第1章　連立一次方程式について

“連立一次方程式” という名前は，すでに耳に馴染んでおり，“あぁ，鶴亀算とも呼ばれる，あれね！” と，思う学生も多いことだろう．ところが大学の授業では，主に線形代数の中の一項目として議論し，そこでは，行列やベクトルを用いた抽象的な議論が中心で，連立一次方程式に限らず，これらの議論全般に対して，“じゃぁ，実際にはどのような場面で使うの？” という感想をもっている人も多いと思う．実用の場面で現れる行列やベクトルとは，理工学等における様々な現象や人為的に設定したモデルのデータを並べて，システマティックに表現したものである．各々の要素について細かく議論することもできるが，行列やベクトル表現にして扱うことで，全体を見渡した議論が可能となる．数学では，これら行列やベクトル等の表現方法に対する規則を定義したり，これらの中から見出される性質を定理や公式などとして議論しており，いわば，実用にあたってのルールブックの様な位置付けとも考えられる．

計算機を用いた数値シミュレーションやアプリケーション開発において,「方程式を解く」ことが頻繁に発生する．実社会において，数値計算が重要な役割を果たす場面としては，実際に実験を行なうことが不可能な問題や，気象予測，流体解析（たとえば，飛行機の翼やエンジン内での気体の動きなど），社会工学問題，オペレーションズリサーチなど多くの場があり，そこでは，物理，工学や社会の現象を表す方程式の求解が主である．それらを実現するために，様々な問題設定に応じて，多くの解法やアルゴリズムが開発されてきた．近年では，画像処理や検索エンジンなど，コンピュータ上でのアプリケーション開発の場面でも，数値計算での研究成果が応用される事例を見かけるようになった．これらは，多くの場合，連立一次方程式

$$A\boldsymbol{x} \quad = \quad \boldsymbol{b} \tag{1.1}$$

や固有値問題を解くことに帰着される．さらには，非線形な方程式についても，線形方程式に近似できる問題もあり，固有値問題の求解でも連立一次方程式の形に変型されるものもある．このように，様々なアプリケーション（「応用」という意味）の根本の部分で，連立一次方程式求解に帰着される場面に遭遇する．

コンピュータを用いて連立一次方程式を解くときに一環する目的は,「いかに精度良く，速く，メモリ量などの計算機資源を少なく抑えて問題を解くか」ということである．それらについて検討する際には，できあがったプログラムそのものやアルゴリズムよりも，上記のような方程式に用いられている行列 A の代数的な性質や構造に着目すると，非常に見通しが良く，結果として，数学的に裏付けられたアルゴリズムが開発されてきた．一方で，数値計算，数値解析では，線形代数とは異なる価値観を持ち合せている．驚くことに，連立一次方程式 (1.1) をコンピュータで解く際には，$\boldsymbol{x} = A^{-1}\boldsymbol{b}$ のような逆行列を作成する演算は行わないし，Cramer(クラメル) の公式を用いた求解を行うこともほとんどない．

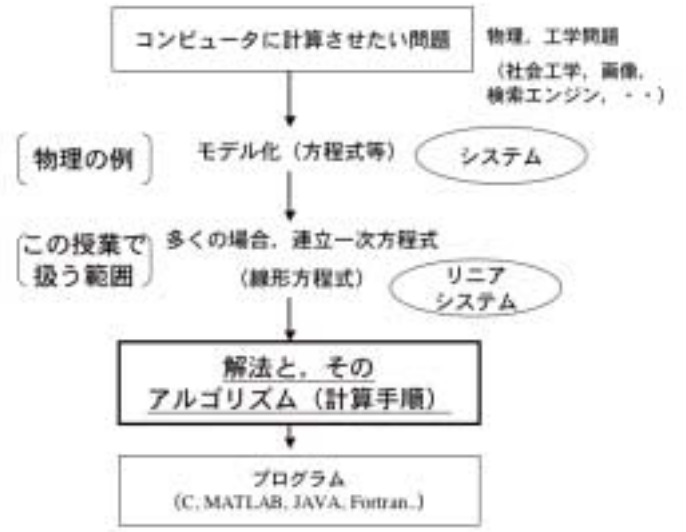

図 1.1: 計算させたい問題からプログラムに辿り着くまでと，数値計算アルゴリズムの位置付け

情報学類はじめコンピュータサイエンス分野の立場からは，様々な数値解法を汎用性のある使い易いライブラリルーチンにしていくこともさることながら，それらの各ルーチンの特徴，性質とその背景，本質を理解しておくことが大切である．様々な応用問題に対しては，対象とする問題に適した解法を選択できることが重要である．どうしても解けない問題に突き当たったときには，その原因がどこにあるかを分析し，新しい解法やアルゴリズムを開発する．また，市販の数学ライブラリパッケージや MATLAB など，数値解法をあつめて使い易い形で提供しているものもすでに多数存在するが，今後も新しいコンピュータ言語やシステム環境を開発し，その上で動作させるアプリケーションプログラムを開発する際には，既存のライブラリ等は使えず，全て０から独自に作成することになる．

