有限要素法特論

2004 年 12 月 20 日

非線形有限要素法特論

- 講義内容は，非線形有限要素法の解説
- 夏学期の,「非線形有限要素法の基礎」を受講していることが望まれる.
- 受講していない場合は,「非線形有限要素法のためのテンソル解析の基礎」久田俊明著，丸善を読んで理解しておくこと.
- 講義資料はhttp://www.sml.k.u-tokyo.ac.jp/members/nabe/lecture2004からダウンロードできるようにします。講義前日までに確定しますので各自プリントアウトして持参してください。
- http://www.sml.k.u-tokyo.ac.jp/members/nabe/lecture2003 から昨年度の講義ノートがダウンロードできます。講義名が「有限要素法特論」ですが、基本的に同じ内容です。
- 今年度（２００４）から新設した演習は、翌週の講義時間に提出してください。必須ではありませんし、採点しませんが、実際に手を動かして計算することは有限要素法の理解を深める上で重要です。
- 質問などは，nabe@sml.k.u-tokyo.ac.jp まで.

非線形有限要素法特論講義予定

1.	10/ 4	微分方程式の境界値問題の有限要素解析
2.	10/18	線形弾性体の有限要素解析
3.	10/25	アイソパラメトリックソリッド要素 (プログラム)
4.	11/ 1	連立一次方程式の数値解法と境界条件処理 （演習あり）
5.	11/ 8	線形有限要素法の基本的なプログラム構造 (プログラム)
6.	11/15	幾何学非線形問題の有限要素定式化１
7.	11/22	幾何学非線形問題の有限要素定式化２
8.	11/29	超弾性体、弾塑性体
9.	12/ 6	休講
10.	12/13	非線形方程式の動的解析手法、固有値解析、構造要素
11.	12/20	連立一次方程式の数値解法 — skyline 法、反復法
12.	1/17	ALE 有限要素流体解析
13.	1/24	ALE 有限要素流体解析

第**??**章でのべた Gauss の消去法は, 正方行列を対象としたものであった. それに対して有限要素法で得られる剛性マトリックスは, 一般的に対角項のまわりだけに非零のデータが集まる帯行列である. ここでは Gauss の消去法を合理化し実際の有限要素解析コードでよく用いられる skyline 法を紹介する.

1 Gauss の消去法の合理化

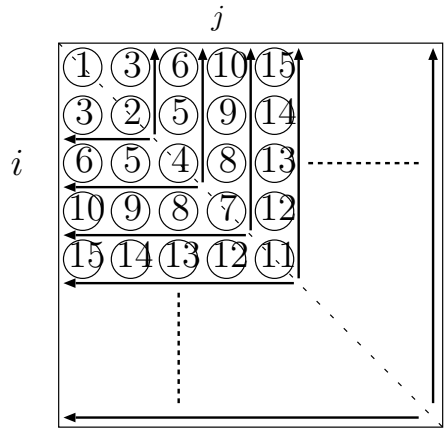
skyline法の詳細は後述するが, ここではskyline法で用いるマトリックスの記憶法を応用して Gauss の消去法の加速を行う. 第**??**章で述べたように fortran でも C でも多次元配列は内部的には 1 次元配列としてあつかわれるので, ループインデックスの回し方によって計算効率がかわり, まったく同じ演算量であるにもかかわらず, 2 次元配列を用いた対称マトリックス版の Gauss の消去法ではマトリックスの上三角を使うか, 下三角を使うかでパフォーマンスが変わった. また, 係数マトリックスが対称ならば三角分解を行うにしても上三角か下三角の領域で事足りるので 2 次元配列を用いてマトリックス全体を記憶するのは合理的ではない.

三角行列の記憶法にはいろいろなものが考えられるが, 有限要素法ではここで示すような方法を用いることが多く, 計算機メーカーから供給される数値計算ライブラリーには, この記憶法に直接対応しているものもある.

上三角と下三角用の二つの 1 次元配列を用意する. A_{11} から順に以下のように記憶していく.

上三角	下三角
au(1) $= A_{11}$	al(1) $= A_{11}$
au(2) $= A_{22}$	al(2) $= A_{22}$
au(3) $= A_{12}$	al(3) $= A_{21}$
au(4) $= A_{33}$	al(4) $= A_{33}$
au(5) $= A_{23}$	al(5) $= A_{32}$
au(6) $= A_{13}$	al(6) $= A_{31}$
au(7) $= A_{44}$	al(7) $= A_{44}$
$\vdots$	$\vdots$

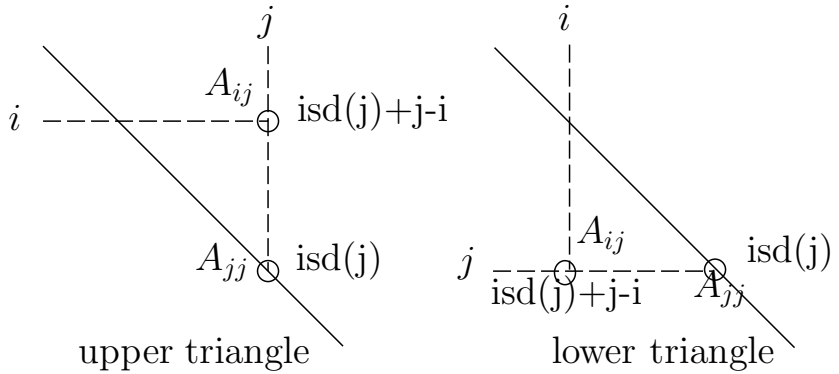
これは以下のような順で係数マトリックスを記憶したことを意味する. 前述の三角分解では最も内側のループは U_{kj} は列を固定, L_{ik} は行を固定してアクセスし内積演算を行うので, このように記憶しておけば上三角も下三角も極端にはなれたデータを参照することはない.



係数マトリックスの対角項 $A_{11}, A_{22}, A_{33}, A_{44}, A_{55}, \ldots$ は配列 au, al の $1, 2, 4, 7, 11, \ldots$ 番目に記憶

される. コーディングを容易にするためにこの対角項の位置も記憶する (配列名 i_sky_diagonal, 略称 isd) この配列を用いると上三角の i, j 成分は配列 au の isd(j)+j-i 番目のデータ, 下三角の j, i 成分は配列 al の isd(j)+j-i 番目のデータとしてアクセスできる. これから分かるように元のマトリクスで対称の位置にある成分は同じインデックスを用いてアクセスできるので一次元配列を用いても比較的分かりやすいコーディングを行うことができる. なお full マトリックスの場合 isd は以下のように計算される.

$$isd(i) = \frac{(i-1)i}{2} + 1 \quad (i = 1, \ldots, n)$$



skyline バージョン 0 としてこの配列を用いたコーディングを行う. 非対称版と対称版をそれぞれ作成し, 1000×1000 のマトリックスで cpu time を計測する. Gauss バージョン 1 に比べてかなり高速化できたことがわかるはずである. このように工夫することによりパフォーマンスが劇的にかわることが数値計算のおもしろさでもある. fortran でのコーディング例を示すと以下のようになる.

ただし, サブルーチン init2 は係数マトリックスを au, al に入れ替え isd をセットする. asky_ver0 (asymmetrical skyline), ssky_ver0 (symmetrical skyline) がそれぞれ非対称, 対称版であり, それ

ぞれ gauss_ver3, gauss_ver4, と同じループ構造で二次元配列 a(i,j), a(j,i) をそれぞれ、au(isd(j)+j-i), al(isd(j)+j-i) と置き換えている. (ただし, $i < j$)

これをさらに合理化したものが asky_ver01, ssky_ver01 である. 具体的な変更内容は, 三角分解の三重ループの内側のインデックスの計算の一部をループの外に出しただけであるが, OS やコンパイラーによってはかなり差が出るものもある. これは RISC cpu がもつスーパースカラーの機能をどれだけ効率よく使えるかにより発生している差であろう.

asky_ver0 :

for $k = 1 \sim i - 1$

$ki =$ `isd(i)` $+ i - k$

$kj =$ `isd(j)` $+ j - k$

$temp\_u +$ `al(ki)au(kj)`

$temp\_l +$ `al(kj)au(ki)`

end for

asky_ver01 :

$ki =$ `isd(i)` $+ i$

$kj =$ `isd(j)` $+ j$

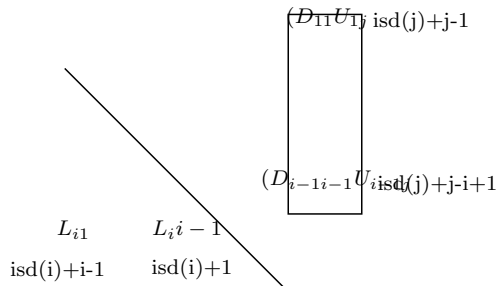
for $k = 1 \sim i - 1$

$temp\_u +$ `al(ki-k)au(kj-k)`

$temp\_l +$ `al(kj-k)au(ki-k)`

end for

上記のループは一般に「内積演算」と呼ばれ, `temp_u` なら `al` の `isd(i)+1` 番目から `isd(i)+i-1` 番目までと `au` の `isd(j)+j-i+1` 番目から `isd(i)+j-1` 番目までをそれぞれベクトルと見做してその内積をとっているとみることができる.



