----------------------------------------------------------------
      if(Parameter_C.n_local_coord.ne.0) then
      do i=1,n_point
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      u(j)=a*past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      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
      end do
c-----------------------------------------------------------
      else
      do i=1,n_point
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.ne.0) then
      u(j)=a*past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      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
      end do
c-----------------------------------------------------------
      endif
      return
      end
C     ------------------------------------------
C     ●   SUBROUTINE /Add_damp2_ld
C       --------------------------------------
C     ●   減衰・整合質量による減衰項のセット(ok)
C     ------------------------------------------
      subroutine Add_damp2_ld(nx,ld_point,Member,n_member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,load_mass)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s    /Newmark_P
      record / member_s     /Member
      record / element_s    /Element
      dimension Member(*),Element(*)
      real*8 ld_point(*),am_member(12,12,*)
      dimension u(12)
      dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
```

```
c-------------------------------------------------------------
      if(load_mass.eq.0) return
c-------------------------------------------------------------
      if(nx.eq.1) then
      a= Newmark_P.alf1_1
      b= Newmark_P.alf1_1*Newmark_P.ddt_1
      ik=1
      else
      a= Newmark_P.alf2_1
      b= Newmark_P.alf2_1*Newmark_P.ddt_1
      ik=2
      endif
c-------------------------------------------------------------
      do i=1,n_member
      ie = Member(i).nm_element
      if(Element(ie).am(ik) .ne. 0.0) then
      do j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)=a*past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
      do  j=1,12
      i1=Member(i).irest(j)
      if(i1.gt.0) then
      sum=0.
      do k=1,12
      sum=sum+am_member(j,k,i)*u(k)
      enddo
      ld_point(i1)=ld_point(i1) - sum
      endif
      end do
c-------------------------------------------------------------
      endif
      end do
      return
      end
C     ---------------------------------------
C     ●   SUBROUTINE /Add_damp3_ld
C       -------------------------------------
C     ●   部材減衰による減衰項のセット(ok)
C     ---------------------------------------
      subroutine Add_damp3_ld(nx,ld_point,Member,n_member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ac_member,Newmark_P,n_damp)
      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(*),ac_member(12,12,*)
```

```
      dimension u(12)
      dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
c-----------------------------------------------------------------
      if(n_damp.eq.0) return
c-----------------------------------------------------------------
      b=Newmark_P.ddt_1
      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)=  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.gt.0) then
      sum=0.
      do k=1,12
      sum=sum+ac_member(j,k,ij)*u(k)
      enddo
      ld_point(irest)=ld_point(irest) - sum
      endif
      end do
      endif
      end do
      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１段階か第２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
```

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。

4.6 予備計算

前節では、数値解析部分を解説したが、ここでは、計算に先立って行うデータ入力と予備計算について説明する。データ入力部は、計算を制御するためのデータ入力と計算に必要なデータ入力がある。前節と同様、この部分のプログラムを以下に示す。

```
c-------------------------------------------------------------
c
c    ★        サブルーチン定義
c
c-------------------------------------------------------------
c-------------------------------------------------------------
      subroutine submain_dynamic_a(i_calnum,iend_code,icontrol,ierr_dat,
     *           T,dt,n_step,ns_step,d_max_v,id_max_v,
     *           i_read_disp,F_disp,i_read_ndbalanceF,F_ndbalanceF,
     *           i_read_spring,F_fay,F_n_spring,F_my_spring,
     *           F_mz_spring,i_stat_spring,n_iterate,
     *           nm_iterate,numb_method)
c-------------------------------------------------------------
      implicit real*8(A-H,O-Z)

c-------------------------------------------------------------
c-------------------------------------------------------------
c
c    ★        システムからの制御情報を取得（以後実行文）
c
c-------------------------------------------------------------
c-------------------------------------------------------------
      if(icontrol .eq. 1  ) goto 9990    ! 解析処理へ
      if(icontrol .eq. 99 ) goto 9997    ! 終了処理へ
      do i=1,10
      M_alloc(i)=0                      ! 動的配列の確保をチェックする配列をゼロセット
      enddo
      open (damp_out,FILE='DOUTPUT')
c-----------------------------------★システムからのコントロール情報を取得(ok)
      call sysnam(FNX_file,N_analysis)
      if(N_analysis.ne.i_calnum) N_analysis=i_calnum  ! 解析番号を変更
      if(N_analysis.le.6) then
      ierr_dat=1
      call err_outf(ierr_dat)
      return
      endif
      ihan = 0
      ierr = 0
      NFILE=100
      ierr_dat =0
      write(damp_out,*) ' System file input ok. No. of analysis:',
     *                  N_analysis
c-----------------------------------★コントロールデータの内容を取得(ok)
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      if(ihan.ne.0) then
      ierr_dat = 2
      call err_outf(ierr_dat)
```

```
      return
      endif
c-----------------------------------★動的解析ダイアログその１のデータを取得(ok)
      call dyctl1(ierr,NINDIT,GINDIS,F1SEC,fs_st,fl_st,ifp_st,IST,
     *           JIKUZERO,G_JIKUZERO_ALPH)
      if(ierr.ne.0) then
       ierr_dat = 3
       call err_outf(ierr_dat)
      endif
c-----------------------------------★動的解析ダイアログその２のデータを取得(ok)
      call dyctl2(ierr,NSTEP,F2SEC,DELT,IGRA,IBETA,BETA,GUMMA,XGAL,
     *           NNTIME,EPSDSP,load_memb_mass,dt_M_filter,IT_ANALYS)
      if(ierr.ne.0) then
       ierr_dat = 4
       call err_outf(ierr_dat)
      endif
c-----------------------------------★解析結果の出力パラメータを取得(ok)
      call doutcl(ierr,IWSTP,SOUTSC,DMAXCK,No_section)
      if(ierr.ne.0) then
       ierr_dat = 5
       call err_outf(ierr_dat)
      endif
c-----------------------------------★減衰ダイアログのデータを取得(ok)
      call damctl(ierr,NREAD,ITYDP,NDMP,NDMP2,NHH,HH,QHH)
      if(ierr.ne.0) then
       ierr_dat = 6
       call err_outf(ierr_dat)
      endif
c-----------------------------------★制御情報の取得に失敗した場合はここで戻る
      if(ierr_dat .ne. 0 ) return
c-------------------------------------------------------------
c
c    ★        制御情報を構造体にセット
c
c-------------------------------------------------------------
      call Set_control(Control,N_analysis,NINDIT,GINDIS,
     *                 IWSTP,SOUTSC,DMAXCK,in_disp,in_stres,
     *                 IT_ANALYS,JIKUZERO,G_JIKUZERO_ALPH)       !ok (in_disp,in_stres  未定義)
c     write(damp_out,*) ' Set_control Ok'
      call Set_model_type(Model_type)
c     write(damp_out,*) ' Set_model_type Ok'
      call Set_newmark(Newmark_P,F1SEC,F2SEC,HH(1),HH(2),
     *                 QHH(1),QHH(2),DELT,BETA,EPSDSP,NNTIME,GUMMA,
     *                 ITYDP,NDMP,NDMP2)
c     write(damp_out,*) ' Set_newmark Ok'
      call Set_dynamic_load(Dynamic_load,IST,IGRA,
     *                      XGAL,load_memb_mass)
c     write(damp_out,*) ' Set_dynamic_load Ok'
c-------------------------------------------------------------
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
c-------------------------------------------------------------
      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_parameters(Parameter_C,ierr)
      close(nfix)
      if(ierr .ne. 0) then
      ierr_dat =11
      call err_outf(ierr_dat)
      return
      endif
c―――――――――――――――――――――――――――――――――★動的解析制限条件チェック
      call Check_analysis_condition(Parameter_C,ierr,
     *    IAnalysis_number_condition,Number_of_nodes,Number_of_members)
      if(ierr .ne. 0) then
      ierr_dat =299
      call err_outf(ierr_dat)
      return
      endif
c―――――――――――――――――――――――――――――――――★地震加速度を予備入力(ok)
      call Get_earth_load(1,acc_earth,Dynamic_load,ierr)
      if(ierr.ne.0) then
      ierr_dat =21
      call err_outf(ierr_dat)
      return
      endif
c―――――――――――――――――――――――――――――――――★節点荷重履歴を予備入力(ok)
      call Get_point_load(1,fdd_point,Dynamic_load,ierr,
     *                    Newmark_P,fs_st,fl_st,ifp_st)
      if(ierr.ne.0) then
      ierr_dat =22
      call err_outf(ierr_dat)
      return
      endif
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★   DLL を使用する場合、ワークエリアとして必要な大きさ及び DLL 番号を得る
c                                      DLL subroutine code No. 1
c
c――――――――――――――――――――――――――――――――――――――――――――――――
c      if(Parameter_C.n_element_dll .ne. 0 )
c     * call Get_DLL_code(dll_code,n_request1,n_request2,
c     *                   n_request3,n_request4)
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_point            ! 節点数
      ALLOCATE (
     *   fll_static_point(3,6,N),am_point(2,N),
     *   fll_force_point(3,N)
     *         )
      N=Parameter_C.n_member           ! 部材数
      ALLOCATE (
     *   am_member(12,12,N),
     *   rot_memb_t(3,3,N),rot_memb(3,3,2,N),
     *   ak_linear(12,12,N),ak_nonlinear(12,12,N)
     *         )
      N=Parameter_C.n_local_coord      ! 局所座標系を使用する場合
      if(N.ne.0)ALLOCATE (
     *   rot_local(3,3,N)
     *)
       M_alloc(1)=1
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)
      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)
       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    ★        制震セミアクティブダンパー用フィルターのパラメータセット
c
c―――――――――――――――――――――――――――――――――――――――――
      if(Model_type.n_m_model(6) .ne. 0) then
      call Set_Maxwell_filter(dt_M_filter,Newmark_P.dt,Model_type,ierr)
```

```
      if(ierr.ne.0) then
       ierr_dat = 20
       call err_outf(ierr_dat)
       return
       endif
c     write(damp_out,*) ' Set_Maxwell_filter Ok'
      endif
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-----------------------------------------------------------
      N=Model_type.n_m_damp
      if(N.ne.0)then
      ALLOCATE (ac_member(12,12,N))
      endif
c-------------------------------★Model_No.1 通常の有限要素弾塑性モデル
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
c-------------------------------★Model_No.3 3 次元軸力弾塑性モデル
c     n=Model_type.n_m_model(3)
c     if(n.ne.0)then
c     ALLOCATE(N_Buckling(n))
c     endif
c-------------------------------★Model_No.4 3 次元ケーブル弾塑性モデル
c-------------------------------★Model_No.5 3 次元免振モデル
      n=Model_type.n_m_model(5)
      if(n.ne.0) then
      ALLOCATE (MSS_work(n))
      endif
c-------------------------------★Model_No.6 3 次元制震 Maxwell モデル
      n=Model_type.n_m_model(6)
      if(n.ne.0) then
      ALLOCATE (E_model6_real(n))
      endif
c-------------------------------★Model_No.7 3 次元バネモデル
c     n=Model_type.n_m_model(7)
c     if(n.ne.0) then
c       ALLOCATE (E_model7(n))
c     endif
c-------------------------------★ファイバーモデル
c-------------------------------★Model_No.11 両端ファイバーモデル
      n= Model_type.n_e_model(11)        !要素モデルの数
      if(n.ne.0) then
      ALLOCATE (E_model11(n))
      endif
      n= Model_type.n_m_model(11)        !部材モデルの数
      if(n.ne.0) then
      ALLOCATE ( M_model11(n))
      endif
c-------------------------------★Model_No.12 両端、中央ファイバーモデル
```

```
      n= Model_type.n_e_model(12)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model12(n))
      endif
      n= Model_type.n_m_model(12)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model12(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.13 両端、中央ファイバーモデル
      n= Model_type.n_e_model(18)        !要素モデルの数
       if(n.ne.0) then
       ALLOCATE (E_model13(n))
      endif
      n= Model_type.n_m_model(18)        !部材モデルの数
       if(n.ne.0) then
       ALLOCATE ( M_model13(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.15 両端ファイバーFEM モデル
      n= Model_type.n_e_model(15)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model15(n))
      endif
      n= Model_type.n_m_model(15)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model15(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.21 両端 MS モデル
      n= Model_type.n_e_model(13)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model21(n))
      endif
      n= Model_type.n_m_model(13)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model21(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.22 両端、中央 MS モデル
      n= Model_type.n_e_model(14)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model22(n))
      endif
      n= Model_type.n_m_model(14)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model22(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.31 両端アナロジーモデル
      n= Model_type.n_e_model(16)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model31(n))
      endif
      n= Model_type.n_m_model(16)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model31(n))
      endif
c――――――――――――――――――――――――――――――――――★Model_No.32 両端、中央アナロジーモデル
```

