5　連立 1 次方程式の解法

連立 1 次方程式は、有限要素法による構造物の応力解析、最小二乗法、微分方程式の差分解法など、数値計算の諸手法の一部分として重要である。ここでは、線形代数の講義で学んだ Cramer(クラメール)の公式を用いた解法と、数値計算の世界では最も基本的な計算法の一つである Gauss(ガウス)の消去法を取り上げる。

5.1　Cramer の公式

連立 2 元 1 次方程式

$$\begin{cases} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{cases}$$

の解は

$$x_1 = \Delta_1/\Delta,\ x_2 = \Delta_2/\Delta$$

と表される。ただしここで、$|A|$ を行列 A の行列式として

$$\Delta = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21},\ \Delta_1 = \begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix} = b_1a_{22} - a_{12}b_2,\ \Delta_2 = \begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix} = a_{11}b_2 - b_1a_{21}$$

などとした。これを Cramer の公式という。3 元以上の連立多元方程式の解も同様に

$$x_k = \frac{\text{係数行列の第 k 列を定数項で置換えた行列の行列式}}{\text{係数行列の行列式}}$$

と書くことができる。Cramer の公式は連立一次方程式を理論的に議論する際には重要な式であるが、実際の数値計算の上では、行列の次数が大きくなると急激に計算時間がかかるようになるので、後に述べる Gauss の消去法など他の方法を用いる方が良い。

練習問題 9: 任意の連立 2 元 1 次方程式を Cramer の公式で解くプログラムを作れ。方程式の各係数と定数項をキーボードから入力出来るようにし、次の連立方程式を解け。

$$\begin{cases} 4x_1 + 5x_2 &= 14 \\ 8x_1 + 28x_2 &= 64 \end{cases}$$

(念のための解答 $x_1 = 1,\ x_2 = 2$)

5.2　Gauss の消去法

n 元の連立 1 次方程式は一般に

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \quad\cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{cases}$$

という形で表すことができる。この第 n 式を x_n について解き、

$$x_n = \frac{1}{a_{nn}} \times [a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{n\,n-1}x_{n-1} - b_n]$$

としたものを第1〜第$n-1$式に代入すれば、それらの式におけるx_nの係数項が消去され、$x_1, x_2 \ldots, x_{n-1}$を未知数とする$n-1$元の方程式

$$\left\{\begin{array}{lllllllll} a'_{11}x_1 & + & a'_{12}x_2 & + & \cdots & + & a'_{1,n-1}x_{n-1} & = & b'_1 \\ a'_{21}x_1 & + & a'_{22}x_2 & + & \cdots & + & a'_{2,n-1}x_{n-1} & = & b'_2 \\ \quad\cdots & & & & & & & & \\ a'_{n-1,1}x_1 & + & a_{n-1,2}x_2 & + & \cdots & + & a'_{n-1,n-1}x_{n-1} & = & b'_{n-1} \end{array}\right.$$

になる。その係数は

$$\begin{array}{rcll} a'_{ij} & = & a_{ij} - a_{in}a_{nj}/a_{nn} & \\ b'_i & = & b_i - a_{in}b_n/a_{nn} & (i, j = 1, 2, \ldots, n-1) \end{array}$$

により計算できる。以下同様に、$x_{n-1}, x_{n-2}, \ldots, x_2$を順に消去すれば、最後に

$$a''_{11}x_1 = b''_1$$

の形の式を得る。これを解いてx_1を求めることができる。x_1が求まれば、上の消去の手順で最後から二番目に導かれた、x_2を消去した時の式を利用してx_2も求めることができ、以下同様にして$x_3, x_4, \ldots, x_n$を順に求めることができる。このような方法をGaussの消去法という。この方法は手順が簡単で規則的なので機械的な処理に向き、他の方法と比較して必要な演算回数や精度の点で有利であるという特長を持つ。アルゴリズムを図式化すると

