第５章　スカイライン行列

ポイント：剛性行列はスカイライン行列を使用する
剛性の選択で計算コストとメモリーコスト

5.1 はじめに

本章では、平面骨組の最終的な釣合式を表す連立方程式をどのように解くのかを考える。特に、係数行列である剛性行列を如何なる形式で保持し、計算するかで、計算スピードと安定性、また、必要メモリー量に大きく関係する。ここでは、現在最も効率の良い手法の一つであるスカイライン行列を使用し、スカイライン行列の LDU 分解法、連立方程式の解法、また、スカイライン行列の保持に必要な表について考える。

キーワード

釣合式の表現　帯行列　スカイライン行列　LDU 分解　計算コストとメモリーコスト

5.2 連立方程式の解法

平面骨組の応力解析では、最終的に、次式の連立方程式を解く必要がある。ここでは、この方程式について考える。

$$[K]\{u\} = \{p\} \qquad \cdots\cdots(5.1)$$

上記の解析手法として、反復解法と直接解法がある。通常では、直接解法が多く用いられており、その基本はガウスの消去法である。同法の拡張として LDU 分解法があり、多くのシステムで使用されている。この分解は一意に行われることが証明されており、剛性行列は、次式のように分解可能となっている。

$$[K]\{u\} = \{p\} \quad \rightarrow \quad [L][D][U]\{u\} = \{p\} \qquad \cdots\cdots(5.2)$$

$$[L] = \begin{bmatrix} 1 & & & & \\ L_{21} & 1 & & 0 & \\ L_{31} & L_{21} & 1 & & \\ L_{41} & & & 1 & \\ & & & & \\ L_{n1} & & & & 1 \end{bmatrix} \quad [D] = \begin{bmatrix} d_1 & 0 & 0 & & \\ 0 & d_2 & 0 & & 0 \\ 0 & & d_3 & & \\ & & & d_4 & \\ & 0 & & & \\ & & & & d_n \end{bmatrix} \qquad \cdots\cdots(5.3a)$$

$$[U]=\begin{bmatrix} 1 & U_{21} & U_{31} & U_{41} & & U_{n1} \\ 0 & 1 & U_{32} & & & \\ 0 & 0 & 1 & & & \\ 0 & & & 1 & & \\ & & & & & \\ 0 & & & & & 1 \end{bmatrix} \qquad \cdots\cdots(5.3b)$$

ここで、$[L]$は下三角行列で対角項は全て１であり、$[D]$は対角行列、また、$[U]$は上三角行列で対角項は全て１である。剛性行列が実・対称行列であるため、次の関係があり、どちらかを求めることになる。

$$[L]^T=[U] \qquad \cdots\cdots(5.4)$$

この分解法には多くのメリットがあり、特に、次に示すように連立方程式を容易に解くことができる。まず、式(5.2)で、

$$[D][U]\{u\}=\{x\} \qquad \cdots\cdots(5.5)$$

と置くと、

$$[L]\{x\}=\{p\} \qquad \cdots\cdots(5.6)$$

となり、最初に、この方程式を解くことになる。上式は、係数行列が下三角行列であることから、以下のように前進代入法で簡単に解が求められる。

$$\begin{bmatrix} 1 & & & & \\ L_{21} & 1 & & 0 & \\ L_{31} & L_{21} & 1 & & \\ L_{41} & & & 1 & \\ & & & & \\ L_{n1} & & & & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \\ \\ x_n \end{Bmatrix} = \begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ \\ \\ p_n \end{Bmatrix} \Rightarrow \begin{aligned} x_1 &= p_1 \\ x_2 &= p_2 - L_{21}x_1 \\ x_3 &= p_3 - L_{31}x_1 - L_{21}x_2 \end{aligned} \Downarrow \qquad \cdots\cdots(5.7)$$

前進代入法で、式(5.6)の解$\{x\}$が求められると、この$\{x\}$を式(5.5)の右辺項として、下式を解くことになる。

$$[U]\{u\}=[D]^{-1}\{x\} \qquad \cdots\cdots(5.8)$$

係数行列$[U]$は、式(5.4)の関係から、

$$[L]^T\{u\}=[D]^{-1}\{x\} \qquad \cdots\cdots(5.9)$$

となり、ここでも下三角行列を使うことができる。式(5.8)の解析も容易であり、次のように後退代入法で次々と解が求められる。

$$\begin{bmatrix} 1 & U_{21} & U_{31} & U_{41} & & & U_{n1} \\ 0 & 1 & U_{32} & & & & \\ 0 & & & & & & \\ 0 & & & 1 & U_{n-1,n-2} & U_{n,n-2} & \\ & & & & 1 & U_{n,n-1} & \\ 0 & & & & & 1 & \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \\ u_{n-3} \\ u_{n-1} \\ u_n \end{Bmatrix} = \begin{Bmatrix} x_1/d_1 \\ x_2/d_2 \\ \\ x_{n-2}/d_{n-2} \\ x_{n-1}/d_{n-1} \\ x_n/d_n \end{Bmatrix} \quad \cdots\cdots(5.10a)$$

$$\begin{aligned} x_{n-2} &= \frac{x_{n-2}}{d_{n-2}} - U_{n,n-2}u_n - U_{n-1,n-2}u_{n-1} \\ x_{n-1} &= \frac{x_{n-1}}{d_{n-1}} - U_{n,n-1}u_n \\ u_n &= \frac{x_n}{d_n} \end{aligned} \quad \cdots\cdots(5.10b)$$

LDU 分解法は、分解できれば、上記のように後の方程式を解くのは容易である。しかも、その解法では、元の三角領域の値は変更されない。そのため、何度でも荷重項を変更して解くことができるメリットがある。さらに、骨組の剛性行列は実対称であり、記憶領域は対角項を含む半分で良い。また、分解アルゴリズムを工夫することで、分解前の剛性行列の記憶領域に、分解後の値を記憶させることが可能であり、メモリーコストを減少させることができる。