```
      n= Model_type.n_e_model(17)        !要素モデルの数
       if(n.ne.0) then
      ALLOCATE (E_model32(n))
      endif
      n= Model_type.n_m_model(17)        !部材モデルの数
       if(n.ne.0) then
      ALLOCATE ( M_model32(n))
      endif
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―――――――――――――――――――――――――――――――――★Model_No.51 3 次元プレテンション動作モデル
c      n=Model_type.n_m_model(51)
c     if(n.ne.0) then
c     ALLOCATE (M_model51(n),
c    *          E_model51(n))
c      endif
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        配列の大きさを動的確保する（その２：DLL 用）
c
c――――――――――――――――――――――――――――――――――――――――――――――
c     if(Parameter_C.n_element_dll .ne. 0 ) then
c      N1=n_request1* Parameter_C.n_element_dll
c      N2=n_request2* Parameter_C.n_element_dll
c       N3=n_request3* Parameter_C.n_member_dll
c       N4=n_request4* Parameter_C.n_member_dll
c      ALLOCATE (
c     *   work1_elememt(N1),work2_elememt(N2),
c     *   work1_member(N3),work2_member(N4)
c     *         )
c     endif
c―――――――――――――――――――――――――――――――――★節点荷重、加速度データ領域を確保
      if(Dynamic_load.n_load_dynamic .ne. 0) then
      ALLOCATE (acc_earth(3,Dynamic_load.n_load_dynamic))
       endif
      if(Dynamic_load.n_load_point .ne. 0) then
      ALLOCATE (fdd_point(3,Dynamic_load.n_load_point))
       endif
       M_alloc(2)=1
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        構造・荷重データを入力し、データの設定を行う（その２）
c
c――――――――――――――――――――――――――――――――――――――――――――――
c―――――――――――――――――――――――――――――――――★地震荷重を入力(ok)
      if(Dynamic_load.load_dynamic(1)  .ne. 0 .or.
     *   Dynamic_load.load_dynamic(2)  .ne. 0 .or.
```

```
     *   Dynamic_load.load_dynamic(3)  .ne. 0 ) then
      call Get_earth_load(2,acc_earth,Dynamic_load,ierr)
       if(ierr.ne.0) then
      ierr_dat =21
       call err_outf(ierr_dat)
       return
       endif
c     write(damp_out,*) ' Get_earth_load Ok'
      endif
c―――――――――――――――――――――――――――――――――★節点荷重分布を入力(ok)
      if(Dynamic_load.load_point(1)  .ne. 0 .or.
     *   Dynamic_load.load_point(2)  .ne. 0 .or.
     *   Dynamic_load.load_point(3)  .ne. 0 ) then
      call Get_point_loadf(fll_static_point,Parameter_C,
     *                     Dynamic_load,ierr)
       if(ierr.ne.0) then
      ierr_dat =22
       call err_outf(ierr_dat)
       return
       endif
c     write(damp_out,*) ' Get_point_loadf Ok'
c―――――――――――――――――――――――――★節点荷重時刻歴を入力およびセット
      call Get_point_load(2,fdd_point,Dynamic_load,ierr,
     *                    Newmark_P,fs_st,fl_st,ifp_st)
       if(ierr.ne.0) then
      ierr_dat = 23
       call err_outf(ierr_dat)
       return
       endif
c     write(damp_out,*) ' Get_point_load Ok'
      endif
c―――――――――――――――――――――――――――――――――★初期不整データを入力(ok)
      if(Control.init_imperfection .ne. 0) then
      nfix=5
       nfi=16
      call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
      ierr_dat = 24
       call err_outf(ierr_dat)
       return
       endif
      call Get_imperfection(Control.amp_imperfection,Point,Parameter_C)
       close(nfix)
       endif
c―――――――――――――――――――――――――――――――――★システム内要素データを入力
c    ★        ファイバーデータ
c―――――――――――――――――――――――――――――★ファイバーデータの予備入力(ok)
      n_element=Model_type.n_e_model(11)+Model_type.n_e_model(12)+
     *          Model_type.n_e_model(15)+
     *          Model_type.n_e_model(14)+Model_type.n_e_model(13)+
     *          Model_type.n_e_model(16)+Model_type.n_e_model(17)+
     *          Model_type.n_e_model(18)++Model_type.n_e_model(19)
      if(n_element .ne. 0) then
      nfix=5
```

```
       nfi=54
       call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
       ierr_dat = 25
       call err_outf(ierr_dat)
       return
       endif
       call Fiber_input(0,ierr,Parameter_C.n_member,
      *      Parameter_C.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)
       close(nfix)
       if(ierr.ne.0)then
       ierr_dat = 26
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' Fiber_input ok'
c―――――――――――――――――――――――――――――★ファイバーモデル領域セット
      n= Model_type.nm_div_fmodel          !要素モデル内のサブ要素の数
       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
      M_alloc(3)=1
c―――――――――――――――――――――――――――――★ファイバーモデルデータ入力
       nfix=5
        nfi=54
       call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
       ierr_dat = 25
       call err_outf(ierr_dat)
       return
       endif
       call Fiber_input(1,ierr,Parameter_C.n_member,
      *      Parameter_C.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)
       close(nfix)
       if(ierr.ne.0) then
       ierr_dat = 26
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' Fiber_input ok'
      endif
c―――――――――――――――――――――――――――――★修正 R0 モデル領域セット
```

```
      n=Model_type.n_m_ro_model
       if(n.ne.0) then
       nfix=5
        nfi=53
       call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
      ierr_dat = 35
       call err_outf(ierr_dat)
       return
       endif
       call RO_data_input(n,RO_work,Element,Parameter_C.n_element,ierr)
       close(nfix)
       if(ierr.ne.0) then
      ierr_dat = 36
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' RO_data_input ok'
       endif
c------------------------------★DLL 内要素データを入力
c                                  DLL subroutine code No. 2
c       n_element=Parameter_C.n_member_dll
c       if(n_element .ne. 0) then
c       nfix=5
c      nfi=60
c       call infile(nfi,nfix,ierr)
c      if(ierr.ne.0) then
c       ierr_dat =13
c      return
c      endif
c       call Get_element_dll(work1_elememt,work2_elememt,Parameter_C)
c      close(nfix)
c      endif
c------------------------------★質量データを入力(ok)
      nfix=5
       nfi=2
      call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
      ierr_dat = 27
       call err_outf(ierr_dat)
       return
       endif
      call Get_mass(ierr,Point,Parameter_C)
       close(nfix)
       if(ierr.ne.0) then
      ierr_dat = 28
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' Get_mass Ok'
c------------------------------★レーリー減衰を入力(ok)
      nfix=5
       nfi=43
      call infile(nfi,nfix,ierr)
```

```
      if(ierr.ne.0) then
      ierr_dat = 29
      call err_outf(ierr_dat)
      return
      endif
      call Get_damp(Newmark_P,ierr)
      close(nfix)
      if(ierr.ne.0) then
      ierr_dat =30
      call err_outf(ierr_dat)
      return
      endif
c     write(damp_out,*) ' Get_damp Ok'
c-----------------------------------------------------------
c
c    ★        動的解析のための予備計算を行う
c
c-----------------------------------------------------------
c-------------------------------★構造物の不安定性チェック
c      call check_structure(Point,Element,Parameter_C)
c-------------------------------★部材長さ計算(ok)
      call Cal_member_length(Member,Point,Parameter_C)
c     write(damp_out,*) ' Cal_member_length Ok'
c-------------------------------★モデルの初期設定
      call Set_initial_data(Element,Member,Parameter_C,Newmark_P,
     *           E_model6_real,Model_type)
c     write(damp_out,*) ' Set_initial_data Ok'
c-------------------------------★DLL 非線形エリアの設定
c      if(Parameter_C.n_member_dll .ne. 0) then
c      call Set_nonlinear_S(Element,Parameter_C,
c     *     work1_elememt,work2_elememt,work1_member,work2_member)
c      endif
c-------------------------------★座標変換行列計算
      call Get_rotate_all(rot_memb_t,Parameter_C,Point,Member)
c                write(damp_out,*) ' Get_rotate_all Ok'
c-------------------------------★局所座標変換行列計算
      call Get_rot_local(rot_local,Parameter_C,Point)
c                write(damp_out,*) ' Get_rot_local Ok'
c-------------------------------★釣合座標系標変換行列計算
      call Get_rotate(rot_memb,rot_memb_t,rot_local,
     *               Parameter_C,Member)
c     write(damp_out,*) ' Get_rotate Ok'
c-------------------------------★節点拘束表の作成
c                                                 未知数等をセット
      call Set_restraint_point(Parameter_C,Point,Control)
c     write(damp_out,*) ' Set_restraint_point Ok'
c-----------------------------------------------------------
c
c    ★        配列の大きさを動的確保する（その３）
c
c-----------------------------------------------------------
      N=Parameter_C.n_unknown
c      ALLOCATE (test_vector(N))
      ALLOCATE (
```

```
*     disp_point(N),vel_point(N),acc_point(N),
*     est_disp_point(N),est_vel_point(N),
*     est_ddisp_point(N)
*         )
 ALLOCATE (
*   result_disp_point(N),result_vel_point(N),result_acc_point(N),
*   past_disp_point(N),past_vel_point(N),past_acc_point(N),
*   past_dacc_point(N)
*         )
 ALLOCATE (
*   ld_point(N),ld_point_repeat(N),fld_static(3,N),
*   a_vector(N),b_vector(N)
*         )
 ALLOCATE (max_h_sky(0:N+1)
*         )
  M_alloc(4)=1
c――――――――――――――――――――――――――――――――――★部材両端の拘束表作成(ok)
      call Set_restraint_member(Parameter_C,Member,Point)
c     write(damp_out,*)' Set_restraint_member Ok'
c――――――――――――――――――――――――――――――――――★スカイライン変換表作成(ok)
c――――――――――――――――――――――――――――――――――★スカイライン行列の領域数等をセット
      call Cal_table_sky(max_h_sky,Parameter_C,Member)
c     write(damp_out,*) ' Cal_table_sky Ok'
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        配列の大きさ（スカイライン用）を動的確保する（その４）
c
c――――――――――――――――――――――――――――――――――――――――――――――
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の数
       if(n.ne.0) then
      ALLOCATE (Concrete_work(n))
      endif
c――――――――――――――――――――――――――――――――――★全体剛性行列など
      ALLOCATE (gskym(Parameter_C.n_skyline))
      ALLOCATE (gskym_d(Parameter_C.n_unknown))
      ALLOCATE (nwork(Parameter_C.n_unknown))
      ALLOCATE (twork(Parameter_C.n_unknown))
       M_alloc(5)=1
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――――――――――――――――――――――――――――――――――★MSS モデルの初期設定
```