$$\left(\begin{array}{l} k = n, n-1, \ldots, 2 \text{の順に} \\ \quad \begin{array}{rcll} \text{新}\, a_{ij} & = \text{前}\, a_{kj}/\text{前}\, a_{kk} & & (j = 1, 2, \ldots, k)) \\ \text{新}\, b_k & = \text{前}\, b_k/\text{前}\, a_{kk} & & \end{array} \\ \quad \left(\begin{array}{l} i = 1, 2, \ldots, k-1 \text{に対して} \\ \quad \begin{array}{rcll} \text{新}\, a_{ij} & = \text{前}\, a_{ij} - \text{前}\, a_{ik}/\text{新}\, a_{kj} & & (j = 1, 2, \ldots, k)) \\ \text{新}\, b_i & = \text{前}\, b_i - \text{前}\, a_{ik}/\text{新}\, b_k & & \end{array} \end{array}\right. \end{array}\right.$$

$$\left(\begin{array}{l} x_1 = \text{最終}\, b_1/\text{最終}\, a_{11} \\ i = 2, 3, \ldots, \text{の順に} \\ \quad x_i = \text{最終}\, b_i - \displaystyle\sum_{j=1}^{i-1}(\text{最終}\, a_{ij})x_j \end{array}\right.$$

となる。[1]

さて、消去法のプログラムをここに示したアルゴリズムに従って全て作り上げても良いのだが、100行程のプログラムを書かねばならず少々大変なので、連立一次方程式 $A\vec{x} = \vec{b}$ の解を求める外部関数 `gsnelm` を用意した。Gaussの消去法 (gaussian elimination) を用いているので、'`gsnelm`' である。以下の練習問題を解く際にはこの関数を用いて良い。[2] 関数 `gsnelm` は、行列とベクトルをその次元と共に与えると連立方程式の解を求め、求められない場合はその理由を示すコードを返すようになっている。内部で色々とややこしいことをしているのだが、利用者は内部の「色々」を全く意識せずに関数を利用することができる。以下に使用例を示す。

```
integer :: dim, mxdm, gsnelm, rc
double precision :: mat(5, 5), vec(5)
dim = 3
mxdm = 5
```

[1] ここに示したアルゴリズムに忠実に従うと、対角要素に0が出て来たときに計算ができなくなってしまう。実際のプログラミングでは同じ列の中で一番絶対値が大きい要素を持つ行を探して、入れ替えを行なう。これをpivot選択と言う。

[2] ファイルのコピーの仕方やコンパイルの方法は別紙を参照。

```
      mat(1, 1) = 1.0d+0
      mat(1, 2) = 3.0d+0
...
      mat(3, 3) = 2.0d+0
      vec(1) = 12.0d+0
...
      rc = gsnelm(mat, vec, dim, mxdm)
      if (rc == 0) then
        print *, 'x(1) =', vec(1)
...
      else
        print *, 'Fail to solve'
      end if
c
      end
```

関数gsnelmは計算に用いるデータとして四つの引数を取って“gsnelm(mat, vec, dim, mxdm)”のように呼び出す。引数のmat、vec、dim、及びmxdmは、それぞれ、係数行列を格納した実数の二次元配列、定数ベクトルを格納した実数の配列、未知数の数(整数)、宣言された配列のサイズ(整数)である。答えは配列vecに入って返される。行列matの内容は計算の中で壊されてしまうので、行列の中身が必要ならばコピーを取っておいてからgsnelmを呼び出さなければならない。

上のサンプルプログラムでは、まず係数行列を記憶するための二次元配列と定数ベクトルを格納する一次元配列を宣言しておいて、行列要素と定数ベクトルに一つずつ値を代入する。行列とベクトルの準備が終ったら関数gsnelmを呼び出し、戻り値を変数“rc”に代入する。関数gsnelmは解が求められると0、途中で計算ができなくなった時は0以外の値を返すので、変数“rc”に代入された戻り値が0の場合は配列要素“v(1)”、“v(2)”、“v(3)”に入っている連立方程式の解を出力する。そうでない場合(計算が途中で行き詰まった場合)は、“Fail to solve”というメッセージをモニタに出力する。この例では係数行列matは5×5の配列なのだが、その内3×3の領域しか使っていない。無駄ではあるのだけれど、未定義の領域を参照しない限り問題を起すことはない。

練習問題 10: 次の連立方程式のいずれかを上記の Gauss の消去法で解きなさい。「mat(1,1) = 1」のようにデータをプログラム中に書き込んで良い。

(1)

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 6 \\ 3x_1 + 10x_2 - 4x_3 = -29 \\ -2x_1 - 4x_2 + x_3 = 9 \end{cases}$$

(解答 $x_1 = 1$, $x_2 = -2$, $x_3 = 3$)