さらに、次のように、骨組の剛性行列は対角項近くにバンド状に集まる傾向がある。しかも、分解後もバンド性を保持することが保障されている。

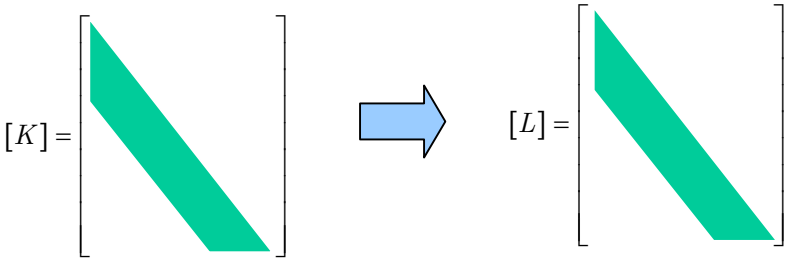


図 5-1 剛性行列のバンド性

この傾向を利用して、剛性行列を上ハーフバンド行列として作成し、分

解後も同じハーフバンド行列に保存する。この手法はコレスキー法と呼ばれる。

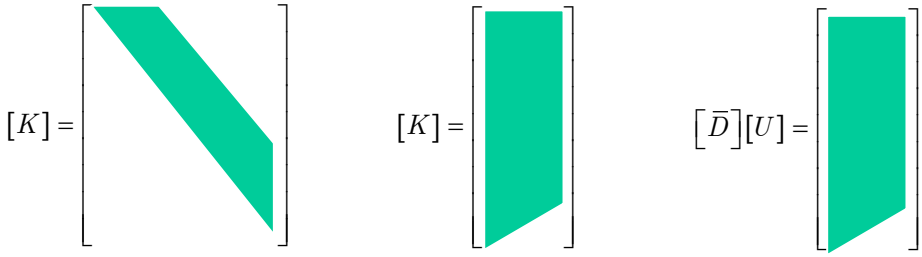


図 5-2 剛性行列のコレスキー分解

上記の図のように、コレスキー行列では、実際の対角項は左端となる。ここでも、分解後は、図のように同じ領域に上三角行列が得られる。コレスキー法の優劣はハーフバンド幅に依存する。従って、このバンド幅が小さいほど計算コストもメモリーコストも減少する。ただ、問題は、一つでも対角項から離れた場所に値が入ると、バンド幅は大きくなり、極端に効率が悪くなる。節点番号の付け方や、未知数番号の付け方に処理能力が大きく依存することになる。

さらに効率の良い方法として開発されたのが、スカイライン法と呼ばれる手法である。同法は、剛性行列の下部分を図 5-3 のように 1 次元で保持する。また、分解後もスカイラインの形状を保持することが保障されており、分解後の値も元の場所に保存することができる。

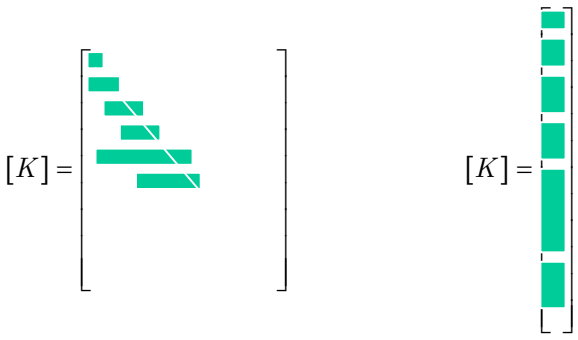


図 5-3 スカイライン分解

スカイライン行列を作成するためには、未知数番号表の他に左より連続番号を振ったとき、対角項の番号を保存する表が必要となる。

骨組の自由度が大きくなり、数千、数万になると、保存メモリーを如何に少なくするか、如何に計算時間を少なくするかが重要な課題となる。他にも多くの方法が開発されているが、このスカイライン法は現在でも多くのプログラムで使用されており、非常に優れた手法の一つといえる。

5.3 境界条件

境界条件を与えないと釣合式(連立方程式)は常に、特異行列となり、LDU 分解が不可能となる。この場合、物理的には剛体変位を許すことになり、骨組が不安定となる。そこで、骨組の境界条件を連立方程式の中に組み込む必要がある。境界条件の与え方は、次のように釣合式を分割して、表現することから始まる。

$$[K]\{u\} = \{p\}$$

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} p_1 \\ p_2 \end{Bmatrix} \quad \cdots\cdots(5.11)$$

ここで、$\{u_1\}$は未知変位ベクトルであり、$\{u_2\}$は境界変位である。一般には、$\{u_2\}$は拘束されており、ゼロとなる。また、$\{p_1\}$は外荷重であり、$\{p_2\}$は境界部分の反力となる。境界変位がゼロベクトルであるとすると、上式は、

$$[K_{11}]\{u_1\} = \{p_1\} \quad \cdots\cdots(5.12)$$

となり、単純に境界部分の自由度を取り除いて、方程式を構成することになる。また、境界部分の反力は次式で与えられる。

$$\{p_2\} = [K_{21}]\{u_1\} \quad \cdots\cdots(5.13)$$

一方、境界変位がゼロベクトルでなく、強制変位として与えられる場合は、

$$[K_{11}]\{u_1\} = \{p_1\} - [K_{12}]\{u_2\} \quad \cdots\cdots(5.14)$$

となり、強制変位項が右辺項に移項される。次に、上式を解いて、未知変位を求める。さらに反力は次式より求められる。

$$\{p_2\} = [K_{21}]\{u_1\} + [K_{22}]\{u_2\} \quad \cdots\cdots(5.15)$$

5.4 スカイライン行列のための表作成

本節では、スカイライン行列について説明する。スカイライン行列は、節点未知番号表だけでは作成できない。次のようなスカイライン用の表（以後この表をスカイライン表と呼ぶ）が別途必要となる。まず、節点拘束表について復習してみよう。第 3 章で用いた例を再度使用する。まず、節点拘束表と未知番号表を再度以下に示す。

節点拘束表　P_rest
(*, 1)： 0, 0, 0
(*, 2)： 1, 1, 1
(*, 3)： 1, 1, 1
(*, 4)： 1, 1, 1
(*, 5)： 0, 0, 0

⇒

未知番号表　P_rest
(*, 1)： 0, 0, 0
(*, 2)： 1, 2, 3
(*, 3)： 4, 5, 6
(*, 4)： 7, 8, 9
(*, 5)： 0, 0, 0

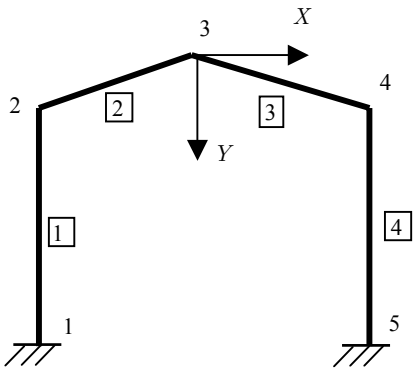
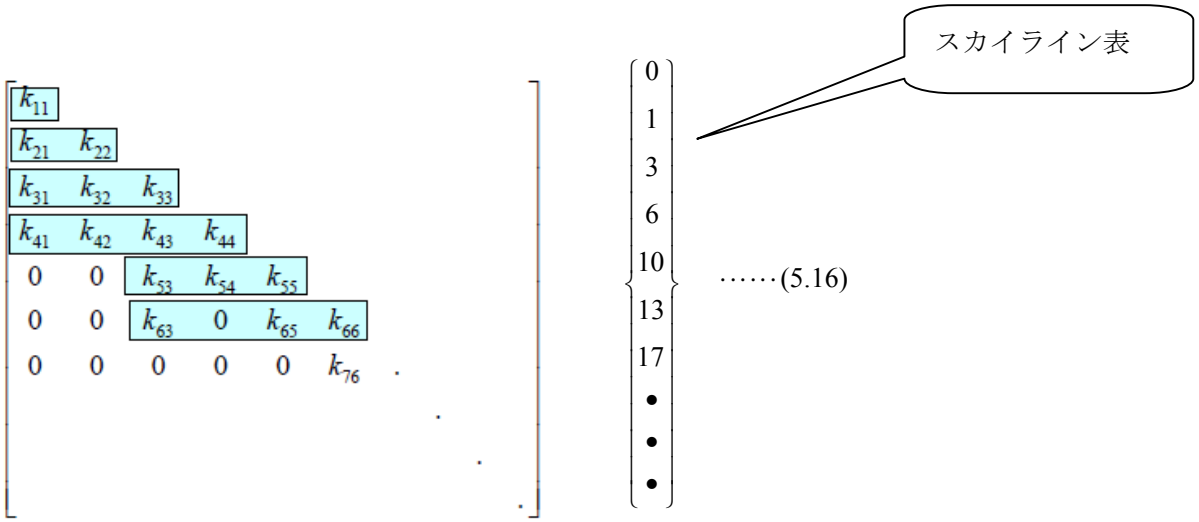