```
      n=Model_type.n_m_model(5)
       if(n.ne.0) then
      call Cal_MSS_dat(Member,Element,Model_type,
     *                 MSS_work,Parameter_C)
c     write(damp_out,*) ' Cal_MSS_dat Ok'
      endif
c―――――――――――――――――――――――――――――――――★平面問題における部材の拘束方向チェック
      call Check_R_direction(Control.analysis_3D,Parameter_C,
     *                       Member,rot_memb)
c     write(damp_out,*) ' Check_R_direction ok'
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'
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)
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'
c―――――――――――――――――――――――――――――――――★節点荷重セット(ok)
      call Set_point_load(fll_static_point,Parameter_C,
     *              Dynamic_load,Point,fld_static,rot_local)
c     write(damp_out,*) ' Set_point_load Ok'
c―――――――――――――――――――――――――――――――――★接線剛性のコピー(ok)
      call Initset_nonlin_stiff(ak_linear,ak_nonlinear,
     *                          Parameter_C.n_member)
c     write(damp_out,*) ' Initset_nonlin_stiff 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
c-------------------------------------------------------------
c
c    ★          静的解析結果等を取り込む
c                （初期変位、初期応力を入力）
c
c-------------------------------------------------------------
c-----------------------------★初期変位を入力
       if(Control.init_disp.ne.0) then
       nfix=5
       nfi=16
c      call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
       ierr_dat = 31
       call err_outf(ierr_dat)
       return
       endif
c      call Get_init_disp(Point,Member,Parameter_C,ierr)
       close(nfix)
       if(ierr.ne.0) then
       ierr_dat =32
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' Get_damp Ok'
       endif
c-----------------------------★初期応力を入力
       if(Control.init_stress.ne.0) then
       nfix=5
       nfi=52
c      call infile(nfi,nfix,ierr)
       if(ierr.ne.0) then
       ierr_dat = 33
       call err_outf(ierr_dat)
       return
       endif
c      call Get_init_stress(Point,Member,Parameter_C,ierr)
       close(nfix)
       if(ierr.ne.0) then
       ierr_dat =34
       call err_outf(ierr_dat)
       return
       endif
c      write(damp_out,*) ' Get_init_stress Ok'
       endif
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
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)
c――――――――――――――――――――――――――――――――――――★ファイル名一覧出力
      write(damp_out,*) ' flcheck Ok'
     *  ,Dynamic_load.load_dynamic(1)
      do i=1,16
      write(damp_out,'(i4,3i6)') i,ifl(i),iflz(i),ifly(i)
      enddo
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
      call Set_preset_disp(0,n_point,past_disp_point,F_disp,Point,
     *                           rot_local,Parameter_C)
c      write(damp_out,*) ' Set_preset_disp ok'
      endif
c――――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        解析ステップをセットする
c
c――――――――――――――――――――――――――――――――――――――――――――――――――
      ns_step=1
      n_step=10
      d_max_v=0.
      id_max_v=0
      stimes=secnds(0.0)                        ! 解析時間を計る
c――――――――――――――――――――――――――――――――――――★解析手法のセット
      N_implicit_method = 1  !陰解法は１回で反復法に戻る
      Iteration_method  = 1  !最初は反復法を使用
c――――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        予備計算はここで終了（GUI に戻る）
c
c――――――――――――――――――――――――――――――――――――――――――――――――――
      return
```

前節と同様に、プログラムの骨組みを用いて、処理の流れを理解しよう。多少長いが、プログラムの構造は単純なので理解し易い。

```
c――――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        サブルーチン定義
c
```

```
c-----------------------------------------------------------------
c-----------------------------------------------------------------
      subroutine submain_dynamic_a(i_calnum,iend_code,icontrol,ierr_dat,          ! 1
     *           T,dt,n_step,ns_step,d_max_v,id_max_v,
     *           i_read_disp,F_disp,i_read_ndbalanceF,F_ndbalanceF,
     *           i_read_spring,F_fay,F_n_spring,F_my_spring,
     *           F_mz_spring,i_stat_spring,n_iterate,
     *           nm_iterate,numb_method)
c-----------------------------------------------------------------
      implicit real*8(A-H,O-Z)                                                   ! 2
c-----------------------------------------------------------------
c-----------------------------------------------------------------
c
c    ★        システムからの制御情報を取得（以後実行文）
c
c-----------------------------------------------------------------
c-----------------------------------------------------------------
      if(icontrol .eq. 1  ) goto 9990    ! 解析処理へ                           ! 3
      if(icontrol .eq. 99 ) goto 9997    ! 終了処理へ                           ! 4
      do i=1,10
      M_alloc(i)=0          ! 動的配列の確保をチェックする配列をゼロセット       ! 5
      enddo
      open (damp_out,FILE='DOUTPUT')                                             ! 6
c-------------------------------★システムからのコントロール情報を取得(ok)
      call sysnam()                                                              ! 7
c-------------------------------★コントロールデータの内容を取得(ok)
      call ctlset()                                                              ! 8
c-------------------------------★動的解析ダイアログその１のデータを取得(ok)
      call dyctl1()                                                              ! 9
c-------------------------------★動的解析ダイアログその２のデータを取得(ok)
      call dyctl2()                                                              ! 10
c-------------------------------★解析結果の出力パラメータを取得(ok)
      call doutcl()                                                              ! 11
c-------------------------------★減衰ダイアログのデータを取得(ok)
      call damctl()                                                              ! 12
c-------------------------------★制御情報の取得に失敗した場合はここで戻る
      if(ierr_dat .ne. 0 ) return                                                ! 13
c-----------------------------------------------------------------
c
c    ★        制御情報を構造体にセット
c
c-----------------------------------------------------------------
      call Set_control()       !ok (in_disp,in_stres  未定義)                  ! 14
      call Set_model_type()                                                      ! 15
      call Set_newmark()                                                         ! 16
      call Set_dynamic_load()                                                    ! 17
c-----------------------------------------------------------------
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
c-----------------------------------------------------------------
      call Get_parameters()                                                      ! 18
c-------------------------------★地震加速度を予備入力(ok)
      call Get_earth_load()                                                      ! 19
```

```
c――――――――――――――――――――――――――――――――★節点荷重履歴を予備入力(ok)
      call Get_point_load()                                         ! 20
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))                      ! 21
      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_point            ! 節点数
      ALLOCATE (
     *   fll_static_point(3,6,N),am_point(2,N),
     *   fll_force_point(3,N)
     *         )
      N=Parameter_C.n_member           ! 部材数
      ALLOCATE (
     *   am_member(12,12,N),
     *   rot_memb_t(3,3,N),rot_memb(3,3,2,N),
     *   ak_linear(12,12,N),ak_nonlinear(12,12,N)
     *         )
      N=Parameter_C.n_local_coord      ! 局所座標系を使用する場合
      if(N.ne.0)ALLOCATE (
     *   rot_local(3,3,N)
     *)
      M_alloc(1)=1                                                  ! 22
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)
      call Get_structure()                                          ! 23
c――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        制震セミアクティブダンパー用フィルターのパラメータセット
c
c――――――――――――――――――――――――――――――――――――――――――――――
      if(Model_type.n_m_model(6) .ne. 0) then                       ! 24
      call Set_Maxwell_filter()                                     ! 25
      endif
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――――――――――――――――――――――――――――――――――――――――――――――
      N=Model_type.n_m_damp
```

```
      if(N.ne.0)then                                                    ! 26
      ALLOCATE (ac_member(12,12,N))                                     ! 27
      endif
c――――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
c――――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
c          以後、処理法は同じ
      ALLOCATE (RO_work(n))
c――――――――――――――――――――――――――――――――――★Model_No.4 3 次元ケーブル弾塑性モデル
c――――――――――――――――――――――――――――――――――★Model_No.5 3 次元免振モデル
      ALLOCATE (MSS_work(n))
c――――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル
      ALLOCATE (E_model6_real(n))
c――――――――――――――――――――――――――――――――――★ファイバーモデル
c――――――――――――――――――――――――――――――――――★Model_No.11 両端ファイバーモデル
      ALLOCATE (E_model11(n))
      ALLOCATE (M_model11(n))
c――――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
      ALLOCATE (E_model12(n))
      ALLOCATE (M_model12(n))
c――――――――――――――――――――――――――――――――――★Model_No.13 両端、中央ファイバーモデル
      ALLOCATE (E_model13(n))
      ALLOCATE (M_model13(n))
c――――――――――――――――――――――――――――――――――★Model_No.15 両端ファイバーFEM モデル
      ALLOCATE (E_model15(n))
      ALLOCATE (M_model15(n))
c――――――――――――――――――――――――――――――――――★Model_No.21 両端 MS モデル
      ALLOCATE (E_model21(n))
      ALLOCATE (M_model21(n))
c――――――――――――――――――――――――――――――――――★Model_No.22 両端、中央 MS モデル
      ALLOCATE (E_model22(n))
      ALLOCATE (M_model22(n))
c――――――――――――――――――――――――――――――――――★Model_No.31 両端アナロジーモデル
      ALLOCATE (E_model31(n))
      ALLOCATE (M_model31(n))
c――――――――――――――――――――――――――――――――――★Model_No.32 両端、中央アナロジーモデル
      ALLOCATE (E_model32(n))
      ALLOCATE (M_model32(n))
c――――――――――――――――――――――――――――――――――★Model_No.33 両端ピン、中央アナロジーモデル
      ALLOCATE (E_model33(n))
      ALLOCATE (M_model33(n))
c――――――――――――――――――――――――――――――――――★節点荷重、加速度データ領域を確保
      ALLOCATE (acc_earth(3,Dynamic_load.n_load_dynamic))
      ALLOCATE (fdd_point(3,Dynamic_load.n_load_point))
      M_alloc(2)=1                                                      ! 28
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★         構造・荷重データを入力し、データの設定を行う（その２）
c
c――――――――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――★地震荷重を入力(ok)
      call Get_earth_load()                                             ! 29
c――――――――――――――――――――――――――――――――――★節点荷重分布を入力(ok)
      call Get_point_loadf()                                            ! 30
c――――――――――――――――――――――――――★節点荷重時刻歴を入力およびセット
```

```
      call Get_point_load()                                          ! 31
c-----------------------------------★初期不整データを入力(ok)
      call Get_imperfection()                                        ! 32
c-----------------------------------★システム内要素データを入力
c    ★        ファイバーデータ
c-----------------------------------★ファイバーデータの予備入力(ok)
      call Fiber_input()                                             ! 33
c-----------------------------------★ファイバーモデル領域セット
      ALLOCATE (M_model_fiber(n))                                    ! 34
      ALLOCATE (E_model_fiber(n))
      M_alloc(3)=1                                                   ! 35
c-----------------------------------★ファイバーモデルデータ入力
      call Fiber_input()                                             ! 36
c-----------------------------------★修正 RO モデル領域セット
      call RO_data_input()                                           ! 37
c-----------------------------------★質量データを入力(ok)
      call Get_mass()                                                ! 38
c-----------------------------------★レーリー減衰を入力(ok)
      call Get_damp()                                                ! 39
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()                                       ! 40
c-----------------------------------★モデルの初期設定
      call Set_initial_data()                                        ! 41
c-----------------------------------★座標変換行列計算
      call Get_rotate_all()                                          ! 42
c-----------------------------------★局所座標変換行列計算
      call Get_rot_local()                                           ! 43
c-----------------------------------★釣合座標系標変換行列計算
      call Get_rotate()                                              ! 44
c-----------------------------------★節点拘束表の作成
c                                     未知数等をセット
      call Set_restraint_point()                                     ! 45
c-------------------------------------------------------------
c
c    ★        配列の大きさを動的確保する（その３）
c
c-------------------------------------------------------------
      ALLOCATE (                                                     ! 46
     *     disp_point(N),vel_point(N),acc_point(N),
     *     est_disp_point(N),est_vel_point(N),
     *     est_ddisp_point(N)
     *         )
      ALLOCATE (
     *   result_disp_point(N),result_vel_point(N),result_acc_point(N),
     *   past_disp_point(N),past_vel_point(N),past_acc_point(N),
     *   past_dacc_point(N)
     *         )
      ALLOCATE (
     *   ld_point(N),ld_point_repeat(N),fld_static(3,N),
```