日頃，コンピュータを専門とする分野では，プログラムの方に目を奪われがちであるが，その中で実現しようとしているものや本来の目的に立ち返ってみると，実は，プログラムの一歩手前の段階である**アルゴリズム**が重要である (図 1.1)．アルゴリズムを設計書として，プログラムは作成される．また，もう一つ手前の段階に目を向けると**解法**というものが存在する．解法は，数学上の議論を式などで表現したものであるが，それに対して，計算手順を意識して表現したものがアルゴリズムである．場合によっては，解法とアルゴリズムは同じもので表されることもあるが，大抵は，解法に対して，計算手順の上での様々な工夫（演算量減少，メモリ量減少，並列化など）に応じて，数種類のアルゴリズムが存在する．

この講義では連立一次方程式を取上げ，それに対する一般的な求解アルゴリズムの導出方法，および，連立一次方程式の作り方について説明する．その際に，いくつか取り上げるアルゴリズムの特徴やどのような点に注目すべきかについて特にウェイトを置いて説明している．連立一次方程式とは式が一次式の場合を指し，線形方程式とも呼ばれ，英語でも “a linear system” と表記されることが多い．具体的な解法の例としては，ガウスの消去法に基づくいくつかの解法と，共役勾配法 (Conjugate gradient method; CG 法) という解法について述べる．

1.1 連立一次方程式を解く場面について

ここでは，どのような問題が連立一次方程式に辿り着いているのかについて，物理の問題を例に取り上げて説明する．ただし，物理の問題自体を議論するわけでは無い．

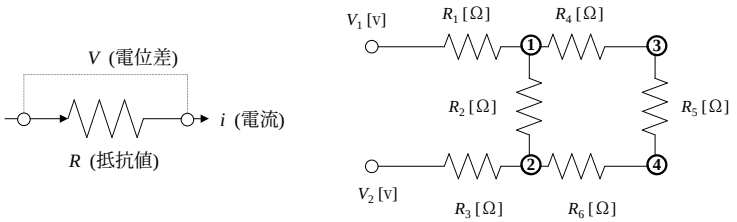

図 1.2: 単一の回路 (左) と複数からなる回路 (右)

【例１：１変数の方程式の例】

まず，図 1.2 の左図を考える．これは，抵抗器 R[Ω] に電流が i[A] 流れている時，抵抗器の両端の電位差（電圧）は，V[V] であることを表しており．

$$\frac{1}{R}\cdot V \quad = \quad i \tag{1.2}$$

という関係が成り立つ（オーム (Ohm) の法則）．ここで，R と i の値が既知だとすると，この式においては，変数 V についての一次方程式である．

【例２：連立一次方程式の例】

次に，図 1.2 の右図が表す電気回路において，節点①〜④ の電圧を求める．

節点①〜④ の電圧を v_1〜v_4 として，キルヒホッフ (Kirchhoff) の法則「各節点に流れ込む電流の和は 0 である」を用いると，

$$\begin{aligned}
\text{節点①} \quad &: \quad \frac{V_1 - v_1}{R_1} + \frac{v_2 - v_1}{R_2} + \frac{v_3 - v_1}{R_4} = 0 \\
\text{節点②} \quad &: \quad \frac{V_2 - v_2}{R_3} + \frac{v_1 - v_2}{R_2} + \frac{v_4 - v_2}{R_6} = 0 \\
\text{節点③} \quad &: \quad \frac{v_1 - v_3}{R_4} + \frac{v_4 - v_3}{R_5} = 0 \\
\text{節点④} \quad &: \quad \frac{v_3 - v_4}{R_5} + \frac{v_2 - v_4}{R_6} = 0
\end{aligned}$$

となる．これらを整理すると，

$$\begin{cases}
(R_2R_4 + R_1R_4 + R_1R_2)v_1 - R_1R_4v_2 - R_1R_2v_3 & = & R_2R_4V_1 \\
-R_3R_6v_1 + (R_2R_6 + R_3R_6 + R_3R_2)v_2 - R_2R_3v_4 & = & R_2R_6V_2 \\
-R_5v_1 + (R_4 + R_5)v_3 - R_4v_4 & = & 0 \\
-R_5v_2 - R_6v_3 + (R_5 + R_6)v_4 & = & 0
\end{cases} \tag{1.3}$$