ループが降順になっているのは au, al は対角項から順に下から上へ, 右から左へ係数マトリックスを記憶しているのに対して定式化では Σ は 1 から順に, つまり上から下, 左から右へアクセスするにようになっているからである. これを昇順に変更したのが, asky_ver02, ssky_ver02 である. これにより計算機メーカーなどから供給される BLAS などに代表される数値演算ライブラリーを活用しやすくなり, また, コンパイラーによっては自動的にライブラリールーチンに置き換えるよう最適化を行うものもある.

asky_ver02 :

$ki = \texttt{isd(i)} + i - i$

$kj = \texttt{isd(j)} + j - i$

for $k = 1 \sim i - 1$

$temp\_u + \texttt{al(ki+k)au(kj+k)}$

$temp\_l + \texttt{al(kj+k)au(ki+k)}$

end for

2　skyline法の基礎

有限要素法の剛性マトリックスの特徴としては

- 一般に次元数が大きい
- 零の成分が多い (sparse)
- 非零の成分は対角項の近くに集まっている (band form)
- 通常の構造解析の問題では対称となる
- 非対称な剛性マトリックスでも対角項を中心として対称な位置に非零の成分は存在する.

などが挙げられる.

これらの特徴を利用してGaussの消去法を改良したのが以下に述べるskyline法である.

例として以下の剛性マトリックスが得られたとする

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & K_{15} & 0 & 0 & K_{18} \\ K_{21} & K_{22} & K_{23} & K_{24} & 0 & 0 & 0 & K_{28} \\ 0 & K_{32} & K_{33} & K_{34} & 0 & 0 & K_{37} & 0 \\ 0 & K_{42} & K_{43} & K_{44} & 0 & K_{46} & K_{47} & 0 \\ K_{51} & 0 & 0 & 0 & K_{55} & K_{56} & K_{57} & 0 \\ 0 & 0 & 0 & K_{64} & K_{65} & K_{66} & K_{67} & 0 \\ 0 & 0 & K_{73} & K_{74} & K_{75} & K_{76} & K_{77} & K_{78} \\ K_{81} & K_{82} & 0 & 0 & 0 & 0 & K_{87} & K_{88} \end{bmatrix} \tag{1}$$

各列 (または行) に対して非零の成分が存在する最小の行 (または列) の番号を並べたものを $m_j \quad (j = 1 \ldots n)$ とする. 例えば式 (1) では $m_j = \{1, 1, 2, 2, 1, 4, 3, 1\}$ である.

この m_j は $[K]$ を LDU 分解したときに得られる $[L]$, $[U]$ についてもかわらない.

実際に式 (1) を三角分解すると以下のようになる.

$$\left[K^{(1)}\right] = \begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} & 0 & 0 & K_{15}^{(1)} & 0 & 0 & K_{18}^{(1)} \\ K_{21}^{(1)} & K_{22}^{(1)} & K_{23}^{(1)} & K_{24}^{(1)} & 0 & 0 & 0 & K_{28}^{(1)} \\ 0 & K_{32}^{(1)} & K_{33}^{(1)} & K_{34}^{(1)} & 0 & 0 & K_{37}^{(1)} & 0 \\ 0 & K_{42}^{(1)} & K_{43}^{(1)} & K_{44}^{(1)} & 0 & K_{46}^{(1)} & K_{47}^{(1)} & 0 \\ K_{51}^{(1)} & 0 & 0 & 0 & K_{55}^{(1)} & K_{56}^{(1)} & K_{57}^{(1)} & 0 \\ 0 & 0 & 0 & K_{64}^{(1)} & K_{65}^{(1)} & K_{66}^{(1)} & K_{67}^{(1)} & 0 \\ 0 & 0 & K_{73}^{(1)} & K_{74}^{(1)} & K_{75}^{(1)} & K_{76}^{(1)} & K_{77}^{(1)} & K_{78}^{(1)} \\ K_{81}^{(1)} & K_{82}^{(1)} & 0 & 0 & 0 & 0 & K_{87}^{(1)} & K_{88}^{(1)} \end{bmatrix} \tag{2}$$

$$\left[L_1^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -L_{21} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -L_{51} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -L_{81} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\left[K^{(2)}\right] = \left[L_1^{-1}\right]\left[K^{(1)}\right] = \begin{bmatrix} K_{11}^{(2)} & K_{12}^{(2)} & 0 & 0 & K_{15}^{(2)} & 0 & 0 & K_{18}^{(2)} \\ 0 & K_{22}^{(2)} & K_{23}^{(2)} & K_{24}^{(2)} & K_{25}^{(2)} & 0 & 0 & K_{28}^{(2)} \\ 0 & K_{32}^{(2)} & K_{33}^{(2)} & K_{34}^{(2)} & 0 & 0 & K_{37}^{(2)} & 0 \\ 0 & K_{42}^{(2)} & K_{43}^{(2)} & K_{44}^{(2)} & 0 & K_{46}^{(2)} & K_{47}^{(2)} & 0 \\ 0 & K_{52}^{(2)} & 0 & 0 & K_{55}^{(2)} & K_{56}^{(2)} & K_{57}^{(2)} & 0 \\ 0 & 0 & 0 & K_{64}^{(2)} & K_{65}^{(2)} & K_{66}^{(2)} & K_{67}^{(2)} & 0 \\ 0 & 0 & K_{73}^{(2)} & K_{74}^{(2)} & K_{75}^{(2)} & K_{76}^{(2)} & K_{77}^{(2)} & K_{78}^{(2)} \\ 0 & K_{82}^{(2)} & 0 & 0 & K_{85}^{(2)} & 0 & K_{87}^{(2)} & K_{88}^{(2)} \end{bmatrix} \tag{4}$$

$$\left[L_2^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -L_{32} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -L_{42} & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -L_{52} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -L_{82} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\left[K^{(3)}\right] = \left[L_2^{-1}\right]\left[K^{(2)}\right] = \begin{bmatrix} K_{11}^{(3)} & K_{12}^{(3)} & 0 & 0 & K_{15}^{(3)} & 0 & 0 & K_{18}^{(3)} \\ 0 & K_{22}^{(3)} & K_{23}^{(3)} & K_{24}^{(3)} & K_{25}^{(3)} & 0 & 0 & K_{28}^{(3)} \\ 0 & 0 & K_{33}^{(3)} & K_{34}^{(3)} & K_{35}^{(3)} & 0 & K_{37}^{(3)} & K_{38}^{(3)} \\ 0 & 0 & K_{43}^{(3)} & K_{44}^{(3)} & K_{45}^{(3)} & K_{46}^{(3)} & K_{47}^{(3)} & K_{48}^{(3)} \\ 0 & 0 & K_{53}^{(3)} & K_{54}^{(3)} & K_{55}^{(3)} & K_{56}^{(3)} & K_{57}^{(3)} & K_{58}^{(3)} \\ 0 & 0 & 0 & K_{64}^{(3)} & K_{65}^{(3)} & K_{66}^{(3)} & K_{67}^{(3)} & 0 \\ 0 & 0 & K_{73}^{(3)} & K_{74}^{(3)} & K_{75}^{(3)} & K_{76}^{(3)} & K_{77}^{(3)} & K_{78}^{(3)} \\ 0 & 0 & K_{83}^{(3)} & K_{84}^{(3)} & K_{85}^{(3)} & 0 & K_{87}^{(3)} & K_{88}^{(3)} \end{bmatrix} \tag{6}$$

