第６章　履歴モデル

6.1 はじめに

SPACE に組み込まれている履歴モデルは、せん断型モデルの履歴やファイバーの履歴、あるいは個材座屈を考慮したトラスモデルのように部材特有の履歴モデルもある。本節では、部材モデル階層構造の最下層に位置する履歴モデルについて解説する。履歴モデルは最下層にあるため、他のプログラムと強く関連することがないことから、複雑なバグを発生させる可能性は少ない。しかし、履歴モデルには、それ特有の難しさやエラーを生じさせる要因を含んでおり、あらゆる場合を想定して徹底的にチェックする必要があろう。

履歴モデルを作成する場合、気を付けなければいけない点が多々ある。理論を正確に表現するコードのみでは正常に動作しない。履歴モデルは多くの場合、増分型で作られるため、一旦正常な履歴ループから外れると、後は全く意味のない計算を行うことになる。このような現象が如何にして生じるのか、その原因を考えてみよう。

履歴モデルを要因とするエラーには、次第に誤差が蓄積し、解析結果に信頼が置けなくなる場合と、一瞬に大きな誤差が発生し、履歴ループを正常に追えなくなってしまう場合とがある。前者は、履歴ループを直線で近似するモデルで、ある直線から他の直線に乗り移るときに生じる。このとき応力は大なり小なり必ず飛び越しを起こし、誤差が生じる。これを避けるために、履歴モデルの中になんらかの処置をしておかなければならない。SPACE での対処法は、各モデルの解説の項で紹介する。一方、ユーザー側は増分時間を小さくしてこれに対処することになる。

後者のエラーは、以後の数値計算を無意味とする。この種のエラーは、増分ひずみが突然大きくなり、履歴ループを正常に追えなくなるときに生じる。例えば、構造物に崩壊メカニズムが発生し、部分的に大きな変位が発生するときにこのような大きなひずみを生じる。また、接線剛性がゼロに近い場合や極端な剛性変化が生じた場合にもこのような現象が起き易い。これら現象を踏まえて、履歴モデルをプログラムコードであらわすとき、何らかの対処法が必要となろう。

現在、SPACE に組み込まれている履歴モデルは、

１．バイリニア型：
２．トリリニア型：
３．最大点指向型
４．武田モデル
５．修正バイリニア型

６．修正 RO モデル
７．スリップバイリニア型
８．個材座屈を考慮したトラス型

があり、特にファイバーの履歴モデルとしては、

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

がある。

SPACE における履歴モデルの組込みは容易である。今後も、履歴モデルを必要に応じて増やしていく予定である。新たな履歴モデルの組み込み法については、第 9 章で説明する。

6.2 ファイバーの履歴

本節では、ファイバーの履歴特性について説明する。ファイバーの履歴に関連するサブルーチンは、初期設定と材料非線形性のチェックに関連する 2 つのサブルーチンである。ここでも、この履歴特性を管理するために階層構造を用いている。まず、この2つのサブルーチンを示すが、ここでは履歴に関連する部分についてのみ示し、他の部分は省く。このサブルーチンは、いずれも両端ファイバーモデルについてである。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_Model_GI11
C     ―――――――――――――――――――――――――――――――――
C     ●   線形剛性行列の計算(ok)
C     ―――――――――――――――――――――――――――――――――
C     ●   Model_No.11 両端ファイバーモデル
C     ―――――――――――――――――――――――――――――――――
      subroutine Fiber_Model_GI11()
      ・
      ・
c―――――――――――――――――――――――――――――――★
      if(it.eq.1) then                                          ! 1
      nmx                 = M_model_fiber(nnm).n_type
      nm_type             = E_model_fiber(nn).nm_type
      goto ( 10,20,30,30,10,20,10,20),nm_type                   ! 2
  10  continue
c―――――――――――――――――――――――★バイリニア型（スチール用）   ! 3
      Bilinear_work(nmx).i_stat  = -1          !  ファイバー要素の状態
```

```
      goto 100
   20 continue
c―――――――――――――――――――――――――★トリリニア型                          ! 4
      Trilinear_work(nmx).i_stat = -1          !  ファイバー要素の状態
      goto 100
   30 continue
c―――――――――――――――――――――――――★コンクリート型                        ! 5
      Concrete_work(nmx).i_stat  = -1          !  ファイバー要素の状態
      goto 100
  100 continue
         enddo
c―――――――――――――――――――――――――――――――――★ファイバー要素のデータセット
        .
        .
c―――――――――――――――――――――――――――――――――★初期剛性行列の計算
      call Fiber_Model_G11(it,ak,alength,Member,Element,                    ! 6
     *      E_model,E_model_fiber,M_model,M_model_fiber)
      return
      end
C     ―――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_check11
C     ―――――――――――――――――――――――――――――――――――――
C     ●   ファイバー要素の材料非線形性チェックし、応力を計算
C     ―――――――――――――――――――――――――――――――――――――
      subroutine Fiber_check(  )
       .
       .
      if(N_analysis.le.8.or.Member.nm_analysis.eq.-1) then                 ! 7
c―――――――――――――――――――――――――★弾性解析
c                                ファイバー軸方向応力計算                  ! 8
      M_model_fiber(nnm).d_stress_x = M_model_fiber(nnm).d_E*du +
     *                                M_model_fiber(nnm).d_stress_x
      else
c―――――――――――――――――――――――――★弾塑性解析
         goto ( 10,20,30,40,50,60,70,80),nm_type                           ! 9
   10 continue
c―――――――――――――――――――――――――★バイリニア型（スチール用）          ! 10
      call BiLinear(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).Q_1,du,
     *              M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1)
      if(Bilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      if(Bilinear_work(nmx).i_stat.ne.0) then
         endif
         goto 100
   20 continue
c―――――――――――――――――――――――――★対称トリリニア型 （スチール用）    ! 11
      call TriLinear(M_model_fiber(nnm).d_E,Trilinear_work(nmx).i_stat,
     *               E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *               E_model_fiber(nn).E_3,
     *               E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *               du,M_model_fiber(nnm).d_stress_x,
     *               Trilinear_work(nmx).P1(1))
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
```

```
      goto 100
  30  continue
c―――――――――――――――――――――――――★直線コンクリート型                    ! 12
      call Concrete(M_model_fiber(nnm).d_E,Concrete_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).E_3,E_model_fiber(nn).Ec_3,
     *              E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *              E_model_fiber(nn).Ec_1,E_model_fiber(nn).Ec_2,
     *              du,  M_model_fiber(nnm).d_stress_x,
     *              Concrete_work(nmx).p1(1),Concrete_work(nmx).P1(2),
     *              Concrete_work(nmx).ipret,Concrete_work(nmx).P1(3),
     *              Concrete_work(nmx).p1(4),Concrete_work(nmx).P1(5),
     *              Concrete_work(nmx).ipre_c)
      if(Concrete_work(nmx).i_stat.ne.0.and.Concrete_work(nmx).i_stat
     *            .ne.3) iistat = iistat + 1
      goto 100
  40  continue
c―――――――――――――――――――――――――★曲線コンクリート型                    ! 13
      call Concrete_e(M_model_fiber(nnm).d_E,Concrete_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_3,
     *              E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *              du,  M_model_fiber(nnm).d_stress_x,
     *              Concrete_work(nmx).P1(1),Concrete_work(nmx).P1(2),
     *              Concrete_work(nmx).ipret,Concrete_work(nmx).ipre_c,
     *              Concrete_work(nmx).P1(3),Concrete_work(nmx).P1(4),
     *              Concrete_work(nmx).P1(5),E_model_fiber(nn).Ec_1,
     *              Concrete_work(nmx).P1(6),E_model_fiber(nn).Ec_2)
      if(Concrete_work(nmx).i_stat.eq.0)then
         if(du.lt.E_model_fiber(nn).Ec_1)then
         iistat = iistat + 1
         endif
         elseif(Concrete_work(nmx).i_stat.ne.3)then
         iistat = iistat + 1
         endif
      goto 100
  50  continue
c―――――――――――――――――――――――――★対称バイリニア型（移動＋等方硬化用）    ! 14
      call BiLinear_h(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).Q_1,du,
     *              M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1,
     *              E_model_fiber(nn).Beta)
      if(Bilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1

         goto 100
  60  continue
c―――――――――――――――――――――――――★対称トリリニア型（移動＋等方硬化用）    ! 15
      call TriLinear_h(M_model_fiber(nnm).d_E,
     *               Trilinear_work(nmx).i_stat,
     *               E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *               E_model_fiber(nn).E_3,
     *               E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *               du,M_model_fiber(nnm).d_stress_x,
     *              Trilinear_work(nmx).P1(1),Trilinear_work(nmx).P1(2),
```

```
     *              E_model_fiber(nn).Beta, E_model_fiber(nn).Beta_2)
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100