という連立一次方程式が得られる．これを行列とベクトルを用いて表すと，

$$\begin{bmatrix} R_2R_4+R_1R_4+R_1R_2 & -R_1R_4 & -R_1R_2 & 0 \\ -R_3R_6 & R_2R_6+R_3R_6+R_3R_2 & 0 & -R_2R_3 \\ -R_5 & 0 & R_4+R_5 & -R_4 \\ 0 & -R_5 & -R_6 & R_5+R_6 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} R_2R_4V_1 \\ R_2R_6V_2 \\ 0 \\ 0 \end{bmatrix} \quad (1.4)$$

となる．

１番目の例題のような１元の一次方程式に対し，係数 $\frac{1}{R}$ の逆数を両辺に掛ければ，求めたい変数 V の解を求めることができる．しかし，２番目の例題のように連立一次方程式の場合，通常は，導出された行列の逆行列を求めるようなことはせず，連立一次方程式の解法に当てはめて求解する．

さらに，式 (1.4) の右辺に着目すると，回路の端の電圧 V_1, V_2 は，この項にある．つまり，V_1, V_2 の値を変化させても，式 (1.4) の係数行列は変化しない．このような特長は，第２章で学ぶ **LU 分解**で有効である．

【例３：もうひとつ別の例】

いま，図 1.3(左) のように，長さ l の糸に重さ m の n 個の質点（大きさ，形を考えないおもりのこと）を等間隔につけて，糸の両端を水平に固定したときの重力の影響による糸のたわみを考える．各質点は，両隣からの張力（一様に T とする）の影響も受けている．

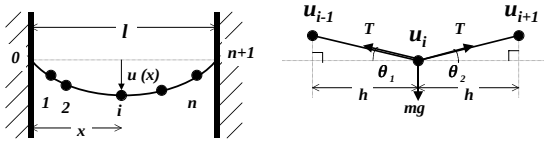

図 1.3: 糸のたわみの例題 (左) と力のつりあい (右).

糸がたわんでいない状態での質点の間隔を記号 h と表すと，$h = \frac{l}{n+1}$ である．図 1.3(左) において，左端からの長さ x における質点のたわみの量を $u(x)$ とする．ここで，質点の番号をあらためて i というインデックスを用いて表すと $x = ih$ である[1]．このとき，質点 u_i(すなわち $u(x)$) に作用する力の関係は，図 1.3(右) で表され，それを方程式で表現すると，

$$mg - T\sin\theta_1 - T\sin\theta_2 = 0 \quad (1.5)$$

である．θ_1, θ_2 が十分に小さいときは，

$$\begin{aligned} \sin\theta_1 &\approx \tan\theta_1 = \frac{u_{i-1}-u_i}{h}, \\ \sin\theta_2 &\approx \tan\theta_2 = \frac{u_{i+1}-u_i}{h} \end{aligned}$$

と考えて良い（記号「≈」は「近似」を意味する．このような近似方法は，物理学的な発想である）．

これにより，方程式 (1.5) は，

$$mg - T\frac{u_{i-1}-u_i}{h} - T\frac{u_{i+1}-u_i}{h} = 0 \quad (1.6)$$

[1] 通常，x という表現は連続量のイメージで用いるので，離散点では i などインデックスを用いた表現にすることが一般的である．また，糸がたわんだ時の間隔 h の変化は微少量であると見做し，h のままで議論しても良い（$\sin\theta \approx \tan\theta$ と同じ考え方）．

と，h に対する u の変化量の**差分**で表される．この式を整理して，

$$-u_{i-1}+2u_i-u_{i+1} \quad = \quad -\frac{mgh}{T} \tag{1.7}$$

という i に関する漸化式が得られる．また，$i=0,n+1$ の点における条件（ここでは，糸の両端にあたる境界部分の条件を指し，一般に**境界条件**と言い，g_0,g_{n+1} などと記述する）は，

$$u_0=g_0=0, \quad u_{n+1}=g_{n+1}=0 \tag{1.8}$$

である（つまり，これら 2 箇所には，たわみが無い）．

式 (1.7) をすべての質点 $(i=1,2,\cdots,n)$ について書き並べると，連立一次方程式が得られる．各質点のたわみを表現したベクトル $\boldsymbol{u}$ を

$$\boldsymbol{u} \quad = \quad (u_1,u_2,u_3,\cdots,u_{n-1},u_n)^T$$

とすると，連立一次方程式 $A\boldsymbol{u}=\boldsymbol{b}$ は，

$$\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & \mathbf{0} & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} c+g_0 \\ c \\ c \\ \vdots \\ c \\ c+g_{n+1} \end{bmatrix} \tag{1.9}$$

と表すことができる．ただし，$c=-\frac{mgh}{T}$ である．

ここでも，式 (1.9) の右辺に着目すると，境界条件の g_0,g_{n+1} は，この項にあり，g_0,g_{n+1} の値を変化させても，式 (1.9) の係数行列は変化しない．

1.2 連立一次方程式について

連立一次方程式は

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ & & & & \vdots & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{cases} \tag{1.10}$$