$$\left[L_3^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -L_{43} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -L_{53} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -L_{73} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -L_{83} & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\left[K^{(4)}\right] = \left[L_3^{-1}\right]\left[K^{(3)}\right] = \begin{bmatrix} K_{11}^{(4)} & K_{12}^{(4)} & 0 & 0 & K_{15}^{(4)} & 0 & 0 & K_{18}^{(4)} \\ 0 & K_{22}^{(4)} & K_{23}^{(4)} & K_{24}^{(4)} & K_{25}^{(4)} & 0 & 0 & K_{28}^{(4)} \\ 0 & 0 & K_{33}^{(4)} & K_{34}^{(4)} & K_{35}^{(4)} & 0 & K_{37}^{(4)} & K_{38}^{(4)} \\ 0 & 0 & 0 & K_{44}^{(4)} & K_{45}^{(4)} & K_{46}^{(4)} & K_{47}^{(4)} & K_{48}^{(4)} \\ 0 & 0 & 0 & K_{54}^{(4)} & K_{55}^{(4)} & K_{56}^{(4)} & K_{57}^{(4)} & K_{58}^{(4)} \\ 0 & 0 & 0 & K_{64}^{(4)} & K_{65}^{(4)} & K_{66}^{(4)} & K_{67}^{(4)} & 0 \\ 0 & 0 & 0 & K_{74}^{(4)} & K_{75}^{(4)} & K_{76}^{(4)} & K_{77}^{(4)} & K_{78}^{(4)} \\ 0 & 0 & 0 & K_{84}^{(4)} & K_{85}^{(4)} & 0 & K_{87}^{(4)} & K_{88}^{(4)} \end{bmatrix} \tag{8}$$

$$\left[L_4^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -L_{54} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -L_{64} & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -L_{74} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -L_{84} & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$\left[K^{(5)}\right] = \left[L_4^{-1}\right]\left[K^{(4)}\right] = \begin{bmatrix} K_{11}^{(5)} & K_{12}^{(5)} & 0 & 0 & K_{15}^{(5)} & 0 & 0 & K_{18}^{(5)} \\ 0 & K_{22}^{(5)} & K_{23}^{(5)} & K_{24}^{(5)} & K_{25}^{(5)} & 0 & 0 & K_{28}^{(5)} \\ 0 & 0 & K_{33}^{(5)} & K_{34}^{(5)} & K_{35}^{(5)} & 0 & K_{37}^{(5)} & K_{38}^{(5)} \\ 0 & 0 & 0 & K_{44}^{(5)} & K_{45}^{(5)} & K_{46}^{(5)} & K_{47}^{(5)} & K_{48}^{(5)} \\ 0 & 0 & 0 & 0 & K_{55}^{(5)} & K_{56}^{(5)} & K_{57}^{(5)} & K_{58}^{(5)} \\ 0 & 0 & 0 & 0 & K_{65}^{(5)} & K_{66}^{(5)} & K_{67}^{(5)} & K_{68}^{(5)} \\ 0 & 0 & 0 & 0 & K_{75}^{(5)} & K_{76}^{(5)} & K_{77}^{(5)} & K_{78}^{(5)} \\ 0 & 0 & 0 & 0 & K_{85}^{(5)} & K_{86}^{(5)} & K_{87}^{(5)} & K_{88}^{(5)} \end{bmatrix} \tag{10}$$

$$\left[L_5^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -L_{65} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -L_{75} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -L_{85} & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$\left[K^{(6)}\right] = \left[L_5^{-1}\right]\left[K^{(5)}\right] = \begin{bmatrix} K_{11}^{(6)} & K_{12}^{(6)} & 0 & 0 & K_{15}^{(6)} & 0 & 0 & K_{18}^{(6)} \\ 0 & K_{22}^{(6)} & K_{23}^{(6)} & K_{24}^{(6)} & K_{25}^{(6)} & 0 & 0 & K_{28}^{(6)} \\ 0 & 0 & K_{33}^{(6)} & K_{34}^{(6)} & K_{35}^{(6)} & 0 & K_{37}^{(6)} & K_{38}^{(6)} \\ 0 & 0 & 0 & K_{44}^{(6)} & K_{45}^{(6)} & K_{46}^{(6)} & K_{47}^{(6)} & K_{48}^{(6)} \\ 0 & 0 & 0 & 0 & K_{55}^{(6)} & K_{56}^{(6)} & K_{57}^{(6)} & K_{58}^{(6)} \\ 0 & 0 & 0 & 0 & 0 & K_{66}^{(6)} & K_{67}^{(6)} & K_{68}^{(6)} \\ 0 & 0 & 0 & 0 & 0 & K_{76}^{(6)} & K_{77}^{(6)} & K_{78}^{(6)} \\ 0 & 0 & 0 & 0 & 0 & K_{86}^{(6)} & K_{87}^{(6)} & K_{88}^{(6)} \end{bmatrix} \tag{12}$$

$$\left[L_6^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -L_{76} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -L_{86} & 0 & 1 \end{bmatrix} \tag{13}$$

$$\left[K^{(7)}\right] = \left[L_6^{-1}\right]\left[K^{(6)}\right] = \begin{bmatrix} K_{11}^{(7)} & K_{12}^{(7)} & 0 & 0 & K_{15}^{(7)} & 0 & 0 & K_{18}^{(7)} \\ 0 & K_{22}^{(7)} & K_{23}^{(7)} & K_{24}^{(7)} & K_{25}^{(7)} & 0 & 0 & K_{28}^{(7)} \\ 0 & 0 & K_{33}^{(7)} & K_{34}^{(7)} & K_{35}^{(7)} & 0 & K_{37}^{(7)} & K_{38}^{(7)} \\ 0 & 0 & 0 & K_{44}^{(7)} & K_{45}^{(7)} & K_{46}^{(7)} & K_{47}^{(7)} & K_{48}^{(7)} \\ 0 & 0 & 0 & 0 & K_{55}^{(7)} & K_{56}^{(7)} & K_{57}^{(7)} & K_{58}^{(7)} \\ 0 & 0 & 0 & 0 & 0 & K_{66}^{(7)} & K_{67}^{(7)} & K_{68}^{(7)} \\ 0 & 0 & 0 & 0 & 0 & 0 & K_{77}^{(7)} & K_{78}^{(7)} \\ 0 & 0 & 0 & 0 & 0 & 0 & K_{87}^{(7)} & K_{88}^{(7)} \end{bmatrix} \tag{14}$$

$$\left[L_7^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -L_{87} & 1 \end{bmatrix} \tag{15}$$

$$\left[K^{(8)}\right] = \left[L_7^{-1}\right]\left[K^{(7)}\right] = \begin{bmatrix} K_{11}^{(8)} & K_{12}^{(8)} & 0 & 0 & K_{15}^{(8)} & 0 & 0 & K_{18}^{(8)} \\ 0 & K_{22}^{(8)} & K_{23}^{(8)} & K_{24}^{(8)} & K_{25}^{(8)} & 0 & 0 & K_{28}^{(8)} \\ 0 & 0 & K_{33}^{(8)} & K_{34}^{(8)} & K_{35}^{(8)} & 0 & K_{37}^{(8)} & K_{38}^{(8)} \\ 0 & 0 & 0 & K_{44}^{(8)} & K_{45}^{(8)} & K_{46}^{(8)} & K_{47}^{(8)} & K_{48}^{(8)} \\ 0 & 0 & 0 & 0 & K_{55}^{(8)} & K_{56}^{(8)} & K_{57}^{(8)} & K_{58}^{(8)} \\ 0 & 0 & 0 & 0 & 0 & K_{66}^{(8)} & K_{67}^{(8)} & K_{68}^{(8)} \\ 0 & 0 & 0 & 0 & 0 & 0 & K_{77}^{(8)} & K_{78}^{(8)} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_{88}^{(8)} \end{bmatrix} \tag{16}$$

したがって $[L]$ は以下のようになる.

$$[L] = [L_1]\,[L_2]\ldots[L_7] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ L_{21} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_{32} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_{42} & L_{43} & 1 & 0 & 0 & 0 & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & L_{64} & L_{65} & 1 & 0 & 0 \\ 0 & 0 & L_{73} & L_{74} & L_{75} & L_{76} & 1 & 0 \\ L_{81} & L_{82} & L_{83} & L_{84} & L_{85} & L_{86} & L_{87} & 1 \end{bmatrix} \tag{17}$$

$[K^{(8)}] = [D][U]$ であることを考えれば, $[L]$, $[U]$ ともに, $m_j = 1,1,2,2,1,4,3$ ともとの $[K]$ の m_j と同じであることが分かる. しかしながら, m_j より対角項に近い成分, たとえば $K_{25}, K_{35}, K_{45}, K_{38}, K_{48}, K_{58}, K_{68}$ については, 三角分解により一般には非零の値になる.