```
*   a_vector(N),b_vector(N)
*         )
 ALLOCATE (max_h_sky(0:N+1)
*         )
 M_alloc(4)=1                                                    ! 47
c――――――――――――――――――――――――――――――――――★部材両端の拘束表作成(ok)
      call Set_restraint_member()                                ! 48
c――――――――――――――――――――――――――――――――――★スカイライン変換表作成(ok)
c――――――――――――――――――――――――――――――――――★スカイライン行列の領域数等をセット
      call Cal_table_sky()                                       ! 49
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★        配列の大きさ（スカイライン用）を動的確保する（その４）
c
c――――――――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――★ファイバー用履歴要素
      ALLOCATE (Bilinear_work(n))                                ! 50
      ALLOCATE (Trilinear_work(n))
      ALLOCATE (Concrete_work(n))
c――――――――――――――――――――――――――――――――――★全体剛性行列など
      ALLOCATE (gskym(Parameter_C.n_skyline))
      ALLOCATE (gskym_d(Parameter_C.n_unknown))
      ALLOCATE (nwork(Parameter_C.n_unknown))
      ALLOCATE (twork(Parameter_C.n_unknown))
      M_alloc(5)=1                                               ! 51
c――――――――――――――――――――――――――――――――――★部材両端中央応力のゼロセット(ok)
      call  Set_pointforce_zero()                                ! 52
c――――――――――――――――――――――――――――――――――★部材応力のゼロセット(ok)
      call  Set_stress_zero()                                    ! 53
c――――――――――――――――――――――――――――――――――★MSS モデルの初期設定
      n=Model_type.n_m_model(5)
       if(n.ne.0) then
      call Cal_MSS_dat()                                         ! 54
c――――――――――――――――――――――――――――――――――★平面問題における部材の拘束方向チェック(ok)
      call Check_R_direction()                                   ! 55
c――――――――――――――――――――――――――――――――――★部材の線形剛性計算(ok)
      call Cal_stiff_linear()                                    ! 56
c――――――――――――――――――――――――――――――――――★剛性の釣合座標系への変換(ok)
      call Rotate_stiffness()                                    ! 57
c――――――――――――――――――――――――――――――――――★部材の減衰行列計算(ok)
      if(Parameter_C.nc_member .ne. 0) then
      call Cal_damp_linear()                                     ! 58
c――――――――――――――――――――――――――――――――――★部材減衰行列の釣合座標系への変換(ok)
      call Rotate_damp()                                         ! 59
      end if
c――――――――――――――――――――――――――――――――――★節点集中質量セット(ok)
      call Set_mass()                                            ! 60
c――――――――――――――――――――――――――――――――――★節点荷重セット(ok)
      call Set_point_load()                                      ! 61
c――――――――――――――――――――――――――――――――――★接線剛性のコピー(ok)
      call Initset_nonlin_stiff()                                ! 62
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()                                          ! 63
```

```
c――――――――――――――――――――――――――――――★最大変位等のゼロセット(ok)
      call Clear_max_disp()                                                  ! 64
c――――――――――――――――――――――――――――――★最大、最小応力等のゼロセット(ok)
      call Clear_max_stress()                                                ! 65
c――――――――――――――――――――――――――――――★不釣合力のゼロセット(ok)
      call Set_zero_pointforce()                                             ! 66
c
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         静的解析結果等を取り込む
c               （初期変位、初期応力を入力）
c
c――――――――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――――――★初期変位を入力
      if(Control.init_disp.ne.0) then
c      call Get_init_disp()                                                  ! 67
       endif
c――――――――――――――――――――――――――――――――――――――★初期応力を入力
      if(Control.init_stress.ne.0) then
c      call Get_init_stress()                                                ! 68
      endif
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()                                                         ! 69
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()                                               ! 70
c――――――――――――――――――――――――――――――――――――――★ファイル名一覧出力
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
      call Set_preset_disp()                                                 ! 71
       endif
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         解析ステップをセットする
c
c――――――――――――――――――――――――――――――――――――――――――――――――
       ns_step=1                                                             ! 72
       n_step=10
       d_max_v=0.
       id_max_v=0
       stimes=secnds(0.0)                        ! 解析時間を計る
c――――――――――――――――――――――――――――――――――――――★解析手法のセット
      N_implicit_method = 1  !陰解法は１回で反復法に戻る
      Iteration_method  = 1  !最初は反復法を使用
c――――――――――――――――――――――――――――――――――――――――――――――――
```

```
c
c    ★          予備計算はここで終了（GUI に戻る）
c
c―――――――――――――――――――――――――――――――――――――――
      return                                                    ! 73
```

予備計算部分について、処理の流れを説明しよう。予備計算は、データ入力部と予備計算部に分かれている。データ入力部は、制御データと構造解析用データの入力に分かれている。また、多くの部分で、動的領域の確保に当てられている。

1. 振動解析プログラムの入り口となっており、このサブルーチンの引数は、解析の制御とグラフィック用に使用するデータである。これらの領域確保は、C++で書かれた動的解析管理ルーチンで行われている。
2. この implicit 文が数値解析全般に用いられており、数値解析で使用する実数は全て倍精度とする。
3. パラメータ icontrol の値が１であると、実際の動的解析へ処理が移行する。
4. 同じく、パラメータ icontrol の値が 99 であると、動的に領域確保した構造体や配列を解放し、動的解析の後処理を行うコードへ処理が移行する。
5. 動的解析では、多くの動的領域を用いて解析を行っている。ただし、動的領域の確保は予備計算が進んで、大きさを決定するパラメータが求められるまで行うことができない。確保する前に、何らかのエラーがあって終了処理にプログラムが移行し、その領域を解放しようすると重大なエラーとなりプログラムがハングすることがある。これを避けるために M_alloc 配列でどこまで動的領域が確保されたかをチェックする。まずは、この配列をゼロセットし、動的領域確保された段階で１を代入する。
6. 解析の進行状況を出力するファイル DOUTPUT をオープンする。このファイルは SPACE の中から見ることができる。
7. SPACE がディスクの中の一次記憶領域である TEMP フォルダーに書き出したコントロールファイル名と動的解析種別を読み込む。そのファイル名は spacesys.xxx である。
8. コントロールファイル(***.ctl)を読み込み、そこに書かれている全ファイルをシステム内にセットする。ここで、***は任意のファイル名を表す。

9. SPACE の中の動的解析ダイアログで定義した解析制御用データを入力する。ここで、その仕様を以下に示す。最初の一行目の 7 は整数データの個数、次の 11 は実数データの個数を表す。各々のデータの持つ意味は、第 7 章のファイル管理で示す。

```
      7          11
   0        0        1        2       10        0        1
0.150000E+02    0.100000E-02    0.100000E+03    0.100000E+03    0.100000E+04
0.100000E+01    0.100000E-04    0.100000E+01    0.200000E+00    0.500000E+00
0.500000E+02
```

10. 動的解析データ用パラメータをファイルより読み込み、データを設定する。
11. 解析結果の出力を制御するファイルを読み込み、データを設定する。
12. 減衰に関するファイルを読み込み、データを設定する。
13. 制御情報の取得に失敗した場合は、ここで解析が終了する。
14. ここからの４つのサブルーチンでは、先に入力した制御情報を各構造体にセットする。このサブルーチンでは解析を制御するパラメータを設定している。構造体は Control であり、その内容は第 7 章を参照されたい。
15. ここでは、解析モデルで使用している部材モデルなどを設定する。構造体は Model_type である。
16. 数値解析で使用するニューマークβ法の係数を計算し、構造体に設定する。構造体は Newmark_P である。
17. 動的荷重に関するデータを構造体に設定する。構造体は Dynamic_load である。
18. このサブルーチン以後の３つのサブルーチンは、動的領域を確保するために、構造データなどを予備入力する。この予備入力 Get_parameters()によって、節点数、要素数、部材数などの構造用パラメータを得ることができる。
19. 地震加速度の予備入力を行い、地震データの大きさを得る。
20. 擬似的静的荷重として用いる節点荷重の大きさを、予備入力によって得る。
21. 上の予備入力で得たパラメータを利用して、ALLOCATE 文によって配列と構造体配列の動的領域を確保する。
22. ここまでが、動的領域確保その１であり、ここで M_alloc(1)に１を設定し、動的領域を確保したことを保障する。

23. 動的に領域確保した構造体配列などを用いて、ここでは、構造物に関するデータを入力する。
24. 構造体 Model_type 中の n_m_model(6)は Maxwell モデルを表しており、それをこの解析モデルが使用しているかどうかチェックしている。使用しておれば、25. を行う。
25. フィルターを使用する場合は、このサブルーチンで初期設定を行う。
26. ここから、28. まで、2 度目の動的領域を確保する。ここでは、解析モデルが各部材モデルを使用した場合、該当する動的領域を確保するように設定されている。使用されているかどうかは、構造体の成分 Model_type.n_m_damp にその部材モデル数がセットされている。以後、使用されているかどうかのチェックは省略されているが、全て同様の方法で各モデルに関する構造体配列が動的確保されている。
27. ALLOCATE 文で、減衰部材モデルに関する配列が動的に領域確保されている。
28. ここまでが、動的領域確保その２であり、ここで M_alloc(2)に１を設定し、動的領域を確保したことを保障する。
29. 地震加速度データをファイルより入力し、セットする。ここでも省略されているが、もしこの解析でこの地震加速度データを使用していればこのサブルーチンが呼ばれることになる。以後の入力用サブルーチンは全て同様である。
30. 擬似的静的荷重としての節点荷重をファイルより入力する。
31. 上記の節点荷重が時刻歴の中でどの様に変化するかを、SPACE のダイアログでユーザーが設定したデータを用いて、具体的に節点荷重の大きさの時刻暦を求める。
32. 解析モデルに初期不整があれば、ファイルより基本的な節点の初期変位を読み込み、次に SPACE のダイアログで設定した初期変位の倍率を掛け、節点の座標に加える。この操作をこのサブルーチンで行う。
33. 解析モデルがファイバー部材を使用しておれば、このサブルーチンでファイバーデータの予備入力を行い、動的領域確保のためのパラメータを取得する。
34. ファイバーに関する構造体配列の動的領域を確保する。
35. この動的確保を行ったことを保障するために、M_alloc(3)を１にセットする。

36. ファイバーデータを入力する。データを入力すると共に、部材データと突合せを行うなど多くの処理を行うため、後節で具体的にその処理を見ていくことにする。
37. 修正 RO モデルを使用する場合、ここでデータ入力を行う。
38. 質量データをこのサブルーチンによって入力する。
39. レーリー減衰を設定するパラメータを入力する。
40. データの入力が終了し、ここからは予備計算に入る。まず、ここでは部材の長さを計算する。
41. 部材モデルの初期設定を行う。
42. 部材座標系から全体座標系へ変換する変換行列を全部材について求める。
43. 節点に局所座標を使用する場合は、ここで、全体座標系から局所座標系に変換する変換行列を計算する。
44. 各部材の部材座標系から、局所座標系を考慮した釣合座標系への変換行列を全部材について求める。
45. 節点拘束データから節点拘束表を作成し、未知番号データを挿入する。
46. 未知数に関連する配列の動的領域確保を行う。
47. ここまでが、動的領域確保その４であり、ここで M_alloc(4)に１を設定し、動的領域を確保したことを保障する。
48. 部材両端の拘束表を作成する。
49. スカイライン行列の変換表を作成する。
50. ファイバー用の動的領域と剛性行列などの配列を動的確保する。
51. ここまでが、動的領域確保その５であり、ここで M_alloc(5)に１を設定し、動的領域を確保したことを保障する。
52. 部材両端と中央の応力をゼロセットする。
53. 部材の応力をゼロセットする。
54. MSS モデルを使用している場合は、ここで初期設定を行う。
55. 平面問題で解析を行う場合、ファイバー断面の y 軸と z 軸のどちらに曲げが生じるかをチェックする。
56. 全部材の線形剛性行列を計算し、配列に保存する。
57. 上で求めた線形の剛性行列を部材座標系から釣合座標系に変換し、配列に保存する。
58. 部材で減衰を使用するモデルに対して、その線形の減衰行列を作成する。
59. 上で求めた減衰行列を、部材座標系から釣合座標系に変換し、保

存する。
60. 節点に集中する質量を、未知数番号順に配列にセットし直す。
61. 全体座標系で入力した節点荷重を釣合座標系に変換し、未知数番号順に配列に保存する。
62. 初期設定として、線形の剛性行列を非線形の剛性行列にコピーする。
63. 増分前の変位、速度、加速度の各ベクトルをゼロセットする。
64. 最大変位、最大速度、最大加速度ベクトルをゼロセットする。
65. 各部材の応力の最大・最小ベクトルをゼロセットする。
66. 不釣合力ベクトルをゼロセットする。
67. 現在では使用不可であるが、部材の初期変位を読み込む。
68. 同じく、現在では使用不可であるが、部材の初期応力を読み込む。
69. ユーザーが指定した出力用ファイルのオープン処理と管理ファイルの初期設定を行う。
70. ユーザーが指定したファイバー断面データが実際にファイバー断面であるかどうかチェックし、そうでない場合は、出力しないように設定する。
71. 画面描画用のデータを初期設定する。
72. 次に解析を実行するための解析パラメータを初期設定する。
73. これで予備計算は終了し、このサブルーチンから抜ける。

4.7 振動解析の後処理

振動解析時では、多くの動的領域を確保しており、処理終了時にこの動的領域を解放しなければならない。動的領域を解放する場合、もっとも大切なことは、領域確保していない配列などを解放しないことである。これを行うと必ずシステムがハングアップすることになる。これを避けるために、SPACE では配列 M_alloc を用いて、動的領域確保を行ったか否かをチェックしている。

ここでは、計算終了時における後処理のプログラムコードを載せる。