と書ける．ここで注意するのは，式の本数と未知数の数が同じこと，$x_1\cdot x_2$ のような未知数の積を含む項がないことである．(1.10) は $n\times n$ の行列 A と n 次元の列ベクトル $\boldsymbol{x}$,$\boldsymbol{b}$ を用いて，式 (1.1) と同じく，

$$A\boldsymbol{x}=\boldsymbol{b}$$

と書ける．このとき，

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \vdots & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},\ \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},\ \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{1.11}$$

である．この連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ が解を持つ条件は A が正則なことである．

このテキストでは，連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を如何に解くかということに焦点を当てており，この式が随所に現れる．

正則

n 次の正方行列 A に対して，

$$AX = XA = I \quad (I : n \text{ 次単位行列}) \tag{1.12}$$

となる正方行列 X が存在するとき，X を A の逆行列といい，A^{-1} で表す．A に逆行列が存在するとき，A は正則であるという．

定理 (正則行列の性質)　n 次正方行列 $A = [a_{ij}]$ に関する次の４つの条件は同値である．

(i) A は正則行列である．

(ii) 連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ はただ１組の解をもつ．

(iii) $\mathrm{rank}(A) = n$

(iv) $\det(A) \neq 0$

その他の連立一次方程式自体についての詳細は，線形代数のテキストで確認して欲しい．

第2章　ガウスの消去法に基づく解法

2.1　ガウス消去法

この節では，連立一次方程式を

$$\left\{\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ & & & & \vdots & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}\right. \tag{2.1}$$

と表現して議論する．

ガウスの消去法では，まず最初に第 1 番方程式を利用して x_1 の係数を消去する．それには，第 1 番方程式を何倍かしたものを，第 2 番方程式〜第 n 番方程式の各々から引いて，x_1 の係数を 0 にすればよい．この操作は同値な変形である．つまり第 i 番方程式用の x_1 の係数を 0 にするために，第 1 番方程式に乗ずる数（乗数）を m_{i1} とすると，

$$\begin{array}{cccccccc} & a_{i1}x_1 & + & \cdots & + & a_{in}x_n & = & b_i \\ -) & m_{i1}(a_{11}x_1 & + & \cdots & + & a_{1n}x_n) & = & m_{i1}b_1 \\ \hline & (a_{i1}-m_{i1}a_{11})x_1 & + & \cdots & + & (a_{in}-m_{i1}a_{1n})x_n & = & b_i - m_{i1}b_1 \end{array}$$

である．ここで，x_1 の係数を $a_{i1} - m_{i1}a_{11} = 0$ とするには

$$m_{i1} = \frac{a_{i1}}{a_{11}}, \quad i = 2, \cdots, n$$

とすればよい．$a_{11} \neq 0$ ならこの操作は実行できて，方程式ごとに異なった乗数 $m_{21},\ m_{31},\ \cdots, m_{n1}$ が求まる．

消去の第 1 段として，第 1 番方程式に m_{i1} を乗じて第 i 番方程式 $(i = 2, \cdots, n)$ に加える操作を繰り返した結果，連立一次方程式 (2.1) は等価な連立一次方程式

$$\left\{\begin{array}{ccccccccc} a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & + & \cdots & + & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & & a_{22}^{(2)}x_2 & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & & & \vdots & & & & \\ & & a_{n2}^{(2)}x_2 & + & \cdots & + & a_{nn}^{(2)}x_n & = & b_n^{(2)} \end{array}\right. \tag{2.2}$$

に変換される．ここで $a_{22}^{(2)}$, $b_2^{(2)}$ などと右上にある () で囲まれた数字は,

$$\begin{aligned} m_{i1} &= \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, & i = 2, \cdots, n, \\ a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, & i, j = 2, \cdots, n, \\ b_i^{(2)} &= b_i^{(1)} - m_{i1} b_1^{(1)}, & i = 2, \cdots, n \end{aligned}$$

によって更新された値である．右上が (1) というのは，元々の方程式から変更されていない値であることを意味する．この操作を第 k 番方程式に対する消去まで繰り返すと,

$$\left\{ \begin{array}{ccccccccccccc} a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & + & \cdots & + & a_{1k}^{(1)}x_k & + & \cdots & + & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & & a_{22}^{(2)}x_2 & + & \cdots & + & a_{2k}^{(2)}x_k & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & & & \ddots & & \vdots & & & & \vdots & & \vdots \\ & & & & & & a_{kk}^{(k)}x_k & + & \cdots & + & a_{kn}^{(k)}x_n & = & b_k^{(k)} \\ & & & & & & \vdots & & & & \vdots & & \vdots \\ & & & & & & a_{nk}^{(k)}x_k & + & \cdots & + & a_{nn}^{(k)}x_n & = & b_n^{(k)} \end{array} \right. \tag{2.3}$$