図 5-4 未知番号表作成用骨組

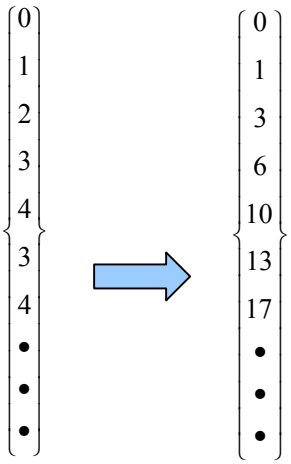
次に、この未知番号表を利用して、スカイライン表を作成する。スカイライン表は、1 次元のスカイライン行列を効率的に作成するための表である。その内容は次のような対角項の保存番号である。



……(5.16)

この表は下三角部分で水平方向に、また、上より順次番号を付けたとき、対角項がスカイライン行列のどの位置にあるかを示す番号である。最初の 0 はダミーであり、後の操作で必要となる。

この表の作成法は、まず、次のように行列の各行の左より最初のゼロでない要素から、対角要素までの数を並べた表を作成する。次に、左の表を上から和をとり、右の表に変換する。この右の表の値は当該行列の対角要素がスカイライン行列で位置する番号となる。



……(5.17)

スカイライン行列は、この表から次のように下部分の剛性を保持する。

$$
\begin{bmatrix}
k_{11} & & & & & \\
k_{21} & k_{22} & & & & \\
k_{31} & k_{32} & k_{33} & & & \\
k_{41} & k_{42} & k_{43} & k_{44} & & \\
0 & 0 & k_{53} & k_{54} & k_{55} & \\
0 & 0 & k_{63} & 0 & k_{65} & k_{66} \\
0 & 0 & 0 & 0 & 0 & k_{76} \\
 & & & & & \ddots
\end{bmatrix}
\begin{Bmatrix}
k_{11} \\ k_{21} \\ k_{22} \\ k_{31} \\ k_{32} \\ k_{33} \\ k_{41} \\ k_{42} \\ k_{43} \\ k_{44} \\ k_{53} \\ k_{54} \\ k_{55} \\ k_{63}
\end{Bmatrix}
\begin{Bmatrix}
0 \\ 1 \\ 3 \\ 6 \\ 10 \\ 13 \\ 17 \\ \bullet \\ \bullet \\ \bullet
\end{Bmatrix}
\quad \cdots\cdots (5.18)
$$

スカイライン行列で、赤色部分は対角項であり、その位置は、スカイライン表の中の値となっている。

次の問題は、式(5.17)の左辺項の表をどのようにして作成するかである。この表は、各行の左より最初のゼロでない要素から、対角要素までの数を並べた表であり、ここでは、節点未知番号表と、節点との結合を表す部材表を利用する。まず、ひとつの部材を取り出し、両端の節点の未知番号表から、部材剛性行列の各要素が全体剛性行列のどの位置に挿入されるかを計算する。その際、全部材に対し、下三角部分に、しかも、各行の最も左に入る要素を検索し、その位置から対角項までの要素数を計算する。

この検索アルゴリズムを図 5-5 に示す。少し長くなっているが、それほど難しくないので、処理の流れを良く理解されたい。最初に、スカイライン表 nsum_d(i)を全自由度に対し１にセットする。次に、全部材について、次の処理を実施する。部材両端の未知番号を取り出し、配列 ff_rest に保存する。

両端の拘束状態（未知番号表）を取り出し、拘束

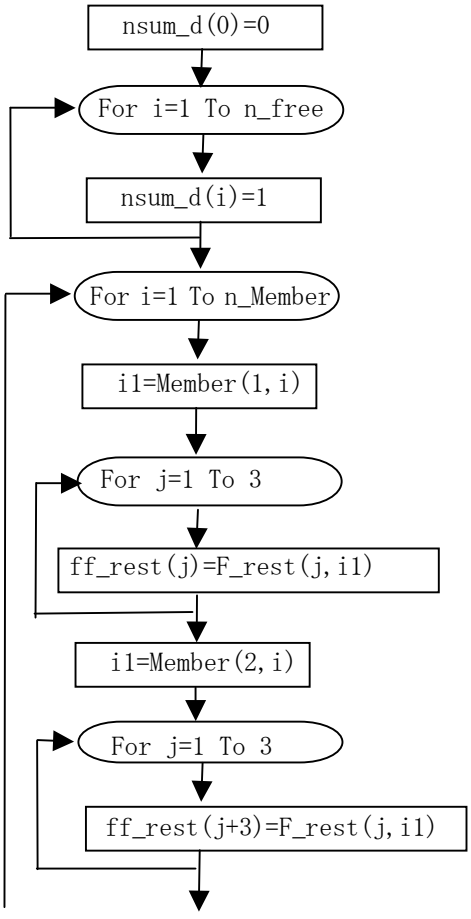


図 5-5(a) スカイライン表の作成流れ図その１

でない場合を確かめる。次に、他の自由度の未知番号を取り出し、その未知番号の値と比較し、自分自身より小さい場合は、下三角に剛性設定であることから、次へ進む。さらに、その列で、最も大きい値かどうかチェックし、大きければ最大バンド幅の値を置き換える。この操作を全部材、全自由度に対し行うと、スカイライン表であるnsum_d()には、その配列位置にその自由度番号（未知番号）に対応した最大バンド幅がセットされる。

最後に、自由度の若い順に和を採り、対角項の位置がセットされることになる。最後の対角項には、スカイライン行列を保存するために必要となる配列個数が求められることになる。