```
c-------------------------------------------------------------
c
c    ★          応答解析終了処理
c
c-------------------------------------------------------------
 9998 continue
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
      call Set_preset_disp(1,n_point,past_disp_point,F_disp,Point,
     *                          rot_local,Parameter_C)
c     write(damp_out,*) ' Set_preset_disp ok'
      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
 9997 continue
c――――――――――――――――――――――――――★最大変位、速度、加速度の出力
      call Out_max_disp(Max_disp,Parameter_C.n_point,
     *                 Point,ifl(12),iflz(12))
c     write(damp_out,*) ' Out_max_disp ok'
c――――――――――――――――――――――――――★最大応力等の出力
      call Out_max_stress(Max_stress,Parameter_C.n_member,
     *                   ifl(16),iflz(16))
c     write(damp_out,*) ' Out_max_stress ok'
c――――――――――――――――――――――――――★ファイルのクローズ
      do i=1,16
      if(ifl(i).eq.1) close(iflz(i))
      enddo
      write(damp_out,*) ' ファイルのクローズ ok'
c――――――――――――――――――――――――――★ファイルのタイムスタンプ
      ihan = 0
      NFILE=100
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      call fltime(ifl,ifly,N_analysis,iflout)
c      write(damp_out,*) ' ファイルのタイムスタンプ ok' ,FNX_file
      ihan = 1
      if(iflout.eq.1) call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
c     write(damp_out,*) ' データセット ok'
c―――――――――――――――――――――――――――――――――――――――
c
c    ★        動的配列の解放
c
c―――――――――――――――――――――――――――――――――――――――
c――――――――――★        配列の動的領域を解放する（その１）
      if(M_alloc(1).eq.1)then
      DEALLOCATE (Point,Element,Member,Max_disp,Max_stress)       !ok
      DEALLOCATE (fll_static_point,am_point,fll_force_point)      !ok
      DEALLOCATE (am_member,rot_memb_t,rot_memb,ak_linear,ak_nonlinear)      !ok
      N=Parameter_C.n_local_coord
      if(N.ne.0) DEALLOCATE (rot_local)       !ok
      endif
```

```
c－－－－－－－－－－★　　配列の動的領域を解放する（その２）
      if(M_alloc(2).eq.1)then
      N=Model_type.n_m_damp
      if(N.ne.0) DEALLOCATE (ac_member)       !ok
      endif
c－－－－－－－－－－★　　配列の動的領域を解放する（その４）
      if(M_alloc(4).eq.1)then
      DEALLOCATE (disp_point,vel_point,acc_point,
     *     est_disp_point,est_vel_point,est_ddisp_point)
      DEALLOCATE (result_disp_point,result_vel_point,result_acc_point,
     *   past_disp_point,past_vel_point,past_acc_point,
     *   past_dacc_point)
      DEALLOCATE (ld_point,ld_point_repeat,fld_static)
      DEALLOCATE (a_vector,b_vector)
      DEALLOCATE (max_h_sky)
      endif
c－－－－－－－－－－★　　配列の動的領域を解放する（その５）
      if( M_alloc(5).eq.1)then
      DEALLOCATE (gskym,gskym_d,nwork,twork)        !ok
      endif
c－－－－－－－－－－★　　配列の動的領域を解放する（その２）
      if(  M_alloc(2).eq.1)then
      n=Model_type.n_m_ro_model
      if(n.ne.0) then
      DEALLOCATE (RO_work )
      endif
      n=Model_type.n_m_model(5)
      if(n.ne.0) then
      DEALLOCATE (MSS_work )            !ok
      endif
      n=Model_type.n_m_model(6)
      if(n.ne.0) then
      DEALLOCATE (E_model6_real )           !ok
      endif
      n= Model_type.n_e_model(11)        !要素モデルの数
      if(n.ne.0) then
      DEALLOCATE (E_model11 )               !ok
      endif
      n= Model_type.n_m_model(11)        !部材モデルの数
      if(n.ne.0) then
      DEALLOCATE ( M_model11 )              !ok
      endif
      n= Model_type.n_e_model(12)        !要素モデルの数
      if(n.ne.0) then
      DEALLOCATE (E_model12 )              !ok
      endif
      n= Model_type.n_m_model(12)        !部材モデルの数
     if(n.ne.0) then
      DEALLOCATE ( M_model12 )              !ok
      endif
      n= Model_type.n_m_model(18)        !部材モデルの数
     if(n.ne.0) then
      DEALLOCATE ( M_model13 )              !ok
      endif
```

```
 n= Model_type.n_e_model(18)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model13 )              !ok
 endif
 n= Model_type.n_e_model(15)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model15 )              !ok
 endif
 n= Model_type.n_m_model(15)        !部材モデルの数
if(n.ne.0) then
 DEALLOCATE ( M_model15 )              !ok
 endif
 n= Model_type.n_e_model(13)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model21 )              !ok
 endif
 n= Model_type.n_m_model(13)        !部材モデルの数
if(n.ne.0) then
  DEALLOCATE ( M_model21 )              !ok
 endif
 n= Model_type.n_e_model(14)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model22 )              !ok
 endif
 n= Model_type.n_m_model(14)        !部材モデルの数
if(n.ne.0) then
 DEALLOCATE ( M_model22 )              !ok
 endif
 n= Model_type.n_e_model(16)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model31)               !ok
 endif
 n= Model_type.n_m_model(16)        !部材モデルの数
if(n.ne.0) then
 DEALLOCATE ( M_model31 )              !ok
 endif
 n= Model_type.n_e_model(17)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model32)               !ok
 endif
 n= Model_type.n_m_model(17)        !部材モデルの数
if(n.ne.0) then
 DEALLOCATE ( M_model32 )              !ok
 endif
 n= Model_type.n_e_model(19)        !要素モデルの数
if(n.ne.0) then
 DEALLOCATE (E_model33)               !ok
 endif
 n= Model_type.n_m_model(19)        !部材モデルの数
if(n.ne.0) then
 DEALLOCATE ( M_model33 )              !ok
 endif
 if(Dynamic_load.n_load_dynamic .ne. 0) then
 DEALLOCATE (acc_earth)               !ok
```

```
      endif
       if(Dynamic_load.n_load_point .ne. 0) then
       DEALLOCATE (fdd_point)              !ok
       endif
       endif
c――――――――――★        配列の動的領域を解放する（その３）
       if( M_alloc(3).eq.1)then
       n= Model_type.nm_div_fmodel         !要素モデル内のサブ要素の数
      if(n.ne.0) then
       DEALLOCATE (M_model_fiber )              !ok
       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
       DEALLOCATE (E_model_fiber )              !ok
       endif
       endif
c――――――――――★        配列の動的領域を解放する（その５）
       if( M_alloc(5).eq.1)then
       n= Model_type.n_m_bilinear          !  バイリニアの履歴要の数
      if(n.ne.0) then
       DEALLOCATE (Bilinear_work )              !ok
       endif
       n= Model_type.n_m_trilinear         !  トリリニアの履歴要の数
      if(n.ne.0) then
       DEALLOCATE (Trilinear_work )              !ok
       endif
       n= Model_type.n_m_Concrete          !  コンクリートの履歴要の数
      if(n.ne.0) then
       DEALLOCATE (Concrete_work )              !ok
       endif
       endif
c――――――――――――――――――――――――――――――★計算終了コードセット
      iend_code =1
       close (damp_out)
       return
c――――――――――――――――――――――――――――――――――――――――
c
c    ★        解析終了
c
c――――――――――――――――――――――――――――――――――――――――
     end
```

ここでは、後処理の流れについて説明する。後処理は、変位や速度、応力などの最大値を該当するファイルに出力すること、また、動的に確保した領域を解放することである。プログラムの骨組みを取り出してみよう。

```
c――――――――――――――――――――――――――――――――――――――――
c
c    ★        応答解析終了処理
c
c――――――――――――――――――――――――――――――――――――――――
```

```
 9998 continue                                                          ! 1
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
      call Set_preset_disp()                                            ! 2
      endif
c-------------------------★応力
      if(i_read_spring .ne. 0) then
      call Set_preset_spring()                                          ! 3
      endif
 9997 continue
c-------------------------------★最大変位、速度、加速度の出力
      call Out_max_disp()                                               ! 4
c-------------------------------★最大応力等の出力
      call Out_max_stress()                                             ! 5
c-------------------------------★ファイルのクローズ
      do i=1,16
      if(ifl(i).eq.1) close(iflz(i))                                    ! 6
      enddo
c-------------------------------★ファイルのタイムスタンプ
      ihan = 0
      NFILE=100
      call ctlset()                                                     ! 7
      call fltime()                                                     ! 8
      if(iflout.eq.1) call ctlset()                                     ! 9
c-------------------------------------------------------------
c
c    ★        動的配列の解放
c
c-------------------------------------------------------------
c----------★        配列の動的領域を解放する（その１）
      if(M_alloc(1).eq.1)then                                           ! 10
      DEALLOCATE (Point,Element,Member,Max_disp,Max_stress)       !ok
      DEALLOCATE (fll_static_point,am_point,fll_force_point)       !ok
      DEALLOCATE (am_member,rot_memb_t,rot_memb,ak_linear,ak_nonlinear)       !ok
       N=Parameter_C.n_local_coord
      if(N.ne.0) DEALLOCATE (rot_local)       !ok
      endif
c----------★        配列の動的領域を解放する（その２）
      if(M_alloc(2).eq.1)then                                           ! 11
      N=Model_type.n_m_damp
      if(N.ne.0) DEALLOCATE (ac_member)       !ok
      endif
       ・
       ・
       ・
c-------------------------★計算終了コードセット
      iend_code =1
      close (damp_out)                                                  ! 12
      return
```

```
c―――――――――――――――――――――――――――――――――――――――
c
c    ★          解析終了
c
c―――――――――――――――――――――――――――――――――――――――
      end
```

後処理の流れについて、プログラムコードの右側の番号にしたがって説明する。

１．ここが後処理の入り口となっており、icontrol が 99 の場合ここに処理が移動する。
２．画面に構造図などを描画指定している場合、ここで、変位の最終設定を行う。
３．同じく、応力の最終設定を行う。
４．最大変位、最大速度、最大加速度をファイルに出力する。
５．部材の最大応力をファイルに出力する。
６．出力ファイルの管理データに従って、ファイルをクローズする。
７．コントロールファイルを読み込む。
８．データを出力したファイルに対して、日時をコントロールデータにセットする。
９．そのコントロールデータをファイルに再度出力する。
10．ここから、解析の中で動的に確保した領域を解放する。まず、M_alloc(1)で管理された構造体や配列を解放する。
11. 同じく、M_alloc(2)で管理された構造体や配列を解放する。以後、同様の処理を行って、構造体や配列を解放する。
12. 計算終了コードを１に設定し、DOUTPUT ファイルをクローズする。これで解析の後処理が終了し、サブルーチンから戻ることになる。

4.8 固有値問題の処理

本節では、固有値問題を数値解析し、固有振動数や振動モード、また刺激係数を求めるプログラムの流れを学ぶ。計算の流れは非常に単純であり、第 4.2 節で説明されているため、おおよそ理解できていると思う。ここでは、実際の FORTRAN コードを用いて説明する。前述したように、ここでは固有値問題を解く計算手法として 2 種類用意されており、また、計算した結果が振動解析のレーリー減衰で利用されていることを頭に入れておいて頂きたい。ここでは、データ入力と予備計算部、及び、後

処理部を除いたプログラムコードを以下に示す。省いた部分のコードは付録を参照されたい。