(2)

$$\begin{cases} 3x_1 + x_2 + x_3 = 10 \\ x_1 + 5x_2 + 2x_3 = 21 \\ x_1 + 2x_2 + 5x_3 = 30 \end{cases}$$

(解答 $x_1 = 1$, $x_2 = 2$, $x_3 = 5$)

(3)

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 8 \\ 3x_1 + 6x_2 - 2x_3 = -9 \\ 2x_1 - 4x_2 + 5x_3 = 9 \end{cases}$$

(解答 $x_1 = -2$, $x_2 = 0.5$, $x_3 = 3$)

結晶の構造解析で重要な役割を果す逆格子ベクトルは、$\vec{G} = n_1\vec{\alpha} + n_2\vec{\beta} + n_3\vec{\gamma}$ と表わすことができる。ここで n_1, n_2, n_3 は整数であり、逆格子の基本並進ベクトル $\vec{\alpha}, \vec{\beta}, \vec{\gamma}$ は結晶の単位格子ベクトルを $\vec{a}, \vec{b}, \vec{c}$ として

$$\begin{aligned}
\vec{\alpha}\cdot\vec{a} &= 2\pi, \quad \vec{\alpha}\cdot\vec{b} = \vec{\alpha}\cdot\vec{c} = 0 \\
\vec{\beta}\cdot\vec{b} &= 2\pi, \quad \vec{\beta}\cdot\vec{c} = \vec{\beta}\cdot\vec{a} = 0 \\
\vec{\gamma}\cdot\vec{c} &= 2\pi, \quad \vec{\gamma}\cdot\vec{a} = \vec{\gamma}\cdot\vec{b} = 0
\end{aligned}$$

を満たすベクトルである。すなわち、ベクトル $\vec{\alpha}$ は $\vec{b}$ 及び $\vec{c}$ と直交し、$\vec{a}$ との内積が 2π になる等々。ベクトルの内積の計算を成分で書いてみるとわかるように、連立一次方程式を解くと逆格子の基本ベクトルを求めることができる。

練習問題 11: 単体金属のアルミニウムは面心立方構造の結晶格子を持ち、その格子定数は $a = 4.05$Å である。アルミニウム結晶の逆格子の基本ベクトルを求めよ。

ヒント

アルミニウムは面心立方構造を持つということなので、結晶の基本ベクトルを成分で書けば、

$$\begin{aligned}
\vec{a} &= (a_x, a_y, a_z) = (0.0, 0.5a, 0.5a) \\
\vec{b} &= (b_x, b_y, b_z) = (0.5a, 0.0, 0.5a) \\
\vec{c} &= (c_x, c_y, c_z) = (0.5a, 0.5a, 0.0)
\end{aligned}$$

となる。逆格子の基本ベクトルは、これらの二つと直交し残りとは内積を取って 2π になるのだから、例えばベクトル $\vec{\gamma} = (\gamma_x, \gamma_y, \gamma_z)$ は連立方程式

$$\begin{aligned}
a_x\gamma_x + a_y\gamma_y + a_z\gamma_z &= 0 \\
b_x\gamma_x + b_y\gamma_y + b_z\gamma_z &= 0 \\
c_x\gamma_x + c_y\gamma_y + c_z\gamma_z &= 2\pi
\end{aligned}$$

を解くことによって見付けることができる。残りの $\vec{\alpha}$ 及び $\vec{\beta}$ についても同様である。

練習問題 10 では解くべき問題 (の係数行列の要素) をプログラムのソースコードの中に直接書き込んでしまったが、それでは汎用性がないので係数行列を入力して問題を解かせるようにする方が良い。

練習問題 12: 次の連立方程式を Gauss の消去法のプログラムで解きなさい

$$\begin{cases}
x_1 + x_2 + x_3 + x_4 &= 10 \\
2x_1 + x_2 + 3x_3 + 2x_4 &= 21 \\
x_1 + 3x_2 + 2x_3 + x_4 &= 17 \\
3x_1 + 2x_2 + x_3 + x_4 &= 14
\end{cases}$$