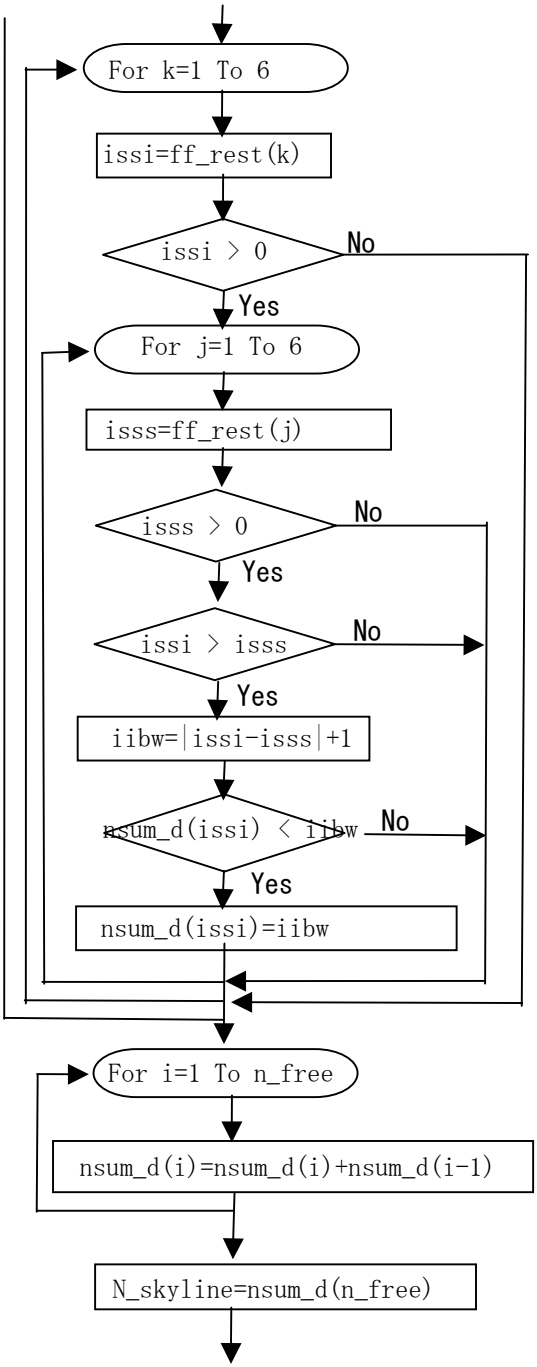


図 5-5(b) スカイライン表の作成流れ図その２

5.5 スカイライン行列の作成

次に、スカイライン剛性行列を作成する部分の処理を説明しよう。既に、対称行列では全体の流れの中で説明したので、詳しいことはそちらを参照されたい。ここでは、スカイライン行列に関連した箇所について説明する。先に図 5-5(b)に示されているように、スカイライン行列に

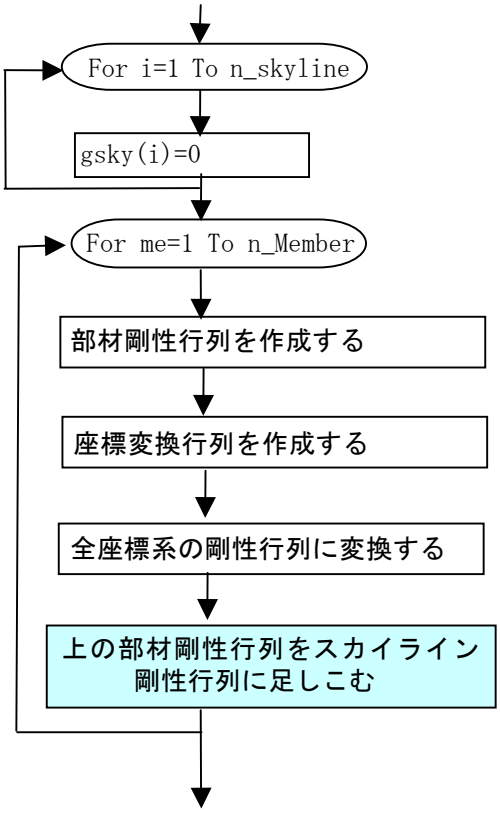


図 5-6 スカイライン剛性行列作成フローチャート

必要となる最大保存領域の値 N_skyline を用いて、１次元のスカイライン行列を動的に確保する。次に図 5-6 に示されるように、まず、スカイライン行列をゼロクリアする。

全体剛性行列作成の手続きは、対称行列と同じであり、異なる箇所は、全体座標系で表される部材剛性行列をスカイライン行列に足しこむ部分である。この手続きは、スカイライン用の表作成と類似しており、対称行列を１次元で保存するため、各要素の保存位置の計算が少し煩雑であるが、それほど難しくはない。まず、スカイライン剛性行列を作成するための全体の流れを理解しよう。

スカイライン表と異なっているのは、スカイライン剛性行列 gsky()の位置を計算する箇所であり、ここで、表を利用して、対角項からのずれを計算して部材剛性行列を１次元のスカイライン行列に足しこんでいる。

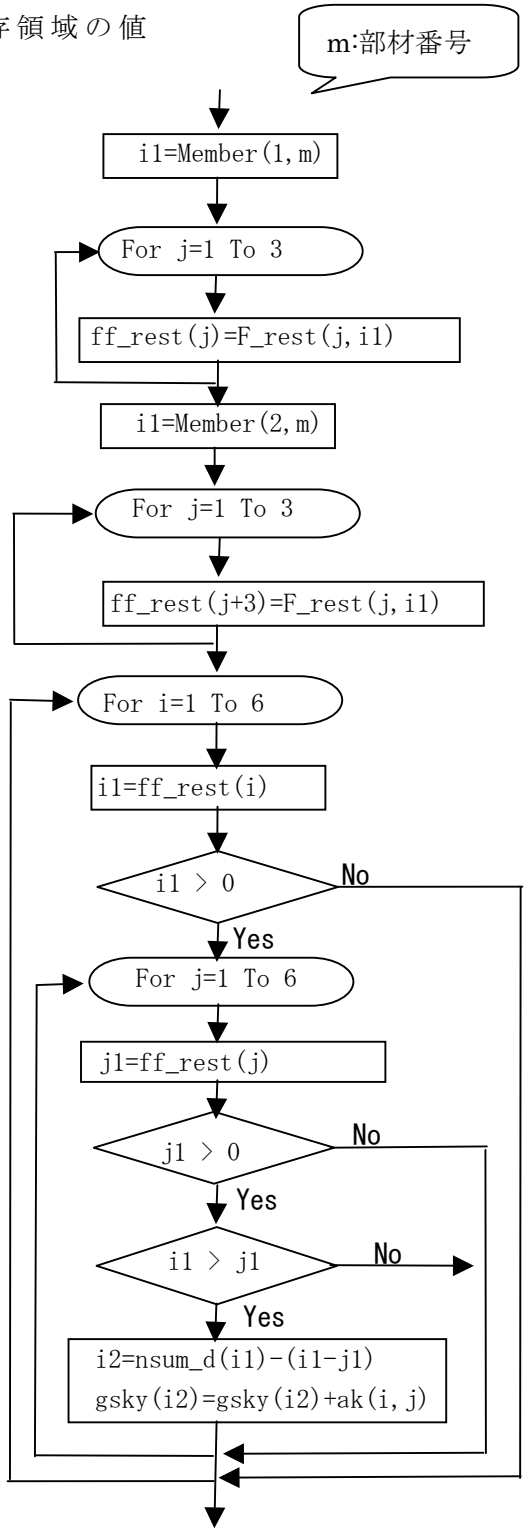


図 5-7 スカイライン剛性行列に部材剛性行列を足しこむ処理の流れ図

5.6スカイライン行列のLDU分解

本節では、1次元で保存されたスカイライン行列をLDU分解する手法を説明する。実対称行列のLDU分解は、一意に分解可能で、しかも、スカイラインの形状を保持する。従って、全体剛性行列の保存領域が、分解された下三角行列の1次元配列を、保存場所として再利用され、結果、メモリー領域が非常に経済的となる。まず、このLDU分解のアルゴリズムについて学習しよう。