```
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★         固有値解析手法の選択
c
c―――――――――――――――――――――――――――――――――――――――――
      if(Eigen_d.n_method .eq. 1) goto 9700
c―――――――――――――――――――――――――――――――――――――――――
c
c    ★         サブスペース法の計算用配列を動的確保する
c
c―――――――――――――――――――――――――――――――――――――――――
      n_member=Parameter_C.n_member
      n_point =Parameter_C.n_point
      n_unknown=Parameter_C.n_unknown
      n_skyline=Parameter_C.n_skyline
      n_local_coord=Parameter_C.n_local_coord
      nroot = Eigen_d.n_modes
      nc = min(2*nroot,nroot+8)
      nnc = nc*(nc+1)/2
      write(76,*)   'n_unknown:',n_unknown
      write(76,*)   'n_skyline:',n_skyline
      write(76,*)   'nroot:',nroot
      write(76,*)   'nc:',nc
      write(76,*)   'nnc:',nnc
      write(76,*)   'method_type',Eigen_d.method_type
      write(76,*)   'Eigen_d.epsj',Eigen_d.epsj
      write(76,*)   'Eigen_d.n_iteration',Eigen_d.n_iteration
      if(nroot.eq.0) then
      ierr_dat=1000
      return
      endif
c―――――――――――――――――――――――――――――――――――――――――
      ALLOCATE (tw_a(n_unknown),tw_b(n_unknown),tw_c(n_unknown))
      ALLOCATE (Eigen_Value(nc),Eigen_Vector(n_unknown,nc))
      ALLOCATE (ar_a(nnc),ar_b(nnc),Omega(nnc))
      ALLOCATE (vec_w1(nc,nc),vec_w2(nc),vec_w3(nc),vec_w4(nc),
     *          vec_w5(nc),vec_w6(nc))
      ALLOCATE (ivec_w1(nc))
c―――――――――――――――――――――――――――――――――――――――――
      do n_step = 1,2
c――――――――――――――――――――――――――★スカイライン行列のゼロセット(ok)
      call Set_sky_zero(gskym,n_skyline)
      call Set_sky_zero(gskymm,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
      call Build_sky_mm_E(n_step,gskymm,
     *           Point,n_point,am_point,rot_local,
     *           n_local_coord,max_h_sky,Eigen_d.load_mass)
c       write(damp_out,*) ' Build_sky_mm ok'
c     write(76,'(12e12.4)')(gskymm(j),j=1,n_unknown)
```

```
c―――――――――――――――――――――★部材の整合質量系の行列への足し込み(ok)
c――――――――――――――――――★部材の整合質量行列計算
c        write(damp_out,*) ' Cal_mass_linear ok',Eigen_d.load_mass
      if(Eigen_d.load_mass .ne. 0) then
      call Cal_mass_linear(n_step,Element,Member,Parameter_C,am_member,
     *            work1_elememt,work2_elememt,work1_member,work2_member,
     *            Eigen_d.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_step,Element,Member,n_member,am_member,
     *                 rot_memb,Eigen_d.load_mass)
c        write(damp_out,*) ' Rotate_mass ok',n_member
c――――――――――――――――――★部材質量系の足し込み(ok)
      call Build_sky_m_E(n_step,gskymm,n_skyline,Member,n_member,
     *                  am_member,max_h_sky,Element)
      endif
c       write(damp_out,*) ' Build_sky_m ok',n_member
c―――――――――――――――――――――★線形剛性の足し込み(ok)
      call Build_sky_k_E(gskym,n_skyline,Member,n_member,
     *                        Ak_linear,max_h_sky)
c      return
c       write(damp_out,*) ' Build_sky_k ok',n_member
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) then
      nm_skyline=n_skyline
      else
      nm_skyline=n_unknown
      endif
c――――――――――――――――――――――――――――――――――――
c
c    ★         固有問題の解析（サブスペース法）
c
c――――――――――――――――――――――――――――――――――――
c      write(damp_out,*) ' SSPACE in ok',n_member

      call SSPACE(gskym,gskymm,max_h_sky,Eigen_Value,Eigen_Vector,
     *         tw_a,tw_b,ar_a,ar_b,vec_w1,vec_w2,vec_w3,vec_w4,vec_w5,
     *         vec_w6,ivec_w1,n_unknown,nnm,n_skyline,nm_skyline,nroot,
     *         Eigen_d.epsj,nc,nnc,Eigen_d.n_iteration,
     *         IFSS,IFPR,Eigen_d.method_type,IEXIT,
     *         Eigen_d.load_mass,NSTIF,gskym_d,nwork,twork,tw_c)
      close(UNIT=NSTIF,STATUS='DELETE')
      if(iexit.ne.0) then
      ierr_dat=998
```

```
      return
      endif
c-----------------------★結果の出力
      call Out_Eigen(n_step,n_unknown,Parameter_C,Eigen_d,
     *           Point,Newmark_P,Eigen_Value,Eigen_Vector,Omega,
     *           gskymm,max_h_sky,rot_local,disp_point_m,
     *           tw_a,tw_b,ifl(7),iflz(7),ifl(8),iflz(8))
      enddo
c-----------------------★計算終了コードセット
      iend_code =1
c-----------------------★ファイルのクローズ
      do i=1,16
      if(ifl(i).eq.1) close(iflz(i))
      enddo
      write(damp_out,*) ' ファイルのクローズ ok'
c-----------------------★ファイルのタイムスタンプ
      ihan = 0
      NFILE=100
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      call fltime(ifl,ifly,N_analysis,iflout)
      write(damp_out,*) ' ファイルのタイムスタンプ ok' ,FNX_file
      ihan = 1
      if(iflout.eq.1) call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
c      write(damp_out,*) ' データセット ok'
c-------------------------------------------------------------
c
c    ★        サブスペース用動的配列の解放
c
c-------------------------------------------------------------
      .
      .
      close (damp_out)
      return
c-------------------------------------------------------------
c
c    ★        せん断型の固有値解析
c
c-------------------------------------------------------------
 9700 continue
c-------------------------------------------------------------
c
c    ★        ヤコビ法の計算用配列を動的確保する
c
c-------------------------------------------------------------
      n_member=Parameter_C.n_member
      n_point =Parameter_C.n_point
      n_unknown=Parameter_C.n_unknown
      n_local_coord=Parameter_C.n_local_coord
      nc=n_unknown
      write(76,*)   'n_unknown:',n_unknown
      write(76,*)   'Eigen_d.epsj',Eigen_d.epsj
      write(76,*)   'Eigen_d.n_iteration',Eigen_d.n_iteration
c-----------------------------------------------------------
      ALLOCATE (gak(n_unknown,n_unknown))
```

```
      ALLOCATE (Eigen_Value(n_unknown),
     *          Eigen_Vector(n_unknown,n_unknown))
      ALLOCATE (Omega(nc))
      ALLOCATE (vec_w5(nc),vec_w2(nc),vec_w3(nc),vec_w4(nc))
c-------------------------------------------------------
      do n_step = 1,2
c-----------------------★剛性行列のゼロセット(ok)
      n=n_unknown*n_unknown
      call Set_sky_zero(gak,n)
      call Set_sky_zero(gskymm,n_unknown)
c      write(damp_out,*) ' Set_gak_zero ok'
c-----------------------★線形剛性の足し込み(ok)
      call Build_gak_E(gak,n_unknown,Member,n_member,
     *                       Ak_linear)
c       write(damp_out,*) ' Build_gak ok',n_member
c-----------------------★集中質量系の行列への足し込み
      call Build_mm_E(n_step,gskymm,
     *           Point,n_point,am_point)
c       write(damp_out,*) ' Build_mm ok'
c-----------------------★固有値問題の変換
      call Trans_Eigen(gak,n_unknown,gskymm,vec_w2,ier)
      if(ier.ne.0) then
      write(damp_out,*) ' err: 節点にゼロ質量があった。'
      return
      endif
c       write(damp_out,*) ' Trans_Eigen ok'
c-------------------------------------------------------
c
c    ★        ヤコビ法の計算用データをセットする
c
c-------------------------------------------------------
c-------------------------------------------------------
c
c    ★        固有問題の解析（ヤコビ法）
c
c-------------------------------------------------------
c      write(damp_out,*) ' Jacobi in ok',n_member
      call Jacobi_E(gak,n_unknown,Eigen_Value,Eigen_Vector,
     *         vec_w5,vec_w2,vec_w3,vec_w4)
c       write(damp_out,*) ' Jacobi out ok',n_member
c-----------------------★固有ベクトルの変換
      call Trans_Eigen_V(Eigen_Vector,n_unknown,gskymm,vec_w2,ier)
c-----------------------★結果の出力
      call Out_Eigen_J(n_step,n_unknown,Parameter_C,Eigen_d,
     *           Point,Newmark_P,Eigen_Value,Eigen_Vector,Omega,
     *           gskymm,rot_local,disp_point_m,
     *           vec_w3,vec_w4,ifl(7),iflz(7),ifl(8),iflz(8))
c       write(damp_out,*) ' Out_Eigen_J ok',n_member
      enddo
c-----------------------★計算終了コードセット
      iend_code =1
c-----------------------★ファイルのクローズ
      do i=1,16
      if(ifl(i).eq.1) close(iflz(i))
```

```
      enddo
      write(damp_out,*) ' ファイルのクローズ ok'
c-----------------------------★ファイルのタイムスタンプ
      ihan = 0
      NFILE=100
      call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
      call fltime(ifl,ifly,N_analysis,iflout)
      write(damp_out,*) ' ファイルのタイムスタンプ ok' ,FNX_file
      ihan = 1
      if(iflout.eq.1) call ctlset(ihan,FNX_file,TITLEX,IDFILE,NFILE)
c      write(damp_out,*) ' データセット ok'
c-------------------------------------------------------------
c
c    ★      ヤコビ法用動的配列の解放
c
c-------------------------------------------------------------
      .
      .
      close (damp_out)
      return
      end
```

続いて、固有値問題に関する計算の流れを、プログラムの骨組みを取り出して詳しく見ていこう。

```
c-------------------------------------------------------------
c
c    ★         固有値解析手法の選択
c
c-------------------------------------------------------------
      if(Eigen_d.n_method .eq. 1) goto 9700                    ! 1
c-------------------------------------------------------------
c
c    ★         サブスペース法の計算用配列を動的確保する
c
c-------------------------------------------------------------
c              サブスペース法で使用するワークエリアを動的確保する
c-------------------------------------------------------------
      ALLOCATE (tw_a(n_unknown),tw_b(n_unknown),tw_c(n_unknown))  ! 2
      ALLOCATE (Eigen_Value(nc),Eigen_Vector(n_unknown,nc))
      ALLOCATE (ar_a(nnc),ar_b(nnc),Omega(nnc))
      ALLOCATE (vec_w1(nc,nc),vec_w2(nc),vec_w3(nc),vec_w4(nc),
     *          vec_w5(nc),vec_w6(nc))
      ALLOCATE (ivec_w1(nc))
c-------------------------------------------------------------
      do n_step = 1,2                                          ! 3
c-----------------------------★スカイライン行列のゼロセット(ok)
      call Set_sky_zero(gskym)                                 ! 4
      call Set_sky_zero(gskymm)
c-----------------------------★集中質量系の行列への足し込み
      call Build_sky_mm_E()                                    ! 5
c-----------------------------★部材の整合質量系の行列への足し込み(ok)
c-----------------------------★部材の整合質量行列計算
```

```
      if(Eigen_d.load_mass .ne. 0) then
      call Cal_mass_linear()                                          ! 6
c-----------------------★整合質量の釣合座標系への変換(ok)
      call Rotate_mass()                                              ! 7
c-----------------------★部材質量系の足し込み(ok)
      call Build_sky_m_E()                                            ! 8
      endif
c---------------------------★線形剛性の足し込み(ok)
      call Build_sky_k_E()                                            ! 9
c-------------------------------------------------------------------
c
c    ★        固有問題の解析（サブスペース法）
c
c-------------------------------------------------------------------
      call SSPACE()                                                   ! 10
c-------------------------★結果の出力
      call Out_Eigen()                                                ! 11
      enddo
c-------------------------------------------------------------------
c
c    ★        サブスペース用動的配列の解放
c
c-------------------------------------------------------------------
      DEALLOCATE (Point,Element,Member)                               ! 12
      DEALLOCATE (gskym,gskym_d,nwork,twork,gskymm)        !ok
      DEALLOCATE (max_h_sky)                               !ok
      DEALLOCATE (tw_a,tw_b,tw_c)                          !ok
       .
       .
      return
c-------------------------------------------------------------------
c
c    ★        せん断型の固有値解析
c
c-------------------------------------------------------------------
 9700 continue
c-------------------------------------------------------------------
c
c    ★        ヤコビ法の計算用配列を動的確保する
c
c-------------------------------------------------------------------
      ALLOCATE (gak(n_unknown,n_unknown))                             ! 13
      ALLOCATE (Eigen_Value(n_unknown),
     *          Eigen_Vector(n_unknown,n_unknown))
      ALLOCATE (Omega(nc))
      ALLOCATE (vec_w5(nc),vec_w2(nc),vec_w3(nc),vec_w4(nc))
c-----------------------------------------------------------------
      do n_step = 1,2                                                 ! 14
c-----------------------★剛性行列のゼロセット(ok)
      call Set_sky_zero(gak)                                          ! 15
      call Set_sky_zero(gskymm)
c---------------------------★線形剛性の足し込み(ok)
      call Build_gak_E()                                              ! 16
c-----------------------★集中質量系の行列への足し込み
```