  70  continue
c―――――――――――――――――――――――――★非対称バイリニア型                    ! 16
      call BiLinear_AS(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).Q_1,E_model_fiber(nn).Ec_1,
     *              E_model_fiber(nn).Ec_2,E_model_fiber(nn).Qc_1,du,
     *              M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1,
     *              Bilinear_work(nmx).P2,Bilinear_work(nmx).Sig_z,
     *              E_model_fiber(nn).Beta)
      if(Bilinear_work(nmx).i_stat.eq.2.or.
     *   Bilinear_work(nmx).i_stat.eq.3) iistat = iistat + 1
      goto 100
  80  continue
c―――――――――――――――――――――――――★非対称トリリニア型                    ! 17
      call TriLinear_AS(M_model_fiber(nnm).d_E,
     *               Trilinear_work(nmx).i_stat,
     *               E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *               E_model_fiber(nn).E_3,E_model_fiber(nn).Ec_1,
     *               E_model_fiber(nn).Ec_2,E_model_fiber(nn).Ec_3,
     *               E_model_fiber(nn).Q_1,E_model_fiber(nn).Q_2,
     *               E_model_fiber(nn).Qc_1,E_model_fiber(nn).Qc_2,
     *               du,M_model_fiber(nnm).d_stress_x,
     *              Trilinear_work(nmx).P1(1),Trilinear_work(nmx).P1(2),
     *              Trilinear_work(nmx).P1(3),
     *               E_model_fiber(nn).Beta, E_model_fiber(nn).Beta_2)
      if(Trilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1
      goto 100
c―――――――――――――――――――――――――★弾塑性解析終了
      endif
  100 continue
      M_model_fiber(nnm).d_eps_x = M_model_fiber(nnm).d_eps_x + du !ファイバー要素の歪  18
      enddo
c―――――――――――――――――――――――――――――――――★ファイバー要素応力の計算
       .
       .
c―――――――――――――――――――――――――――――――――★j 端部ファイバー
c―――――――――――――――――――――――――――――――――★歪のセット
      elseif(it.eq.2) then
       .
       .
      endif
      return
      end
```

1. このサブルーチンは両端ファイバーモデルに関する線形剛性を求めるサブルーチンであり、ファイバーに関する初期設定を行うためにパラメータを設定する。まず、変数 it が１の場合は、i 端のファイ

バー断面について剛性を求める。このファイバーの履歴型 nm_type を取得する。

2. ファイバーの履歴型に基づいて初期設定を分類する。ただし、ファイバーの履歴は、現在、8 つに分類されているが、そのワーク領域が 3 つに分類されているため、ここでは、ワーク領域の分類にしたがって初期設定を行っている。
3. ここでは、バイリニア型の履歴特性の初期設定を行う場所であり、状態を表す構造体成分 Bilinear_work(nmx).i_stat を-1 にセットする。

   この構造体に-1 をセットすることで、次節で示すサブルーチンの中で、ワーク領域の初期設定が行われることになる。
4. ここでは、トリリニア型の履歴特性の初期設定を行う場所であり、状態を表す構造体成分 Trilinear_work(nmx).i_stat を-1 にセットする。
5. ここでは、直線コンクリート型の履歴特性の初期設定を行う場所であり、状態を表す構造体成分 Concrete_work(nmx).i_stat を-1 にセットする。
6. このサブルーチンによってファイバー断面の線形剛性行列を求める。
7. このサブルーチン Fiber_check()は、ファイバーの非線形性をチェックし、ファイバーの接線剛性を求める。まず、解析の種類とこの部材の解析種別をチェックし、塑性解析を行う必要がない場合、つまり、弾性解析の場合は次の処理を行い、弾塑性解析を行う場合は、9 以降の処理を行う。
8. ファイバーの軸方向ひずみから弾性の軸方向応力を計算する。
9. ファイバーの履歴特性に合わせて、弾塑性チェックを行う。ここで、履歴モデルの階層構造が見られることになる。
10. ここでは、対称バイリニア型の履歴特性を有するファイバーの弾塑性チェックを行い、増分後の軸方向応力と接線剛性を求める。
11. ここでは、対称トリリニア型の履歴特性を有するファイバーの弾塑性チェックを行い、増分後の軸方向応力と接線剛性を求める。
12. ここでは、直線コンクリート型の履歴特性を有するファイバーの弾塑性チェックを行い、増分後の軸方向応力と接線剛性を求める。
13. ここでは、曲線コンクリート型の履歴特性を有するファイバーの弾塑性チェックを行い、同様の処理を行う。
14. ここでは、対称バイリニア型（移動+等方硬化）の履歴特性を有するファイバーの弾塑性チェックを行い、同様の処理を行う。

15. ここでは、対称トリリニア型（移動+等方硬化）の履歴特性を有するファイバーの弾塑性チェックを行い、同様の処理を行う。
16. ここでは、非対称バイリニア型の履歴特性を有するファイバーの弾塑性チェックを行い、同様の処理を行う。
17. ここでは、非対称トリリニア型の履歴特性を有するファイバーの弾塑性チェックを行い、同様の処理を行う。
18. ファイバーの増分ひずみを、増分前のひずみに足しこむ。これで、i 端の弾塑性チェック処理は終了し、次に j 端の処理を行う。処理の内容は上記に示した内容と同様である。

6.2.1 バイリニア型履歴モデル

本節では、履歴特性としては、図 6-1 に示す最も単純なファイバーの履歴ルールであるバイリニア型について説明する。図中の記号は、

σ_1：降伏軸方向応力
E_1：初期剛性
E_2：第２剛性

である。

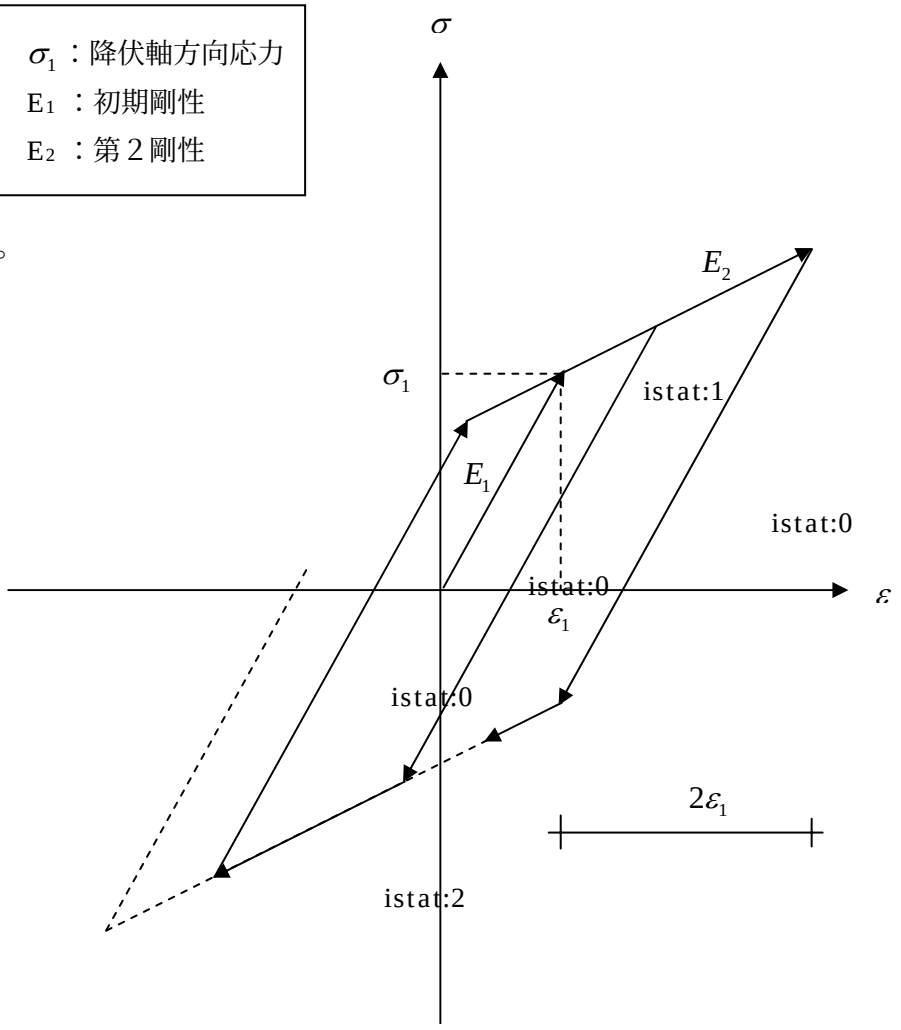

図 6-1 ファイバーモデルのバイリニア型履歴特性

履歴特性は骨格曲線と履歴ルールによって決定される。ここでは、最も単純なバイリニア Masing 型の履歴特性について説明する。この履歴特性はスチールの履歴特性として用いられる。骨格曲線は図 6-1 に示すようなバイリニアであり、また、ファイバーの弾塑性状態は、istat:0 などと表されている。このバイリニア型では、状態 istat:0 は弾性状態を表し、状態 istat:1 は引張降伏以後の状態を、状態 istat:2 は圧縮降伏以後の状態を表す。また、除荷後の状態は、弾性状態と同じであるとする。

履歴特性をプログラム化する場合、全ての弾塑性状態を把握した後、各状態の特性を正確に理解することが大切である。特に、ある状態から他の状態に移り変わるルールと、その時点におけるパラメータなどの設定を的確に行うことである。骨格曲線が折れ線で与えられている場合、状態が変移するとき、図 6-2 に示されるように必ず飛びこしが生じる。これを避けるために反復計算を導入して誤差を小さくすることも考えられるが、SPACE では、計算効率と解析誤差を考慮して、次の 3 方法のいずれかを用いている。

図 6-2 の左のタイプ（タイプ１）は、剛性が強から弱に変移する場合で使用し、元の剛性で増分変位 du を求め、その後、弱の剛性で増分応力を求める。同図の中のタイプ（タイプ２）は、除荷の場合で、剛性が弱から強に変移する場合に使用する。弱の剛性を用いて増分応力を求め、その後、強の剛性を用いて増分変位を計算する。図 6-2 の右のタイプ（タイプ３）は、変移する前の剛性と後の剛性を用いてエネルギーを計算し、このエネルギーが同一となるように増分変位と増分応力を求める。

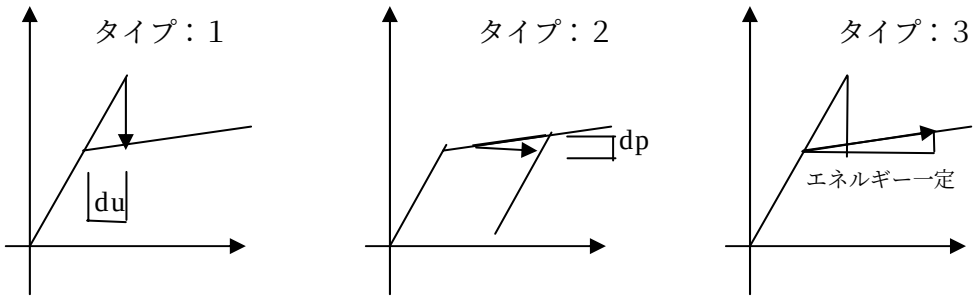

図 6-2 飛び越しの修正法

バイリニア Masing 型では、弾塑性状態が 3 つに分類されており、その状態は以下のようである。

１）istat:0　引張側と圧縮側の両方に境界を持つ。最初は、弾性状態であるため、その境界は降伏軸方向応力となるが、塑性状態に入った場合は新たに設定する必要がある。他の状態への移動は、引張側は istat:1 へ、圧縮側は istat:2 へとなる。

２）istat:1　増分ひずみが引張ひずみである場合は境界がなく、圧縮ひずみの場合は、その点が反曲点となり、弾性状態 istat:0 となる。ここで、istat:0 状態における両境界値を設定する必要がある。

３）istat:2　増分ひずみが圧縮ひずみである場合は境界がなく、引張ひずみの場合は、その点が反曲点となり、弾性状態 istat:0 となる。ここで、istat:0 状態における両境界値を設定する必要がある。

次に、上記の履歴ルールにしたがって記述されたサブルーチン BiLinear()を具体的に見てみよう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / BiLinear
C     ―――――――――――――――――――――――――――――――――
C     ●   BiLiner_Masing 履歴モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine BiLinear(AK,istat,AK_1,AK_2,Q_1,du,P,P1)
      implicit real*8(A-H,O-Z)
      real*8  AK,AK_1,AK_2,Q_1,P,P1
c―――――――――――――――――――――――――――――――――――――――――――
c    AK               :接線剛性
c    istat            :現在の状態(Work)
c    AK_1             :第一勾配
c    AK_2             :第二勾配
c    Q_1              :第一折れ点のせん断力
c    du               :増分変位
c    P                :現在のせん断力(Work)
c    P1               :istat=0 における反曲点上端(Work)
C―――――――――――――――――――――――――――――――――――――――――――
c
  100 continue
      if(istat.eq.0) then                                          ! 1
      p = AK*du + p                                                ! 2
      if(du.gt.0.) then                                            ! 3
      if(p.lt.p1) return                                           ! 4
c―――――――――――――――――――――――――――――★istat = 0 から istat=1 へ
      istat=1                                                      ! 5
      du2= (p - p1)/AK
      AK=AK_2
      p = p1 + AK*du2
      return
      else                                                         ! 6
      p2= p1 - 2.* Q_1                                             ! 7
      if(p.gt.p2) return                                           ! 8
c―――――――――――――――――――――――――――――★istat = 0 から istat=2 へ
```

```
      istat=2                                                          ! 9
      du2= (p - p2 )/AK                                                ! 10
      AK=AK_2                                                          ! 11
      p = p2 + AK*du2                                                  ! 12
      return
      endif
      elseif(istat.eq.1) then                                          ! 13
      p = AK*du + p                                                    ! 14
      if(du.ge.0.) return
c―――――――――――――――――――――――――――――★istat =1 から istat=0 へ
      istat=0                                                          ! 15
      dp = AK*du                                                       ! 16
c                          誤差が大きくなるため、除荷の場合、
c                          一回は元の剛性を使用して、歪を計算する。
      AK=AK_1                                                          ! 17
      du=dp/AK
      p=p-dp
      p1 = p
      p = AK*du + p
      return
      elseif(istat.eq.2) then                                          ! 18
      p = AK*du + p
      if(du.le.0.) return                                              ! 19
c―――――――――――――――――――――――――――――★istat =2 から istat=0 へ
      istat=0                                                          ! 20
      dp =  AK*du
      p=p-dp
      p1 = p + 2. * Q_1
      AK=AK_1
      du=dp/AK
      p = AK*du + p
      return
      else
c―――――――――――――――――――――――――――――★初期設定
      istat = 0                                                        ! 21
      AK=AK_1
      p1=Q_1
      p=0.
      goto 100
      endif
      return
      end
```

1. 状態が弾性の場合（istat=0)、以下の処理を行う。
2. 増分ひずみと剛性を用いて増分応力を計算し、その値を増分前の応力に足しこむ。
3. 増分ひずみが引張の場合（du>0)、以下の処理を行う。
4. 増分後の応力が引張側の境界値 p1 より小さい場合は、未だこの状態内であるとして、このサブルーチンより戻る。
5. 増分後の応力が引張側の境界値 p1 より大きい場合、状態が istat:1

となる。そこで、飛び越し修正法のタイプ１を用いて状態が変化したときの処理を行う。まず、応力が境界値 p1 より飛び出た部分のひずみを求め、du2 にセットする。次に、接線剛性を第 2 勾配の値とし、さらに、その接線剛性と先ほどの飛び出した部分のひずみ du2 とで、増分応力を計算し、増分後の応力をセットする。その後このサブルーチンから戻る。

6. 増分ひずみが圧縮の場合、以下の処理を行う。
7. 現在の引張側の境界値と引張側降伏応力を用いて、圧縮側の境界値を求め、p2 にセットする。
8. 増分後の応力が境界値より大きい場合、弾性範囲内であることから、このサブルーチンより戻る。
9. 増分後の応力が境界値より小さい場合、状態が変化し、istat:2 となる。
10. 飛び出た部分のひずみを求め、du2 にセットする。
11. ユーザー設定の剛性第 2 勾配を接線剛性の値としてセットする。
12. さらに、その接線剛性と先ほどの飛び出した部分のひずみ du2 とで、増分応力を計算し、増分後の応力をセットする。その後、このサブルーチンから戻る。ここで、状態 istat:0 の処理が終了する。
13. ここからは、状態 istat:1 の処理が始まる。
14. 増分応力を計算し、増分後の応力を計算する。
15. 増分ひずみ du が正の場合、つまり、引張ひずみの場合は、このサブルーチンより戻る。
16. 増分ひずみ du が負の場合、つまり、除荷が生じた場合、状態パラメータ istat を 0 に戻す。次に、タイプ２の飛び越し修正処理を用いて、増分変位と増分応力を修正する。まず、増分変位から増分応力を求める。
17. 接線剛性を第一勾配の剛性にセットする。その剛性と先に求めた増分応力を用いて増分変位を求め直す。また、状態 istat:0 における境界値 p1 をこの除荷位置の応力とする。求めた増分変位と増分応力より増分後の応力をセットし、サブルーチンから戻る。
18. ここからは、状態 istat:2 の処理が始まる。増分応力を計算し、増分後の応力を計算する。
19. 増分ひずみが負である場合、つまり、圧縮ひずみが進むとき、処理は終了し、サブルーチンより戻ることになる。
20. 増分ひずみが正である場合、応力が反転するため、状態パラメータを 0 に戻す。次に、タイプ２の飛び越し修正処理を用いて、増分変

位と増分応力を修正する。まず、増分応力を求め、増分後の応力から増分応力を引くことによって増分前の応力を求める。この増分前の応力から状態 istat:0 における境界値 p1 を計算する。このとき、反転応力 p は圧縮応力であり、境界値 p1 は引張側の境界値である。次に、接線剛性として第一勾配の剛性をセットし、その剛性を用いて先に求めた増分応力から増分変位を求める。この増分変位と第一勾配の剛性によって増分応力を求め、その増分応力を増分前の応力に足しこみ、増分後の応力を求める。これで、状態 istat:2 の処理が終了し、サブルーチンから戻ることになる。

21. ここでは、このファイバーに関する初期設定を行う。まず、状態パラメータを istat:0 とする。次に、接線剛性 AK に第一勾配の剛性をセットする。また、現在の応力 p をゼロにし、この状態の引張側の境界値 p1 を降伏応力とする。以上で初期設定処理を終了し、処理１に戻る。