線形の連立方程式は次式で表されるものとする。

$$[G]\{u\} = \{p\} \qquad \cdots\cdots(5.19)$$

係数行列は実対称とすると、次のように一意に分解可能である。

$$[G] = [L][D][L]^T \qquad \cdots\cdots(5.20)$$

ここで、$[L]$は下三角行列であり、対角項は全て1である。また、$[D]$は対角行列である。ここで上式はアルゴリズムを述べる都合で次のように表す。

$$[G] = [L][U]; \quad [U] = [D][L]^T \qquad \cdots\cdots(5.21)$$

上式は行列の成分で次のように表される。

$$\left.\begin{aligned} g_{ij} &= \sum_{k=1}^{i} l_{ik}u_{kj} = \sum_{k=1}^{i-1} l_{ik}u_{kj} + u_{ij} \quad (i \le j) \\ g_{ij} &= \sum_{k=1}^{j} l_{ik}u_{kj} = \sum_{k=1}^{j-1} l_{ik}u_{kj} + l_{ij}u_{jj} \quad (i \ge j) \end{aligned}\right\} \cdots\cdots(5.22)$$

上式を少し変更すると、下三角行列の成分l_{ij}が次のように求められる。

$$l_{ij} = \frac{1}{u_{jj}}(g_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}) \quad (i > j) \qquad \cdots\cdots(5.23)$$

式(5.22)の上で、i=jとすると、

$$u_{ii} = g_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{ki} \qquad \cdots\cdots(5.24)$$

となる。ここで、$[U]$も行列の成分で表すと、$[D]$が対角行列であることを考慮すると

$$u_{ij} = d_{ii}l_{ji} \qquad \cdots\cdots(5.25)$$

であり、上式よりスカイライン用の LDU 分解のアルゴリズム基礎式が以下のように得られる。

$$\left.\begin{aligned} l_{ij} &= \frac{1}{d_{jj}}\left(g_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}\right) \quad (i > j) \\ d_{ii} &= g_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk} \end{aligned}\right\} \cdots\cdots(5.26)$$

次に、スカイライン用の分解アルゴリズムがどのように動作するかについて見ていこう。まず、1 行目では、下より $d_1 = g_{11}$ となり、第 2 行目では、上より、

$$l_{21} = \frac{1}{d_{11}}(g_{12}) \qquad \cdots\cdots(5.27)$$

となる。l_{11} は定義より、１である。また、d_{22} は、式(5.26)の下より

$$d_{22} = g_{22} - l_{21}^2 d_{11} \qquad \cdots\cdots(5.28)$$

となる。既に l_{21} と d_{11} が求められていることから、d_{22} は簡単に計算できる。さらに、第 3 行では、下三角行列の成分は、

$$\left.\begin{aligned} l_{31} &= \frac{1}{d_{11}}(g_{31}) \\ l_{32} &= \frac{1}{d_{22}}(g_{32} - l_{31} d_{11} l_{21}) \\ l_{31} &= 1 \end{aligned}\right\} \cdots\cdots(5.29)$$

として得られ、d_{33} は

$$d_{33} = g_{33} - l_{31}^2 d_{11} - l_{32}^2 d_{22} \qquad \cdots\cdots(5.30)$$

となる。以上のように、下三角行列と対角行列の全ての成分は、第 1 行目から順次計算することによって、容易に求められることが分かる。

スカイライン法アルゴリズムについて、さらに理解を深めるために図 5-8 を用いて説明しよう。下式は、図のように第 i 行と第 j に関連して計算が行われ、式中の積和は両行のバンド幅外では当然行われないことになる。求めた下三角行列の成分は、元の係数行列と同じ配列位置に置き換えて保存される。

$$\left.\begin{aligned} l_{ij} &= \frac{1}{d_{jj}}\left(g_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}\right) \quad (i > j) \\ d_{ii} &= g_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk} \end{aligned}\right\} \cdots\cdots(5.31)$$

特に、計算過程で、第i行を次のように置くと、

$$\bar{l}_{ij} = g_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk} \quad (i > j) \qquad \cdots\cdots(5.32)$$

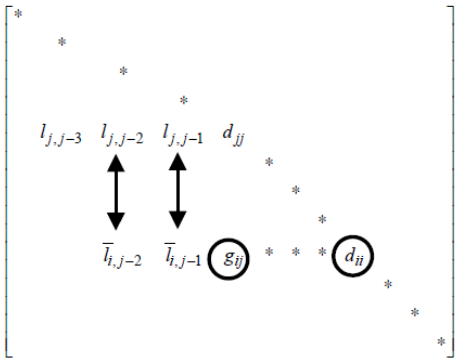


図 5-8 スカイライン分解アルゴリズム

下式の三重積の和から、次のように単純な積和に置き直すことができる。

$$\bar{l}_{ij} = g_{ij} - \sum_{k=1}^{j-1} \bar{l}_{ik} l_{jk} \quad (i > j) \qquad \cdots\cdots(5.33)$$

$$d_{ii} = g_{ii} - \sum_{k=1}^{i-1} \bar{l}_{ik} l_{ik} \qquad \cdots\cdots(5.34)$$

$$l_{ij} = \frac{\bar{l}_{ij}}{d_{jj}} \qquad \cdots\cdots(5.35)$$

スカイライン法の分解アルゴリズムでは、式(5.33)により第 i 行 j 列の下三角行列の計算過程を求めて保存し、この操作を i-1 列まで全て実施する。その後、式(5.34)を用いて対角行列の対角項を求める。最後に、式(5.35)を計算して、第 i 行の下三角行列成分を求めることになる。

以上のアルゴリズムを、流れ図を用いてさらに理解を深めよう。この処理は、スカイライン分解サブルーチンで実施される。

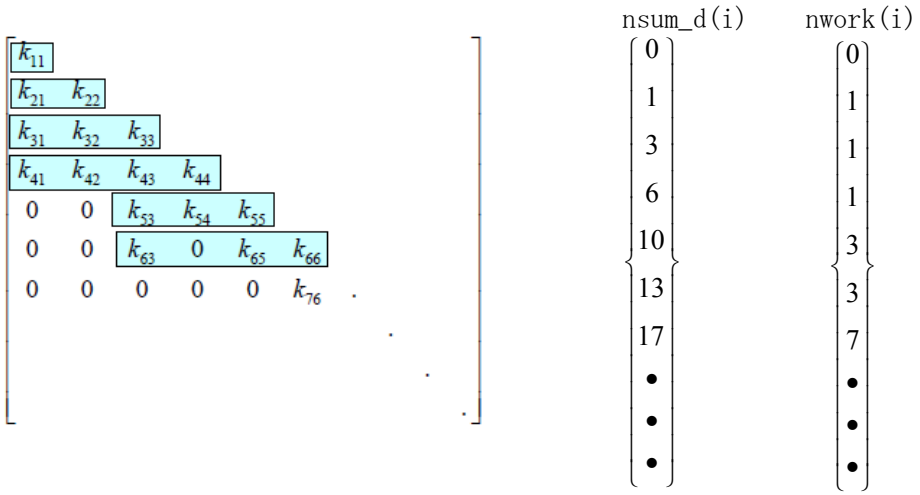


図 5-9 スカイライン行列と表の関係

ここで、nwork(i)はワーク領域であるが、ここでは、i行のスカイラインの開始列番号を表す。まず、このワーク領域であるnwork()をスカイライン表から作成する。フローチャートは左から右に続く。右側の部分で３重のループが掛かっていることが分かる。