プログラムは、任意の次数の連立方程式を解くことができるように、まずは変数の数、引き続いて各係数および定数項のデータを読むように書くこと。

5.3 応用: 塑性変形と転位のすべり

材料科学のトピックの中からの関連テーマとして、塑性変形と転位のすべりについて考えよう。また も計算機言語と関係のない補足説明の部分は小さな活字で組んでおく。

例えば針金を引張ると伸びるというように、物質に力を加えると変形する。加えた力が余り大きなものでなければ、この変形は一時的なもので、力を取り去ると元の形に戻る。しかし「仏の顔も三度まで」と言うように、ものには限度があって、度が過ぎると元に戻らなくなる。大きな力を加えたために、原子の並び順が変ってしまうのである。こうした原子の並び方の変化は、転位と呼ばれる局所的な結晶構造の乱れが、結晶中を移動することによって起きると考えられている(図 1)。転位線が移動する結晶面をすべり面、移動する方向をすべり方向と呼ぶ。多くの場合、転位は原子が一番密に並んでいる面上を、原子が一番密に並んでいる方向へ移動するので、例えば面心立方構造を持つ物質では $(1\,1\,1)$ 面と等価な面上を $[\bar{1}\,1\,0]$ と等価な方向にすべる。$(1\,1\,1)$ と等価な面は 4 つ存在し、それぞれの面上で可能なすべり方向は三つずつあるので、すべり面とすべり方向の組み合わせ(「すべり系」と呼ばれる) は全部で 12 存在する。[3]

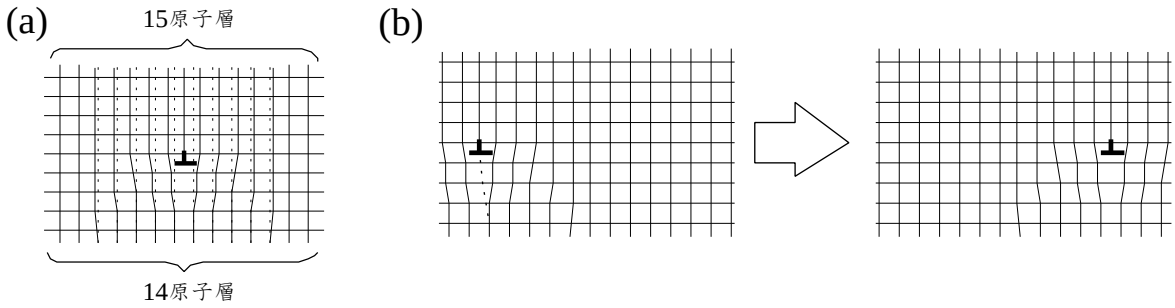

図 1: 二次元正方格子上の刃状転位 (a) と転位の運動による変形の概念図 (b)。結晶の上半分には余計な原子面が存在して格子が局所的に歪んでいる。(b) 図の点線のように原子のつなぎ換えが起ると、転位は結晶の中を移動することができる。左端から右端まで転位が動くと、結晶の下半分は少しだけ左側にずれている。

これらのすべり系の一つが活動した時に結晶に起きる歪を考えよう。図 2 は直方体状の試料を引張った時、$(h\,k\,l)$ 面上を一本の刃状転位がすべって、結晶が Burgers ベクトル一つ分だけずれた様子を示す。影をつけた面がすべり面、白抜きの矢印が転位の Burgers ベクトルである。すべり面の法線方向及びすべり方向が引張り軸となす角度を、それぞれ λ、θ とすると、剪断変形を引き起す方向への応力の成分 (転位線が感じている力) は、試料を引張る応力を $\sigma_{\rm applied}$ として $\sigma = \sigma_{\rm applied} \cos\lambda \cos\theta$ となる。因子 $\cos\lambda\cos\theta$ を Schmid 因子と呼ぶ。転位が動き出すのに必要な応力 (臨界分解剪断応力) は引張りの方向に依存しないので、活動することのできる複数のすべり系がある場合には、Schmid 因子が最大になるものがまず活動する。すべり面より右の部分が紙面の斜め奥の方向に動くので、結晶は形が変るだけでなく、上から見て反時計方向に回転することが分る。