さて、サブルーチン BiLinear()を用いて、ファイバーの弾塑性状態と接線剛性を求めたわけであるが、この情報がどのように利用されるかについて述べよう。先に示したサブルーチン Fiber_check()をもう少し詳細に検討する。

```
C     ―――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Fiber_check11
C     ―――――――――――――――――――――――――――――――
C     ●   ファイバー要素の材料非線形性チェックし、応力を計算
C     ―――――――――――――――――――――――――――――――
      subroutine Fiber_check()
c――――――――――――――――――――――――――――――★i 端部ファイバー
c――――――――――――――――――――――――――――――★歪のセット
      if(it.eq.1) then
      do i=1,nm_div
      if(N_analysis.le.8.or.Member.nm_analysis.eq.-1) then
c――――――――――――――――――――――――★弾性解析
c                                    ファイバー軸力計算
      M_model_fiber(nnm).d_stress_x = M_model_fiber(nnm).d_E*du +
     *                                M_model_fiber(nnm).d_stress_x
      else
c――――――――――――――――――――――――★弾塑性解析
      goto ( 10,20,30,40,50,60,70,80),nm_type
  10  continue
c――――――――――――――――――――――――★バイリニア型（スチール用）
      call BiLinear(M_model_fiber(nnm).d_E,Bilinear_work(nmx).i_stat,          ! 1
     *              E_model_fiber(nn).E_1,E_model_fiber(nn).E_2,
     *              E_model_fiber(nn).Q_1,du,
     *              M_model_fiber(nnm).d_stress_x,Bilinear_work(nmx).P1)
      if(Bilinear_work(nmx).i_stat.ne.0) iistat = iistat + 1                   ! 2
```

```
      goto 100
      ・
      ・
  100 continue
      M_model_fiber(nnm).d_eps_x = M_model_fiber(nnm).d_eps_x + du !ファイバー要素の歪
      enddo
c―――――――――――――――――――――――――――――――――★ファイバー要素応力の計算
      d_state = float(iistat)/float(nm_div)   ! 塑性した面積の計算              ! 3
      if(d_state .eq.0) then
      Member.d_stat(1) = 0
      elseif(d_state .ge.0.8) then
      Member.d_stat(1) = 2
      else
      Member.d_stat(1) = 1
      endif
      nm_div  = E_model.n_section_1
      nn      = E_model.nm_section_1 - 1
      nnm     = M_model.nm_section_1 - 1
      ra      = 0.                                                            ! 4
      ray     = 0.
      raz     = 0.
      raz2    = 0.
      ray2    = 0.
      rayz    = 0.
      gg      = 0.
      aa      = 0.
      AN      = 0.
      AMy     = 0.
      AMz     = 0.
      do i=1,nm_div                                                           ! 5
      nn      = nn  + 1
      nnm     = nnm + 1
      A       = E_model_fiber(nn).A
      E       = M_model_fiber(nnm).d_E                                        ! 6
      ra      = ra   + E*A
      ray     = ray  + E*E_model_fiber(nn).Arz
      raz     = raz  + E*E_model_fiber(nn).Ary
      ray2    = ray2 + E*E_model_fiber(nn).Arz2
      raz2    = raz2 + E*E_model_fiber(nn).Ary2
      rayz    = rayz + E*E_model_fiber(nn).Aryz
      aa      = aa + A
      gg      = gg + A*E_model_fiber(nn).G
      ANN     = M_model_fiber(nnm).d_stress_x*E_model_fiber(nn).A
      AN      = AN + ANN
      AMy     = AMy + ANN * E_model_fiber(nn).rz
      AMz     = AMz + ANN * E_model_fiber(nn).ry
      enddo
      nn=E_model.nm_section_1
      call jikuzero_control(Control,ra,E_model_fiber(nn).E_1,
     *                      E_model_fiber(nn).A)
      M_model.d_aa_1   = aa              !  断面積の和                          ! 7
      M_model.d_ra_1   = ra              !  E*断面積の和
      M_model.d_ray_1  = ray             !  E*A*z
      M_model.d_raz_1  = raz             !  E*A*y
```

```
      M_model.d_raz2_1 = raz2            !  E*A*y*y
      M_model.d_ray2_1 = ray2            !  E*A*z*z
      M_model.d_rayz_1 = rayz            !  E*A*z*y
      M_model.d_gg_1   = gg              !  G*A
c――――――――――――――――――――――――――――――――★j 端部ファイバー          ! 8
      elseif(it.eq.2) then
      •
      •
      endif
      return
      end
```

1. サブルーチン BiLinear()をコールし、ファイバーの弾塑性チェック処理を行う。接線剛性とファイバーの状態を、このサブルーチンの引数であるヤング係数 M_model_fiber().d_E と状態パラメータ Bilinear_work().i_stat として取得する。
2. ファイバーの状態をチェックし、弾性状態でないファイバー数を計算する。
3. 断面内で塑性になったファイバーの比率を求め、80%以上の場合は、構造体成分 Member.d_stat を 2 に設定し、ひとつでも塑性化したファイバーがある場合は１に設定する。この値はファイルに出力され、プレゼンターで塑性化と塑性ヒンジを表示する場合のパラメータとして使用される。
4. 断面内の各ファイバーのデータを用いて断面特性を求める。まず、ここでは、各特性値のゼロクリアを行う。
5. 断面内の全ファイバーについて、各特性値の和を取る。
6. ファイバーのヤング係数 M_model_fiber().d_E をセットする。
7. 求めた断面特性を構造体にセットする。

6.2.2 直線コンクリート型の履歴モデル

本節では、ファイバーに関する履歴ルールで、直線コンクリート型の履歴モデルについて説明する。図中の記号は以下のようである。

圧縮と引張の第一勾配	AK_1
圧縮第二勾配	AK_2
圧縮第三勾配	AK_3
引張第二勾配	AK_4
引張強度	Q_1
圧縮側第一折れ点の応力	Q_2
圧縮強度	Q_3
圧縮流れ点	Q_4

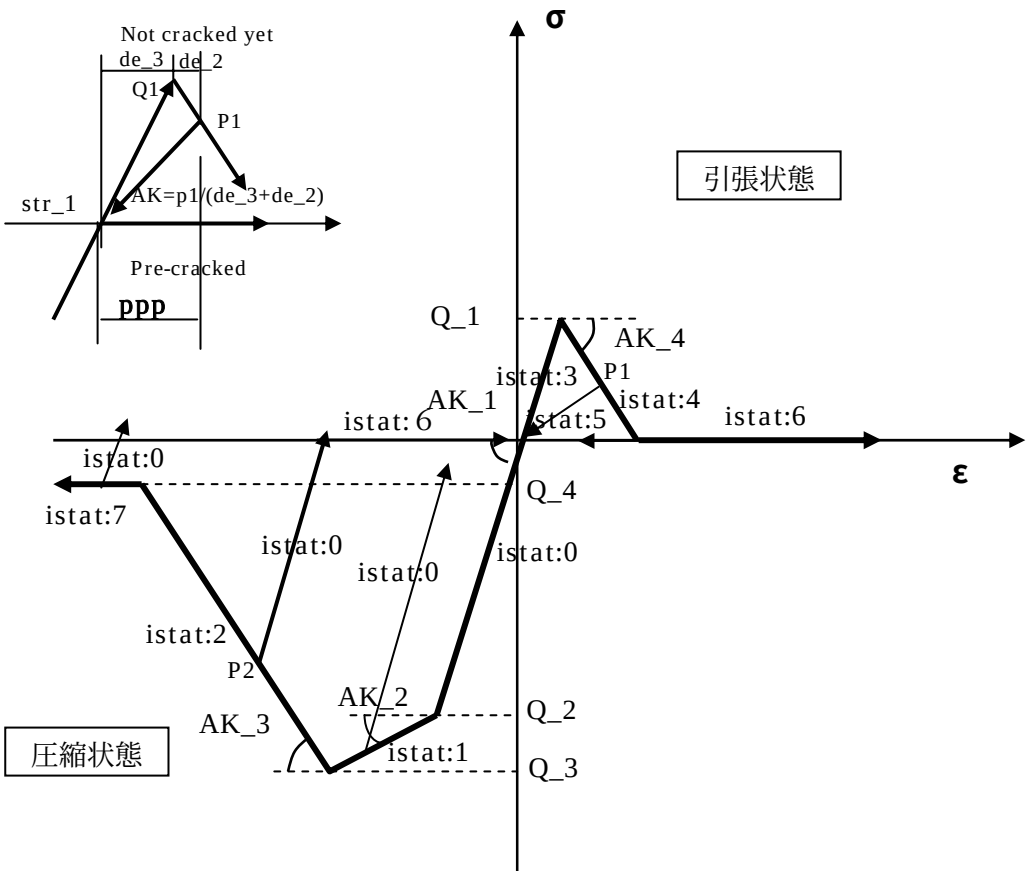

図 6-3 直線コンクリート型の履歴特性

コンクリートの履歴特性は、非常に複雑で、正確に模擬することはかなり難しい。例えば、骨格曲線はほとんど直線部分がなく、全てが非線形処理を行う必要がある。また、引張亀裂や圧縮破壊を経験すると、その後の履歴はそれ以前と大きく異なった挙動を示す。

SPACE の直線コンクリート型モデルは、このようなコンクリートの複雑な挙動を、近似的にしかも効率よく模擬できることを目的として開発されたものである。このモデルでは、コンクリートの履歴は全て直線で表す。また、履歴のルールは圧縮側と引張側の最大耐力を経験したかどうかで変化するため、複雑で理解し難い。そこで、ここでは履歴ルールを大きく４つに分けて検討する。

１．圧縮側と引張側、共に最大耐力を経験していない履歴
２．先に圧縮側の最大耐力を経験する履歴
３．上記と逆で、先に引張側の最大耐力を経験する履歴
４．両者共に、最大耐力を経験した後の履歴

履歴ルールを表すために、状態を表すパラメータ istat の他に６つのパラメータが用いられており、それらが重要な役割を果たしている。その６つのパラメータとは、(str_1,P1)、(str_2,P2)、(ipret,ipre_c)である。括弧でくくられているパラメータは対になっており、一つ目は、引張側のひずみと応力、二つ目は圧縮側のひずみと応力、最後は引張側及び圧縮側の崩壊経験を表すパラメータである。ここで説明する履歴ルールの中で、どのようにこれらのパラメータが利用されているのか、また、実際のプログラムの中でどのように使われているかを理解されたい。

最初に、1）の圧縮側と引張側、共に最大耐力を経験していない場合ついて、図 6-4 を用いて説明する。

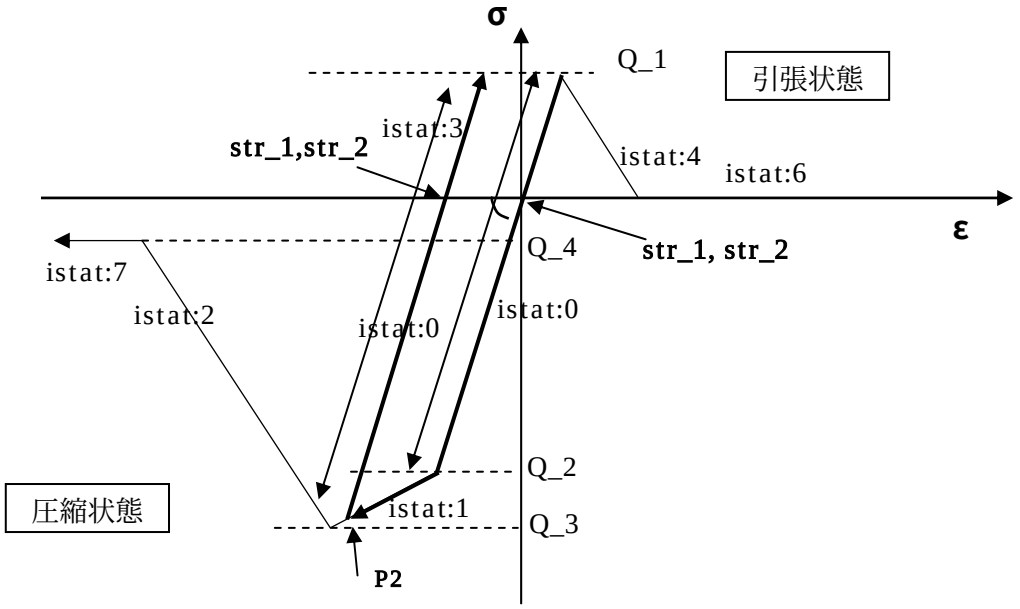

図 6-4　圧縮側と引張側、共に最大耐力を経験していない履歴

応力が、Q_1>σ>Q_2 の場合（引張側、圧縮側の最大耐力を表す P1 と P2 は、最初 Q_1 と Q_2 に設定されている）は、通常の弾性範囲と同様で、応力とひずみの関係を表す履歴は非常に単純である。圧縮側で、応力が Q_2 を超えた場合、履歴は、状態１の骨格曲線上に沿って動くが、その時点で増分ひずみが反転すると、その応力を P2 として、Q_1>σ>P2 の間を直線的に動くことになる。コンクリートの崩壊経験を表すパラメータ ipret（引張）と ipre_c（圧縮）は、いずれも 0 である。また、状態：0 における直線のパラメータとして、その直線の引張側最大値は Q_1、応力 0 におけるひずみは str_2 である。

次に、2)の先に圧縮側の最大耐力を経験する場合について説明する。

図 6-5 に示すように、ひずみが圧縮状態を進むと履歴は状態１から状態２へと骨格曲線上を移動し、圧縮側の最大耐力を経験する。状態 2 上で増分ひずみが反転すると、その応力を P2 として、その後の履歴は Q_1>σ>P2 の間を直線で移動することになる。これは、前に説明した状態 1 からの反転した場合と同様である。ここでは、コンクリートの崩壊経験を表すパラメータ ipre_c（圧縮）は 1 となる。また、状態：0 における直線のパラメータとして、未だ引張最大耐力を経験していないので、その直線の引張側最大値は P1、応力 0 におけるひずみは str_2 である。