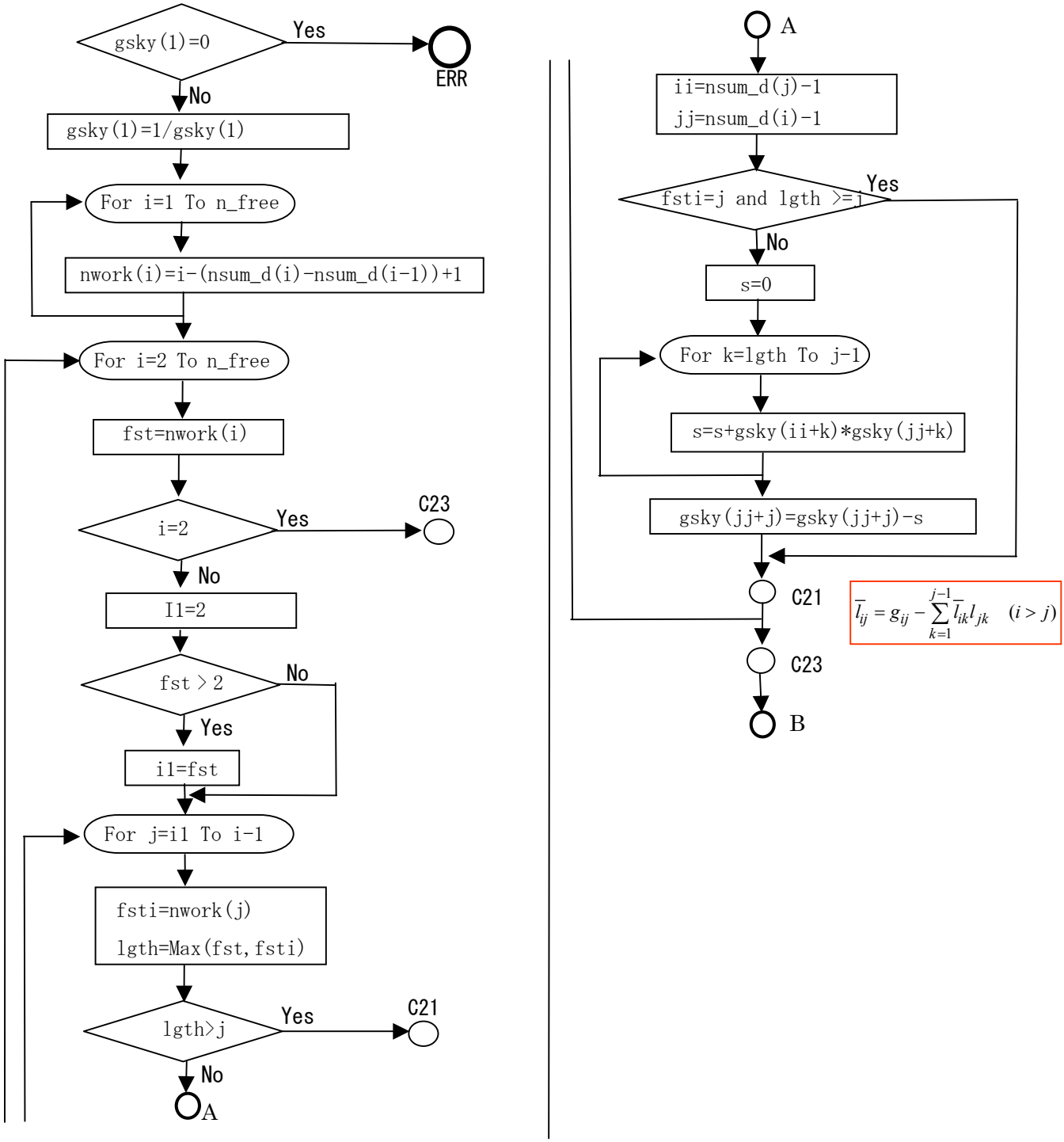


図 5-10(a)　スカイライン行列の分解処理の流れ図

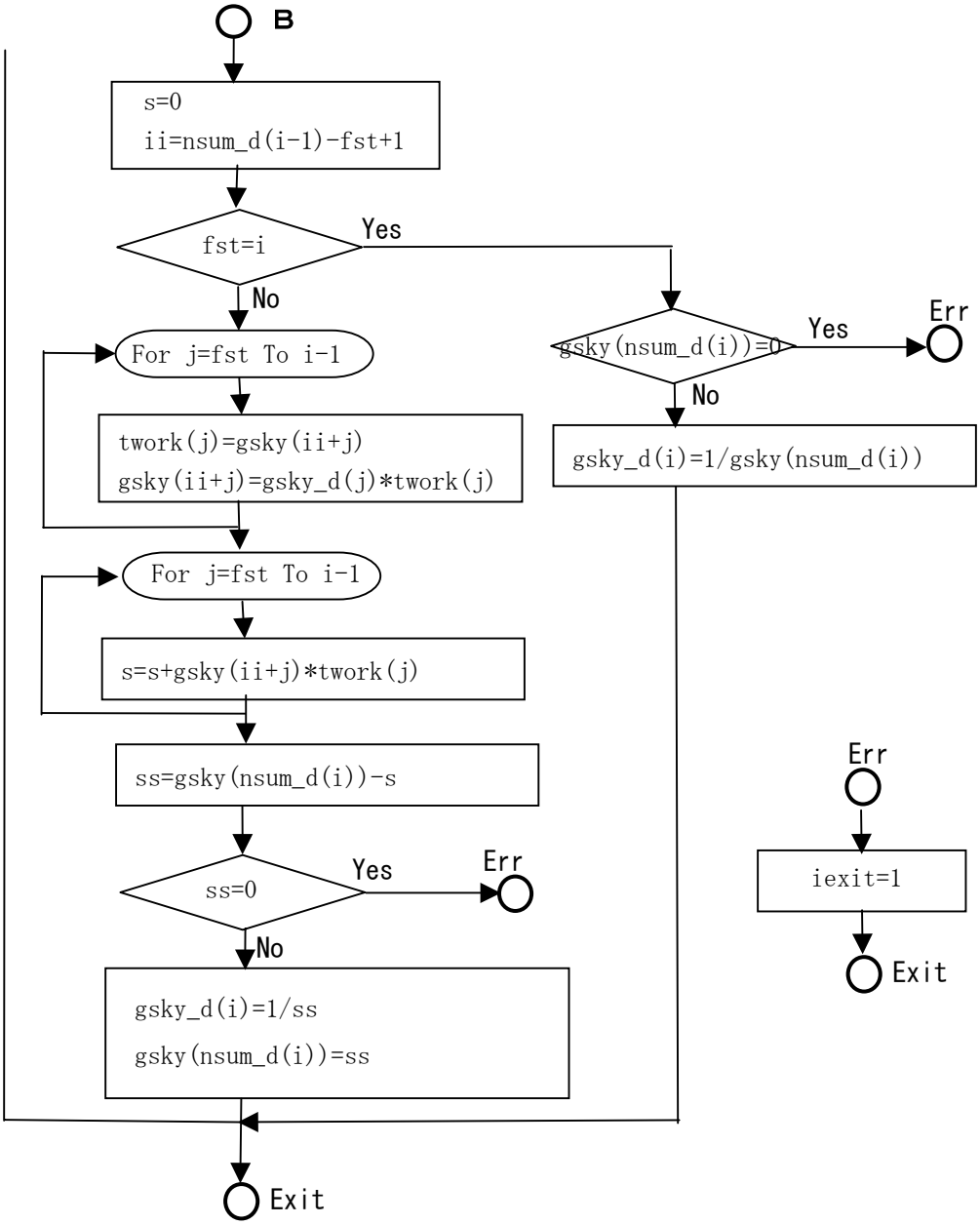


図 5-10(b) スカイライン行列の分解処理の流れ図

5.7 LDU 分解された係数行列を有する方程式の解法

係数行列が LDL^T 分解された線形方程式は以下のように表される。

$$[L][D][L]^T\{v\}=\{p\} \qquad \cdots\cdots(5.36)$$

ここで、

$$\left.\begin{aligned}[D][L]^T\{v\}&=\{x\}\\ [L]^T\{v\}&=[D]^{-1}\{x\}\end{aligned}\right\}\cdots\cdots(5.37)$$