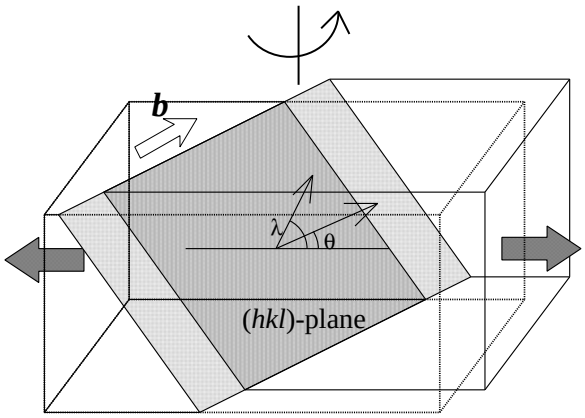

図 2: 一本の転位線のすべりによる結晶の変形。

[3]後の例題を解く際の便宜に 12 のすべり系を列挙すると、$(1\,1\,1)$ 面上の $[\bar{1}\,1\,0]$、$[0\,\bar{1}\,1]$、$[1\,0\,\bar{1}]$ 方向、$(\bar{1}\,1\,1)$ 面上の $[\bar{1}\,\bar{1}\,0]$、$[1\,0\,1]$、$[0\,1\,\bar{1}]$ 方向、$(1\,\bar{1}\,1)$ 面上の $[1\,1\,0]$、$[\bar{1}\,0\,1]$、$[0\,\bar{1}\,\bar{1}]$ 方向、$(\bar{1}\,\bar{1}\,1)$ 面上の $[1\,\bar{1}\,0]$、$[0\,1\,1]$、$[\bar{1}\,0\,\bar{1}]$ 方向である。

こうしたすべりが複数起きると、結晶の変形の様子は図 3 のようになる。すべり面を $(h\,k\,l)$ 面、格子の単位ベクトルを $\mathbf{a}_1$、$\mathbf{a}_2$、$\mathbf{a}_3$ とする。以下では、単位ベクトルが立方体をなす立方晶系の格子を仮定する (他の結晶構造の時にどうなるかは宿題としておこう)。また、すべり方向と Burgers ベクトルの方向が一致する刃状転位を考え、すべり方向を $[u\,v\,w]$ 方向とする。Miller 指数の定義からベクトル $\mathbf{a}_1$ は h 回すべり面と交差するので、平均して一本の転位線が $(h\,k\,l)$ 面をすべるなら、ベクトル $\mathbf{a}_1$ の場所にあった原子は $\mathbf{a}_1 - h\mathbf{b}$ へと移動するであろう。他の単位ベクトルについても同様なので、転位がすべったことによりもともとの単位格子は、

$$\left(\begin{array}{ccc}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3\end{array}\right) = \left(\begin{array}{ccc} a_1^x & a_2^x & a_3^x \\ a_1^y & a_2^y & a_3^y \\ a_1^z & a_2^z & a_3^z \end{array}\right) \longrightarrow \left(\begin{array}{ccc} a_1^x + hua & a_2^x + kua & a_3^x + lua \\ a_1^y + hva & a_2^y + kva & a_3^y + lva \\ a_1^z + hwa & a_2^z + kwa & a_3^z + lwa \end{array}\right)$$

と歪むであろう。立方晶を仮定していたので、元の格子は対角行列となることに注意して、この変形を行列で記述するなら、

$$\left(\begin{array}{ccc} a+hua & kua & lua \\ hva & a+kva & lva \\ hwa & kwa & a+lwa \end{array}\right) = \left(\begin{array}{ccc} 1+hu & ku & lu \\ hv & 1+kv & lv \\ hw & kw & 1+lw \end{array}\right) \left(\begin{array}{ccc} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{array}\right)$$

$$= \left[\left(\begin{array}{ccc} 1+\dfrac{hu+kv+lw}{3} & 0 & 0 \\ 0 & 1+\dfrac{hu+kv+lw}{3} & 0 \\ 0 & 0 & 1+\dfrac{hu+kv+lw}{3} \end{array}\right)\right.$$

$$+ \left(\begin{array}{ccc} 0 & \dfrac{ku-hv}{2} & \dfrac{lu-hw}{2} \\ \dfrac{hv-ku}{2} & 0 & \dfrac{lv-kw}{2} \\ \dfrac{hw-lu}{2} & \dfrac{kw-lv}{2} & 0 \end{array}\right)$$

$$\left. + \left(\begin{array}{ccc} \dfrac{2hu-kv-lw}{3} & \dfrac{ku+hv}{2} & \dfrac{lu+hw}{2} \\ \dfrac{hv+ku}{2} & \dfrac{2kv-lw-hu}{3} & \dfrac{lv+kw}{2} \\ \dfrac{hw+lu}{2} & \dfrac{kw+lv}{2} & \dfrac{2lw-hu-kv}{3} \end{array}\right)\right] \left(\begin{array}{ccc} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{array}\right) \qquad (1)$$