この性質を用いて Gauss の消去法の合理化を図る. この例について $[L]$, $[U]$ は以下のようになる.

$$[L] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ L_{21} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_{32} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_{42} & L_{43} & 1 & 0 & 0 & 0 & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & L_{64} & L_{65} & 1 & 0 & 0 \\ 0 & 0 & L_{73} & L_{74} & L_{75} & L_{76} & 1 & 0 \\ L_{81} & L_{82} & L_{83} & L_{84} & L_{85} & L_{86} & L_{87} & 1 \end{bmatrix} \tag{18}$$

$$[U] = \begin{bmatrix} 1 & U_{12} & 0 & 0 & U_{15} & 0 & 0 & U_{18} \\ 0 & 1 & U_{23} & U_{24} & U_{25} & 0 & 0 & U_{28} \\ 0 & 0 & 1 & U_{34} & U_{35} & 0 & U_{37} & U_{38} \\ 0 & 0 & 0 & 1 & U_{45} & U_{46} & U_{47} & U_{48} \\ 0 & 0 & 0 & 0 & 1 & U_{56} & U_{57} & U_{58} \\ 0 & 0 & 0 & 0 & 0 & 1 & U_{67} & U_{68} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & U_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

この $[L], [U]$ を用いて $[K] = [L][D][U]$ を改めて計算すると以下のようになる.

1, 2, 3 列

$$\begin{bmatrix} D_{11} & D_{11}U_{12} & 0 \\ L_{21}D_{11} & L_{21}D_{11}U_{12}+D_{22} & D_{22}U_{23} \\ 0 & L_{32}D_{22} & L_{32}D_{22}U_{23}+D_{33} \\ 0 & L_{42}D_{22} & L_{42}D_{22}U_{23}+L_{43}D_{33} \\ L_{51}D_{11} & L_{51}D_{11}U_{12}+L_{52}D_{22} & L_{52}D_{22}U_{23}+L_{53}D_{33} \\ 0 & 0 & 0 \\ 0 & 0 & L_{73}D_{33} \\ L_{81}D_{11} & L_{81}D_{11}U_{12}+L_{82}D_{22} & L_{82}D_{22}U_{23}+L_{83}D_{33} \end{bmatrix} \tag{20}$$

4, 5 列

$$\begin{bmatrix} D_{11}U_{15} & 0 \\ L_{21}D_{11}U_{15}+D_{22}U_{25} & 0 \\ L_{32}D_{22}U_{25}+D_{33}U_{35} & 0 \\ L_{42}D_{22}U_{25}+L_{43}D_{33}U_{35}+D_{44}U_{45} & D_{44}U_{46} \\ L_{51}D_{11}U_{15}+L_{52}D_{22}U_{25}+L_{53}D_{33}U_{35}+L_{54}D_{44}U_{45}+D_{55} & L_{54}D_{44}U_{46}+D_{55}U_{56} \\ L_{64}D_{44}U_{45}+L_{65}D_{55} & L_{64}D_{44}U_{46}+L_{65}D_{55}U_{56}+D_{66} \\ L_{73}D_{33}U_{35}+L_{74}D_{44}U_{45}+L_{75}D_{55} & L_{74}D_{44}U_{46}+L_{75}D_{55}U_{56}+L_{76}D_{66} \\ L_{81}D_{11}U_{15}+L_{82}D_{22}U_{25}+L_{83}D_{33}U_{35}+L_{84}D_{44}U_{45}+L_{85}D_{55} & L_{84}D_{44}U_{46}+L_{85}D_{55}U_{56}+L_{86}D_{66} \end{bmatrix} \tag{21}$$

7 列

$$\begin{bmatrix} 0 \\ 0 \\ D_{33}U_{37} \\ L_{43}D_{33}U_{27} + D_{44}U_{47} \\ L_{53}D_{33}U_{37} + L_{54}D_{44}U_{47} + D_{55}U_{57} \\ L_{64}D_{44}U_{47} + L_{65}D_{55}U_{57} + D_{66}U_{67} \\ L_{73}D_{33}U_{35} + L_{74}D_{44}U_{45} + L_{75}D_{55}U_{57} + L_{76}D_{66}U_{67} + D_{77} \\ L_{83}D_{33}U_{37} + L_{84}D_{44}U_{47} + L_{85}D_{55}U_{57} + L_{86}D_{66}U_{67} + L_{87}D_{77} \end{bmatrix} \tag{22}$$

8 列

$$\begin{bmatrix} D_{11}U_{18} \\ L_{21}D_{11}U_{18} + D_{22}U_{28} \\ L_{32}D_{22}U_{28} + D_{33}U_{38} \\ L_{42}D_{22}U_{28} + L_{43}D_{33}U_{38} + D_{44}U_{48} \\ L_{51}D_{11}U_{18} + L_{52}D_{22}U_{28} + L_{53}D_{33}U_{38} + L_{52}D_{44}U_{48} + D_{55}U_{58} \\ L_{64}D_{44}U_{48} + L_{65}D_{55}U_{58} + D_{66}U_{68} \\ L_{73}D_{33}U_{38} + L_{74}D_{44}U_{48} + L_{75}D_{55}U_{58} + L_{76}D_{66}U_{68} + D_{77}U_{78} \\ L_{81}D_{11}U_{18} + L_{82}D_{22}U_{28} + L_{83}D_{33}U_{38} + L_{84}D_{44}U_{48} + L_{85}D_{55}U_{58} + L_{86}D_{66}U_{68} + L_{87}D_{77}U_{78} + D_{88} \end{bmatrix} \tag{23}$$

これは一般に以下のようにインデックスを用いて表すことができる.

$$\left\{ \begin{array}{l} K_{ij} = D_{ii}U_{ij} \quad \text{上三角} \\ K_{ji} = L_{ji}D_{ii} \quad \text{下三角} \end{array} \right\} \quad i = m_j$$

$$\left\{ \begin{array}{l} K_{ij} = \displaystyle\sum_{k=\max(m_i,m_j)}^{i-1} L_{ik}D_{kk}U_{kj} + D_{ii}U_{ij} \quad \text{上三角} \\ K_{ji} = \displaystyle\sum_{k=\max(m_i,m_j)}^{i-1} L_{jk}D_{kk}U_{ki} + L_{ji}D_{ii} \quad \text{下三角} \end{array} \right\} \quad i = m_j + 1 \ldots j - 1$$

$$K_{jj} = \sum_{k=m_j}^{j-1} L_{jk}D_{kk}U_{kj} + D_{jj} \quad \text{対角項}$$

この関係を用いると, $[L], [D], [U]$ は以下のように順次計算できる.

$$\begin{aligned} &\textbf{for} \quad j = 1 \\ &\qquad D_{11} = K_{11} \\ &\textbf{for} \quad j = 2 \\ &\qquad U_{12} = K_{12}/D_{11} \\ &\qquad L_{21} = K_{21}/D_{11} \\ &\qquad D_{22} = K_{22} - L_{21}D_{11}U_{12} \end{aligned}$$

for $j = 3 \sim n$

for $i = m_j + 1 \sim j - 1$

for $k = \max(m_i, m_j) \sim i - 1$

$temp\_u = temp\_u + L_{ik}\{D_{kk}U_{kj}\}$

$temp\_l = temp\_l + \{L_{jk}D_{kk}\}U_{ki}$

end for

$(D_{ii}U_{ij}) = K_{ij} - temp\_u$

$(L_{ji}D_{ii}) = K_{ji} - temp\_l$

end for

for $i = m_j \sim j - 1$

$U_{ij} = (D_{ii}U_{ij})/D_{ii}$

$L_{ji} = (L_{ji}D_{ii})/D_{ii}$

end for

for $k = m_j \sim j - 1$

$temp = temp + L_{jk}D_{kk}U_{kj}$

end for

$D_{jj} = K_{jj} - temp$

end for

以上で得られた三角行列から未知ベクトルを求める. ここでは, 下三角行列は前進代入, 上三角行列は後退消去を用いる. 上下三角行列の対角項は 1 であることを考えると, 以下のコーディングになる.