```
      call Build_mm_E()                                          ! 17
c―――――――――――――――――――――――――★固有値問題の変換
      call Trans_Eigen()                                         ! 18
c――――――――――――――――――――――――――――――――――――――――
c
c    ★          固有問題の解析（ヤコビ法）
c
c――――――――――――――――――――――――――――――――――――――――
      call Jacobi_E()                                            ! 19
c―――――――――――――――――――――――――★固有ベクトルの変換
      call Trans_Eigen_V()                                       ! 20
c―――――――――――――――――――――――――★結果の出力
      call Out_Eigen_J()                                         ! 21
      enddo
c――――――――――――――――――――――――――――――――――――――――
c
c    ★          ヤコビ法用動的配列の解放
c
c――――――――――――――――――――――――――――――――――――――――

      DEALLOCATE (Point,Element,Member)                  !ok     ! 22
      DEALLOCATE (gskym,gskym_d,nwork,twork,gskymm)      !ok
      DEALLOCATE (gak)          !ok
        •
        •
     return
      end
```

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の内容を概説する。

1. データ入力した後、計算手法によって分岐する。ここでは、ヤコビ法を選択した場合、文番号 9700 に飛ぶ。
2. これ以後は、サブスペース法で数値解析を行う。ここでは、サブスペース法で必要になるワーク領域を動的確保している。
3. 解析が 2 段階で行われるため、固有値解析も 2 度行われる。
4. スカイライン行列である質量行列 gskymm と剛性行列 gskym をゼロクリアする。
5. 以下の処理で質量行列を作成する。まず、節点集中質量を行列に組み込む。
6. 部材分布質量がある場合は、次の 7 と 8 も処理する。ここでは、部材座標系で質量行列を計算する。
7. 質量行列を部材座標系から釣合座標系に変換する。
8. スカイライン行列である質量行列に、部材の整合質量を組み込む。
9. 部材の線形剛性をスカイライン行列に組み込む。

10. 固有値問題をサブスペース法で解析する。
11. 振動モード、振動数、固有周期などを求め、仕様に合わせてファイルに出力する。
12. 解析の中で動的確保した領域を解放する。
13. ここからヤコビ法を用いた解析を行う。まず、ワーク領域を動的確保する。
14. 解析が 2 段階で行われるため、固有値解析も 2 度行われることになる。
15. 正方行列である質量行列 gskymm と剛性行列 gskym をゼロクリアする。
16. 部材の線形剛性を正方行列に組み込む。
17. 質量行列を作成する。ここでは節点集中型のみ扱う。節点集中質量を正方行列に組み込む。
18. 一般固有値問題から標準固有値問題に変換する。
19. ヤコビ法を用いて固有値問題を解く。
20. 固有ベクトルを変換する。
21. 振動モード、振動数、固有周期などを求め、仕様に合わせてファイルに出力する。
22. 解析の中で動的確保した領域を解放する。

以上で、固有値問題を解析する流れを説明した。固有値問題を数値解析するサブスペース法 SSPACE()とヤコビ法 Jacobi_E()については、参考文献を参照されたい[3)]。また、他のサブルーチンは、他の節で説明されている。ここでは、省略するが、具体的なコードは付録を見られたい。

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
```

4.14 部材の弾塑性チェック

本節では、前節に続いて、部材モデルの階層最上位の仕組みを持つサブルーチン Check_stress()について説明する。このサブルーチンは、部材断面の弾塑性解析を行う。まず、このサブルーチンをコールするコードを以下に示す。多くの部材モデルを含むためサブルーチンの引数が多いことが特徴である。Maxwell モデルの塑性チェックは、次のサブルーチン Check_Maxwell_stress()で行う。

```
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――――――――――――――――――――――――――――――――★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)
```

上記２つのプログラムコードを以下に示す。

```
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 )
      implicit real*8(A-H,O-Z)
      include "submain.h"
      include "submainx.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.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)
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
     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                                                          ! 2
      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)         ! 3
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)       ! 4
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                                       ! 5
      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
      do j=1,6                                                           ! 6
      f(j)=-Member(i).stress(j)
      enddo
      do j=7,12
      f(j)=Member(i).stress(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
   12 continue
c―――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      if(N_analysis.ge.9) then                                           ! 7
      call Cal_check_stiff_M2(Member(i),Element(ie),RO_work,
     *     vv,vpp )
      endif
      do j=1,3                                                           ! 8
      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)         ! 9
      do j=1,12                                                          ! 10
      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
      iee = Member(i).n_model_type                                       ! 11
      call Cal_check_stiff_M3(Member(i),Element(ie),
     *     vv,vpp,N_analysis)
      do j=1,12
      f(j)=0.
      enddo
      do j=1,3
      f(j)=-Member(i).stress(j)
```

```
      f(j+6)=Member(i).stress(j+3)
      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
   14 continue
c-----------------------------------★Model_No.4 3 次元ケーブル弾塑性モデル
      call Cal_check_stiff_M4(Member(i),Element(ie),
     *     vv,vpp ,N_analysis)
      f(1)=-Member(i).stress(1)
      f(7)=-f(1)
      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
   15 continue
c-----------------------------------★Model_No.5 3 次元免振モデル
      if(N_analysis.ge.9) then
      iee=Member(i).n_model_type
      call Cal_check_stiff_M5(Member(i),Element(ie),MSS_work(iee),
     *     vv,vpp,Model_type.n_spring,Model_type.cosin(1,1) )
      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
   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
      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_check_stiff_M11(Control,N_analysis,                    ! 12
     *      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)
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)      ! 13
      do j=1,12
       Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
  112 continue
c―――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
      call Cal_check_stiff_M12(Control,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)
      do j=1,12
       Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
  113 continue
c―――――――――――――――――――――――――――――――――★Model_No.13 両端 MS モデル
      call Cal_check_stiff_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)
      do j=1,12
       Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
  114 continue
c―――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
      call Cal_check_stiff_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)
      do j=1,12
       Member(i).force(j)=Member(i).force(j)+ff(j)
      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_check_stiff_M15(Control,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
      do j=1,6
      f(j)=-Member(i).stress(j)
      enddo
      do j=7,12
      f(j)=Member(i).stress(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
  116 continue
c――――――――――――――――――――――――――――――――★Model_No.16 両端アナロジーモデル
      call Cal_check_stiff_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)
      do j=1,12
      Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
  117 continue
c――――――――――――――――――――――――――――――――★Model_No.17 両端、中央アナロジーモデル
      call Cal_check_stiff_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)
      do j=1,12
      Member(i).force(j)=Member(i).force(j)+ff(j)
      Enddo
      goto 100
  118 continue
c――――――――――――――――――――――――――――――――★Model_No.18
      call Cal_check_stiff_M13(Control,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)
      do j=1,12
      Member(i).force(j)=Member(i).force(j)+ff(j)
      enddo
      goto 100
  119 continue
c――――――――――――――――――――――――――――――――★Model_No.19 両端、中央アナロジーモデル
```

```
c      call Cal_check_stiff_M33(N_analysis,
c     *      mem,Model_type,Member(i),Element(ie),
c     *      E_model33, E_model_fiber,
c     *      M_model33, M_model_fiber,
c     *      vv,vpp,f,me_x)
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
  120 continue
c――――――――――――――――――――――――――――――――――――――★Model_No.20
      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)
c――――――――――――――――――――――――――――――――――――――★部材の接線剛性を釣合系に変換
  100 continue
      end do
      return
      end
C     ――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Check_Maxwell_stress
C       ――――――――――――――――――――――――――――――――――――――――
C     ●   Maxwell model の応力計算
C     ――――――――――――――――――――――――――――――――――――――――
      subroutine Check_Maxwell_stress(Member,n_member,
     *                          Element,E_model6_real,Newmark_P)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s    / Member
      record / element_s   / Element
      record / E_model6_real_s   / E_model6_real
      record / newmark_s     / Newmark_P
      dimension Member(*),Element(*),E_model6_real(*)
c――――――――――――――――――――――――――――――――――――――――――――――――
      do i=1,n_member
c――――――――――――――――――――――――――――――――――――――★要素及びモデルのセット
      iet  = Member(i).element_type
      ie   = Member(i).nm_element
      ien  = Member(i).n_model_type
      if(iet.eq.6)then                                                    ! 14
c――――――――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル(ok)
      call  Check_maxwelldamp(E_model6_real(ien),Element(ie),Newmark_P)
      endif
      end do
      return
      end
```

上記のサブルーチンの構造は、手続き上多少長くなっているが、前節で説明したプログラムの構造とほとんど同じとなっている。従って、全体の流れを理解することは容易である。前節のサブルーチンと異なっている部分を中心に、このサブルーチンの説明を行う。

1. 最初に、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後の変位をセットする。
2. 部材両端の材端力 f をゼロクリアする。
3. 増分後の変位 v を釣合座標系から部材座標系 vv に変換する。
4. 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。
5. 要素番号を構造体の成分 Element(ie).n_element から取得する。他のパラメータは、第 4.12 節を参照されたい。
6. 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処理が行われる。ただし、弾性モデルであるので、チェックを行う処理はなく、両端の応力 Member(i).stress(j)から材端節点力 f にデータをコピーする。さらに、この材端節点力をサブルーチン RotateL_y()を用いて、釣合座標系に変換する。最後に、この材端力を構造体の成分 Member(i).force(j)にセットする。この構造体を用いて、式(3.54)の中の $Q(y_n)$が求められる。
7. ここでは、せん断型弾塑性モデルの履歴をチェックする。まず、解析種別番号が 9 以上の場合に、サブルーチン Cal_check_M2()がコールされ、処理が実行される。解析種別が弾性解析もしくは幾何学的非線形解析の場合はこの処理は行われない。
8. 次に、材端応力を材端力にコピーする。
9. 材端力を部材座標系から釣合座標系に変換する。
10. 変換した材端力を構造体成分 Member(i).force(j)にセットする。
11. ここでは、座屈を考慮した３次元軸力モデルの履歴をチェックする。後の処理は前モデルと同一である。以後は、同様な処理が続く。
12. ここでは、両端ファイバーモデルの履歴をチェックする。この内容については、次章の部材モデルで詳細に解説する。増分の材端力をこのサブルーチンから得る。
13. 増分の材端力 f を部材座標系から釣合座標系に変換し、構造体に足しこむ。以下、他の部材モデルに対して同様な処理が続く。
14. ここでは、部材モデル番号が 6 である Maxwell モデルについてのみ、履歴チェックを行う。

解析種別番号 9、10 は、
９：弾塑性解析
10：幾何学的非線形＋弾塑性解析
である。

部材モデルの第２層に関する例として、せん断弾塑性モデルについて説明する。このサブルーチン Cal_check_stiff_M2()は、せん断型弾塑性モデルの履歴特性を管理するシステムであり、最上位層と同様の構造を有している。また、部材モデルとして、さらに深い階層構造を有する静的縮合モデルについては、次章で詳細に述べるつもりである。

