5　剛性方程式の解法

5.1　一般的な剛性方程式の解法

全体剛性マトリックスが求まれば，各節点の変位と節点に作用する節点力との関係が次式のように表される。

$$[K]\{U\}=\{F\} \tag{5.1}$$

ここに，$[K]$は全体剛性マトリックス，$\{U\}$は節点変位ベクトル，$\{F\}$は節点力ベクトルである。

しかしながら，(5.1)式はこのままでは解けない。なぜならば，(5.1)式には境界条件が考慮されていないからである（剛体変位モードが含まれる）。すなわち，幾何学的境界上にある節点では変位が規定され，力学的境界上にある節点では節点力が規定される。そこで，(5.1)式の行と列を並べ替えて，次のように表す。

$$\begin{bmatrix}[K_{uu}] & [K_{u\bar{u}}]\\ [K_{\bar{u}u}] & [K_{\bar{u}\bar{u}}]\end{bmatrix}\begin{Bmatrix}\{u\}\\ \{\bar{u}\}\end{Bmatrix}=\begin{Bmatrix}\{\bar{f}\}\\ \{f\}\end{Bmatrix} \tag{5.2}$$

ここで，バー付きのベクトルは既知量，バーが付いていないものは未知量である。

そして，(5.2)式は，まず，上側の式から，未知の節点変位を求める。すなわち，

$$[K_{uu}]\{u\}=\{\bar{f}\}-[K_{u\bar{u}}]\{\bar{u}\} \tag{5.3}$$

そして，求まった節点変位を用いれば，(5.2)式の下側の式により未知の反力が求まる。

$$\{f\}=[K_{\bar{u}u}]\{u\}+[K_{\bar{u}\bar{u}}]\{\bar{u}\} \tag{5.4}$$

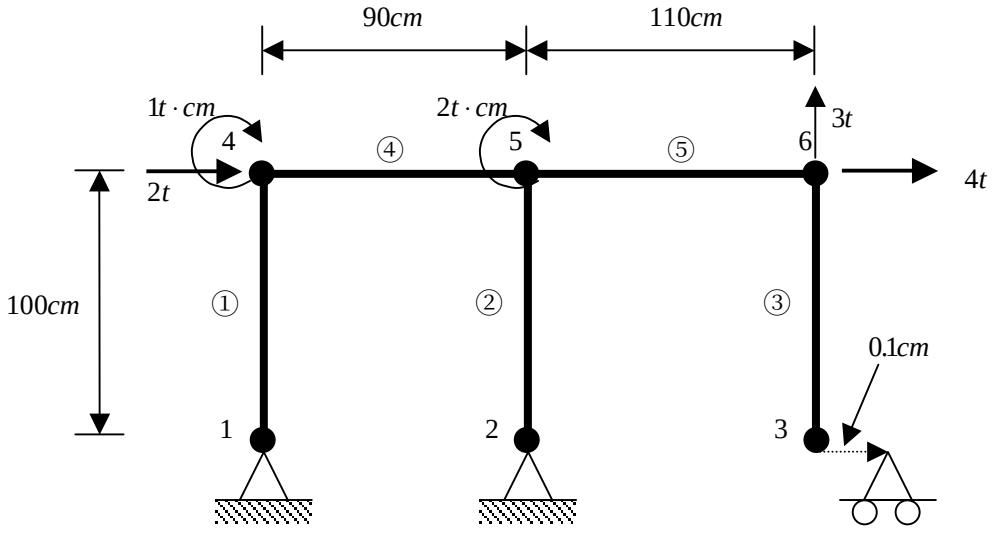

図 5.1　骨組有限要素モデルと境界条件

ここで，図 5.1 に示すように，骨組のモデルに境界条件と節点力が設定された問題を考えよう。

ただし，節点 3 では水平方向に強制変位を与えるものとする。この場合の変位ベクトル$\{U\}$は次のようになる。

$$\{U\}^T=\{u_1\quad v_1\quad \theta_1\quad u_2\quad v_2\quad \theta_2\quad u_3\quad v_3\quad \theta_3\quad u_4\quad v_4\quad \theta_4\quad u_5\quad v_5\quad \theta_5\quad u_6\quad v_6\quad \theta_6\} \tag{5.5}$$

この変位ベクトルを, 変位が規定されているものと, そうでないものに分けると次のようになる。

$$\begin{aligned}\{\bar{\delta}\}^T&=\{u_1\quad v_1\quad u_2\quad v_2\quad u_3\quad v_3\}=\{0\quad 0\quad 0\quad 0\quad 0.1\quad 0\}\\ \{\delta\}^T&=\{\theta_1\quad \theta_2\quad \theta_3\quad u_4\quad v_4\quad \theta_4\quad u_5\quad v_5\quad \theta_5\quad u_6\quad v_6\quad \theta_6\}\end{aligned} \tag{5.6}$$