となる．次の第 k 段では「第 k 番方程式を用いて第 $(k+1)$ 番以降の方程式の x_k を 消去」する．すなわち $a_{kk}^{(k)} \neq 0$ のとき

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, \cdots, n \tag{2.4}$$

を用いて

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \qquad i, j = k+1, \cdots, n, \tag{2.5}$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}, \qquad i = k+1, \cdots, n \tag{2.6}$$

という操作を施せばよい．このようにして第 n 番方程式による消去までを繰り返すと，連立した方程式は

$$\left\{ \begin{array}{ccccccccccccc} a_{11}^{(1)}x_1 & + & \cdots & + & a_{1k}^{(1)}x_k & + & \cdots & + & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & & a_{22}^{(2)}x_2 & + & a_{2k}^{(2)}x_k & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & \ddots & & \vdots & & & & \vdots & & \vdots \\ & & & & a_{kk}^{(k)}x_k & + & \cdots & + & a_{kn}^{(k)}x_n & = & b_k^{(k)} \\ & & & & & \ddots & & & \vdots & & \vdots \\ & & & & & & a_{n-1,n-1}^{(n-1)}x_{n-1} & + & a_{n-1,n}^{(n-1)}x_n & = & b_{n-1}^{(n-1)} \\ & & & & & & & & a_{nn}^{(n)}x_n & = & b_n^{(n)} \end{array} \right. \tag{2.7}$$

となる[1]．ここまでの過程を**前進消去** (forward elimination)，または**上三角化** という．

[1] 本テキストでの添字の約束として，a_{ij} という記述は，a の (i,j) 成分を表し，$a_{n-1,j}$ という記述は，a の $(n-1,j)$ 成分を表している（カンマの有無は，添字の表すものが誤解されないよう，適宜付けている点に注意）.

一番下の第 n 番方程式には，未知数が x_n だけしか含まれていないので，$a_{nn}^{(n)} \neq 0$ なら

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}} \tag{2.8}$$

と解ける．第 n 番方程式から x_n が求まれば，第 $(n-1)$ 番方程式より

$$x_{n-1} = \frac{1}{a_{n-1,n-1}^{(n-1)}} \left(b_{n-1}^{(n-1)} - a_{n-1,n}^{(n-1)} x_n \right) \tag{2.9}$$

が求まる．このように下から順に繰り返せば，x_k を計算する時点で，$x_{k+1}, \cdots, x_n$ は既知なので，

$$x_k = \frac{1}{a_{kk}^{(k)}} \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j \right), k = n-1, \cdots, 1 \tag{2.10}$$

として求められる．この過程を **後退代入** (backward substitution) という．前進消去が問題なく終了していれば $a_{kk}^{(k)} \neq 0$ である．

ガウスの消去法を用いれば，解が存在する多くの連立一次方程式は解けるが，以下のようなとき

$$\begin{cases} 0x_1 + 1x_2 + 2x_3 &= 2 \\ -1x_1 + 0x_2 + 3x_3 &= 4 \\ 3x_1 + 1x_2 + 0x_3 &= -3 \end{cases} \tag{2.11}$$

は a_{11} が 0 となって，解が存在するのにもかかわらず解けない．このような場合でもアルゴリズムが破綻しないよう，**部分軸選択** (partial pivoting) を導入する．

部分軸選択 (1)

部分軸選択とは第 k 段において，第 k 番方程式をそのまま使うのではなく，第 k 番方程式から第 n 番方程式までの x_k の係数 $a_{kk}^{(k)}, \cdots, a_{nk}^{(k)}$ から “絶対値が最大” のものを探索し，その係数を持つ方程式と第 k 番方程式を入れ換えてから消去を行う操作のことである．すなわち，対角要素 ($a_{kk}^{(k)}$：枢軸，ピボット要素とも呼ぶ) が絶対値最大となるようにする．これによって，単に解けるようになるだけでなく，各段の作業において $|m_{ik}| \leq 1$ が保証され，コンピュータを使用する際の丸め誤差の増幅を防ぐことにも役立っている．消去結果のすべての対角要素がゼロでなければ，この行列は正則である．

例として (2.11) を部分軸選択つきガウスの消去法で解いてみよう．

前進消去

$k = 1$ **段:**

1 列目 1 行以下で絶対値最大の要素は 3 行目にあるので，第 1 番方程式と第 3 番方程式を交換する．

$$\begin{cases} 3x_1 + 1x_2 + 0x_3 = -3 \\ -1x_1 + 0x_2 + 3x_3 = 4 \\ 0x_1 + 1x_2 + 2x_3 = 2 \end{cases}$$

第 2 番方程式の x_1 を消去するため 第 1 番方程式 $\times\,(-\frac{1}{3})$ を第 2 番方程式に加える ($m_{21} = -\frac{1}{3}$ ということ).