サブルーチン Cal_check_stiff_M2()のプログラムコードを、まず見てみよう。

```
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"
      record / member_s2    / Member
      record / element_s2   / Element
      record / RO_work_s    /  RO_work
      dimension vv(12),vpp(12),RO_work(*)
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
c       structure / element_s2/
c       integer  element_type   !  要素タイプ(6)
c       integer  n_element      !  非線形要素番号
c       real*8   AK_1           !  第一剛性
c       real*8   AK_2           !  第二剛性
c       real*8   AK_3           !  第三剛性
c       real*8   Q_1            !  第一折れ点のせん断力
c       real*8   Q_2            !  第二折れ点のせん断力
c       real*8   AKu            !  軸方向バネ
c       real*8   dm1            !  ダミー
c       real*8   dm2            !  ダミー
c       real*8   dm3            !  ダミー
c       real*8   dm4            !  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       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       end structure
c      record /element_s2/ Element
c     ALLOCATABLE ::Element(:)
c     ALLOCATE (Element(n_element))
```

```
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
c      structure / member_s2/
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_so              !  部材の層番号
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  istat_v            !  y 方向:履歴特性の状態(*)
c      integer  istat_w            !  z 方向:履歴特性の状態(*)
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*8   alength            !  長さ
c      real*8   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（釣合座標系）
c      real*8   stress(6)          !  部材中央の応力（部材座標系）
c      real*8   AKv_tan            !  接線剛性(*)
c      real*8   AKw_tan            !  接線剛性(*)
c      real*8   dmv(5)             !  y 方向耐力:履歴特性で使用(*)
c      real*8   dmw(5)             !  z 方向耐力:履歴特性で使用(*)
c      end structure
c
c     ALLOCATABLE :: Member (:)
c     ALLOCATE (Member (n_member))
c            Element              structure
c            Member               structure
c            ak_linear            real*8   線形剛性行列
c―――――――――――――――――――――――――――――――――――――――――――――
      No_rireki=Element.nm_type                                          ! 1
      if(No_rireki/10.eq.0) then                                         ! 2
      goto(5,10,20,30,40,50,60),No_rireki+1                              ! 3
    5 continue
c―――――――――――――――――――――――――――――――――★規定モデル：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)                        ! 4
      goto 999
   10 continue
c―――――――――――――――――――――――――――――――――★トリリニア：Nomal
      call Mesing_TriLiner(Member,Element,vv,vpp)                        ! 5
      goto 999
   20 continue
c―――――――――――――――――――――――――――――――――★トリリニア：最大点指向型
      call DirecMax_TriLiner(Member,Element,vv,vpp)                      ! 6
      goto 999
   30 continue
c―――――――――――――――――――――――――――――――――★トリリニア：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)                        ! 7
      goto 999
```

```
   40 continue
c――――――――――――――――――――――――――――――――――★バイリニア：Nomal
      if(Member.istat_v.eq.0.and.Member.istat_w.eq.0) then                  ! 8
      Element.AK_2=Element.AK_1
      Element.Q_2 =Element.Q_1
      endif
      call Mesing_TriLiner(Member,Element,vv,vpp)
      goto 999
   50 continue
c――――――――――――――――――――――――――――――――――★ 欠番
      goto 999                                                             ! 9
   60 continue
c――――――――――――――――――――――――――――――――――★欠番
      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)                                             ! 10
      write(222,'(A20,I5)') 'Modify Bi-Liner', mro
      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)                                             ! 11
      write(222,'(A20,I5)') 'Modify R-O', mro
      call Modify_RO1(Member,Element,RO_work(mro),vv,vpp)
      goto 999
      endif
  999 continue
      return
      end
```

プログラムで分かるように、せん断型モデルの履歴特性が階層として並んでいる。この部材モデル第２層の構造を確認しながら、プログラムコードの内容を見られたい。なお、タイプ別の履歴モデル番号は以下の用である。

タイプ別の履歴モデル番号
規定値０は、武田モデルとする。
１：トリリニア
２：最大点指向型
３：武田モデル
４：バイリニア
11：修正バイリニア
12：修正 R-O

1. 当該部材の要素構造体成分 Element.nm_typ より、履歴番号を取得する。

2. 履歴番号 No_rireki が 10 以内であるかどうかチェックし、以内であれば次の処理を行う。
3. 履歴番号 No_rireki にしたがって処理を分類する。
4. 履歴番号にしたがって処理を分類し、ここでは規定値:0 である武田モデルの履歴を処理するサブルーチン Takeda_TriLiner()がコールされる。
5. ここでは履歴番号:1 であるトリリニア型モデルの履歴を処理するサブルーチン Mesing_TriLiner()がコールされる。
6. ここでは履歴番号:2 である最大点指向型モデルの履歴を処理するサブルーチン DirecMax_TriLiner()がコールされる。
7. ここでは履歴番号:3 である武田モデルの履歴を処理するサブルーチン Takeda_TriLiner()がコールされる。
8. ここでは履歴番号:4 であるバイリニア型モデルの履歴を処理する場所であるが、第 3 剛性と第 2 折れ点のデータを変更して、サブルーチン Mesing_TriLiner()がコールされる。
9. これ以降のシングル番号は、現在欠番になっており、ダミー処理を行っている。
10. ここでは履歴番号:11 である修正バイリニアモデルの履歴を処理するサブルーチン Modify_Bi_Liner1()がコールされる。
11. ここでは履歴番号:12 である修正 RO モデルの履歴を処理するサブルーチン Modify_RO1 ()がコールされる。

せん断弾塑性モデルの履歴階層は、非常に単純な構造なので直ぐに理解できよう。せん断弾塑性モデルに新たな履歴特性を加えるには、この階層に仕様を考慮してサブルーチンを付け加えることになる。これについては、第 9 章で詳細に解説する。

4.15 接線剛性

本節では、最後の部材モデルの階層構造を有するサブルーチンとして、非線形剛性を作成するサブルーチンについて説明する。このサブルーチン Get_nonlinear_stiff()も部材モデルの第 1 階層を有している。まず、このサブルーチンをコールするコードを示す。

```
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 )
```

このサブルーチンコールも多数の引数を有しており、今後も部材モデルが増加するごとに増加する。次に、このサブルーチンの内容を以下に示す。

```
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,
     *     work1_elememt,work2_elememt, work1_member, work2_member)
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
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
c      record /member2_s  / Member
c――――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
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 Member(*),Element(*),MSS_work(*)
      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(*),v(12),ak(12,12)
      dimension vv(12)
C     ――――――――――――――――――――――――――――――――――――――――――――――――――――
      if(N_analysis.eq.7) return     !   線形解析;7                                1
      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) 2
c―――――――――――――――――――――――――――――――――★部材両端の変位取得
      do j=1,12                                                          ! 3
      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)         ! 4
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                            ! 5
   11 continue
c―――――――――――――――――――――――――――――――――★Model_No.1 通常の有限要素弾塑性モデル
      if(N_analysis.eq.9) goto 9999   ! 弾塑性解析はなし
      Member(i).an_vv(1)=vv(8)-vv(2)  ! 非線形剛性計算のための変位セット
      Member(i).an_ww(1)=vv(9)-vv(3)
      call Cal_nonlin_stiff_M1(Member(i),Element(ie),                    ! 6
     *                         Member(i).stress(7),ak )
      if(Member(i).i_rigid_length.ne.0..or.
     *          Member(i).j_rigid_length.ne.0.)
     *           call Deal_Rigid_element(ak,
     *             Member(i).i_rigid_length,Member(i).j_rigid_length)      ! 7
      goto 100
   12 continue
c―――――――――――――――――――――――――――――――――★Model_No.2 3 次元せん断弾塑性モデル
      if(N_analysis.eq.8) goto 9999   ! 幾何学的非線型解析はなし          ! 8
      call Cal_nonlin_stiff_M2(Member(i),Element(ie),                    ! 9
     *     ak )
      goto 100
   13 continue
c―――――――――――――――――――――――――――――――――★Model_No.3 3 次元軸力弾塑性モデル
      if(N_analysis.eq.9)then                                  ! 弾塑性解析
      call Cal_nonlin_stiff_M3(Member(i),Element(ie),ak)                 ! 10
      endif
      if(N_analysis.eq.8.or.N_analysis.eq.10)then              ! 幾何学的非線形解析
      call Cal_geomet_stiff_Nx(Member(i).stress(1),vv,
     *             Member(i),Element(ie),ak)
      endif
      goto 100
   14 continue
c―――――――――――――――――――――――――――――――――★Model_No.4 3 次元ケーブル弾塑性モデル
      if(N_analysis.eq.8) goto 9999   ! 幾何学的非線型解析はなし
```

```
      call Cal_nonlin_stiff_M4(Member(i),Element(ie),
     *     ak )
      goto 100
   15 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.5 3 次元免振モデル
      if(N_analysis.eq.8) goto 9999   ! 幾何学的非線型解析はなし
      iee=Member(i).n_model_type
      call Cal_nonlin_stiff_M5(Member(i),Element(ie),
     *                  Model_type.n_spring,Model_type.cosin(1,1),
     *                  MSS_work(iee),ak )
      goto 100
   16 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.6 3 次元制震 Maxwell モデル(ok)
      if(N_analysis.eq.8) goto 9999   ! 幾何学的非線型解析はなし
      goto 9999
   17 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.7 3 次元プレテンション動作モデル
      if(N_analysis.eq.8) goto 9999   ! 幾何学的非線型解析はなし
      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_nonlin_stiff_M11(N_analysis,                                    ! 11
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,E_model11, E_model_fiber,
     *      M_model11 , M_model_fiber)
      goto 100
  112 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.12 両端、中央ファイバーモデル
      call Cal_nonlin_stiff_M12(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,E_model12, E_model_fiber,
     *      M_model12 , M_model_fiber)
      goto 100
  113 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.13 両端 MS モデル
      call Cal_nonlin_stiff_M21(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,E_model21, E_model_fiber,
     *      M_model21 , M_model_fiber)
      goto 100
  114 continue
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★Model_No.14 両端、中央 MS モデル
      call Cal_nonlin_stiff_M22(N_analysis,
```

```
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,E_model22, E_model_fiber,
     *      M_model22, M_model_fiber)
      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_nonlin_stiff_M15(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,E_model15, E_model_fiber,
     *      M_model15 , M_model_fiber)
      goto 100
  116 continue
c-------------------------------★Model_No.16 両端、中央アナロジーモデル
      call Cal_nonlin_stiff_M31(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,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_nonlin_stiff_M32(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak ,ier,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_nonlin_stiff_M13(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak,ier,E_model13, E_model_fiber,
     *      M_model13,M_model_fiber)
      goto 100
  119 continue
c-------------------------------★Model_No.19
      call Cal_nonlin_stiff_M33(N_analysis,
     *      Model_type,Member(i),Element(ie),
     *      ak,ier,E_model33, E_model_fiber,
     *      M_model33,M_model_fiber)
      goto 100
  120 continue
c-------------------------------★Model_No.20
      goto 100
      endif
      goto 100
 9998 continue
c-------------------------------★Model_No.DLL
c      call Cal_nonlin_stiff_dll(mem,Member(i),Element(ie),
c     *   work1_elememt,work2_elememt,work1_member,work2_member,
c     *     vv,ak)
  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),                                  ! 12
     *              rot_memb(1,1,2,i),ak_nonlinear(1,1,i))
 9999 continue
```

```
return
end
```

上記のサブルーチンの構造は、手続き上多少長くなっているが、前節で説明したプログラムの構造とほとんど同じとなっている。前節のサブルーチンと異なっている部分を中心に、このサブルーチンの説明を行う。

1. 解析種別で線形解析の場合は、このサブルーチンをスキップする。
2. 当該の部材が線形解析を指定されている場合は、この部材の非線形剛性の再計算を取りやめる。
3. 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、部材両端の増分後の変位をセットする。
4. 両端節点の変位を釣合座標系から部材座標系に変換する。
5. 部材モデル番号で処理が分類され、ここでは、幾何学的非線形解析を行う場合の処理が行われる。まず、非線形剛性を計算するために、面外方向の変位をセットする。
6. 非線形剛性 ak をサブルーチン Cal_lonlin_stiff_M1()で求める。
7. 部材両端及びどちらか一方に剛域がある場合は、非線形剛性 ak を、サブルーチン Deal_Rigid_element()を用いて変換する。
8. ここでは、せん断型弾塑性モデルについて非線形剛性 ak を求める。ただし、このモデルは幾何学的非線形性は考慮しないため、解析種別が幾何学的非線形解析の場合はここでの処理をスキップする。
9. サブルーチン Cal_lonlin_stiff_M2()を用いて非線形剛性 ak を求める。
10. ここでは、座屈を考慮した３次元軸力弾塑性モデルの非線形剛性 ak を求める。弾塑性解析では、サブルーチン Cal_nonlin_stiff_M3()を用いて、さらに幾何学的非線形性を考慮する場合は、サブルーチン Cal_geomet_stiff_Nx()を用いて、非線形剛性 ak に剛性を足しこむ。これ以降の部材モデルは同様の処理を行う。
11. ここでは、両端ファイバーモデルの非線形剛性 ak を、サブルーチン Cal_nonlin_stiff_M11()より求める。静的縮合モデルであるこれ以降の部材モデルに対しては、次章で詳しく解説する。
12. 部材モデル番号で処理を分担して求めた非線形剛性 ak を、サブルーチン Rotate_K()を用いて、部材座標系から釣合座標系に変換する。また、その剛性を配列 ak_nonlinear に保存する。