これに対応する外力は次のようになる。

$$\begin{aligned}\{f\}^T&=\{f_{x1}\quad f_{y1}\quad f_{x2}\quad f_{y2}\quad f_{x3}\quad f_{y3}\}\\ \{\bar{f}\}^T&=\{0\quad 0\quad 0\quad 2\quad 0\quad 1\quad 0\quad 0\quad 2\quad 4\quad 3\quad 0\}\end{aligned} \tag{5.7}$$

この時，(5.3)式の$[K_{uu}]$は 12×12，$[K_{u\bar{u}}]$は 12×6 のマトリックスとなり，(5.4)式の$[K_{\bar{u}u}]$は，6×12，$[K_{\bar{u}\bar{u}}]$は 6×6 のマトリックスとなる。

以上の計算を一般的に行うには，以下のようにプログラムすればよい。

```
      Parameter(mnod=1000,mnel=1000,mnnd=2,mndg=3,mnmt=100)
      Dimension ngeobc(mndg),igeobc(mndg,mnod),vgeobc(mndg,mnod)
     1         ,nforce(mndg),iforce(mndg,mnod),vforce(mndg,mnod)
     1         ,SKGaa(mnod*mndg,mnod*mndg),SKGab(mnod*mndg,mnod*mndg)
     1         ,SKGba(mnod*mndg,mnod*mndg),SKGbb(mnod*mndg,mnod*mndg)
     1         ,D(mnod*mndg),Da(mnod*mndg),Db(mnod*mndg)
     1         ,F(mnod*mndg),Fa(mnod*mndg),Fb(mnod*mndg)
     1         ,icrs(mnod*mndg)
c
      nod  = 6                  !節点数
      ndg  = 3                  !一節点の自由度数
      ndgt = ndg*nod            !全自由度数
c
c 各自由度の変位規定節点の数
      ngeobc(1) = 3             ! x 方向の変位規定節点数
      ngeobc(2) = 3             ! y 方向の変位規定節点数
      ngeobc(3) = 0             ! z 軸回転変位規定節点数
c 各自由度の変位規定節点番号
      igeobc(1,1) = 1           ! x 方向の変位規定節点番号
      igeobc(1,2) = 2           ! x 方向の変位規定節点番号
      igeobc(1,3) = 3           ! x 方向の変位規定節点番号
      igeobc(2,1) = 1           ! y 方向の変位規定節点番号
      igeobc(2,2) = 2           ! y 方向の変位規定節点番号
      igeobc(2,3) = 3           ! y 方向の変位規定節点番号
c 各自由度の変位規定節点の変位値
      vgeobc(1,1) = 0.d0        ! x 方向の変位規定節点の変位値
      vgeobc(1,2) = 0.d0        ! x 方向の変位規定節点の変位値
      vgeobc(1,3) = 0.1d0       ! x 方向の変位規定節点の変位値
      vgeobc(2,1) = 0.d0        ! y 方向の変位規定節点の変位値
      vgeobc(2,2) = 0.d0        ! y 方向の変位規定節点の変位値
      vgeobc(2,3) = 0.d0        ! y 方向の変位規定節点の変位値
c
c 各自由度の 0 以外の外力の作用する節点数
      nforce(1) = 2             ! x 方向の外力が作用する節点数
      nforce(2) = 1             ! y 方向の外力が作用する節点数
```

```
      nforce(3) = 2              ! z 軸回転の外力が作用する節点数
c 各自由度の外力が作用する節点番号
      iforce(1,1) = 4            ! x 方向の外力が作用する節点番号
      iforce(1,2) = 6            ! x 方向の外力が作用する節点番号
      iforce(2,1) = 6            ! y 方向の外力が作用する節点番号
      iforce(3,1) = 4            ! z 軸回転の外力が作用する節点番号
      iforce(3,2) = 5            ! z 軸回転の外力が作用する節点番号
c 各自由度の外力の作用する節点の外力値
      vforce(1,1) = 2.d0         ! x 方向の外力が作用する節点の外力値
      vforce(1,2) = 4.d0         ! x 方向の外力が作用する節点の外力値
      vforce(2,1) = 3.d0         ! y 方向の外力が作用する節点の外力値
      vforce(3,1) = 1.d0         ! z 軸回転の外力が作用する節点の外力値
      vforce(3,2) = 2.d0         ! z 軸回転の外力が作用する節点の外力値
c
c 全自由度の番号付けと変位ベクトルの 0 クリアー
      do 20 i = 1,ndgt
      icrs(i) = i                !自由度番号
   20 D(i) = 0.d0                !変位ベクトル{U}
c 変位規定自由度の認識と変位値の代入
      do 200 i = 1,ndg
      nc = ngeobc(i)
      if( nc.eq.0 ) go to 200
      do 210 j = 1,nc
      n = ndg*( igeobc(i,j)-1 ) + i
      icrs(n) = - icrs(n)        !変位が規定される自由度番号はマイナスにする