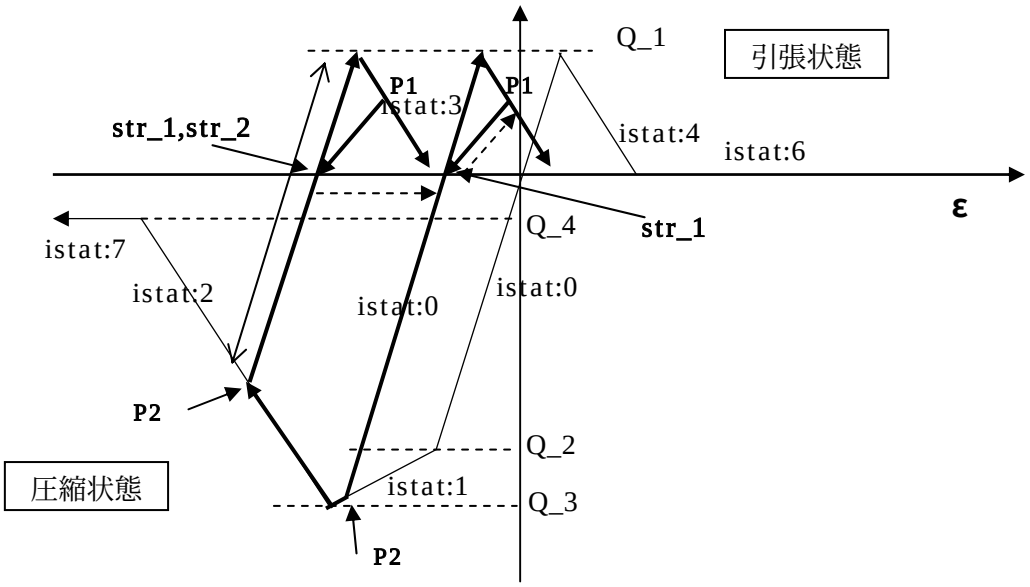

図 6-5　先に圧縮側の最大耐力を経験する履歴

ここで、先に圧縮側の最大耐力を超えてから、引張側の最大耐力を経験する場合について考える。無論、これ以前には引張側最大耐力を超える経験をしておらず、最初の引張側最大耐力経験である。もし、図 6-5 で示すように状態１から反転した履歴が、状態 0、状態 3 を通過した後、引張側の最大耐力を経験し、状態 4 において増分ひずみが再度圧縮になったとする。この時、最初に圧縮状態から引張状態に移ったときのひずみを str_1 とする。状態４において応力 P1 で増分ひずみが圧縮となり、ひずみが反転することで状態 5 となる。この状態 5 では、ファイバーの接線剛性は、ひずみ str_1 と応力 P1 を結ぶ直線の勾配となる。さらに同方向にひずみが進むと、ひずみ str_1 から状態 0 に戻ることになる。履歴が状態 5 から状態 0 に進む理由は、最初の引張側最大耐力経験であるためで、ひずみ str_1 と str_2 が同じ値であることによる。この値が異なる場合については、次に説明する。

さらにひずみが進み、図 6-5 のように状態 0、１、2 へと進んだ後、

再度、状態２で反転し、状態 0 に進んだ場合について考える。一度、引張最大耐力を経験しているため､応力 0 の位置(ここでのひずみが str_2 となる）から、点線で示す状態６となり、ひずみが str_1 になるまで進む。ひずみが str_1 に到達すると、引張剛性が発生して、P1 方向に向かう履歴となる。

次に、3)の先に引張側の最大耐力を経験する履歴について考えよう。図 6-6 に示されるように、ひずみがまず引張となり、さらに同方向に進行して最大引張耐力を超え、状態 4 に達する。ここで、増分ひずみが圧縮となり、応力 P1 で反転するとしよう。ここでは、状態 5 となり、接線剛性は、応力 P1 とひずみ str_1 によって計算される。さらに同方向にひずみが進むと、履歴は、ひずみ str_2（最初の最大耐力経験であるからひずみ str_1 と同じ）から状態 0 となり、つづいて、状態 1、状態２へと進む。ここで、増分ひずみが引張となると、反転して状態 0 となる。状態 2 で応力が 0 となると、状態が 6 となり、ひずみが str_1 まで進むことになる。このとき、折れ曲がる応力 0 の位置のひずみを新たに str_2 とする。ひずみが str_1 まで達すると、状態が 5 となり、P1 に向かって進むことになる。

上記の状態で、今度は、P1 から、ひずみが圧縮方向に進むとき、状態 5 からひずみ str_1 で状態 6 となり、ひずみが str_2 の方向に進むことになる。ひずみが str_2 に達した後、圧縮側に剛性が生じ、状態 0 となって、ひずみが進むことになる。無論、引張破壊が進み、状態 6 になった後､ひずみが逆転して逆方向に進むと､状態 6 のままひずみが str_2 まで進み、同じく圧縮剛性が生じ、状態 0 となって P2 方向に進む。

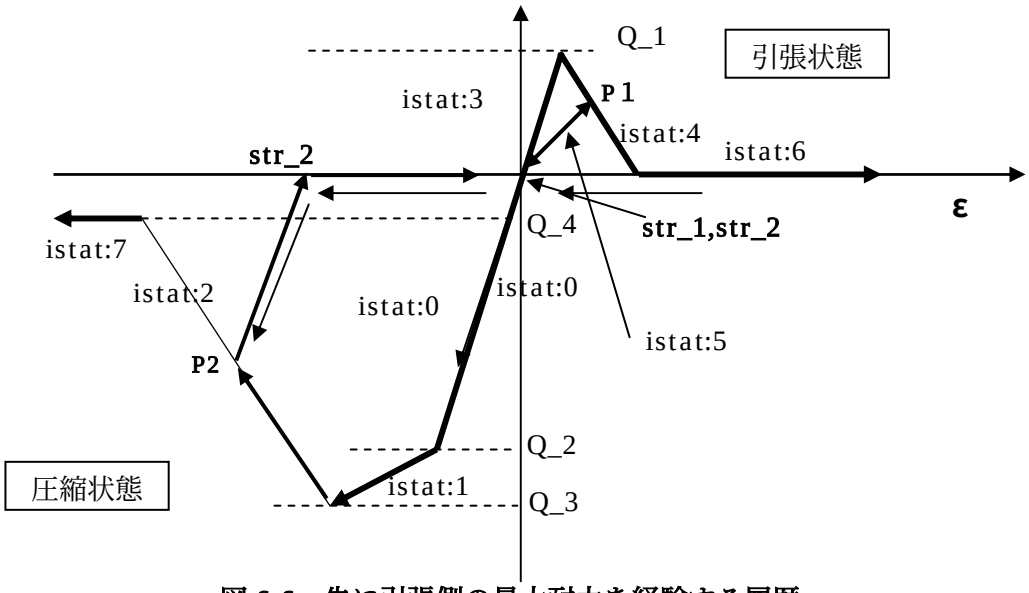

図 6-6　先に引張側の最大耐力を経験する履歴

最後に、4)の図 6-7 で示す圧縮側も引張側も最大耐力を経験した後の履歴特性について説明する。まず、圧縮の状態 2 において応力 P2 で増分ひずみが引張になり、反転して状態 0 となる場合を考えよう。さらに、ひずみが同方向に進行し、応力が 0 となって状態が 6 に変化する。この折れ曲がった点のひずみを str_2 として設定する。そのままひずみが進行すると、ひずみ str_1 に到達する。ここで、引張破壊パラメータ ipret（引張）が 1 の場合は状態が 5 となり、応力 P1 に向かってひずみが進行する。さらに、P1 を過ぎると状態４を経て、状態 6 へと移っていく。この時、引張破壊パラメータ ipret（引張）は 2 となる。一方、既に引張破壊パラメータ ipret（引張）が 2 の場合は、そのまま状態が 6 で変化せず、ひずみは同方向に進むことになる。

次に、逆進する場合について考える。引張破壊パラメータ ipret（引張）が 1 の場合は、状態 4 で反転し、状態 5 となってひずみが str_1 に到達すると状態 6 となる。一方、引張破壊パラメータ ipret（引張）が 2 の場合は、状態６に既に入っており、ひずみ str_1 を通過して、str_2 まで進行する。次に、ひずみ str_2 で、圧縮側の剛性が発生し、状態 0 に変化する。さらに、ひずみが同方向に進行すると、応力 P2 で状態 2 へと移り、最後にコンクリートの圧壊状態である状態７へと進行する。ここで、ひずみが反転すると状態 0 を経由して、状態 6 へと移っていく。この状態を経験すると引張側の剛性は期待できなくなり、ipre_c（圧縮）を 2 に設定する。

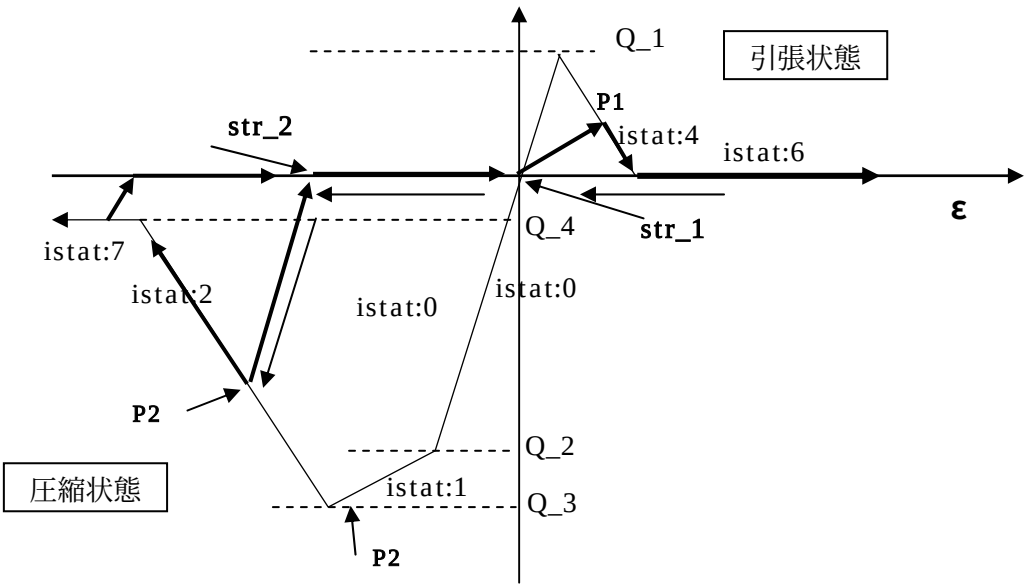

図 6-7 両者共に、最大耐力を経験した後の履歴

以上の説明をまとめると、直線コンクリート型の履歴特性は、次の 8 つの状態で表現される。

istat：0　圧縮側の弾性状態もしくは状態 1、2 から反転して戻るときの状態であり、境界値としては、P2＜σ＜0 である。圧縮側は、骨格曲線との交点 P2 の位置で異なり、状態：1 か 2 または 7 へと移動する。これは崩壊経験を表すパラメータ ipre_c（圧縮）の値によって区別する。パラメータ ipre_c が 0 の場合、状態 1 へ、ipre_c が 1 では状態 2 へ、ipre_c が 3 では状態 7 へ変化する。一方、引張側は以前に引張崩壊を経験しているか否かによって移動する状態が異なる。まず、経験していない場合は状態：3 に移動する。この時、応力 0 のひずみが str_1 および str_2 となる。状態：4 を経験し、しかも str_1 と str_2 が同じ場合は最大引張耐力 P1 に向かって移動する。この状態は 5 である。状態：6 を経験している場合は状態：6 へ移動する。これも、崩壊経験を表すパラメータ ipret（引張）の値によって区別する。パラメータ ipret が 0 の場合、状態 3 へ、ipret が 1 でしかも str_1 と str_2 が同じ場合は状態 5 へ、str_1 と str_2 が異なる場合もしくは ipret が 2 では状態 6 へ変化する。

istat：1　圧縮側の第 2 勾配であり、境界値として増分ひずみが圧縮の場合は Q_3＜σで、引張の場合はひずみが反転する。この場合、状態：0 となる。

istat：2　圧縮側の第 3 勾配であり、特徴として接線剛性が負勾配を有する。境界値として増分ひずみが圧縮の場合は Q_4＞σで、引張の場合は、ひずみが反転する。この場合、状態：0 となる。崩壊経験を表すパラメータ ipre_c は 1 となり、圧縮耐力を示す応力 P2 は、反転する応力となる。

istat：3　引張側の第 1 勾配であり、境界は 0＜σ＜Q_1 である。増分ひずみが引張で応力が Q_1 を超える場合は、状態は 4 に移動する。逆に、増分ひずみが圧縮の場合で、応力が 0 を超えるとき、str_1 と str_2 が同じ場合は状態 0 へ、str_1 と str_2 が異なる場合は状態 6 へ変化する。

istat：4　引張側の第 2 勾配であり、特徴として接線剛性が負勾配を有する。境界値としては、0＜σであり、増分ひずみが境界を越えて大きくなると状態は 6 へと変化する。逆に、反転して圧