下三角行列 (前進代入)

for $i = 1$

$y_1 = c_1$

for $i = 2 \sim n$

for $k = m_i \sim i - 1$

$temp = temp + L_{ik} y_k$

end for

$x_i = c_i - temp$

end for

対角項

for $i = 1 \sim n$

$y_i = x_i / D_{ii}$

end for

上三角行列 (後退消去)

for $i = n$

$b_n = y_n$

for $i = n \sim 2 step = -1$

for $j = m_i \sim i - 1$

$y_j = y_j - U_{ji} b_i$

end for

$b_{i-1} = y_{i-1}$

end for

有限要素法の剛性マトリックスは対角項付近のみ非零で, 残りの成分が零である場合が多いため, このような手続きにより大幅に効率化ができる.

3 skyline 法のコーディング

skyline 法では剛性マトリックスを 1 次元配列に記憶する. 例として式 (1) を用いると, マトリックスの成分を以下のような順に記憶する.

$$\left[K^{(1)}\right] = \begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} & 0 & 0 & K_{15}^{(1)} & 0 & 0 & K_{18}^{(1)} \\ K_{21}^{(1)} & K_{22}^{(1)} & K_{23}^{(1)} & K_{24}^{(1)} & 0 & 0 & 0 & K_{28}^{(1)} \\ 0 & K_{32}^{(1)} & K_{33}^{(1)} & K_{34}^{(1)} & 0 & 0 & K_{37}^{(1)} & 0 \\ 0 & K_{42}^{(1)} & K_{43}^{(1)} & K_{44}^{(1)} & 0 & K_{46}^{(1)} & K_{47}^{(1)} & 0 \\ K_{51}^{(1)} & 0 & 0 & 0 & K_{55}^{(1)} & K_{56}^{(1)} & K_{57}^{(1)} & 0 \\ 0 & 0 & 0 & K_{64}^{(1)} & K_{65}^{(1)} & K_{66}^{(1)} & K_{67}^{(1)} & 0 \\ 0 & 0 & K_{73}^{(1)} & K_{74}^{(1)} & K_{75}^{(1)} & K_{76}^{(1)} & K_{77}^{(1)} & K_{78}^{(1)} \\ K_{81}^{(1)} & K_{82}^{(1)} & 0 & 0 & 0 & 0 & K_{87}^{(1)} & K_{88}^{(1)} \end{bmatrix} \tag{24}$$

```
[ au(1),al(1)     au(3)                                  au(13)                                  au(29) ]
[    al(3)     au(2),al(2)     au(5)       au(8)        au(12)                                  au(28) ]
[               al(5)      au(4),al(4)     au(7)        au(11)                 au(21)           au(27) ]
[               al(8)        al(7)     au(6),al(6)     au(10)       au(16)    au(20)           au(26) ]
[   al(13)      al(12)       al(11)       al(10)     au(9),al(9)     au(15)    au(19)           au(25) ]
[                                         al(16)       al(15)   au(14),al(14)  au(18)           au(24) ]
[                            al(21)       al(20)       al(19)       al(18)  au(17),al(17)       au(23) ]
[   al(29)       al(28)      al(27)       al(26)       al(25)       al(24)     al(23)     au(22),al(22) ]
```
(25)

記憶すべき非零の部分の形状がビルの輪郭線 (skyline) に見えることから一般に skykine 法と呼ばれている. また skyline と対角項に挟まれた部分をプロファイルと呼ぶ.

係数マトリックスの対角項 $K_{11}, K_{22}, \ldots$ は配列 `au, al` の 1, 2, 4, 6, ... 番目に記憶されている. そこで対角項の位置を記憶した配列 `i_sky_diagonal` (略称 `isd`) を用意する. j 列目では, $K_{m_j j}$ から K_{jj} までの $j - m_j + 1$ 個の成分を記憶する必要があるので `isd` は以下のように定められる.

$$\texttt{isd(1)} = 1 \qquad\qquad \texttt{isd(j+1)} = \texttt{isd(j)} + j - m_j + 1 \qquad (j = 1, \ldots, n)$$

なお, $j - m_j + 1$ を列の高さと呼ぶ. ここで $j = 1, \ldots, n+1$ としたのは, m_n を同じ手続きで求めるためである. 三角分解などで用いる m_j は

$$m_j = j - \texttt{isd(j+1)} + \texttt{isd(j)} + 1$$

として求められる.

また isd を用いると, 上三角の i, j 成分, 下三角の j, i 成分はそれぞれ au, al の isd(j)+j-i 番目の要素としてアクセスできる.

課題

以上に説明したことをもとに skyline バージョン 1 を作成する. 以下のような m 重対角行列を係数マトリックスにもつ連立一次方程式を取り上げる. (例では $m = 4$)

$$\left[K^{(1)}\right] = \begin{bmatrix} 1 & 2 & 3 & 4 & 0 & 0 & & \dots & & 0 \\ 2 & 2 & 3 & 4 & 5 & 0 & & & & 0 \\ 3 & 3 & 3 & 4 & 5 & 6 & & & & 0 \\ 4 & 4 & 4 & 4 & 5 & 6 & & & & 0 \\ 0 & 5 & 5 & 5 & 5 & 6 & & & & \\ 0 & 0 & 6 & 6 & 6 & 6 & & & & \vdots \\ & & & & & & \ddots & & & 0 \\ \vdots & & & & & & & & & n \\ & & & & & & & & & n \\ & & & & & & & & & n \\ 0 & 0 & 0 & 0 & & \dots & 0 & n & n & n & n \end{bmatrix} \tag{26}$$

Gauss の消去法バージョン 1 と同様に $\{b\} = \{1, 2, 3, \dots, n\}^T$ として

$$[A]\{b\} = \{c\} \tag{27}$$

により $\{c\}$ を求める. この $\{c\}$ を用いて $[A]\{b\} = \{c\}$ を解き $\{b\}$ をもとめ $\{1, 2, 3, \dots, n\}^T$ となっ

ていることを確認する. そのうえで 1000×1000 のマトリックスに対していくつかの m について cpu time を計測する.

`isd` の計算は以下のようにすると分かりやすい. まず `isd(i)` に i 列の高さを求めて代入しておく. この場合であれば以下のようになる.

for $i = 1 \sim m$
    `isd(i)` $= i$
end for
for $i = m + 1 \sim n$
    `isd(i)` $= m$
end for

それを元に実際の `isd` を計算する. たとえば以下のようになる.

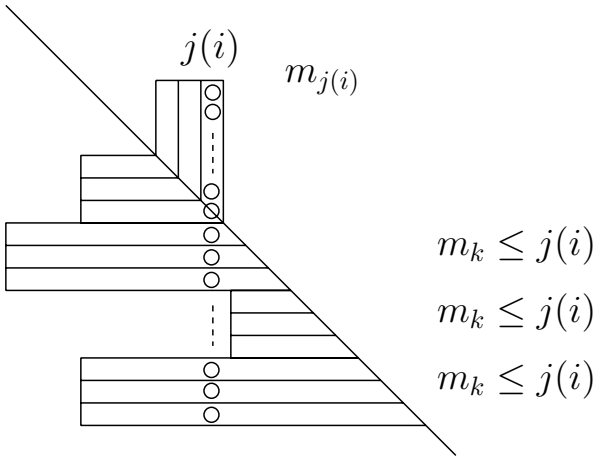
```
isd(1) = 1
isd(2) = 2
n_previous = isd(2)
for   i = 3 ~ n + 1
    n_current = isd(i)
    isd(i) = isd(i-1) + n_previous
    n_previous = n_current
end for
```

なお有限要素法で用いるメッシュに skyline 法を適用する場合にもここで用いたものと同様の方法をとる.

?? 節で示した, asky_ver0, ssky_ver0, asky_ver01, ssky_ver01, asky_ver02, ssky_ver02 をもとにコーディングした例を示す.

4 skyline 法の境界条件処理

?? 節で説明した境界条件処理の方法は skyline 法にも適用することができる. ただし係数マトリックスを上下三角に分割して記憶しているので, そのままの形で実行することができない. たとえば, 下図のような係数マトリックスの場合, 上三角行列に関しては, $A_{1j(i)} = A_{2j(i)} = \ldots = A_{m_{j(i)}-1j(i)} = 0$ であり記憶されていない. また下三角行列に関しては, k 行めが $m_k > j(i)$ であれば $A_{kj(i)} = 0$ であり記憶されていない.