      D(n) = vgeobc(i,j)         !変位ベクトルに規定変位値を代入
  210 continue
  200 continue
c
c 節点力ベクトルへの節点力値の代入
      do 30 i = 1,ndgt
   30 F(i) = 0.d0                !節点力ベクトル{F}
      do 300 i = 1,ndg
      nc = nforce(i)
      if( nc.eq.0 ) go to 300
      do 310 j = 1,nc
      n = ndg*( iforce(i,j)-1 ) + i
      F(n) = vforce(i,j)         !節点力値の代入
  310 continue
  300 continue
c
c 全体剛性マトリックスの分解
      ia = 0
      ib = 0
      do 400 i = 1,ndgt
      is = icrs(i)
      if( is.lt.0 ) to go 450
      ia = ia + 1
      ja = 0
      jb = 0
      do 410 j = 1,ndgt
      js = icrs(j)
      if( js.gt.0 ) then
       ja = ja + 1
       SKGaa(ia,ja) = SKG(i,j)  !マトリックス[K_uu]
      else
       jb = jb + 1
       SKGab(ia,jb) = SKG(i,j)  !マトリックス[K_uū]
      end if
  410 continue
      go to 400
  450 ib = ib +1
      ja = 0
      jb = 0
      do 420 j = 1,ndgt
      js = icrs(j)
      if( js.gt.0 ) then
       ja = ja + 1
```

```
       SKGba(ib,ja) = SKG(i,j)  !マトリックス[K_ūu]
      else
       jb = jb + 1
       SKGbb(ib,jb) = SKG(i,j)  !マトリックス[K_ūū]
      end if
  420 continue
  400 continue
c
c 変位ベクトルと節点力ベクトルの分解
      ia = 0
      ib = 0
      do 500 i = 1,ndgt
      is = icrs(i)
      if( is.gt.0 ) then
       ia = ia + 1
       Fa(ia) = F(i)                    !既知節点力ベクトル{f̄}
      else
       ib = ib + 1
       Db(ib) = D(i)                    !既知変位ベクトル{ū}
      end if
  500 continue
c 右辺ベクトルの計算
      na = ia                           !未知変位の自由度数
      nb = ib                           !未知反力の自由度数
      do 600 i = 1,na
      s = 0.d0
      do 610 j = 1,nb
  610 s = s + SKGab(i,j)*Db(j)  ![K_uū]{ū}
      Fa(i) = Fa(i) - s                 !{f̄}-[K_uū]{ū}
  600 continue
c
c 連立方程式の解法
      call solve(SKGaa,Fa,na,mnod*mndg)  !{u}=[K_uu]^-1({f̄}-[K_uū]{ū})
      do 650 i = 1,na
  650 Da(i) = Fa(i)
c
c 反力の計算
      do 700 i = 1,nb
      Fb(i) = 0.d0
      do 710 j = 1,na
  710 Fb(i) = Fb(i) + SKGba(i,j)*Da(j)          ![K_ūu]{u}
      do 720 j = 1,nb
  720 Fb(i) = Fb(i) + SKGbb(i,j)*Db(i)          !{f}=[K_ūu]{u}+[K_ūū]{ū}
  700 continue
      stop
      end
c
      SUBROUTINE SOLVE(A,B,N,ND)
      DIMENSION A(ND,1),B(ND)
      N1 = N-1
      IF( N1.NE.0 ) GO TO 10
      B(1) = B(1)/A(1,1)
      GO TO 200
   10 DO 100 K = 1,N1
      K1 = K+1
      C = A(K,K)
      DO 1 J = K1,N
    1 A(K,J) = A(K,J)/C
      B(K) = B(K)/C
      DO 2 I = K1,N
      C = A(I,K)
      DO 3 J = K1,N
    3 A(I,J) = A(I,J) - C*A(K,J)
    2 B(I) = B(I) - C*B(K)
  100 CONTINUE
      B(N) = B(N)/A(N,N)
      DO 4 L = 1,N1
      K = N-L
```

```
      K1 = K+1
      DO 5 J = K1,N
    5 B(K) = B(K) - A(K,J)*B(J)
    4 CONTINUE
  200 RETURN
      END
```

5.2 計算効率の良い剛性方程式の解法

5.2.1 バンド幅の縮小

図 5.1 のような節点番号を付した場合，すでに示したように全体剛性マトリックスの成分は図 5.2 のように配置される。しかし，節点番号の付け方を工夫すると，マトリックスの成分は，より対角に近い部分に集まってくる。図に示す対角線と一番外側にある線との間の列数をバンド幅と呼ぶが, このバンド幅が小さくなるほどコンピュータで必要とされるメモリーを節約でき, また, 0 の掛け算を行う回数が少なくなるため計算効率が高まる。