となる。最終行では、変形を表わす行列を大括弧の中に示すように三つの成分に分解した。一つ目の成分は単位行列の整数倍になっており、膨張や収縮などの体積の変化を表わす項である。二つ目は反対称行列 (転置すると符号が変る行列) であり、結晶の回転運動を表わしている。三つ目は対称行列 (転置しても符号が変らない行列) であり、この行列が結晶の (剪断) 変形を表わす項である。歪テンソルの定義を思い出すと、定数倍を除いて、この三つ目の行列が変形を記述する歪テンソルになっている。

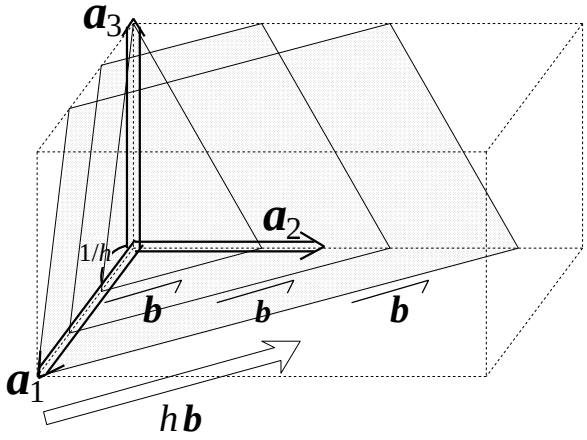

図 3: 同一のすべり系に属する転位線が複数すべることによる結晶の変形。

塑性変形の際には、Schmid 因子が大きいすべり系から活動するので、全てのすべり系が活動するわけではない。また、面心立方格子では 12 のすべり系があるが、全てが独立なわけではない。例えば $(1\,1\,1)$ 面上の $[1\,0\,\bar{1}]$ すべりは、同じ面上の $[\bar{1}\,1\,0]$ および $[0\,\bar{1}\,1]$ の和と等しい。そんなわけで、特定の応力を加えたときに、どのすべり系が活動するかを予測するという問題が出て来る。どんな歪テンソルで表現される変形にも対応できるためには、独立なすべり系が五つあればよい。規格化された直交基底を見付けるには Schmidt の直交化の手続きがあるが、独立なすべり系を見出すだけなら掃き出し法を用いることもできる。

応用問題 4: 面心立方格子を持つ試料を東経 141 度、北緯 43 度 (札幌市の位置) の方向に引張ったときに活動するすべり系を掃き出し法で見付けよ。

ヒント

面倒な仕事に取りかかるときは、すべき仕事を細かく分ける。例えば、

1. 12 のすべり系を用意する。
2. 引張り方向を入力して各すべり系に対する Schmid 因子を計算。
3. Schmid 因子の大きい順にすべり系を並べ換える。
4. 歪テンソルの計算。
5. 掃き出し

こんな風に仕事を分けて考える。今回の掃き出し法は、連立方程式を解くわけではないので、関数 `gsnelm` をそのまま使うことはできないが、これに手を加えることで問題を解いてみよう。

手順 1. のすべり系の用意は、前のページの脚注に列挙しておいた 12 のすべり面とすべり方向の組を配列に代入すれば良い。三次元空間ですべり面の法線方向とすべり方向を格納するには一つのすべり系あたり 6 次元のベクトルが必要で、すべり系は 12 あるので、12×6 の二次元配列を利用する。この解説では、6 次元のベクトルはすべる面の法線方向を先、すべり方向を後に格納する約束として書くけれど、どちらが先でも問題はない。72 の要素を一つ一つ代入すると、それだけで 72 行にもなってしまうので、多分ファイルに書いて読むのが楽。外部関数の形にすると、

```
      function setss(ss)
...
      do i = 1, 12
        read *, ss(i,1), ss(i,2), ss(i,3), ss(i,4), ss(i,5), ss(i,6)
      end do
c
      setss = 0
      end
```

こんな感じ。

手順 2. の Schmid 因子の計算は、引張り方向とすべり方向のなす角度の余弦 $(\cos\theta)$ やすべり面の法線とのなす角度の余弦 $(\cos\lambda)$ を求めなければならない。この時活躍するのが高校で教わった内積の公式