そこでまず, 右辺の修正を行うループ

for $i = 1 \sim m$
  for $k = 1 \sim n$
    $c_k = c_k - \tilde{b}_{j(i)} A_{kj(i)}$
  end for
end for

については, 以下のように変更する.

for $i = 1 \sim m$

$m_{j(i)} = j(i) - \mathtt{isd}(j(i)+1) + \mathtt{isd}(j(i)) + 1$

for $k = m_{j(i)} \sim j(i)$

$kj = \mathtt{isd}(j(i)) + j(i) - k$

$c_k = c_k - \tilde{b}_{j(i)}\mathtt{au}(kj)$

end for

for $k = j(i)+1 \sim n$

$m_k = k - \mathtt{isd}(k+1) + \mathtt{isd}(k) + 1$

if$(m_k \leq j(i))$**then**

$jk = \mathtt{isd}(k) + k - j(i)$

$c_k = c_k - \tilde{b}_{j(i)}\mathtt{al}(jk)$

end if

end for

end for

また係数マトリックスを $0, 1$ に変更するループ

for $i = 1 \sim m$

  for $k = 1 \sim n$

    $A_{kj(i)} = 0$

    $A_{j(i)k} = 0$

  end for

  $A_{j(i)j(i)} = 1$

  $c_{j(i)} = 0$

end for

については以下のように変更する.

for $i = 1 \sim m$

  $m_{j(i)} = j(i) - \mathtt{isd}(j(i)+1) + \mathtt{isd}(j(i)) + 1$

  for $k = m_{j(i)} \sim j(i) - 1$

    $kj = \mathtt{isd}(j(i)) + j(i) - k$

    $\mathtt{au}(kj) = 0$

    $\mathtt{al}(kj) = 0$

  end for

for $k = j(i)+1 \sim n$

$m_k = k - \mathtt{isd}(k+1) + \mathtt{isd}(k) + 1$

if$(m_k \leq j(i))$**then**

$jk = \mathtt{isd}(k) + k - j(i)$

$\mathtt{au}(jk) = 0$

$\mathtt{al}(jk) = 0$

end if

end for

$jj = \mathtt{isd}(j(i))$

$\mathtt{au}(jj) = 0$

$\mathtt{al}(jj) = 0$

end for

課題??で示したバネ–質点系の問題を skyline 法により解く. fortran でのコーディング例を示すと以下のようになる.

5 skyline 法の有限要素法への適用 (1)

以上に説明した skyline 法を実際の有限要素解析コードに適用するには (i) `isd` を計算する, (ii) merge を skyline に対応させる, 必要がある.

isd は各列の非零成分の最小行番号を求めれば構成できる. 下図のようなメッシュが与えられたとする. ただし簡単のために各節点の自由度は1 とする.

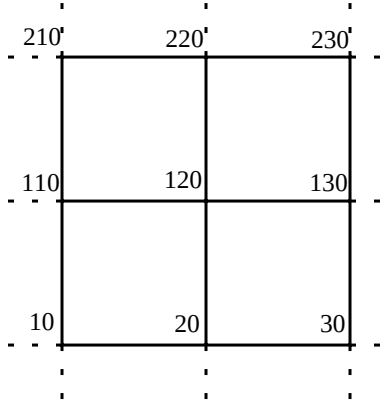


図 1: メッシュの例

ここで節点 120 について考える. この節点は (10, 20, 120, 110), (20, 30, 130, 120), (110, 120, 220, 210), (120, 130, 230, 220) の四つの要素に含まれており, これ以外の要素には含まれていない. まず, (10, 20, 120, 110) の要素剛性マトリックスの 16 成分は全体剛性マトリックスの (10, 10), (10, 20), (10, 120), (10, 110), (20, 10), (20, 20), (20, 120), (20, 110), (120, 10), (120, 20), (120, 120), (120, 110), (110, 10), (110, 20), (110, 120), (110, 110) 成分に重ね合わされる. 120 列に注目すると (10, 120), (20, 120), (120, 120), (110, 120) の４成分であるがこのうちで行番号が最小の成分は, (10, 120) である. 同様に (20, 30, 130, 120), (110, 120, 220, 210), (120, 130, 230, 220) の要素についても 120 列の行番号最小の成分を求めると, それぞれ (20, 120), (110, 120), (120, 120) である.

以上から $m_{120} = 10$ であることがわかる. この作業は一般化でき, m_j は自由度 j を含むすべての要素に含まれる自由度のうち最小のものであることがわかる.

m_j を効率よくすべての自由度 j について求めるためのアルゴリズムの一例を示す.

要素 1 における最小自由度は, 要素 i に含まれるすべての自由度 j について m_j の候補であり, したがって列の高さの候補も n_height $= j - m_j + 1$ で得られる. そこでまず要素 i に含まれるすべての自由度について isd(j) に列の高さの候補 n_height を代入する.

要素 2 についても同様に n_height を求めることができるが, 要素 2 に含まれる自由度には要素 1 に含まれる自由度が存在する可能性がある. そこで isd(j) に代入されている値と, 要素 2 で得られた n_height を比較して, 要素 2 で得られた n_height のほうが大きかった場合のみ isd(j) を要素 2 で得られた n_height に更新する.

以下同様にすべての要素についてこれを行えば, 最終的に得られた isd には各自由度の列の高さが代入されている. これを元に isd(j) を構成する.

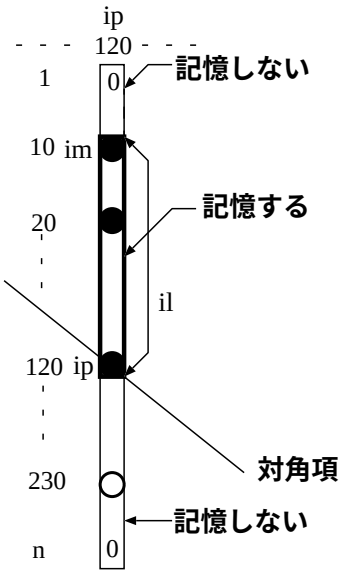


図 2: 剛性マトリックスの様子

この手続きをインデックスを用いて表すと以下のようになる.

総節点数 : `npoin` (number of point)
総要素数 : `nelem` (number of element)
1節点あたりの自由度数 : `ndofn` (number of dof of node)
要素結合条件 : `lnods` (link of nodes)
各要素あたりの節点数 : `ntnoel`
全自由度 : `ntotdf` = `ndofn` × `npoin`

for $i = 1 \sim ntotdf + 1$
  `isd(i)` $= 1$
end for
for $ielem = 1 \sim$ `nelem`
  $min_node =$ `lnods`$(1, ielem)$
  for $inode = 2 \sim$`mnode`
    if (`lnods`$(inode, ielem) < min\_node$)
      $min\_node =$ `lnods`$(i, ielem)$
    end if
  end for

$m_j = (min\_node - 1) \times \texttt{ndofn} + 1$

for $inode = 1 \sim \texttt{ntnoel(ielem)}$

for $idof = 1 \sim \texttt{ndofn}$

$j = (\texttt{lnods}(i, ielem) - 1) \times \texttt{ndofn} + idof$

$n\_height = j - m_j + 1$

if $(\texttt{isd}(j) < n\_height)$

$\texttt{isd}(j) = n\_height$

end if

end for

end for

end for

$\texttt{n\_previous} = \texttt{isd}(2)$

$\texttt{isd}(1) = 1$

$\texttt{isd}(2) = 2$

for $i = 3 \sim \texttt{ntotdf} + 1$

$\texttt{n\_current} = \texttt{isd}(i)$

$\texttt{isd}(i) = \texttt{isd}(i-1) + \texttt{n\_previous}$

$\texttt{n\_previous} = \texttt{n\_current}$

end for

4.1 節で導入した merge を skyline 法に対応させるには, 全体剛性で上三角部分すなわち K_{ij} で $i \leq j$ であるような成分の場合は上三角用の配列に, 下三角部分すなわち K_{ij} で $i \geq j$ であるような成分の場合は下三角用の配列に merge する必要がある.