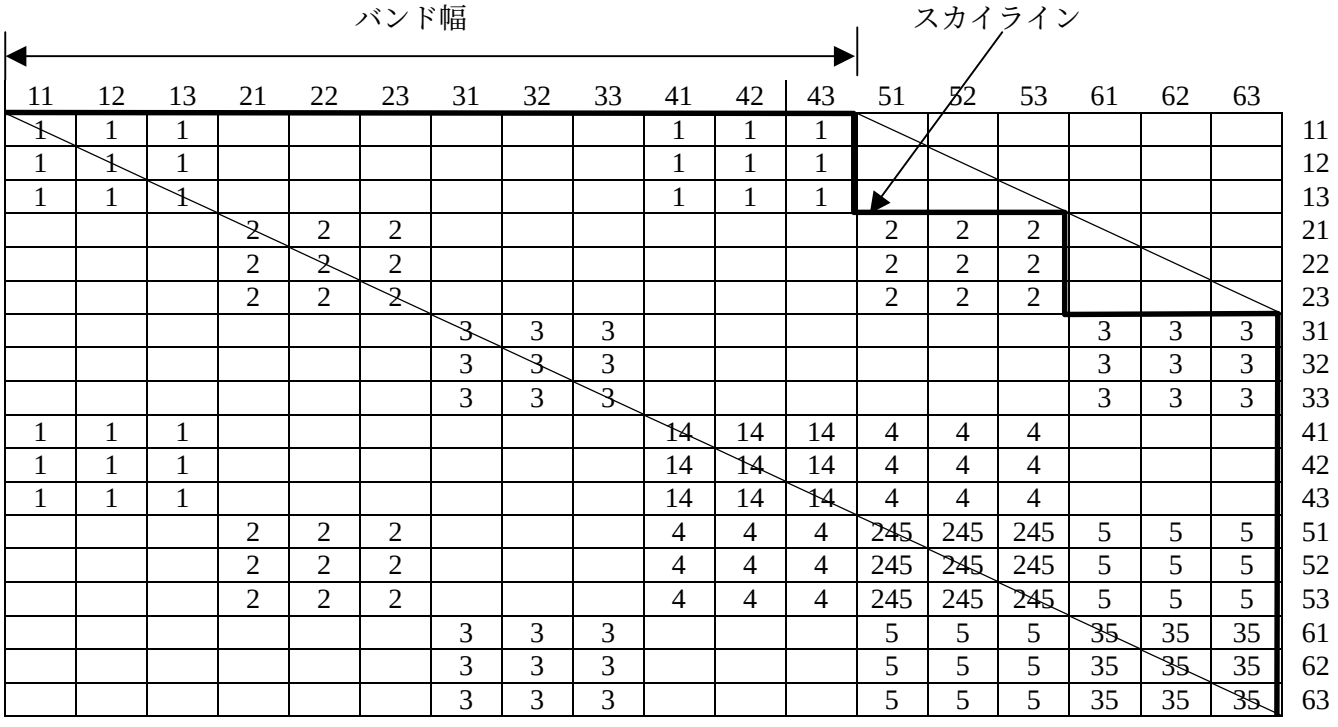

図 5.2　全体剛性マトリックスの成分とバンド幅およびスカイライン

ここでは，文献に示されるバンド幅を最小化するプログラムを用いて節点番号の付け替えを行ってみる。このプログラムでは，節点数，要素数，各要素両端の節点番号の情報を与えれば自動的に番号の付け替えを行う。図 5.1 を例にプログラムの利用法を示す。

```
      Parameter(mnod=10000,mnel=10000,mnnd=2)
      Dimension Jnt(mnod),indv(mnel,mnnd)
      nod = 6
      nel = 5
```

```
      nnd = 2
c
c 要素両端の節点番号
c
      indv(1,1) = 1
      indv(1,2) = 4
      indv(2,1) = 2
      indv(2,2) = 5
      indv(3,1) = 3
      indv(3,2) = 6
      indv(4,1) = 4
      indv(4,2) = 5
      indv(5,1) = 5
      indv(5,2) = 6
c
c バンド幅の最小化
      call Optnum(nod,nel,nnd,indv,jnt,mnod,mnel)
c
      do 10 i = 1,nod
      write(*,300) i, jnt(i)
  300 format(1x,'original = ',i5,3x,'renumber = ',i5)
   10 continue
c
      stop
      end
c
      SUBROUTINE OPTNUM(NOD,NEL,NND,INDV,JNT,mnod,mnel)
      DIMENSION INDV(mnel,1),JNT(1),JMEM(mnod),MEMJT(mnod*8)
     1         ,NEWJT(mnod),JOINT(mnod)
      IDIFF = NOD
      DO 10 J = 1,NOD
   10 JMEM(J) = 0
      DO 60 J = 1,NEL
      DO 50 I = 1,NND
      JNTI = INDV(J,I)
      IF( JNTI.EQ.0 ) GO TO 60
      JSUB = (JNTI-1)*8
      DO 40 II=1,2
      IF( II.EQ.I ) GO TO 40
      JJT = INDV(J,II)
      IF( JJT.EQ.0 ) GO TO 50
      MEM1 = JMEM(JNTI)
      IF(MEM1.EQ.0) GO TO 30
      DO 20 III = 1,MEM1
      IF( MEMJT(JSUB+III).EQ.JJT ) GO TO 40
   20 CONTINUE
   30 JMEM(JNTI) = JMEM(JNTI) + 1
      MEMJT(JSUB+JMEM(JNTI)) = JJT
      IF(IABS(JNTI-JJT).GT.IDIFF) IDIFF=IABS(JNTI-JJT)
   40 CONTINUE
   50 CONTINUE
   60 CONTINUE
c
      MINMAX = IDIFF
      DO 160 IK = 1,NOD
      DO 120 J  = 1,NOD
      JOINT(J) = 0
  120 NEWJT(J) = 0
      MAX = 0
      I = 1
      NEWJT(1)  = IK
      JOINT(IK) = 1
      K=1
  130 K4=JMEM(NEWJT(I))
      IF( K4.EQ.0 ) GO TO 145
      JSUB = ( NEWJT(I)-1 )*8
      DO 140 JJ=1,K4
      K5 = MEMJT(JSUB+JJ)
```

```
      IF( JOINT(K5).GT.0 ) GO TO 140
      K = K+1
      NEWJT(K)  = K5
      JOINT(K5) = K
      NDIFF = IABS(I-K)
      IF( NDIFF.GE.MINMAX ) GO TO 160
      IF( NDIFF.GT.MAX ) MAX = NDIFF
  140 CONTINUE
      IF( K.EQ.NOD ) GO TO 150
  145 I = I + 1
      GO TO 130
  150 MINMAX = MAX
      DO 155 J = 1,NOD
  155 JNT(J) = JOINT(J)
  160 CONTINUE
      RETURN
      END
```

以上のプログラムを実行すると，以下のような結果が得られる。したがって，節点番号は図 5.3 のように付け変わる。

```
 original =    1    renumber =     1
 original =    2    renumber =     4
 original =    3    renumber =     6
 original =    4    renumber =     2
 original =    5    renumber =     3
 original =    6    renumber =     5
```

このとき，全体マトリックスの成分は，図 5.4 のように配置され，図 5.2 との比較から明らかなようにバンド幅は縮小される。また，図 5.2，図 5.4 の太線で示されているのは，マトリックスのスカイラインで，マトリックスのスカイラインはさらに縮小されていることがわかる。