縮増分ひずみが生じるばあいは、状態：5 となる。状態：5 における接線剛性は、状態：3 で引張側に入ったひずみ str_1 に向かう直線より決定する。

istat：5 ここでの接線勾配は計算によって求める。境界値としては、圧縮側は、最大耐力 P1 であり、引張側は、応力 0 である。圧縮側は状態：4 になり、引張側は、ひずみが str_1 と str_2 等しいと状態：0 となり、等しくないと状態：6 となる。

istat：6　引張側で勾配は 0 となる。境界値としては、引張側はなしで、圧縮側はひずみが str_2 になるまでである。ひずみが str_2 を超えると圧縮側では、弾性状態：0 に変化する。

istat：7　圧縮側で勾配は 0 で応力値は Q_4 となる。圧縮側の境界値はなく、増分ひずみが反転して引張増分ひずみが生じると、弾性状態：0 となる。この状態を経験すると引張側の剛性はないことになる。

以上で、各状態の説明を終えるが、ここで、この履歴ルールを表現するサブルーチンを具体的に見ておこう。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE / Concrete (Ver.3.01)
C     ―――――――――――――――――――――――――――――――――
C     ●   コンクリート履歴モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine Concrete(AK,istat,AK_1,AK_2,AK_3,AK_4,Q_1,Q_2,Q_3,Q_4,
     +                    de,P,P1,P2,IPRET,STR,STR_1,STR_2,IPRE_C)
      implicit real*8(A-H,O-Z)
c―――――――――――――――――――――――――――――――――――――――――――――――
c     AK              :接線剛性
c     istat           :現在の状態(Work)
c     AK_1            :圧縮＆引張　第一勾配
c     AK_2            :圧縮第二勾配
c     AK_3            :圧縮第三勾配
c     AK_4            :引張第二勾配
c     Q_1             :引張強度
c     Q_2             :圧縮第一折れ点の応力
c     Q_3             :圧縮強度
c     Q_4             :圧縮流れ点
c     de              :増分ひずみ
c     P               :現在の応力(Work)
c     P1              :istat=4 における圧縮に切り替わったときの反曲点(Work)
c     P2              :istat=1,2 における引張に切り替わったときの反曲点(Work)
c     IPRET           :過去の引張履歴(破壊したら 1,流れたら 2)(Work)
c     IPRE_C          :過去の圧縮履歴(流れたら 1)(Work)
c     STR             :現在のひずみ量(Work)
c     STR_1           :応力が最初に圧縮状態から引張状態になったときのひずみ(Work)
```

```
c    STR_2                 :最近の圧縮状態から引張状態になったときのひずみ(Work)
C————————————————————————————————————
c
   99 continue
      go to (10,100,110,120,130,140,150,160,170),istat+2                           ! 1
c
c————————————————————————◎初期節点
  10  str = 0.                                                                     ! 2
      str_1 = 0.
      str_2 = 0.
      p=0.
      AK = AK_1
      ipret= 0
      ipre_c=0
      p1 = Q_1         ! 引張側の最大耐力：引張亀裂を経験すると低下する
      p2 = Q_2         ! 圧縮側の最大耐力：圧壊を経験すると低下する
      istat=0
      if(de.gt.0.) istat=3                                                         ! 3
      goto 99
c————————————————————————◎istat = 0
  100 p = AK*de + p                                                                ! 4
      str = str + de
      if(de.lt.0.) then                    ! 圧縮側                                ! 5
      if(p.ge.p2) return                   ! 圧縮側耐力より小                      ! 6
      if(ipre_c.eq.0)then                                                          ! 7
c————————————————————————★istat = 0 から istat=1 へ
      istat=1
      de_1= (p - p2)/AK                                                            ! 8
      AK=AK_2                                                                      ! 9
      p = p2 + AK*de_1
      elseif(ipre_c.eq.1)then                                                      ! 10
c————————————————————————★istat = 0 から istat=2 へ
      istat=2                                                                      ! 11
      de_1= (p - p2)/AK                                                            ! 12
      AK=AK_3
      p = p2 + AK*de_1
      elseif(ipre_c.eq.2)then                                                      ! 13
c————————————————————————★istat = 0 から istat=7 へ
      istat=7                                                                      ! 14
      AK =  AK_1*0.000000001
      p = Q_4
      endif                                                                        ! 15
      else                                                                         ! 16
      if(p.lt.0.) return                                                           ! 17
      if(ipret.eq.0)then                                                           ! 18
c————————————————————————★istat = 0 から istat=3 へ
      istat=3                                                                      ! 19
      de_1 = p / AK          !  pは引張応力
      str_1 = str - de_1                                                           ! 20
      str_2 = str_1
      elseif(ipret.eq.1) then                                                      ! 21
      if(str .gt. str_1)then                                                       ! 22
c————————————————————————★istat = 0 から istat=5 へ
      istat=5
```

```
      de_1 = p / AK                                                    ! 23
      de_2 =( p1 - Q_1 ) / AK_4                                        ! 24
      de_3 =Q_1 / AK_1                                                 ! 25
      AK = p1 / ( de_2 + de_3 )                                        ! 26
      p = AK*de_1                                                      ! 27
      else                                                             ! 28
c-----------------------------★istat = 0 から istat=6 へ
      istat=6
      de_1 = p / AK                                                    ! 29
      AK = AK_1*0.000000001
      p = 0.
      str_2 = str - de_1
      endif
      elseif(ipret.eq.2)then                                           ! 30
c-----------------------------★istat = 0 から istat=6 へ
      istat=6
      de_1 = p / AK
      AK = AK_1*0.000000001
      p = 0.
      str_2 = str - de_1
      endif
      endif
      return
c-----------------------------◎istat = 1
  110 p = AK*de + p                                                    ! 31
      str = str + de
      if(de.gt.0.) then                                                ! 32
c-----------------------------★istat =1 から istat=0 へ
      istat=0
      pw=AK*de
      p = p - pw
      str = str - de
      p2 = p
      AK=AK_1
      de=pw/ak
      str = str + de
      p = AK*de + p
      return
      elseif(p.le.Q_3)then                                             ! 33
c-----------------------------★istat =1 から istat=2 へ
      istat=2                                                          ! 34
      de_1 = (p - Q_3) / AK
      AK=AK_3
      p = Q_3 + AK*de_1
      ipre_c=1
      endif
      return
c-----------------------------◎istat = 2
  120 p = AK*de + p                                                    ! 35
      str = str + de
      if(de.gt.0.) then                                                ! 36
c-----------------------------★istat =2 から istat=0 へ
      istat=0
      p = p - AK*de                                                    ! 37
```

```
      pw=AK*de                                                    ! 38
      p2 = p                                                      ! 39
      AK=AK_1
      str = str -  de                                             ! 40
      de=-pw/ak                                                   ! 41
      str = str + de
      p = AK*de + p
c―――――――――――――――――――――――★istat =0 におけるチェック
      if(ipret.le.1) then                                         ! 42
      if(p.lt.p1) return                                          ! 43
      str = str -  de                                             ! 44
      p = -AK*de + p                                              ! 45
      pw=p1-p                                                     ! 46
      de=pw/AK
      str = str + de
      p=p1
      else                                                        ! 47
      if(p.lt.0.) return                                          ! 48
      str = str -  de                                             ! 49
      p = -AK*de + p
      pw= -p
      de=pw/ak
      str = str + de
      p=0.
      endif
      return                                                      ! 50
      elseif(p.gt.Q_4)then                                        ! 51
c―――――――――――――――――――――――――――――★istat =2 から istat=7 へ
      istat=7                                                     ! 52
      AK = AK_1*0.000000001
      p = Q_4
      ipret = 2
      ipre_c = 2
      p2 = Q_4
      endif
      return                                                      ! 53
c―――――――――――――――――――――――――――――◎istat = 3
  130 p = AK*de + p                                               ! 54
      str = str + de
      if(p.gt.Q_1) then                                           ! 55
c―――――――――――――――――――――――――――――★istat =3 から istat=4 へ
      istat=4
      de_1 = (p - Q_1) / AK
      AK=AK_4
      p = AK*de_1 + Q_1
      ipret = 1
      elseif(p.lt.0.)then                                         ! 56
c―――――――――――――――――――――――――――――★istat =3 から istat=0 へ
      istat=0
      endif
      return
c―――――――――――――――――――――――――――――◎istat = 4
  140 p = AK*de + p                                               ! 57
      str = str + de
```

```
      if(de.lt.0.) then                                          ! 58
c-----------------------------★istat =4 から istat=5 へ
      istat=5                                                    ! 59
      p = p - AK*de
      p1 = p
      ppp=((str - str_1) - de)
      if(abs(ppp).gt.0.000000000001) then                        ! 60
      AK = p / ppp
      p = AK*de + p
      else                                                       ! 61
      AK= AK_1
      p = AK*de + p
      endif
      if(p.lt.0. )then                                           ! 62
c-----------------------------★境界値の飛び越しチェック
      p = -AK*de + p
      str = str - de
      if(AK.ne.0.)then                                           ! 63
      de =-p/ak
      else                                                       ! 64
      de=-(str-str_1)
      endif
      p=0.                                                       ! 65
      str = str + de
      endif
      elseif(p.lt.0.)then                                        ! 66
c-----------------------------★istat =4 から istat=6 へ
      istat=6                                                    ! 67
      AK = AK_1*0.000000001
      p = 0.
      ipret = 2                                                  ! 68
      endif
      return
c-----------------------------◎istat = 5
  150 p = AK*de + p                                              ! 69
      str = str + de
      if(p.gt.p1) then                                           ! 70
c-----------------------------★istat =5 から istat=4 へ
      istat=4                                                    ! 71
      de_1 = (p - p1) / AK
      AK=AK_4
      p = AK*de_1 + p1
      elseif(p.lt.0.)then                                        ! 72
      if(str.lt.str_2)then
c-----------------------------★istat =5 から istat=0 へ
      istat=0                                                    ! 73
      str = str_2
      AK=AK_1
      else                                                       ! 74
c-----------------------------★istat =5 から istat=6 へ
      istat=6                                                    ! 75
      AK = AK_1*0.000000001
      p = 0.
      endif
```