とおくと、方程式は以下のようになる。上式を、成分を用いて表すと、第 i 行は以下のようになる。

$$\left.\begin{aligned}\sum_{k=1}^{i-1} l_{ik}x_k + x_i &= p_i \\ \sum_{k=n}^{i+1} l_{ik}v_k + v_i &= \frac{x_i}{d_{ii}}\end{aligned}\right\}\cdots\cdots(5.38)$$

さらに、移項すると、

$$\left.\begin{aligned}x_i &= p_i - \sum_{k=1}^{i-1} l_{ik}x_k \\ v_i &= \frac{x_i}{d_{ii}} - \sum_{k=n}^{i+1} l_{ki}v_k\end{aligned}\right\}\cdots\cdots(5.39)$$

上式が前進代入と後退代入の基礎式であり、前者は i=1 から n に向かって逐次演算することで解が求まり、また後者は i=n から 1 に向かって代入することで解が求められることになる。

前進代入法と後退代入法のアルゴリズムが、下の流れ図に示される。ワーク領域として、式(5.39)の上の x_i の値を、twork()に保存している。

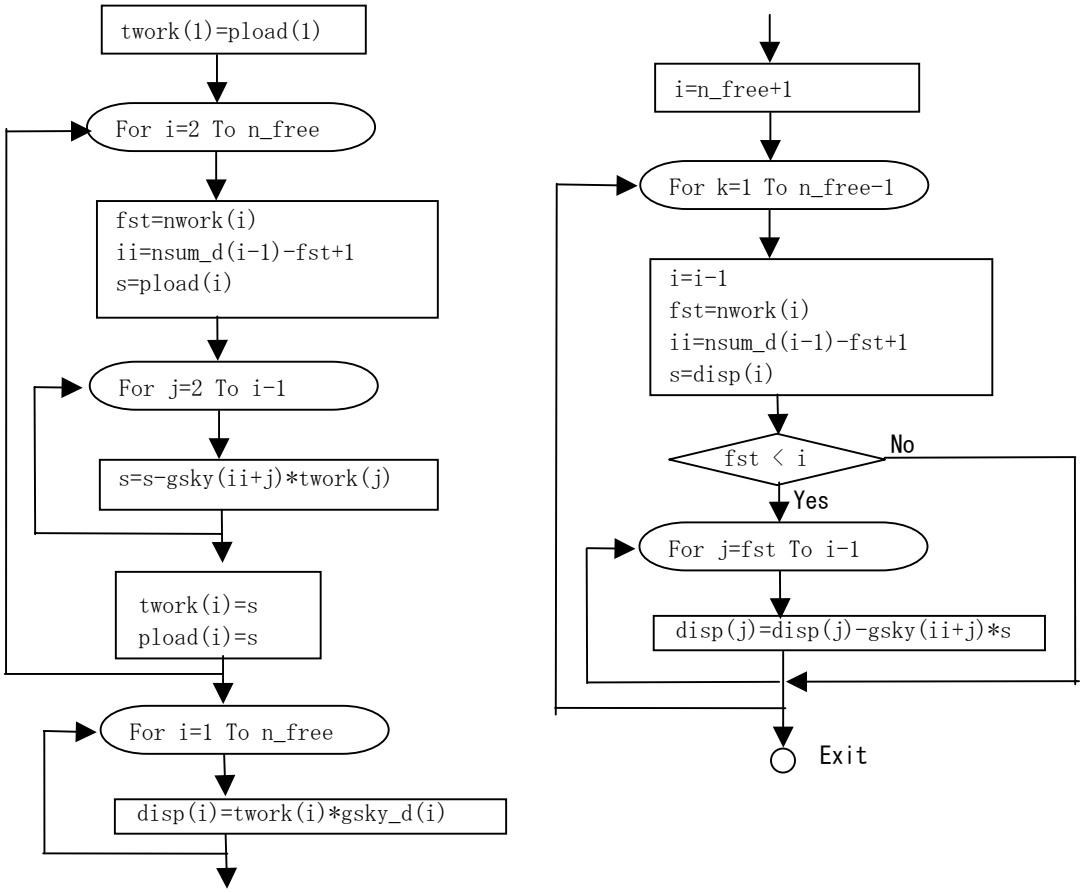


図 5-11 前進・後退代入法のアルゴリズムフローチャート

5.8 スカイライン行列の LDU 分解と方程式の解法プログラム

本節では、ここで示したスカイライン行列に関連したサブルーチンを示す。まず、スカイライン表を作成するサブルーチン Set_sky_table()、スカイライン行列に部材剛性行列を足しこむサブルーチン Build_sky_k()、スカイライン行列を LDU 分解するサブルーチン Decomp_sky()、方程式を解くサブルーチン solv_sky()である。前節までで説明した各サブルーチンの処理内容を参考に、プログラムの動作を良く理解されたい。

```
'-----------------------------------------------------
'       スカイライン行列用対角項番号表の作成
'-----------------------------------------------------
Private Sub Set_sky_table(n_free, F_rest, nsum_d, n_Member, Member,
n_skyline)
Dim i As Integer
Dim j As Integer
Dim k1 As Integer
Dim k2 As Integer
Dim iset As Integer
Dim i1 As Integer
Dim ff_rest(6)
'---------------------------------------引数仕様
'   n_free 自由度数
'   nsum_d() スカイライン用対角項の番号表
'---------------------------------------
  nsum_d(0) = 0
  For i = 1 To n_free
  nsum_d(i) = 1
  Next
  For i = 1 To n_Member
  i1 = Member(1, i)
  For j = 1 To 3
  ff_rest(j) = F_rest(j, i1)
  Next
  i1 = Member(2, i)
  For j = 1 To 3
  ff_rest(j + 3) = F_rest(j, i1)
  Next
'---------------------------------------
   For k = 1 To 6
    If (ff_rest(k) > 0) Then
     issi = ff_rest(k)
    For j = 1 To 6
     If (ff_rest(j) > 0) Then
      isss = ff_rest(j)
      If (issi > isss) Then
       iibw = Abs(isss - issi) + 1
       If (nsum_d(issi) < iibw) Then
       nsum_d(issi) = iibw
       End If
      End If
```

```
      End If
     Next
     End If
   Next
   Next
'---------------------------------------
 For i = 1 To n_free
 nsum_d(i) = nsum_d(i - 1) + nsum_d(i)
 Next
 n_skyline = nsum_d(n_free)      '最大スカイライン行列の数
End Sub
```

```
 '-----------------------------------------------------
 '       スカイライン剛性 gsky()の作成
 '-----------------------------------------------------
 Private Sub Build_sky_k(m, gsky, ak, F_rest, nsum_d, Member)
 Dim i As Integer
 Dim j As Integer
 Dim i1 As Integer
 Dim j1 As Integer
 Dim i2 As Integer
 Dim ff_rest(6)
 '---------------------------------------引数仕様
 '    m  部材番号
 '    gsky() スカイライン行列
 '    ak()   部材剛性行列
 '    F_irest() 節点拘束表
 '    nsum_d() スカイライン表
 '---------------------------------------部材両端の拘束表作成
   i1 = Member(1, m)
   For j = 1 To 3
   ff_rest(j) = F_rest(j, i1)
   Next
   i1 = Member(2, m)
   For j = 1 To 3
   ff_rest(j + 3) = F_rest(j, i1)
   Next
 '---------------------------------------
 For i = 1 To 6
  i1 = ff_rest(i)
  If (i1 > 0) Then
   For j = 1 To 6
     j1 = ff_rest(j)
     If (j1 > 0) Then
      If (j1 <= i1) Then
       i2 = nsum_d(i1) - (i1 - j1)
       gsky(i2) = gsky(i2) + ak(i, j)
      End If
     End If
   Next
  End If
 Next
 '---------------------------------------
 End Sub
```

```
'-------------------------------------------------------------------------------
'       2 つの値から大きい値を取り出す関数（Decomp_sky で使用）
'-------------------------------------------------------------------------------
Private Function Max_V(a, b)
Max_V = a
If (a < b) Then Max_V = b
End Function
```

```
'-------------------------------------------------------------------------------
'       スカイライン剛性の LDU 分解（分解後は同一領域に保存、対角項は gsky_d に保存）
'-------------------------------------------------------------------------------
Private Sub Decomp_sky(n_free, nsum_d, gsky, gsky_d, nwork, twork, iexit, istable)