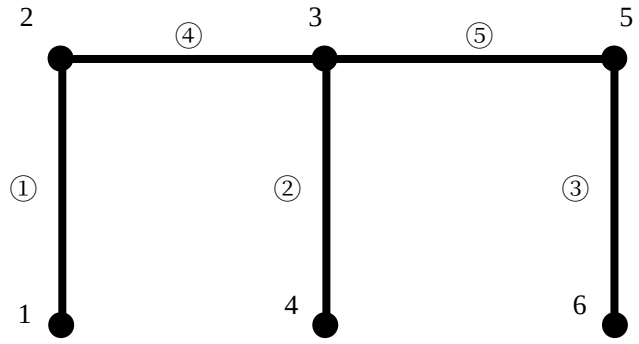

図 5.3　節点番号の付け替え

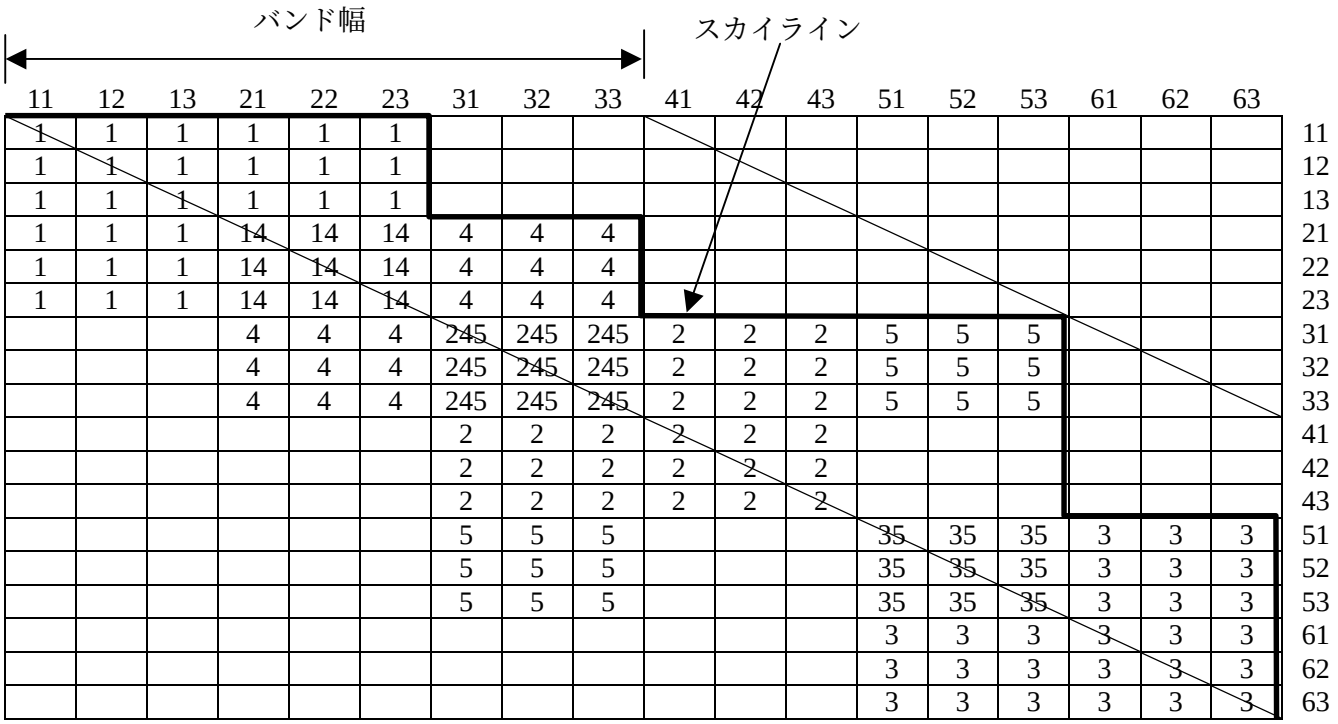

図 5.4　全体剛性マトリックスの成分とバンド幅およびスカイライン

5.2.2　スカイライン法による連立方程式の解法

まず，スカイライン法では，マトリックスのスカイラインをプログラムに認識させるために，対角項からマトリックスのスカイライン高さまでの列の高さを計算する。文献に掲載されているサブルーチンでは，図 5.6 に示すようにマトリックスの成分に番号を付けて，この対角項の番号を事前に計算するようになっている。この場合の各対角成分の番号は，

1，3，6，10，15，21，25，30，36，40，45，51，58，66，75，79，84，90

の 18 個となる。また，必要となるマトリックスの容量は，最終列の対角成分の値で 90 となる。

この例題に関する，このサブルーチンの使い方を以下に示す。

```
      Parameter(mnod=1000,mnel=1000,mnnd=2,mndg=3,mnmt=100)
      Dimension indv(mnel,mnnd),jnt(mnod),indsk(mnod*mndg)
      nod = 6                 !節点数
      nel = 5                 !要素数
      nnd = 2                 !一要素の節点数
      ndg = 3                 !一節点の自由度数
      ndgt = ndg*nod          !全自由度数
c 節点番号の付け替え
      do 40 i = 1,nel
      do 40 j = 1,nnd
      ir = jnt( indv(i,j) )
   40 indv(i,j) = ir
c
      call Mhight(indv,indsk,nel,nnd,ndg,ndgt,mnel)
c
```

11	12	13	21	22	23	31	32	33	41	42	43	51	52	53	61	62	63	
1	2	4	7	11	16													11
	3	5	8	12	17													12
		6	9	13	18													13
			10	14	19	22	26	31										21
				15	20	23	27	32										22
					21	24	28	33										23
						25	29	34	37	41	46	52	59	67				31