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta = a_x b_x + a_y b_y + a_z b_z$$

である。引張方向は東経 141 度、北緯 43 度ということなので、この方向の単位ベクトルは $\mathbf{e} = (\cos 141^\circ \cos 43^\circ, \sin 141^\circ \cos 43^\circ, \sin 43^\circ)$ という成分を持つ。よってこの引張り方向とすべり方向 $[\bar{1}\,0\,1]$ のなす角度の余弦は

$$\cos\theta = \frac{(-1)\times e_x + 0\times e_y + 1\times e_z}{\sqrt{(-1)^2+0^2+1^2}}$$

と計算できる。ただしここで、e_x などは引張方向の単位ベクトルの成分である。プログラムにすると

```
      function gtsmdf(lngd, lttd, schmd, ss)
...
      read *, lngd, lttd
      lngd = lngd / 45.0d+0 * atan(1.0d+0)
      lttd = lttd / 45.0d+0 * atan(1.0d+0)
c
      rad(1) = cos(lttd) * cos(lngd)
      rad(2) = cos(lttd) * sin(lngd)
      rad(3) = sin(lttd)
c
      do i = 1, 12
        cslmbd = (rad(1)*ss(i,1) + rad(2)*ss(i,2) + rad(3)*ss(i,3))
     $         / sqrt(ss(i,1)**2 + ss(i,2)**2 + ss(i,3)**2)
        cstht  = ...
        schmd(i) = cslmbd * cstht
      end do
...
```

こんな感じ。組込み関数 atan は三角関数 tan の逆関数で、atan(1.0d+0) は $\pi/4$ になる。手順 3. のすべり系の並べ換えは、論じ始めると大変なことになる。例えば google などのインターネットの検索サイトでは、毎日何百万件もの問合せのたびに、これまた何百万ものウェブページに得点をつけて、点数の順に表示を行なうという荒技をこなしている。答えが 1 秒以内に返るのは驚異的である。ソーティングの問題は一冊の本が書けてしまうほど大きな問題なのだけれど、今回は高々12 個の要素を並べ換えるだけだから、効率を極大化する方策を考えるより、さっさと並べ換えた方が良い。

```
      function sortss(schmd, ss)
...
      do i = 1, 11
      do j = i + 1, 12
        if (abs(schmd(i)) < abs(schmd(j))) then
          tmp = schmd(i)
          schmd(i) = schmd(j)
          schmd(j) = tmp
          do k = 1, 6
            tmp = ss(i,k)
            ss(i,k) = ss(j,k)
            ss(j,k) = tmp
          end do
        end if
      end do
      end do
...
```

1 番目と 2 番目を比べ、1 番目と 3 番目を比べ、1 番目と 4 番目を比べ、(中略)、2 番目と 3 番目を比べ、2 番目と 4 番目を比べ、2 番目と 5 番目を比べ、(中略)、10 番目と 11 番目を比べ、10 番目と 12 番目を比べ、11 番目と 12 番目を比べるという具合に $12\times 11/2 = 66$ 回の大小比較を行ない、後に出て来る Schmid 因子の方が大きいような組み合わせを見付ける度に、大きな Schmid 因子のすべり系を前の方に移動するということを繰り返せば、最後には Schmid 因子の順に整列されることになる。ただし絶対値が大きな負の数になるような Schmid 因子を持つすべり系は、すべり方向を逆転すると正の大きな Schmid 因子を持つようになるので、大小比較の際は絶対値を用いて評価した。

手順 4. の歪テンソルの計算は、式 (1) を計算するだけである。

```
      function geteps(ss, eps)
...
      do i = 1, 12
        eps(i,1,1) = (2.0d+0 * ss(i,1) * ss(i,4)
     $             - ss(i,2) * ss(i,5) - ss(i,3) * ss(i,6)) / 3.0d+0
        eps(i,1,2) = (ss(i,2) * ss(i,4) + ss(i,1) * ss(i,5)) / 2.0d+0
        eps(i,1,3) = (ss(i,3) * ss(i,4) + ss(i,1) * ss(i,6)) / 2.0d+0
        eps(i,2,1) = eps(i,1,2)
...
        eps(i,3,3) = (2.0d+0 * ss(i,3) * ss(i,6)
     $             - ss(i,1) * ss(i,4) - ss(i,2) * ss(i,5)) / 3.0d+0
      end do
...
```