$$\begin{cases} 3x_1 + 1x_2 + 0x_3 = -3 \\ 0x_1 + \frac{1}{3}x_2 + 3x_3 = 3 \\ 0x_1 + 1x_2 + 2x_3 = 2 \end{cases}$$

$k = 2$ **段:**

2 列目 2 行以下で絶対値最大の要素は 3 行目にあるので，第 2 番方程式と第 3 番方程式を交換する．

$$\begin{cases} 3x_1 + 1x_2 + 0x_3 = -3 \\ 0x_1 + 1x_2 + 2x_3 = 2 \\ 0x_1 + \frac{1}{3}x_2 + 3x_3 = 3 \end{cases}$$

第 3 番方程式の x_2 を消去するため，第 2 番方程式 $\times\,(\frac{1}{3})$ を第 3 番方程式に加える ($m_{32} = \frac{1}{3}$).

$$\begin{cases} 3x_1 + 1x_2 + 0x_3 = -3 \\ 0x_1 + 1x_2 + 2x_3 = 2 \\ 0x_1 + 0x_2 + \frac{7}{3}x_3 = \frac{7}{3} \end{cases}$$

$k = 3$ **段:**

第 3 番方程式の x_3 の係数はゼロではない．前進消去ですべての対角要素 $\neq 0$ となったので，行列 A は正則である．ただし，$k = 3$ 段では，実際の計算は行われない．

後退代入

$k = 3$ **段:** $\frac{7}{3}x_3 = \frac{7}{3}$ より $x_3 = 1$.

$k = 2$ **段:** $x_2 + 2x_3 = 2$ を移項した $x_2 = 2 - 2x_3$ に $x_3 = 1$ を代入して $x_2 = 0$.

$k = 1$ **段:** $3x_1 + x_2 = -3$ より $x_1 = \frac{1}{3}\,(-3 - x_2)$，$x_2 = 0$ を代入して $x_1 = -1$.

このように無事に解が求まった．部分軸選択は，枢軸が 0 になるとき以外にも効果がある．これについては，後の節 (p.17) であらためて説明している．

部分軸選択つきガウス消去法のアルゴリズム

以上の計算手順をまとめた部分軸選択つきガウス消去法を，アルゴリズム 2.1 で表した[2] ．記号は

- $=$ は右辺の値を左辺に代入
- $\rightleftharpoons$ は値の交換
- **for** $k = 1,\ 2,\ \cdots,\ n$ は，変数 k を 1 から n まで変えて，**end** までの演算を繰り返す.

を意味する．斜体文字は変数，太字のローマン体はアルゴリズムの制御文を表す.

[2] アルゴリズムは計算手順を示すものであり，任意のプログラミング言語に対応できるものでなければならない．数式を用いて表現すると，どのような計算を行なっているのか確認し易い．実際に数式からプログラミングする際のポイントについては，付録の章を参照.

アルゴリズム: 2.1 (部分軸選択付きガウス消去法)

for $k = 1, 2, \cdots, n-1$
　　/* ピボッティング開始 */
　　$amax = \mid a_{kk} \mid ; p = k$
　　for $i = k+1, k+2, \cdots, n$
　　　　if $\mid a_{ik} \mid > amax$ **then**
　　　　　　$amax = \mid a_{ik} \mid ; p = i$
　　　　end if
　　end
　　for $j = k, k+1, \cdots, n$
　　　　$a_{pj} \rightleftharpoons a_{kj}$
　　end
　　$b_p \rightleftharpoons b_k$
　　/* ピボッティング終了 */
　　/* 前進消去 */
　　for $i = k+1, k+2, \cdots, n$
　　　　$m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$
　　　　for $j = k+1, k+2, \cdots, n$
　　　　　　$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$　　(*)
　　　　end
　　　　$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$　　(**)
　　end
end
/* 後退代入 */
　　$x_n = b_n^{(n)} / a_{nn}^{(n)}$　　(***)
for $k = n-1, \cdots, 1$
　　$x_k = (b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j) / a_{kk}^{(k)}$　　(***)
end

実際のプログラミングでは，$amax$ が許容値より小さい時には，ゼロとみなすなどの処理も記述しておく．
アルゴリズム 2.1 に必要な浮動小数演算 (四則演算) の概数を見積ると

前進消去	左辺	$\frac{2}{3}n^3$ 回	$(*)$ 部分の演算数	((2.5) 式の計算)
	右辺	n^2 回	$(**)$ 〃	((2.6) 〃)
後退代入	右辺	n^2 回	$(***)$ 〃	((2.8)〜(2.10) 〃)