							30	35	38	42	47	53	60	68				32
								36	39	43	48	54	61	69				33
									40	44	49	55	62	70				41
										45	50	56	63	71				42
											51	57	64	72				43
												58	65	73	76	80	85	51
													66	74	77	81	86	52
														75	78	82	87	53
															79	83	88	61
																84	89	62
																	90	63

図 5.6　スカイライン成分の番号付け

```
      write(*,50) ( indsk(i),i = ndgt )
   50 format(10i5)
      stop
      end
C
      SUBROUTINE MHIGHT(INDV,INDSK,NEL,NS,NF,NDF,ND)
      DIMENSION INDV(ND,1),INDSK(1),LO(100)
      DO 10 I=1,NDF
   10 INDSK(I)=0
      NSF=NS*NF
      DO 20 NE=1,NEL
      CALL LOCATE(NE,LO,INDV,NS,NF,ND)
      DO 30 JS=1,NSF
      JT=LO(JS)
      J=JT
      JH=0
      DO 40 IS=1,NSF
      IT=LO(IS)
      I=IT
      JJ=J-I
      IF(JJ.GT.JH) JH=JJ
   40 CONTINUE
      IF(JH.GT.INDSK(J)) INDSK(J)=JH
   30 CONTINUE
   20 CONTINUE
      JJ=0
      DO 50 I=1,NDF
      JJ=JJ+INDSK(I)+1
      INDSK(I)=JJ
   50 CONTINUE
      RETURN
      END
C
      SUBROUTINE LOCATE(NE,LO,INDV,NS,NF,ND)
      DIMENSION LO(1),INDV(ND,1)
      DO 10 J=1,NS
      JB=NF*(J-1)
```

```
      L=INDV(NE,J)
      N=NF*(L-1)
      DO 20 K=1,NF
   20 LO(JB+K)=N+K
   10 CONTINUE
      RETURN
      END
```