こんな感じ。歪テンソルは対称行列であることが利用できる。
最後の掃き出し法は、これまでの手順で得られた個々のすべり系がつくる歪テンソルの成分を一行に並べたものを、12 行集めて 12×9 の行列を作り、行同士の引き算をする。今回の仕事と連立方程式の解を求める計算の違いは、今回の問題では Schmid 因子の順に行が並んでいるので、行の順番を入れ換えてはならないということである。その代りに、対角要素を残すことに拘らなくて良い。そこで、消去を進めようとしている列の中で絶対値が最大の pivot 行を選択をする代りに、着目している行の中で絶対値が最大の要素を探して、その要素の存在する列を消去する。全ての列の要素が零なら絶対値の最大の要素は零になってしまうが、そういう時はそのすべり系は線型独立ではないということである。また関数 `gsnelm` では最終行から上に向って消去をしているが、今回は上から下に向って消去をすべきである。
具体的なプログラムコードは、まず歪テンソルの要素を並べて行列の設定をする外部関数から考えると、

```
      function setmat(eps, mat, dim, mxdm)
...
      do i = 1, 12
        mat(i,1) = eps(i,1,1)
        mat(i,2) = eps(i,1,2)
        mat(i,3) = eps(i,1,3)
        mat(i,4) = eps(i,2,1)
...
        mat(i,9) = eps(i,3,3)
      end do
...
```

これは簡単。3×3 の二次元正方行列を 9 次元のベクトルに並べ換えているイメージである。
次に、pivot 行を探す関数 `pivot(mat, k, dim, mxdm)` に代えて、pivot 列を探す関数 `pivot2(mat, k, dim, mxdm)` を定義する。

```
      function pivot2(mat, k, dim, mxdm)
...
      pivot2 = 0
      do i = 1, dim
        print *, 'mat(',k,i, ') :', mat(k,i)
        if(pivot2 == 0 .and. abs(mat(k,i)) > 1.0d-10) then
          pivot2 = i
        else if(abs(mat(k,pivot2)) < abs(mat(k,i))) then
          pivot2 = i
        end if
      end do
c
      end
```

関数 pivot2 は着目する行の要素が全て零になると 0 を返すので、線型従属なすべり系を区別するのに利用できる。
残りの仕事は行の消去の順番を変更するなど、細かい修正を施すことである。着目している行の中で絶対値が最大の要素が 1 になるように行の要素を割り算する関数 sclrw は、

```
      function sclrw2(mat, k, ip, dim, mxdm)
...
      pv = mat(k,ip)
      do j = 1, dim
        mat(k,j) = mat(k,j) / pv
      end do
...
```

他の行の pivot 列目を消去する関数 elmrw2 は

```
      function elmrw2(mat, ip, k, i, dim, mxdm)
...
      pv = mat(i,ip)
      do j = 1, dim
        mat(i,j) = mat(i,j) - mat(k,j) * pv
      end do
...
```

以上二つは規格化や消去を行なう範囲を 1 から 9(dim) までに変えた。元の関数 gsnelm では、0 を何かの数で割ったり、0 から 0 を引く無駄な計算を避けるために、計算をする範囲を限定してあったが、今回はそれができない。
最後にこれまでの外部関数を呼び出して、線型従属なすべり系を消去する関数は

```
      function gsnlm2(mat, dim, mxdm)
...
      do k = 1, mxdm - 1
        ip = pivot2(mat, k, dim, mxdm)
        if (ip /= 0) then
          rc = sclrw2(mat, k, ip, dim, mxdm)
          do i = k + 1, mxdm
            rc = elmrw2(mat, ip, k, i, dim, mxdm)
          end do
        end if
      end do
...
```

となる。

さて、こうしてできたプログラムを実行すると、五つのすべり系が独立なものとしてリストアップされる。すべり系はすべり面の法線方向とすべり方向から構成されるので、3 次元+3 次元の自由度を持つが、これらは直交していなければならないので、自由度は 1 減って 5 になる。歪テンソルにしても対称行列で対角和が 0 なので自由度は 5 である。よって掃き出しを行なう行列のサイズは、実は 5×12 で十分なのであった。