```
      endif
      return
c―――――――――――――――――――――――――――◎istat = 6
  160 p = 0.                                                        ! 76
      str = str + de
      if(str.lt.str_2) then                                         ! 77
c―――――――――――――――――――――――――――★istat =6 から istat=0 へ
      istat=0
      de=str-str_2
      str = str_2
      AK=AK_1
      endif
      if(ipret.eq.1)then                                            ! 78
      if(str.lt.str_1)return                                        ! 79
c―――――――――――――――――――――――――――★istat = 6 から istat=5 へ
      istat=5                                                       ! 80
      de_1 = str - str_1
      de_2 =( p1 - Q_1 ) / AK_4
      de_3 =Q_1 / AK_1
      AK = p1 / ( de_2 + de_3 )
      p = AK*de_1
      endif
      return
c―――――――――――――――――――――――――――◎istat = 7
  170 p = Q_4                                                       ! 81
      str = str + de
      if(de.gt.0.) then                                             ! 82
c―――――――――――――――――――――――――――★istat =7 から istat=0 へ
      istat=0
      AK=AK_1
      str = str - de      ! 弾性復帰は一回無視する                  ! 83
      endif
      return
      end
```

1. 状態パラメータにしたがって処理を分類する。状態パラメータが-1のときは初期設定を行う。
2. ここからは初期設定を行う。まず各種のワーク用データをゼロクリアする。接線剛性を第一剛性とする。状態パラメータを 0 にする。
3. 増分変位が正の場合は引張応力となり、状態パラメータを 3 にする。負の場合は状態パラメータを 0 とする。
4. ここからは状態パラメータが 0 の場合の処理を行う。圧縮側の弾性部分の処理を行う。まず、増分ひずみより増分後の応力を計算する。増分後のひずみもセットする。
5. まず、増分ひずみをチェックし、以降では圧縮の場合の処理を行う。増分ひずみが引張の場合、処理は 16. に飛ぶ。
6. 増分後の応力が圧縮耐力 P2 より小さい場合は、この状態内であると

して処理を終了し、このサブルーチンから戻る。

7. 増分後の応力が圧縮耐力より大きい場合は、これ以降では状態変化に関する処理を行う。まず、過去の圧縮履歴 ipre_c をチェックして、コンクリートが圧壊していない場合は、状態パラメータを 1 にセットする。
8. 飛び越した部分のひずみを計算する。
9. 接線剛性に圧縮側第 2 勾配の剛性をセットする。また、飛び越した部分のひずみと圧縮側第 2 勾配の剛性を用いて、増分後の応力を再計算する。
10. 過去の圧縮履歴 ipre_c が 1 でコンクリートが圧壊した経験を有する場合は以下の処理を行う。
11. 状態パラメータを 2 にセットする。
12. 飛び越した部分のひずみを計算する。接線剛性に圧縮側第 3 勾配の剛性をセットする。また、飛び越した部分のひずみと圧縮側第 3 勾配の剛性を用いて、増分後の応力を再計算する。
13. 過去の圧縮履歴 ipre_c が 2 でコンクリートが完全圧壊した経験を有する場合は、以下の処理を行う。
14. 状態パラメータを 7 にセットし、接線剛性をほとんどゼロにする。また、増分後の応力は第 3 折れ点の応力とする。
15. ここで、状態パラメータが 0 で圧縮側の処理が終了する。
16. ここからは、状態パラメータが 0 で引張側の処理を行う。
17. 増分後の応力が圧縮の場合、この状態内であるとして処理を終了し、このサブルーチンより戻る。その他は以降の処理へ移る。
18. 過去の引張履歴 ipret=0 で、引張亀裂が生じていない場合は、以下の処理を行う。
19. 引張側の状態になったとして、状態パラメータを 3 にセットする。引張側のひずみ de_1 を計算する。
20. 現在のひずみから引張側にひずみ de_1 を引いて、引張応力が働き出したときのひずみを str_1 として保存する。また、初期設定として、圧縮状態で引張破壊して、応力が流れ出したときのひずみ str_2 として、str_1 をコピーする。
21. 過去の引張履歴 ipret=1 で、引張亀裂が生じている場合は、以下の処理を行う。
22. 現在のひずみが str_1 より大きい場合は状態パラメータが 5 となる。
23. 引張側のひずみ de_1 を計算する。
24. 引張側最大耐力 Q_1、過去の反転位置の応力 P1 並びに状態 4 の剛

性勾配を用いて、最大耐力点から反転位置までのひずみ de_2 を計算する。

25. 原点から引張最大耐力までのひずみ de_3 を計算する。
26. 引張応力へ入る位置から、反転する位置までの直線の傾き、つまり、状態 5 の接線剛性 AK を計算する（図 6-3 の左上の図参照)。
27. 引張方向に飛び出したひずみと接線剛性を用いて、増分後の応力を求める。
28. 現在のひずみが str_1 より小さい場合は、状態パラメータは 6 となる。
29. 引張破壊して応力が流れ出したときのひずみ str_2 を、現在のひずみから飛びだしたひずみを引いて設定する。
30. 過去の引張履歴 ipret=2 で、応力が流れた状態の場合は、以下の処理を行う。状態パラメータは 6 となる。引張破壊して流れ出したときのひずみ str_2 は、現在のひずみから飛び出したひずみを引いて設定する。ここまでで、状態：0 に関する処理が終了する。
31. ここからは、状態：1 で圧縮側第 2 勾配に関する処理を行う。増分後の応力と増分後のひずみを計算する。
32. 増分ひずみが正となると、ひずみが反転する場合であり、弾性状態である istat=0 とする。また、一旦増分後の応力とひずみを増分前の応力とひずみに戻す。反転位置の応力 P2 にこの増分前の応力をセットする。接線剛性を圧縮側の第 1 勾配の剛性をセットする。前に求めた増分応力を用いて増分ひずみを計算する。この値より増分後の応力とひずみを計算し直す。ここでサブルーチンより戻る。
33. 応力が第 2 折れ点の応力より小さい場合、この第 2 折れ点を飛び越したことになり、以下の処理を行う。
34. 状態パラメータを istat=2 とする。飛び出した部分のひずみ de_1 を計算し、接線剛性に圧縮側第 3 勾配の剛性をセットする。この飛び出した部分のひずみ de_1 と接線剛性を用いて増分後の応力を計算する。最後に圧縮崩壊を表すパラメータを ipre_c=1 とする。これで状態: 1 の処理が終了し、このサブルーチンから戻ることになる。
35. ここからは、状態 2 の処理を行う。まず、増分ひずみを用いて増分後の応力を計算する。
36. 増分ひずみが正のとき、ひずみは反転し、状態は 0 に戻る。
37. 一旦、増分前の応力を計算する。
38. 増分応力を計算する。
39. 状態 2 で、反転する位置の応力を P2 として保存する。また、接線

剛性に圧縮第１勾配の剛性をセットする。

40. 増分前のひずみに戻す。
41. 新しい接線剛性を用いて、増分ひずみを計算する。ただし、前の接線剛性と符号が逆なので、増分ひずみに負符号を付ける。この増分ひずみを用いて、増分後のひずみと応力を計算する。
42. ここでは、反転して状態 0 となるが、飛び出したひずみがあまりにも大きいとき、状態 0 の境界を越えてしまう場合がある。このような場合に備えて以下の処理を行う。まず、引張側の崩壊パラメータが 2 でない場合の処理を行う。
43. このとき、増分後の応力が境界値 P1 を超えない場合は、状態に変化なしとして、このサブルーチンより抜ける。
44. 増分後の応力が境界値 P1 を超える場合は、応力とひずみを増分前に戻す。
45. 飛び出した応力 pw を求める。
46. それに対応した増分ひずみを計算する。最後に、増分後の応力として P1 にセットする。
47. 引張側の崩壊パラメータが 2 の場合、以降の処理を行う。
48. 増分後の応力が境界値である 0 より小さいとき、このサブルーチンより戻る。
49. 飛び出した応力 pw を求め、それに対応した増分ひずみを計算する。最後に、増分後の応力として 0 をセットする。
50. ここまでで、状態 2 の反転処理を終了する。
51. 状態 2 で、増分ひずみが負の場合の処理を行う。まず、増分後の応力と境界応力 Q_4 を比較し、大きいときは状態 2 の範囲内であることよりサブルーチンから抜ける。
52. 小さいときは状態 2 の範囲から抜け、状態 7 に移る。状態パラメータを istat=7 とする。接線剛性をほとんど 0 とする。増分後の応力を Q_4 とする。また、崩壊パラメータを ipret=2 とする。また、圧縮側の崩壊パラメータも ipre_c=2 とする。最後に p2 の値を Q_4 とする。
53. これで、状態 2 の処理を終了する。
54. ここからは、状態 3 についての処理を行う。まず、増分後の応力を計算し、また、増分後のひずみも計算する。ここでの境界値応力は、Q_1 と 0 である。
55. 増分後の応力が引張側の最大耐力 Q_1 より大きい場合は、以下の処理を行う。まず、状態パラメータを istat=4 とし、飛び出したひず

みde_1を計算する。接線剛性を引張側の第2勾配の剛性をセットし、増分後の応力を再計算する。ここで、引張側の崩壊パラメータである ipret を 1 にセットする。

56. 一方、増分後の応力が 0 より小さい場合は、状態を 0 にセットする。接線剛性は、状態 3 と同じであることから変更しない。

57. ここからは、状態 4 についての処理を行う。まず、増分後の応力と増分後のひずみを計算する。引張増分ひずみに対しては、境界値応力は 0 で状態が 6 に変化する。また、増分圧縮ひずみが生じると反転して状態 5 となる。

58. 増分ひずみが負の場合、反転処理を行う。

59. まず、状態パラメータを istat=5 とする。増分前の応力を計算し、その値を P1 とする。さらに、増分前のひずみからひずみ str_1 を引き、図 6-3 の左上の図に示す ppp の値を求める。

60. 求めた ppp がゼロでない場合は、接線剛性を計算し、その値を用いて増分後の応力を求める。

61. 求めた ppp がほとんどゼロの場合は（この場合はほとんどない）接線剛性として弾性剛性の K_1 をセットする。その値を用いて増分後の応力を求める。

62. ここで計算した応力と境界応力である 0 とを比較し、増分応力が大きくて境界応力を飛び越した場合は、次の処理を行う。まず、増分前の応力とひずみを求める。

63. 次に、増分ひずみを求めるが、接線剛性 AK がゼロであるかどうかチェックし、ゼロでない場合は割り算して増分ひずみを求める。

64. 接線剛性がゼロの場合は（P1 がほとんど 0 の場合)、予備のためにこの式で増分ひずみをセットしておく。

65. 増分後の応力を 0 にセットし、同様に増分後のひずみをセットする。

66. 一方、増分ひずみが正の場合は、増分後の応力が境界応力である 0 より小さくなって飛び越した場合は、次の処理を行う。

67. 状態パラメータを istat=6 にセットする。接線剛性をほとんどゼロにする。増分後の応力を 0 にする。

68. 引張側の崩壊パラメータを ipret=2 にセットする。これで、状態４の処理は全て終了する。

69. ここからは、状態 5 についての処理を行う。まず、増分後の応力を計算し、また、増分後のひずみも計算する。

70. 応力 p が P1 より大きいとき、状態は 4 に変化する。

71. 状態パラメータを istat=4 にする。次に、飛び出したひずみ de_1

を求め、接線剛性を引張側の第 2 勾配の剛性にセットする。この剛性と飛び出したひずみを用いて、応力を再計算する。

72. 一方、応力が 0 より小さくなる場合については、以下の処理を行う。この状態は、増分ひずみが圧縮で、しかも、状態 5 の境界値である応力 0 を超えた場合に相当する。
73. 計算した増分後のひずみが str_2 を飛び越えて小さくなる場合は、以下の処理を行い、まだ、飛び越えない場合は、処理 75 へ制御が移る。ここでは状態 6 となる。
74. 状態パラメータを istat=0 とする。増分後のひずみを str_2 に変更する。また、接線剛性を圧縮の第 1 勾配の剛性にセットする。
75. 状態パラメータを istat=6 とする。増分後の応力を 0 に変更する。また、接線剛性をほとんどゼロにセットする。
76. ここからは、引張崩壊状態：6 の処理を行う。まず、応力を 0 とする。次に、増分後のひずみを計算する。また、ここでの境界値は、増分ひずみが引張状態で、しかも、境界ひずみ str_2 を飛び越して小さくなったときである。
77. 増分後のひずみが境界値ひずみ str_2 より小さくなったときは、再度圧縮剛性が発生するため、状態を istat=0 に戻す。増分後のひずみを str_2 とする。これは、増分ひずみ de が非常に大きくなって、応力とひずみの関係を図化したとき、ユーザーに誤解を与えないためである。接線剛性を圧縮側の第 1 勾配の剛性を用いる。
78. ここからの処理は、増分ひずみが引張に対するものであり、まず、引張側の崩壊パラメータが ipret=1 の場合、つまり、引張剛性が多少残っている場合に以下の処理を行う。ただし、ipret=2 の場合は、このサブルーチンより戻る。
79. 増分後のひずみが、引張開始点のひずみ str_1 より小さい場合は、サブルーチンから戻る。このひずみを飛び越すと引張側の剛性が発生して以下の処理を行うことになる。
80. 状態を 5 とし、飛び越したひずみ de_1 を計算する。次に、図 6-3 の左上の図を参考にして、ひずみ de_2、de_3 を求め、それらを利用して、状態 5 における剛性を計算する。さらに、増分後の応力も計算する。
81. ここからは、圧縮崩壊状態：7 の処理を行う。まず、この状態の応力は一定であり、Q_4 である。また、増分後のひずみを計算する。
82. 増分ひずみが負であれば、ひずみが進行したとして、サブルーチンを抜ける。また、正であれば、ひずみが反転したことより、状態：0

に変化する。接線剛性に圧縮側第１勾配の剛性をセットする。

83. 弾性復帰する場合、この一回の増分ひずみを足しこむことを取りやめる。これは、計算誤差によって状態 0 と状態：7 が交互に現れるとき、図として表示する場合、ひずみの履歴が微細な振動状態を呈するように見えるのを防ぐためである。

6.3 せん断型モデルの履歴

本節では、せん断型モデルの履歴特性について説明する。せん断型モデルの履歴に関連するサブルーチンは、初期設定と材料非線形性のチェックに関連する２つのサブルーチンである。ここでも、この履歴特性を管理するために階層構造を用いている。まず、この２つのサブルーチンを示すが、ここでは履歴に関連する部分についてのみ示し、他の部分は省く。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /Cal_lin_stiff_M2
C       ―――――――――――――――――――――――――――――――――
C     ●   Model_No.2 3次元せん断弾塑性モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine Cal_lin_stiff_M2(Member,Element,ak_linear)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / member_s2    / Member
      record / element_s2   / Element
      dimension ak_linear(12,12)
      data  BI_MODEL_NUMBER/11/
      data  RO_MODEL_NUMBER/12/
C     ―――――――――――――――――――――――――――――――――
      do i=1,12                                                    ! 1
      do j=1,12
        ak_linear(j,i) = 0.0
      end do
      end do
      ak=Element.Aku                                               ! 2
      ak_linear(1,1)= ak
      ak_linear(1,7)=-ak
      ak_linear(7,7)= ak
      ak_linear(7,1)=-ak
      if (Element.nm_type.eq.BI_MODEL_NUMBER) then                 ! 3
        ak=Element.AK_1*Member.alength
      else if(Element.nm_type.eq.RO_MODEL_NUMBER ) then            ! 4
        ak=Element.AK_1/Member.alength
      else                                                         ! 5
        ak=Element.AK_1
      endif
      ak_linear(2,2)= ak                                           ! 6
      ak_linear(2,8)=-ak
      ak_linear(8,8)= ak
      ak_linear(8,2)=-ak
      ak_linear(3,3)= ak
      ak_linear(3,9)=-ak
      ak_linear(9,9)= ak
      ak_linear(9,3)=-ak
c―――――――――――――――――――――――――――――★履歴特性の初期設定
      Member.AKv_tan=ak          ! Member.AKv_tan=Element.AK_1     ! 7
      Member.AKw_tan=ak          ! Member.AKw_tan=Element.AK_1
      Member.istat_v=0
      Member.istat_w=0
```

```
      return
      end
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     ―――――――――――――――――――――――――――――――――――
      No_rireki=Element.nm_type                                    ! 8
      if(No_rireki/10.eq.0) then
      goto(5,10,20,30,40,50,60),No_rireki+1                        ! 9
    5 continue
c―――――――――――――――――――――――――――――――★規定モデル：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)                  ! 10
      goto 999
   10 continue
c―――――――――――――――――――――――――――――――★トリリニア：Nomal
      call Mesing_TriLiner(Member,Element,vv,vpp)                  ! 11
      goto 999
   20 continue
c―――――――――――――――――――――――――――――――★トリリニア：最大点指向型
      call DirecMax_TriLiner(Member,Element,vv,vpp)                ! 12
      goto 999
   30 continue
c―――――――――――――――――――――――――――――――★トリリニア：武田モデル
      call Takeda_TriLiner(Member,Element,vv,vpp)                  ! 13
      goto 999
   40 continue
c―――――――――――――――――――――――――――――――★
      goto 999
   50 continue
c―――――――――――――――――――――――――――――――★
      goto 999
   60 continue
c―――――――――――――――――――――――――――――――★履歴モデル
      goto 999
      elseif(No_rireki/10.eq.1) then
      goto(101,102),No_rireki - 10
  101 continue
c―――――――――――――――――――――――――――――――★修正バイリニアモデル
      mro=Element.n_section(1)
      call Modify_Bi_Liner1(Member,Element,RO_work(mro),vv,vpp)    ! 14
      goto 999
  102 continue
```

```
c―――――――――――――――――――――――――――――――★修正 RO モデル
      mro=Element.n_section(1)
      call Modify_RO1(Member,Element,RO_work(mro),vv,vpp)                    ! 15
      goto 999
      endif
  999 continue
      return
      end
```