である．

2.2 LU 分解

この節では，連立一次方程式を第 1 章の式 (1.1) で示したように，

$$A\boldsymbol{x} \quad = \quad \boldsymbol{b} \tag{2.12}$$

と表現して議論する．

この方程式に対して，係数行列は同じだが右辺項 $\boldsymbol{b}$ が異なるような問題を解くことが多い．例えば，1.1 節の例 2 のように同じ回路に対して，与える電圧 (V_1, V_2) のみを変化させるような場合である．このとき，各々に対してガウス消去法を適用していると，計算コストの高い係数行列部分に対して同じ演算を施すこととなり，無駄な計算が生ずる．ここでは，あらかじめ係数行列に対する演算のみを行ない，結果的にガウス消去法と等価な計算を行なう．このときの操作では，行列を用いて表現し，行列演算による操作を施すことでガウス消去法と等価な “LU 分解” を用いた計算方法について説明する．

また，行列を用いると，様々な演算を簡略された記号で表すことができ，各操作に対する見通しが良くなるのも大きな利点である．

ガウスの消去法 第 k 段の操作 (ピボット無しの場合)

$$\text{行列 } M_k \quad : \quad k \text{ 番方程式を } m_{i,k} \text{倍して } i \text{ 番方程式 } (i = k+1, \cdots, n) \text{ に加える}$$

この操作は行列の基本変形を応用して，次の行列で表せる．

$$M_k = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \overset{\overset{k \text{ 列}}{\downarrow}}{\begin{bmatrix} 1 & & & & & & & \\ 0 & \ddots & & & & & & \\ \vdots & 0 & 1 & & & & & \\ \vdots & \vdots & -m_{k+1,k} & 1 & & & & \\ \vdots & \vdots & \vdots & 0 & 1 & & & \\ \vdots & \vdots & -m_{i,k} & \vdots & & 1 & & \\ \vdots & \vdots & \vdots & \vdots & & & \ddots & \\ 0 & 0 & -m_{n,k} & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ k+1 \text{ 行} \end{array} \tag{2.13}$$

ここで，$m_{i,k}$ は，(2.4) と同じく，

$$m_{i,k} = \frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}, \qquad i = k+1, \cdots, n$$

である．

ガウス消去法の前進消去は M_k を使って

$$M_{n-1} \ \cdots \ M_k \cdots \ M_1 A\boldsymbol{x} = M_{n-1} \ \cdots M_k \cdots \ M_1 \boldsymbol{b} \tag{2.14}$$

と書ける．$M_{n-1}\ \cdots\ M_k\ \cdots\ M_1A$ を計算すると，上三角行列 (Upper triangular matrix)

$$U = \begin{bmatrix} a_{11}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \cdots & \cdots \\ & \Large 0 & & & a_{nn}^{(n)} \end{bmatrix} \tag{2.15}$$

となる．ここでの $a_{ij}^{(k+1)}$ の計算方法も，ガウス消去法の (2.5) と同じく，

$$a_{ij}^{(k+1)} \quad = \quad a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \qquad i,j = k+1,\cdots,n$$

である．ここでは，右辺項に対する (2.6) 式の演算は行わない点に注意．式 (2.15) の行列 U は，

$$M_{n-1}\ \cdots\ M_1A = U$$

であり，

$$A = (M_{n-1}\ \cdots\ M_1)^{-1}U$$

すなわち

$$A = (M_1^{-1}M_2^{-1}\ \cdots M_{n-1}^{-1})U \tag{2.16}$$

である．

ここで，M_k の逆行列 M_k^{-1} は，要素 $m_{i,k}(i = k+1,\cdots,n)$ の符号を変えたもの

$$M_k^{-1} = \begin{bmatrix} 1 & & & & & & & \\ 0 & \ddots & & & & & \Large 0 & \\ \vdots & 0 & 1 & & & & & \\ \vdots & \vdots & m_{k+1,k} & 1 & & & & \\ \vdots & \vdots & \vdots & 0 & 1 & & & \\ \vdots & \vdots & m_{i,k} & \vdots & & 1 & & \\ \vdots & \vdots & \vdots & \vdots & & & \ddots & \\ 0 & 0 & m_{n,k} & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \tag{2.17}$$

であり（各自で確かめてみよう），式 (2.16) の $M_1^{-1}M_2^{-1}\cdots M_{n-1}^{-1}$ 部分は，下三角行列 (Lower tridiagonal matrix)

$$
L = \begin{bmatrix}
1 & & & & & & \\
m_{2,1} & \ddots & & & & & \\
m_{3,1} & & 1 & & & & \\
\vdots & & m_{k+1,k} & 1 & & & \\
\vdots & & \vdots & m_{k+2,k+1} & 1 & & \\
\vdots & & m_{i,k} & \vdots & & \ddots & \\
\vdots & & \vdots & \vdots & & & 1 \\
m_{n,1} & & m_{n,k} & m_{n,k+1} & \cdots & m_{n,n-1} & 1
\end{bmatrix}
\qquad (2.18)
$$