以上のプログラムを前節のプログラムと組み合わせて（メインルーチンは１つに統一する），実行することにより以下のような結果を得る。

```
  1   3   6  10  15  21  25  30  36  40
 45  51  58  66  75  79  84  90
```

この **indsk** の情報を用いて，全体剛性マトリックスの成分を一次元ベクトルに格納する。この部分をプログラムで書くと次のようになる。なお，節点座標，変位が規定される節点番号，外力が作用する節点番号に対しても **jnt** を利用して節点番号の付け替えを行っておく必要がある。

```
      Parameter(mnod=1000,mnvol=100000)
      Dimension SKG(mnvol),xrn(mnod),yrn(mnod)
c
      nvol = indsk(ndgt)
c
c 節点座標番号の付け替え
      do 10 i = 1,nod
      xrn(jnt(i)) = x(i)
   10 yrn(jnt(i)) = y(i)
      do 20 i = 1,nod
      x(i) = xrn(i)
   20 y(i) = yrn(i)
c
c 変位規定節点番号の付け替え
      do 30 i = 1,ndg
      n = ngeobc(i)
      do 40 j = 1,n
      ir = jnt( igeobc(i,j) )
      igeobc(i,j) = ir
   40 continue
   30 continue
c
c 外力が作用する節点番号の付け替え
      do 50 i = 1,ndg
      n = nforce(i)
      do 60 j = 1,n
      ir = jnt( iforce(i,j) )
      iforce(i,j) = ir
   60 continue
   50 continue
c
c 全体剛性マトリックスのゼロクリアー
c
      do 70 i = 1,nvol
   70 SKG(i) = 0.d0
c
c 要素剛性マトリックスの重ね合わせ
c
      do 100 n = 1,nel
      call Elmat(x,y,Eyg,Are,Siy,indv,mte,SKL,n,mnel,mnnd,mndg)
      do 110 ip = 1,nnd
```

```
      do 110 i  = 1,ndg
      iL = ndg*( ip-1 ) + i
      iG = ndg*( indv(n,ip)-1 ) + i
      do 120 jp = 1,nnd
      do 120 j  = 1,ndg
      jL = ndg*( jp-1 ) + j
      jG = ndg*( indv(n,jp)-1 ) + j
      if( iG.gt.jG ) go to 120
      kG = indsk(jG) + iG - jG          !通常の重ね合わせと異なる部分
      SKG(kG) = SKG(kG) + SKL(iL,jL)
  120 continue
  110 continue
  100 continue
```

４章の重ね合わせのプログラムの一部を以上のものに入れ換えれば，スカイライン法の全体剛性マトリックスを作ることができる。

右辺ベクトル$\{\bar{f}\}-[K_{\delta\bar{\delta}}]\{\bar{\delta}\}$は，スカイラインマトリックスとベクトルの掛け算を行うサブルーチンを用いて行う。

```
      Idx = -1        ：icrs がマイナスになる列成分をベクトルと掛ける
      Idy =  1        ：icrs がプラスになる行成分のみ計算を行う

      Isw = -1        ：結果を{f}-[K_uu]{u} の形で返す

      call SKXMLU(SKG,icrs,ndgt,indsk,D,F,Idx,Idy,Isw)
      stop
      end
C
      SUBROUTINE SKXMLU(A,ICRS,ND,INDSK,X,Y,IDX,IDY,ISW)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),ICRS(1),X(1),Y(1),INDSK(1)
      DO 10 I = 1,ND
      IF( ICRS(I)*IDY.LT.0 ) GO TO 10
      YI = 0.D0
      IF( I.EQ.1 ) GO TO 20
      INDI = INDSK(I)
      IK = INDI-INDSK(I-1)
      JJ1 = I-IK+1
      IF( JJ1.EQ.I ) GO TO 20
      I1 = I-1
      DO 30 J = JJ1,I1
      IF( ICRS(J)*IDX.LT.0 ) GO TO 40
      K = INDI+J-I
      YI = YI+X(J)*A(K)
   40 K = K+1
   30 CONTINUE
   20 DO 50 J=I,ND
      IF( ICRS(J)*IDX.LT.0 ) GO TO 50
      INDJ = INDSK(J)
      IF( J.EQ.1 ) GO TO 60
      JT = J-INDJ+INDSK(J-1)+1
      IF( I.LT.JT ) GO TO 50
   60 K = INDJ+I-J
      YI = YI+X(J)*A(K)
   50 CONTINUE
      IF(ISW) 70,80,90
   70 Y(I) = Y(I)-YI
      GO TO 10
   80 Y(I) = YI
```

```
      GO TO 10
   90 Y(I) = Y(I)+YI
   10 CONTINUE
      RETURN
      END
```