1. このサブルーチンは、せん断型モデル部材の線形剛性行列を求めるもので、ここで、このモデルに含まれる全履歴モデル共通の初期設定を行う。各履歴モデルで必要となる初期設定は他の部分で行う。これについては、直ぐ後で説明する。まず、線形剛性行列を計算するために、剛性行列をゼロクリアする。
2. 軸方向剛性を Element 構造体から取得する。せん断型モデルは一般には軸方向の変形場を含まないが、ここでは、平面骨組みに組み込まれることを考慮して軸方向剛性をセットする。この軸方向剛性を剛性行列の適切な位置にセットする。
3. せん断剛性を構造体より取得する。ここでは、修正バイリニアモデルの線形剛性を取り出す。
4. ここでは、修正 RO モデルの線形剛性を取り出す。
5. 一般のせん断型の線形剛性を取得し、ak にセットする。
6. 線形のせん断剛性を y 方向と z 方向のバネ定数として剛性行列に配置する。現在は、入力仕様によって、y 方向と z 方向の線形剛性並びに履歴特性が同じとしているが、せん断型の立体振動を行うためには、異なったデータをセットする必要がある。いずれ、入力仕様を変更して対処することになる。
7. ここで、履歴チェックのための初期設定を行う。接線剛性に線形のせん断剛性をセットする。また、y 方向と z 方向の状態パラメータを 0（弾性）とする。
8. このサブルーチンでは、せん断型の履歴モデルの弾塑性チェックを行い、その接線剛性を求める。まず、最初に部材の履歴モデル番号を取得する。
9. その番号にしたがって履歴モデルに関係するサブルーチンをコールする。
10. ここは、履歴モデル番号：0 番であり、既定モデルとなっている。現在の仕様では、武田モデルとなっている。
11. 履歴モデル番号：1 番は、トリリニア型の履歴特性となっている。

12. 履歴モデル番号：2 番は、最大点指向型トリリニア型の履歴特性となっている。
13. 履歴モデル番号：3 番は、武田モデルの履歴特性となっている。
14. 履歴モデル番号：11 番は、修正バイリニア型の履歴特性となっている。
15. 履歴モデル番号：12 番は、修正 RO モデルの履歴特性となっている。

6.3.1 最大点指向型トリリニアの履歴

本節では、図 6-8 に示すせん断型モデルの履歴ルールである最大点指向型トリリニア履歴モデルについて説明する。

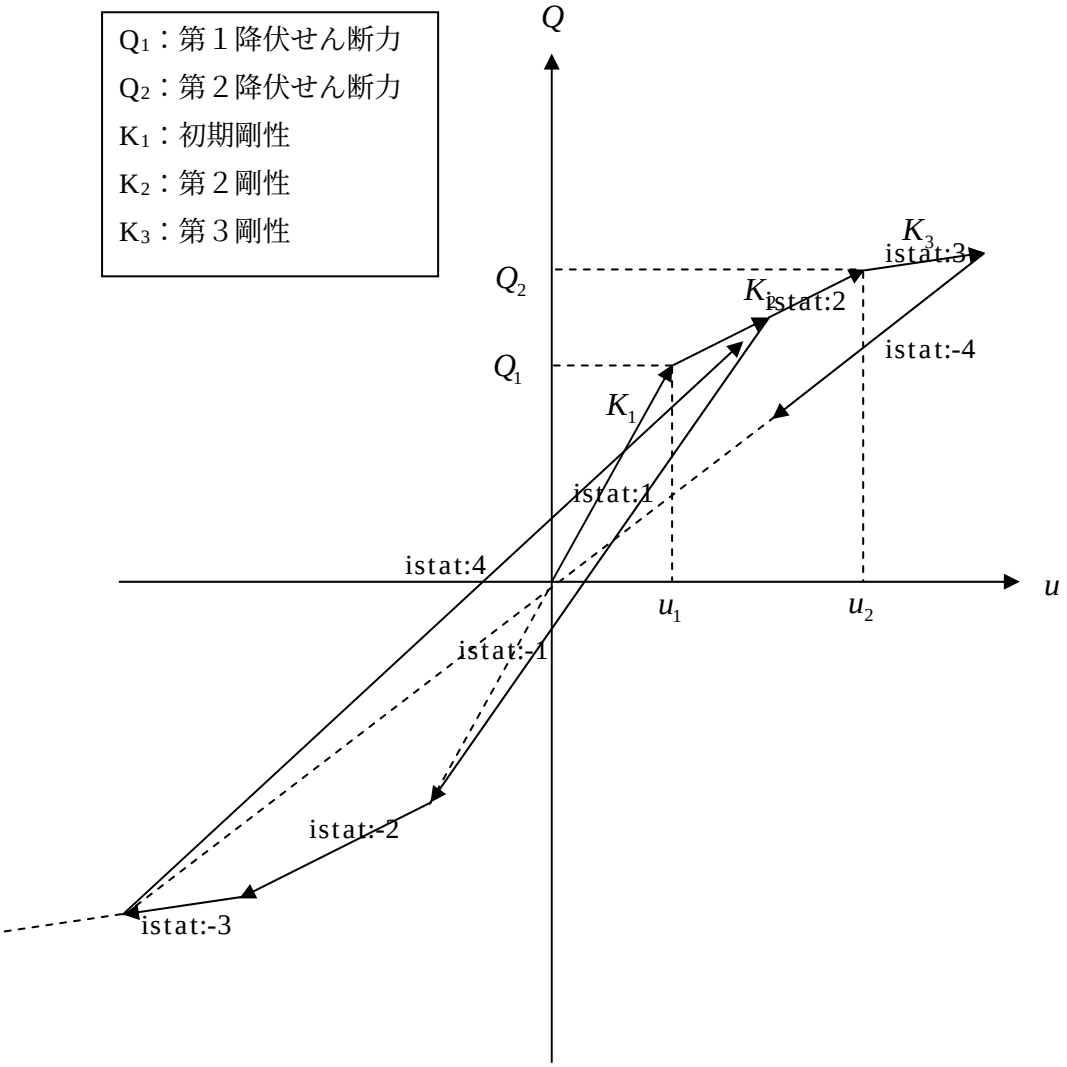

図 6-8 最大点指向型モデルの履歴特性

最大点指向型の履歴ルールを次のようにまとめる。

1. 骨格曲線はトリリニア型である。
2. 初めて一方の変位が最大変位 u_1 を超えて折り返した場合は、反対側の第１降伏点を目指す。
3. 変位が共に最大変位 u_1 を超えて折り返した場合は、反対側の最大変位点を目指す。