これをふまえてインデックスで表すと以下のようになる.

for $inode = 1, ntnoel(ielem)$

  for $idof = 1, ndofn$

    $ip((inode-1) * ndofn + idof) = (lnods(inode, ielem) - 1) * ndofn + idof$

  end for

end for

for $i = 1, ntnoel(ielem) \times ndofn$

  for $j = 1, ntnoel(ielem) \times ndofn$

    if $(ip(i) \leq ip(j))$**then**

      $ij = isd(ip(j)) + ip(j) - ip(i)$

      $au(ij) = au(ij) + astiff(i, j)$

    end if

    if $(ip(i) \geq ip(j))$**then**

      $ij = isd(ip(i)) + ip(i) - ip(j)$

      $al(ij) = al(ij) + astiff(i, j)$

    end if

  end for

end for

一般に有限要素法で構造解析を行う場合, 剛性マトリックスは対称になり, この性質をいかして剛性マトリックスは上下三角のうち片側のみを記憶する. 当然ながら要素剛性マトリックスも対称なので, これも上下三角のうち片側だけを計算する場合が多い. この時要素剛性マトリックスは上三角行列あるいは下三角行列になる.

たとえば 4 節点, 各節点あたりの自由度 1 コネクティビティ (4, 1, 2, 3) の要素剛性マトリックス A の上三角部分のみを求めたとする. このとき A の各成分が全体剛性 K のどの成分に対応するかを示すと以下のようになる.

$$\begin{bmatrix} A_{11}=K_{44} & A_{12}=K_{41} & A_{13}=K_{42} & A_{14}=K_{43} \\ 0 & A_{22}=K_{11} & A_{23}=K_{12} & A_{24}=K_{13} \\ 0 & 0 & A_{33}=K_{22} & A_{34}=K_{23} \\ 0 & 0 & 0 & A_{44}=K_{33} \end{bmatrix} \tag{28}$$

一方全体剛性は以下のようになっている.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \\ 0 & A_{22} & A_{23} & A_{24} & \dots \\ 0 & 0 & A_{33} & A_{34} & \\ 0 & 0 & 0 & A_{44} & \\ & \vdots & & & \ddots \end{bmatrix} \tag{29}$$

これからわかるように, $A_{22}, A_{23}, A_{24}, A_{33}, A_{34}, A_{44}, A_{11}$ については, $K_{11}, K_{12}, K_{13}, K_{22}, K_{23}, K_{33}, K_{44}$ としてmergeされる際の行き先があるが, A_{12}, A_{13}, A_{14} については全体剛性マトリックスでは下三角になるため行き先がない. これらは対称性を考慮して K_{14}, K_{24}, K_{34} にmergeする. 即ち全体剛性で下三角に対応する成分については対称な位置にmergeする. これをインデックスで書くと以下のようになる.

for $inode = 1, ntnoel(ielem)$

  for $idof = 1, ndofn$

    $ip((inode-1) \times ndofn + idof) = (lnods(inode, ielem) - 1) * ndofn + idof$

  end for

end for

for $i = 1, ntnoel(ielem) \times ndofn$

  for $j = i, ntnoel(ielem) \times ndofn$

    if$(ip(i) \leq ip(j))then$

      $ij = isd(ip(j)) + ip(j) - ip(i)$

    else

      $ij = isd(ip(i)) + ip(i) - ip(j)$

    end if

    $au(ij) = au(ij) + astiff(i, j)$

  end for

end for

6 skyline 法の有限要素法への適用 (2)

たとえば非圧縮性固体を変位 / 不定静水圧の混合型有限要素定式化を行うと, 変位が 1 節点あたり 3 自由度であるのに対して不定静水圧は1 自由度である. あるいは接触解析を Lagrange 未定乗数法により行う時には, 接触している節点は通常の自由度に加え Lagrange 未定乗数の自由度が追加されるが離れれば元の通常の自由度に戻る. また, シェル要素や梁要素とソリッド要素を混在させて解析を行う場合には, それぞれの節点あたりの自由度は異なっている. このように有限要素解析では節点ごとに自由度が異なることは多い.

このような状況への対応としては, 1 節点あたりの自由度として最大のものを用い必要のない自由度は変位境界条件で固定にするという方法がある. たとえば非圧縮性物質の混合型有限要素定式化ならすべての節点を 3 自由度に設定し, 不定静水圧にあたる節点は 3 自由度のうち冗長な二つの自由度を固定にすれば良い. しかしながらこの方法だと, 1 節点あたり 18 自由度の配管要素と 3 自由度のソリッド要素を混在させる場合, 無駄な自由度が多くなり過ぎる.

そこでここでは一連の skyline の手続きのなかで自由度番号をつけ直すことによりこれに対応する. まず配列 `ndofn2(1:npoin)` を用意し各節点の自由度を記憶する. たとえば, 非圧縮性物質の混合型有限要素定式化なら以下のようになる.

$$\texttt{ndofn2(i)} = \begin{cases} 3 & \text{変位節点} \\ 1 & \text{不定静水圧節点} \end{cases}$$

次に配列 `i_sky_renumbering(1:ntotdf)` (略称:`isr`) を用意し元の自由度番号と新しい自由度番号の対応付をする. 新しい自由度番号は不定静水圧節点の冗長な 2 自由度を縮退させることにより得られる. `isr` はもとの自由度番号をインデックスとして新しい自由度番号を代入しておく. ただし縮退される自由度については区別のため 0 にしておく. 縮退前後の自由度と `isr` の関係を下図に示す.

縮退前後の全自由度数は以下のようになる.

$$\begin{aligned} \text{縮退前の全自由度} \quad \texttt{ntotdf} &= \texttt{ndofn} \times \texttt{npoin} \\ \text{縮退後の全自由度} \quad \texttt{nskydf} &= \sum_{i=1}^{\texttt{npoin}} \texttt{ndofn2}(i) \end{aligned}$$

この手続きをインデックスで表すと, 以下のようになる. 6.5 節で示したものとの違いはまず最初に `isr` をつくること, 次に列の高さなどの計算において用いる自由度番号を `isr` を用いて変換

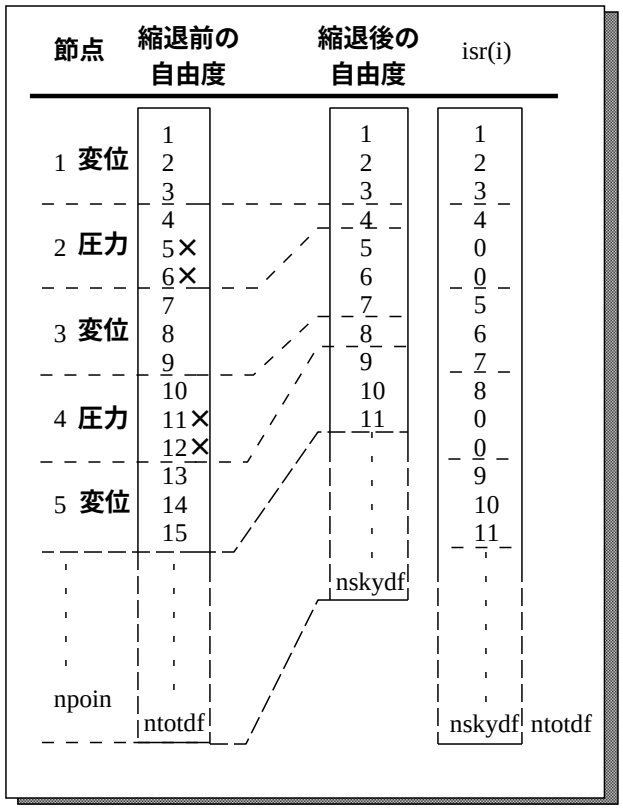


図 3: iskyid の例

していることである.

for $i = 1, ntotdf$

$isr(i) = 0$

end do

$nskydf = 0$

for $inode = 1, npoin$

  for $idof = 1, ndofn2(inode)$

    $nskydf = nskydf + 1$

    $isr((inode - 1) * ndofn + idof) = nskydf$

  end do

end do

for $i = 1, ntotdf + 1$

  $isd(i) = 1$

end do

for $ielem = 1, nelem$

  $min_node = lnods(1, ielem)$

  for $inode = 2, ntnoel(ielem)$

    if$(lnods(inode, ielem) \leq min_node)then$

      $min_node = lnods(inode, ielem)$

    end if

  end do

$mj = isr((min_node - 1) * ndofn + 1)$

for $inode = 1, ntnoel(ielem)$

for $idof = 1, ndofn$

$j = isr((lnods(inode, ielem) - 1) * ndofn + idof)$

$n_height = j - mj + 1$

if$(isd(j) < n_height)then$

$isd(j) = n_height$

end if

end do

end do

end do

$n_previous = isd(2)$

$isd(1) = 1$

$isd(2) = 2$

for $i = 3, ntotdf + 1$

$n_current = isd(i)$

$isd(i) = isd(i-1) + n_previous$

$n_previous = n_current$

end do

これに伴って merge も変更することになる. 変更点は, 全体剛性マトリックス中での行番号, 列番号を表す配列 `ip` をつくるときに縮退した自由度に番号をつけ変えること, および冗長な自由度についてはskip することである. 具体的な手続きを示すと以下のようになる.