そして，$[K_{uu}]\{u\}=\{\bar{f}\}-[K_{u\bar{u}}]\{\bar{u}\}$ の計算は以下のサブルーチンを用いて行う。この場合，与えた外力ベクトル F に計算された変位が代入されて返される。得られた変位を出力する場合，節点番号が付け替えられていることに注意して，元の節点番号で出力するようにする。

```
      call SKSOLV(SKG,F,icrs,indsk,ndgt)
      do 200 i = 1,nod
      write(*,210)( F( ndg*( jnt(i)-1 )+j ),j=1,ndg )
  210 format(1x,'u = ',e12.4,2x,'w = ',e12.4,2x,'ud = ',e12.4)
  200 continue
      stop
      end
c
      SUBROUTINE SKSOLV(A,Y,ICRS,INDSK,ND)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),Y(1),ICRS(1),INDSK(1)
      DO 10 J=2,ND
      IF(ICRS(J).LE.0) GO TO 10
      INDJ=INDSK(J)
      JH=INDJ-INDSK(J-1)
      JJH=J-JH+1
      DO 20 I=JJH,J
      IF(ICRS(I).LE.0) GO TO 20
      IF(I.EQ.1) GO TO 20
      INDI=INDSK(I)
      IH=INDI-INDSK(I-1)
      JHP=JH+I-J
      IS=MINO(IH,JHP)
      IF(IS.LE.1) GO TO 20
      ISS=I-IS+1
      KH=INDJ+I-J
      KH1=INDJ+ISS-J
      KH2=INDI+ISS-I
      IM1=I-1
      DO 30 N=ISS,IM1
      IF(ICRS(N).LE.0) GO TO 60
      INDN=INDSK(N)
      IF(A(INDN).NE.0.D0) GO TO 40
      WRITE(6,50) N
   50 FORMAT(5X,'****** INSTABILITY ERROR ******  AT=',I8)
      RETURN
   40 A(KH)=A(KH)-A(KH1)*A(KH2)/A(INDN)
   60 KH1=KH1+1
      KH2=KH2+1
   30 CONTINUE
   20 CONTINUE
   10 CONTINUE
      DO 100 I=1,ND
      IF(ICRS(I).LE.0) GO TO 100
      INDI=INDSK(I)
      IF(I.EQ.1) GO TO 110
      INDH=INDSK(I-1)
      JK=INDI-INDH
      JH=I-JK+1
      K=INDH+1
      IF(JH.EQ.I) GO TO 110
      IM1=I-1
```

```
      DO 120 J=JH,IM1
      IF(ICRS(J).LE.0) GO TO 130
      Y(I)=Y(I)-A(K)*Y(J)
  130 K=K+1
  120 CONTINUE
  110 Y(I)=Y(I)/A(INDI)
  100 CONTINUE
      DO 200 IC=2,ND
      I=ND-IC+1
      IF(ICRS(I).LE.0) GO TO 200
      SUM=0.D0
      J1=I+1
      DO 210 J=J1,ND
      IF(ICRS(J).LE.0) GO TO 210
      INDJ=INDSK(J)
      JT=J-INDJ+INDSK(J-1)+1
      IF(I.LT.JT) GO TO 210
      K=INDJ+I-J
      SUM=SUM+A(K)*Y(J)
  210 CONTINUE
      INDI=INDSK(I)
      Y(I)=Y(I)-SUM/A(INDI)
  200 CONTINUE
      RETURN
      END
```

最後に反力$\{f\}=[K_{\bar{u}u}]\{u\}+[K_{\bar{u}\bar{u}}]\{\bar{u}\}$の計算は，先に用いたサブルーチン SKXMLU を用いて計算できる。ただし，SKSOLV の計算を行うと，全体剛性マトリックスは LU 分解されて保存されているので，再度重ね合わせを行って全体剛性マトリックスを作る必要がある。また，反力を出力する場合も節点番号が付け替えられていることに注意を要する。

```
      Idx =  0        ：マトリックスの全列成分とベクトルの成分を掛ける
      Idy = -1        ：icrs がマイナスになる行成分のみ計算を行う
      Isw =  0        ：結果を{f}の形で返す
```

```
      call SKXMLU(SKG,icrs,ndgt,indsk,F,REF,Idx,Idy,Isw)
c
      do 300 i = 1,nod
      do 300 j = 1,ndg
      n = ndg*( jnt(i)-1 )+j
      if( icrs(n).lt.0 ) then
       write(*,310) i, j, F(n)
  310  format(1x,'nod No. = ',i5,2x,'direc. = ',i3,2x,'React. = ',e12.4)
      end if
  300 continue
      stop
      end
```