この履歴ルールにしたがって次のサブルーチンが設計されている。このサブルーチン DirecMax_TriLiner()を具体的に示そう。このサブルーチンは、前節のファイバーの弾塑性チェックプログラムと、かなり異なった方法で作られている。特に、せん断力を増分型で求めていないのが特徴である。また、状態パラメータは、正方向に向かうときは正に、負方向に向かうときは負としている。骨格曲線を形成する 3 つの直線で、状態パラメータは、第１勾配を１に、第 2 勾配を 2 に、第 3 勾配を 3 に、また、最大点に向かう直線を 4 とする。良く読んで理解されたい。

```
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /DirecMax_TriLiner
C       ―――――――――――――――――――――――――――――――――
C     ●   Direc. Max TriLiner 履歴モデル
C     ―――――――――――――――――――――――――――――――――
      subroutine DirecMax_TriLiner(Member,Element,vv,vpp)
      implicit real*8(V)
      include "submain.h"
      record / member_s2_DirecMax / Member
      record / element_s2         / Element
      integer           i          ! カウンター
      dimension vv(12)   ! 部材座標系増分変位
      dimension vpp(12)  ! 部材座標系直前変位
      real*8            dU               ! 現在の相対増分変位
      real*8            Up               ! 直前の相対変位
C     ―――――――――――――――――――――――――――――――――
c
c     部材
      structure / member_s2_DirecMax /
      integer  nm_element         !  要素番号
      integer  element_type       !  要素タイプ
      integer  n_model            !  モデルの入れ物番号
      integer  n_model_type       !  モデル別の通し番号
      integer  n_element_type     !  要素タイプ別番号
      integer  analysis_3D        !  解析型 0:3D 1:2D(x-z) 2:2D(y-z)
      integer  nm_so              !  部材の層番号
      integer  nm_dll_element     !  DLL を用いた要素か（０；システム内要素、１：DLL 要素）
      integer  nm_point(2)        !  節点番号
```

```
      integer  irest(12)          !  部材両端の自由度番号表
                                  !istat は履歴特性で使用（インデント 1:y 方向，2:z 方向）
      integer  istat(2)           !  履歴特性の状態(*)
      integer  nm_analysis        !  部材解析種別
      integer  nm_group           !  部材グループ
      integer  nm_local_coord(2)  !  局所座標系の有無とその回転行列の番号
      integer  nm_damp            !  部材減衰の有無とその減衰行列の番号
      real*8   alength            !  長さ
      real*8   i_rigid_length     !  i 端剛域長さ
      real*8   j_rigid_length     !  j 端剛域長さ
      real*8   i_shear_G          !  i 端せん断剛性
      real*8   j_shear_G          !  j 端せん断剛性
      real*8   rot_x              !  部材主軸の回転角度（度）
      real*8   force(12)          !  部材両端の部材端力（釣合座標系）
      real*8   stress(6)          !  部材中央の応力（部材座標系）
                                  !  AK_tan は履歴特性で使用（インデント 1:y 方向，2:z 方向）
      real*8   AK_tan(2)          !  接線剛性(*)
                                  !  以下，履歴特性で使用（インデント 1:y 方向，2:z 方向）
      real*8   Uo(2)              !  目標点の変位
      real*8   Qo(2)              !  目標点の剪断力
      real*8   Ur(2)              !  荷重反転時の目標点の変位(istat=1,4)
      real*8   Qr(2)              !  荷重反転時の目標点の剪断力(istat=1,4)
      real*8   UmaxP(2)           !  正載荷側の最大変位
      real*8   QmaxP(2)           !  正載荷側の最大剪断力
      real*8   UmaxM(2)           !  負載荷側の最大変位
      real*8   QmaxM(2)           !  負載荷側の最大剪断力
      real*8   dmw(4)             !  ダミー
      integer   d_stat(3)         !  断面の弾塑性状態　(1) i 端 (2) j 端　(3) 中央
      real*8   an_vv(10)          !  部材軸力（計算用内部変位ｖ）
      real*8   an_ww(10)          !  部材軸力（計算用内部変位w）
      end structure
C
C     ―――――――――――――――――――――――――――――――――
C       第一ステップ処理
      if(Member.istat(1).eq.0.and.Member.istat(2).eq.0) then                    ! 1
      call Initial_DirecMax_TriLiner(Element,Member)                           ! 2
      return
      endif
C     ―――――――――――――――――――――――――――――――――
C       第 n ステップ処理
      do i=1,2                                                                 ! 3
      dU=vv(7+i)-vv(1+i)                ! 現在の相対増分変位の計算
      Up=vpp(7+i)-vpp(1+i)              ! 直前の相対変位の計算
      call Cal_DirecMax_TriLiner(i, Element, Member, dU, Up)                   ! 4
      end do
      return
      end
C     ―――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE Initial_DirecMax_TriLiner
C       ―――――――――――――――――――――――――――――――――
C     ●   1 ステップの初期化
C     ―――――――――――――――――――――――――――――――――
      subroutine Initial_DirecMax_TriLiner(Element,Member)
      include "submain.h"
```

```
      record / member_s2_DirecMax / Member
      record / element_s2         / Element
      integer    i       ! カウンター
C     ―――――――――――――――――――――――――――――――――――
                                                        !エラー入力処理
      if(Element.AK_1.le.0.0) then                                    ! 5
      Element.AK_1=1.0
      endif
      if(Element.AK_2.le.0.0) then
      Element.AK_2=Element.AK_1
      endif
      if(Element.Q_2.le.Element.Q_1) then                             ! 6
      Element.Q_2=Element.Q_1
      endif
                                                        !フラグの初期化
      do i=1,2                                                        ! 7
      Member.istat(i)  = 1                          ! 状態パラメータ
      Member.AK_tan(i) = Element.AK_1               ! 接線剛性
      Member.Uo(i)     = Element.Q_1/Element.AK_1    ! 第４状態における正側の直線式の変位
      Member.Qo(i)     = Element.Q_1                 ! 第４状態における正側の直線式のせん断力
      Member.Ur(i)     =-Element.Q_1/Element.AK_1    ! 第４状態における負側の直線式の変位
      Member.Qr(i)     =-Element.Q_1                 ! 第４状態における負側の直線式の応力
      Member.UmaxP(i)  = Element.Q_1/Element.AK_1    ! 正側の最大変位点の初期設定
      Member.QmaxP(i)  = Element.Q_1                 ! 正側の最大応力点の初期設定
      Member.UmaxM(i)  =-Element.Q_1/Element.AK_1    ! 負側の最大変位点の初期設定
      Member.QmaxM(i)  =-Element.Q_1                 ! 負側の最大応力点の初期設定
      end do
      Element.U_1=Element.Q_1/Element.AK_1                            ! 8
      Element.U_2=(Element.Q_2-Element.Q_1)/Element.AK_2
     *           +Element.U_1
      return
      end
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE Cal_DirecMax_TriLiner
C       ―――――――――――――――――――――――――――――――――――
C     ●   n ステップ
C     ―――――――――――――――――――――――――――――――――――
      subroutine Cal_DirecMax_TriLiner(id,Element,Member,dU,Up)
      include "submain.h"
      record / member_s2_DirecMax / Member
      record / element_s2         / Element
      integer           id                    ! 方向 ID
      real*8            dU                    ! 増分相対変位
      real*8            Up                    ! 直前の相対変位
      real*8            Qp                    ! 直前の剪断力
      real*8            U                     ! 現在の相対変位
      real*8            Q                     ! 現在の剪断力
      real*8            tmp(10)               ! 作業領域
      integer           iflag                 ! 接線剛性変化フラグ
C     ―――――――――――――――――――――――――――――――――――
                                                        !現在の変位の計算
      U=Up+dU                                                         ! 9
                                                        !載荷方向の反転
      if( dU*real(Member.istat(id)).lt.0.0 ) then                     ! 10
```

```
        tmp(1)=Member.Uo(id)
        tmp(2)=Member.Qo(id)
        if( iabs(Member.istat(id)).eq.1.or.                              ! 11
     *      iabs(Member.istat(id)).eq.4 ) then
          Member.Uo(id)=Member.Ur(id)                                    ! 12
          Member.Qo(id)=Member.Qr(id)
          Member.Ur(id)=tmp(1)
          Member.Qr(id)=tmp(2)
          Member.istat(id)=-1*Member.istat(id)
        else                                                             ! 13
                                                          !直前の剪断力の計算
          Qp=CalQ_DirecMax_TriLiner(id,Element,Member,Up)                ! 14
          Member.Ur(id)=Up                                               ! 15
          Member.Qr(id)=Qp
          if( Member.istat(id).gt.0 ) then                               ! 16
            Member.Uo(id)=Member.UmaxM(id)
            Member.Qo(id)=Member.QmaxM(id)
            Member.istat(id)=-4
          else                                                           ! 17
            Member.Uo(id)=Member.UmaxP(id)
            Member.Qo(id)=Member.QmaxP(id)
            Member.istat(id)=4
          endif
        Member.AK_tan(id) = (Member.Qo(id) - Member.Qr(id))               ! 18
     *                    / (Member.Uo(id) - Member.Ur(id))
        endif
        return
      endif
C     ―――――――――――――――――――――――――――――――――――
                                                          !接線剛性の変化のチェック
      if( iabs(Member.istat(id)).ne.3 ) then                             ! 19
C     ―――――――――――――――――――――――――――――――――――
        if( Member.istat(id).gt.0 ) then                                 ! 20
          if( Member.Uo(id).lt.U ) then
            iflag=1
          else
            iflag=0
          endif
        else                                                             ! 21
          if( Member.Uo(id).gt.U ) then
            iflag=1
          else
            iflag=0
          endif
        endif
C     ―――――――――――――――――――――――――――――――――――
      if( iflag.eq.1 ) then                                              ! 22
        if( iabs(Member.istat(id)).eq.1 ) then                           ! 23
          Member.AK_tan(id)=Element.AK_2
          if ( Member.istat(id).gt.0 ) then                              ! 24
            Member.istat(id)= 2
            Member.Uo(id)   = Element.U_2
            Member.Qo(id)   = Element.Q_2
          Else                                                           ! 25
```

```
          Member.istat(id)=-2
          Member.Uo(id)   =-Element.U_2
          Member.Qo(id)   =-Element.Q_2
        endif
      else if( iabs(Member.istat(id)).eq.2 ) then                    ! 26
        Member.AK_tan(id)=Element.AK_3
        if ( Member.istat(id).gt.0 ) then                            ! 27
          Member.istat(id)=3
        else                                                         ! 28
          Member.istat(id)=-3
        endif
      else                                                           ! 29
        if( dabs(U).lt.dabs(Element.U_2) ) then                      ! 30
          Member.AK_tan(id)=Element.AK_2
          if ( Member.istat(id).gt.0 ) then                          ! 31
           Member.istat(id)= 2
            Member.Uo(id)   = Element.U_2
            Member.Qo(id)   = Element.Q_2
          else                                                       ! 32
            Member.istat(id)=-2
            Member.Uo(id)   =-Element.U_2
            Member.Qo(id)   =-Element.Q_2
          endif
        else                                                         ! 33
            Member.AK_tan(id)=Element.AK_3
          if ( Member.istat(id).gt.0 ) then                          ! 34
            Member.istat(id)=3
          else                                                       ! 35
            Member.istat(id)=-3
          endif
        endif
      endif
      Member.stress(1+id)                                            ! 36
     *      =CalQ_DirecMax_TriLiner(id,Element,Member,U)
      endif
      endif
C     ――――――――――――――――――――――――――――――――――――
                                                    !現在の剪断力の計算
      Q=CalQ_DirecMax_TriLiner(id,Element,Member,U)                  ! 37
                                                    !最大振幅値のチェック
      if( Member.istat(id).gt.0 ) then                               ! 38
        if( Member.UmaxP(id).lt.U ) then                             ! 39
          Member.UmaxP(id) = U
          Member.QmaxP(id) = Q
        endif
      else                                                           ! 40
        if( Member.UmaxM(id).gt.U ) then
          Member.UmaxM(id) = U
          Member.QmaxM(id) = Q
        endif
      endif
      return
      end
C     ――――――――――――――――――――――――――――――――――――
```

```
C     ●   real*8 function CalQ_DirecMax_TriLiner
C       ―――――――――――――――――――――――――――――――――
C     ●   変位 U に対応する剪断力 Q の計算
C     ―――――――――――――――――――――――――――――――――
      real*8 function CalQ_DirecMax_TriLiner(id,Element,Member,U)
      include "submain.h"
      record / member_s2_DirecMax / Member
      record / element_s2         / Element
      integer           id                    ! 方向 ID
      real*8            U                     ! 現在の相対変位
      real*8            Q                     ! 現在の剪断力
C     ―――――――――――――――――――――――――――――――――
      if( iabs(Member.istat(id)).eq.1) then                          ! 41
        Q=Element.AK_1*U
      else if( iabs(Member.istat(id)).eq.2) then                     ! 42
        if( Member.istat(id).gt.0 ) then                             ! 43
          Q=Element.AK_2*(U-Element.U_1) +Element.Q_1
        else                                                         ! 44
          Q=Element.AK_2*(U+Element.U_1) -Element.Q_1
        endif
      else if( iabs(Member.istat(id)).eq.3) then                     ! 45
        if( Member.istat(id).gt.0 ) then                             ! 46
          Q=Element.AK_3*(U-Element.U_2) +Element.Q_2
        else                                                         ! 47
          Q=Element.AK_3*(U+Element.U_2) -Element.Q_2
        endif
      else                                                           ! 48
          Q=Member.AK_tan(id)*(U-Member.Ur(id)) +Member.Qr(id)
      endif
          CalQ_DirecMax_TriLiner=Q                                   ! 49
      return
      end
```

1. 両方向の初期設定を行うかどうかチェックする。
2. せん断型モデルの履歴特性用ワーク領域を初期設定するために、サブルーチン Initial_DirecMax_TriLiner()をコールする。
3. 両方向の処理を行う。増分相対変位と増分前の相対変位をセットする。
4. 履歴チェックを行うサブルーチン Cal_DirecMax_TriLiner()をコールする。
5. このサブルーチンでは初期設定を行う。まず、初期剛性が適切でない場合、剛性の値として 1.0 を設定する。
6. 第 1 折れ点と第 2 折れ点のせん断力が逆転している場合は、第 2 折れ点のせん断力を第 1 折れ点の値とする。
7. 両方向についてワーク用構造体 Member の成分を初期設定する。
8. 図に示す第 1 折れ点の変位 u1 と第 2 折れ点の変位 u2 を計算し、構

造体 Element の成分にセットする。

9. このサブルーチンでせん断型モデルの履歴チェックを行う。まず、増分後の相対変位をセットする。
10. 状態パラメータの方向と増分変位の関係を調べ、変位が反転すると次の処理を行う。まず、正側の第 1 折れ点の座標をワーク配列 tmp にコピーする。
11. 状態パラメータが 1 もしくは 4 の場合は、次の処理を行う。
12. 正側と負側の基本座標を入れ替える。
13. 状態パラメータが 2 もしくは 3 の場合は、履歴が反転し、次の処理を行う。
14. サブルーチン Cal_DirecMax_TriLiner()を用いて、増分前のせん断力を計算する。
15. 反転位置の座標(Up,Qp)を構造体成分 Member.Ur(id)、Member.Qr(id)にセットする。
16. 増分前の状態で、どちら方向に変位が進んでいたかチェックする。ここでは、正方向に進んでいたので、状態パラメータを-4 にセットする。また、直線式の相手側の座標を、負側の最大点の座標としてセットする。
17. ここでは、負方向に進んでいたので、状態パラメータを 4 にセットする。また、直線式の相手側の座標を、正側の最大点の座標としてセットする。
18. 骨格曲線より反転したため、最大点を指向する直線式の傾きを計算し、その値を接線剛性とする。その接線剛性の値を構造体成分 Member.AK_tan(id)に保存する。これで、骨格曲線より反転した処理を終了し、このサブルーチンより戻る。
19. ここからは、骨格曲線からの反転処理以外の処理について行う。したがって、状態パラメータの方向と増分変位の方向は同じである。まず、状態パラメータが 3 以外の場合、つまり、1、2、4 の場合について次の処理を行う。状態パラメータが 3 の場合は、境界がないため、境界を越えたかどうかのチェックを必要としない。
20. まず、変位の進行方向が正の場合について処理を行う。増分後の変位がその状態の境界変位 Member.Uo(id)より大きい場合、状態を変更しなければならないので、変更フラグを iflag=1 とする。変位が境界を超えない場合は、iflag=0 とする。
21. ここからは、変位の進行方向が負の場合について処理を行う。増分後の変位がその状態の境界変位 Member.Uo(id)より小さい場合、状態を変更しなければならないので、変更フラグを iflag=1 とする。変

位が境界を超えない場合は、iflag=0 とする。
22. 状態変更フラグが 1 の場合では、ここ以降の処理で、接線剛性や状態パラメータの変更を行う。
23. 状態パラメータが 1 か -1 の場合、まず、接線剛性を第 2 勾配の剛性をセットする。
24. 次に、変位の進行方向を調査し、正方向の場合は状態パラメータを 2 にセットし、境界座標として第 2 折れ点の座標をセットする。
25. 変位の進行方向が負方向の場合、状態パラメータを-2 にセットし、境界座標として負の第 2 折れ点の座標をセットする。
26. 状態パラメータが 2 か -2 の場合、まず、接線剛性を第 3 勾配の剛性をセットする。
27. 次に、変位の進行方向を調査し、正方向の場合は状態パラメータを 3 にセットする。
28. 変位の進行方向が負方向の場合､状態パラメータを-3 にセットする。
29. ここでは、状態パラメータの絶対値が 1 か 4 の場合についてチェックする。
30. 変位 U が第 2 折れ点 Element.U_2 より小さい場合、状態変更フラグが 1 であるので、まず、接線剛性に第 2 勾配の剛性をセットする。
31. 変位の進行方向が正の場合、状態パラメータを 2 にセットし、境界座標として第 3 折れ点の座標をセットする。
32. 変位の進行方向が負方向の場合、状態パラメータを-2 にセットし、境界座標として、負の第 3 折れ点の座標をセットする。
33. 変位 U が第 2 折れ点 Element.U_2 より大きい場合、状態変更フラグが 1 であるので、まず、接線剛性に第 3 勾配の剛性をセットする。
34. 変位の進行方向が正の場合、状態パラメータを 3 にセットし、境界座標は必要としないので設定しない。
35. 変位の進行方向が負の場合、状態パラメータを-3 にセットし、境界座標は必要としないので設定しない。
36. サブルーチン CalQ_DirecMax_TriLiner()を用いて、増分後のせん断力を求め、構造体の応力にセットする。ここで、状態変更フラグが 1 の場合の処理が終了する。
37. 最大変位と最大せん断力をセットするために、現時点のせん断力を求める。
38. 変位の進行方向をチェックする。
39. 変位が正方向の変位より、大きい場合は最大値をセットする。
40. 変位が負方向の変位より、小さい場合は最小値をセットする。

41. このサブルーチン CalQ_DirecMax_TriLiner()では、状態パラメータにしたがって増分後のせん断力を計算する（図 6-8 参照)。状態パラメータが１の場合は、簡単で、線形剛性に変位を掛けて求める。
42. 状態パラメータが絶対値で 2 の場合の処理を行う。
43. 状態パラメータが 2 の場合、正側第 2 勾配上のせん断力 Q を計算する。
44. 状態パラメータが-2 の場合、負側第 2 勾配上のせん断力 Q を計算する。
45. 状態パラメータが絶対値で 3 の場合の処理を行う。
46. 状態パラメータが 3 の場合、正側第 3 勾配上のせん断力 Q を計算する。
47. 状態パラメータが-3 の場合、負側第 3 勾配上のせん断力 Q を計算する。
48. 状態パラメータが-4 の場合、図に示すような状態４の直線を決定する式からせん断力 Q を求める。
49. 求めたせん断力 Q を関数名 CalQ_DirecMax_TriLiner にセットする。

これで、履歴特性を表す代表的なサブルーチンの説明を終えるが、他の履歴については、付録に収録されているサブルーチンを見て、学習されたい。無論、ここで示した手法だけがこの履歴ルールを実現するわけではない。読者も他の手法で書いてみるとより理解が深まることと思う。是非、試されたい。また、他の履歴ルールを実現し、この SPACE に組み込みたい場合は、第 9 章を参考にして実行されると良い。