[**非対称版**]

for $inode = 1, ntnoel$

for $idof = 1, ndofn$

$ip((inode-1) * ndofn + idof) = isr((lnods(inode)-1) * ndofn + idof)$

end for

end for

for $i = 1, ntnoel * ndofn$

if $(ip(i).ne.0)then$

for $j = 1, ntnoel * ndofn$

if $(ip(j).ne.0)then$

if $(ip(i).le.ip(j))then$

$ij = isd(ip(j)) + ip(j) - ip(i)$

$au(ij) = au(ij) + astiff(i, j)$

end if

if $(ip(i).ge.ip(j))then$

$ij = isd(ip(i)) + ip(i) - ip(j)$

$al(ij) = al(ij) + astiff(i, j)$

end if

end if

end for

end if

end for

[**対称版**]

for $inode = 1, ntnoel$

for $idof = 1, ndofn$

$ip((inode-1)*ndofn+idof) = isr((lnods(inode)-1)*ndofn+idof)$

end for

end for

for $i = 1, ntnoel*ndofn$

if $(ip(i).ne.0)then$

for $j = i, ntnoel*ndofn$

if $(ip(j).ne.0)then$

if $(ip(i).le.ip(j))then$

$ij = isd(ip(j)) + ip(j) - ip(i)$

$else$

$ij = isd(ip(i)) + ip(i) - ip(j)$

end if

$au(ij) = au(ij) + astiff(i,j)$

end if

end for

end if

end for

7　sskyline 法の有限要素法への適用 (3)

下図に示すような 1 要素の単純引っ張りの解析を考える. 節点 5, 6, 7, 8 の x_3 方向に外力を作用させるとする. 全体で 24 自由度あるので解くべき連立一次方程式の数は 24 であるが, 実際には変位境界条件処理により, 節点 1 の x_1, x_2, x_3 方向節点 2 の x_2, x_3 方向, 節点 3 の x_3 方向, 節点 4 の x_1, x_3 方向, 節点 5 の x_1, x_2 方向節点 6 の x_2 方向, 節点 8 の x_1 方向の合計して 12 自由度は自明な方程式になっている. また節点 5, 6, 7, 8 の x_3 方向に強制変位を与える場合なら先の 12 自由度に加え, 4 自由度が自明な方程式になる.

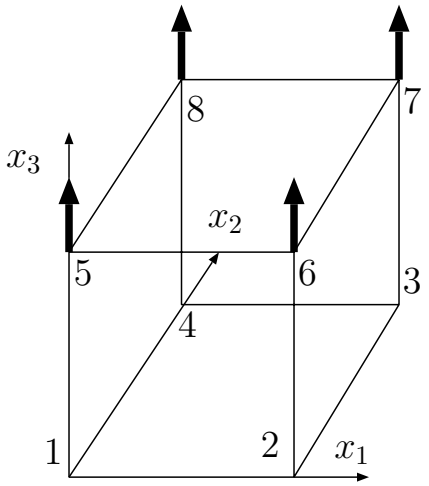


図 4: 1 要素の単純引っ張り

そこでここでは, 前説で導入した自由度番号の並べ替えをさらに進めて (i) 変位境界条件が与えられておらす, 連立一次方程式を解く必要のある自由度, (ii) 強制変位が与えられ, 連立一次方程式を解く必要はないが, 右辺の計算に必要なため剛性マトリックスは記憶する必要がある自由度, (iii) 変位が零に固定されているため skip して良い自由度の順に並べ替える.

自由度番号並べ替えのための配列 `isr` を求める手続きをインデックスで示すと以下のようになる.

for $i = 1, ntotdf$
  $isr(i) = 0$
end for
for $i = 1, n\_bc\_given$
  $isr(i\_bc\_given(i)) = -1$
end for
$nskydf = 0$
for $inode = 1, npoin$
  for $idof = 1, ndofn2(inode)$
    if $(isr((inode-1) * ndofn + idof).eq.0)then$
      $nskydf = nskydf + 1$
      $isr((inode-1) * ndofn + idof) = nskydf$
    end if

end for

end for

for $i = 1, n\_bc\_nonzero$

$nskydf = nskydf + 1$

$isr(i\_bc\_nonzero(i)) = nskydf$

end for

for $inode = 1, npoin$

for $idof = 1, ndofn2(inode)$

if $(isr((inode-1) * ndofn + idof).eq.-1)then$

$nskydf = nskydf + 1$

$isr((inode-1) * ndofn + idof) = nskydf$

end if

end for

end for

`isd` をもとめる手続きではこれまでは各要素でもっとも小さい節点番号の第一自由度が最小自由度であったのに対して, ここでは自由度の並べ替えによりこの関係が成立しない. そこで以下のような手続きにより直接最小自由度を求め `isd` を求める.

for $i = 1, ntotdf + 1$

$isd(i) = 1$

end for

for $ielem = 1, nelem$

$mj = ntotdf$

for $inode = 1, ntnoel(ielem)$

for $idof = 1, ndofn2(lnods(inode, ielem))$

$mj\_temp = isr((lnods(inode, ielem) - 1) * ndofn + idof)$

if $(mj\_temp < mj) mj = mj\_temp$

end for

end for

for $inode = 1, ntnoel(ielem)$

for $idof = 1, ndofn$

$j = isr((lnods(inode, ielem) - 1) * ndofn + idof)$

$n\_height = j - mj + 1$

if $(isd(j) < n\_height) then$

$isd(j) = n\_height$

end if

end for

end for

end for

$n\_previous = isd(2)$

$isd(1) = 1$

$isd(2) = 2$

for $i = 3, ntotdf + 1$

$n\_current = isd(i)$

$isd(i) = isd(i - 1) + n\_previous$

$n\_previous = n\_current$

end for

merge の変更点は固定されている自由度は剛性マトリックスを記憶しないことであるが, もし何らかの理由で記憶する必要があるならば, バージョン 2 の merge と同じである.

[非対称版]

$n\_active = nskydf - n\_bc\_given + n\_bc\_nonzero$

for $inode = 1, ntnoel$

  for $idof = 1, ndofn$

    $ip((inode-1)*ndofn+idof) = isr((lnods(inode)-1)*ndofn+idof)$

  end for

end for

for $i = 1, ntnoel * ndofn$

if $(ip(i) \neq 0 \& ip(i) \leq n\_active) then$

for $j = 1, ntnoel * ndofn$

if $(ip(j) \neq 0 \& ip(j) \leq n\_active) then$

if $(ip(i) \leq ip(j)) then$

$ij = isd(ip(j)) + ip(j) - ip(i)$

$au(ij) = au(ij) + astiff(i, j)$

end if

if $(ip(i) \geq ip(j)) then$

$ij = isd(ip(i)) + ip(i) - ip(j)$

$al(ij) = al(ij) + astiff(i, j)$

end if

end if

end for

end if

end for

[対称版]

$n\_active = nskydf - n\_bc\_given + n\_bc\_nonzero$

for $inode = 1, ntnoel$

  for $idof = 1, ndofn$

    $ip((inode-1)*ndofn+idof) = isr((lnods(inode)-1)*ndofn+idof)$

  end for

end for

for $i = 1, ntnoel*ndofn$

  if $(ip(i) \neq 0 \& ip(i) \leq n\_active) then$

    for $j = i, ntnoel*ndofn$

      if $(ip(j) \neq 0 \& ip(j) \leq n\_active) then$

        if $(ip(i) \leq ip(j)) then$

          $ij = isd(ip(j)) + ip(j) - ip(i)$

        $else$

          $ij = isd(ip(i)) + ip(i) - ip(j)$

        end if

        $au(ij) = au(ij) + astiff(i,j)$

      end if

    end for

end if

end for