Dim i As Integer
Dim j As Integer
Dim fst As Integer
Dim fsti As Integer
Dim lgth As Integer
Dim i1 As Integer
Dim ii As Integer
Dim jj As Integer
Dim s As Double
Dim ss As Double
'---------------------------------------引数仕様
'   n_free 自由度数
'   gsky() スカイライン行列
'   gsky_d() スカイライン行列分解　対角項
'   nsum_d() スカイライン用未知数表
'   nwork() ワーク領域
'   twork() ワーク領域
'   iexit  終了コード
'   istable 不安定次数
'---------------------------------------
'---------------------------------------
iexit = 1
i = 0
'---------------------------------------
If (gsky(1) = 0#) Then GoTo Err_C
gsky_d(1) = 1# / gsky(1)
For i = 1 To n_free
nwork(i) = i - (nsum_d(i) - nsum_d(i - 1)) + 1
Next
'---------------------------------------
For i = 2 To n_free
 fst = nwork(i)
 If (i = 2) Then GoTo C_23
  i1 = 2
  If (fst > 2) Then i1 = fst
  For j = i1 To i - 1
   fsti = nwork(j)
   lgth = Max_V(fst, fsti)
    If (lgth > j) Then GoTo C_21
    ii = nsum_d(j) - j
    jj = nsum_d(i) - i
```

```
        If (fsti = j) Then GoTo C_21
         If (lgth >= j) Then GoTo C_21
            s = 0#
          For k = lgth To j - 1
            s = s + gsky(ii + k) * gsky(jj + k)
          Next
          gsky(jj + j) = gsky(jj + j) - s
C_21:
  Next
C_23:
 '---------------------------------------
 s = 0#
 ii = nsum_d(i - 1) - fst + 1
 If (fst <> i) Then
  For j = fst To i - 1
   twork(j) = gsky(ii + j)
   gsky(ii + j) = gsky_d(j) * twork(j)
  Next
  For j = fst To i - 1
   s = s + gsky(ii + j) * twork(j)
  Next
  ss = gsky(nsum_d(i)) - s
  If (ss = 0#) Then GoTo Err_C
   gsky_d(i) = 1# / ss
   gsky(nsum_d(i)) = 1# / gsky_d(i)
 Else
  If (gsky(nsum_d(i)) = 0#) Then GoTo Err_C
     gsky_d(i) = 1# / gsky(nsum_d(i))
 End If
 Next
 '---------------------------------------
 iexit = 0
 istable = 0
 For i = 1 To n_free
 If (gsky_d(i) < 0#) Then istable = istable + 1
 Next
 GoTo End_cc
 '--------------------------------------- エラーコード１で終了
Err_C:
 iexit = 1
End_cc:
End Sub
```

```
'-----------------------------------------------------
'       方程式の解法（LDU 分解された係数行列から方程式の解を求める）
'-----------------------------------------------------
Private Sub solv_sky(n_free, nsum_d, gsky, gsky_d, nwork, twork, pload, disp)
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim fst As Integer
Dim ii As Integer
Dim s As Double
'---------------------------------------引数仕様
```

```
'    n_free 自由度数
'    gsky() スカイライン行列
'    gsky_d() スカイライン行列分解　対角項
'    nsum_d() スカイライン表
'    nwork() ワーク領域
'    twork() ワーク領域
'    pload() 荷重項
'    disp() 変位項
'---------------------------------------
'       前進代入
'---------------------------------------
  twork(1) = pload(1)
  For i = 2 To n_free
   fst = nwork(i)
   ii = nsum_d(i - 1) - fst + 1
   s = pload(i)
   For j = fst To i - 1
    s = s - gsky(ii + j) * twork(j)
   Next
   pload(i) = s
   twork(i) = s
  Next
'---------------------------------------
'       後退代入
'---------------------------------------
  For i = 1 To n_free
  disp(i) = twork(i) * gsky_d(i)
  Next
  i = n_free + 1
  For k = 1 To n_free - 1
   i = i - 1
   fst = nwork(i)
   ii = nsum_d(i - 1) - fst + 1
   s = disp(i)
   If (fst < i) Then
    For j = fst To i - 1
    disp(j) = disp(j) - gsky(ii + j) * s
    Next
   End If
  Next
'---------------------------------------
'F_cel = "A36"
'For i = 1 To n_free
' Range(F_cel).Offset(i, 0).Value = disp(i)
'Next
End Sub
```

5.9 課題

骨組の応力解析を実施するためには、連立方程式を解く必要がある。その計算コストとメモリーコストは、剛性行列をどのような形式で保存

するかにかかっている。ここでは、Excel の VBA ではあるが、一般に良く用いられているスカイライン行列を用い、その操作方法について学んだ。そこで、平面骨組プログラムにこのスカイライン行列に関連するサブルーチンを組み込むことにする。プログラムコードは既に示されているので、これを参照して解析プログラムに組み込む。さらに、簡単なモデルで、解析結果が適切かどうか確認されたい。

5.10 まとめ

本章では、一般に応力解析システムで使用されるスカイライン行列について学んだ。ここでは、スカイライン行列の LDU 分解法、連立方程式の解法、また、スカイライン行列の保持に必要な表について、その作成方法を詳細に学習した。また、これらのアルゴリズムをコード化し、その内容についても学んだ。