となる．(2.18) の行列 L は，$L = (M_{n-1} \cdots M_1)^{-1}$ である．

このようにすると，行列 A は (2.16) のとおり，$A = LU$ と分解できる．これを，A の **LU 分解** (LU decomposition または LU factorization) と呼ぶ．

LU 分解では，このように係数行列 A を下三角行列 L と上三角行列 U に分解しておき，方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の右辺に対する計算を $LU\boldsymbol{x} = \boldsymbol{b}$ と考え，補助的なベクトル $\boldsymbol{y}$ を用いて，

$$
\begin{cases}
L\boldsymbol{y} = \boldsymbol{b}, \\
U\boldsymbol{x} = \boldsymbol{y}
\end{cases}
\qquad (2.19)
$$

の 2 段階に分けて計算する．すなわち L に対する前進代入と U に対する後退代入で済ます．

具体的な計算は，次の手順で行なう．

【前進代入】

$$
\begin{aligned}
y_1 &= b_1, && (k = 1 \text{ のとき}), \\
y_k &= b_k - \sum_{j=1}^{k-1} m_{kj} y_j, && k = 2, 3, \cdots, n
\end{aligned}
\qquad (2.20)
$$

この前進代入の式と，ガウス消去法における (2.6) 式とを比較してみよう．

【後退代入】

$$
\begin{aligned}
x_n &= \frac{y_n}{a_{nn}^{(n)}}, && (k = n \text{ のとき}), \\
x_k &= \frac{1}{a_{kk}^{(k)}} \left(y_k - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j \right), && k = n-1, n-2, \cdots, 1
\end{aligned}
\qquad (2.21)
$$

この後退代入の式と，ガウス消去法における (2.8)〜(2.10) 式とを比較してみよう．

ここで，ガウス消去法と LU 分解を用いた求解法の演算量について比較してみる．

前述のとおり，ガウス消去法では，

・前進消去の演算に対して，$\frac{2}{3}n^3 + n^2$

・後退代入に対して，n^2

の演算量が必要である．一方，LU 分解を用いた方法では，

・係数行列の LU 分解に対して，$\frac{2}{3}n^3$

・前進代入と後退代入の演算はほぼ同じであり，合わせると，$2n^2$

である．

LU 分解のメリットと求解において逆行列を用いない理由

LU 分解を用いれば，複数の右辺に対しても左辺の前進消去は 1 回でよい．したがって，係数行列 A は同じだが，異なる右辺項ベクトル $\boldsymbol{b}$ が s 個ある n 元連立一次方程式を解く（すなわち，s 回解く）ための浮動小数演算回数はおよそ

$$\frac{2}{3}n^3 + s \times 2n^2 \text{ 回}$$

であり，ガウス消去法を s 回行うよりも計算量がグンッと少なくなる．

このようにして考えていくと，$s = n$ のとき，すなわち $AX = B$ を満足する行列 X ，あるいは逆行列 $(AX = I)$ を求めるとき必要な浮動小数演算回数はおよそ $\frac{8}{3}n^3$ 回である．(1.1) の解を逆行列を使って書けば

$$\boldsymbol{x} = A^{-1}\boldsymbol{b}$$

だが，コンピュータでこの式のとおりに逆行列 A^{-1} を作って解こうとすると，

$$\frac{8}{3}n^3 \text{ 回}$$

の浮動小数演算が必要となる．逆行列を作る場合の演算量と，ガウス消去法や LU 分解を用いて直接計算する場合の $\frac{2}{3}n^3 + 2n^2$ とを比べると，いかに無駄な計算をしているかがわかる．しかも，逆行列を使うと誤差が入りやすい．このため，特に逆行列の要素の値を知りたいとき以外，**数値計算では逆行列を使わない**のが常識である．

部分軸選択 (ピボッティング) 付き LU 分解

LU 分解の k 段におけるピボッティング

行列 P_k ： k 番方程式と p 番方程式を入れ換える

※ 実際には, p の値は各段に応じて異なることに留意

これは次の行列で表せる．

$$
P_k = \begin{array}{c}
\begin{array}{cccccccccccc}
 & & & k\text{列} & & & & p\text{列} & & & &
\end{array} \\
\left[\begin{array}{cccccccccccc}
1 & & & \vdots & & & & \vdots & & & & \\
 & \ddots & & \vdots & & & & \vdots & & & & \\
 & & 1 & \vdots & & & & \vdots & & & & \\
\cdots & \cdots & \cdots & 0 & \cdots & \cdots & \cdots & 1 & \cdots & \cdots & \cdots & k\text{行} \\
 & & & \vdots & 1 & & & \vdots & & & & \\
 & & & \vdots & & \ddots & & \vdots & & & & \\
 & & & \vdots & & & 1 & \vdots & & & & \\
\cdots & \cdots & \cdots & 1 & \cdots & \cdots & \cdots & 0 & \cdots & \cdots & \cdots & p\text{行} \\
 & & & \vdots & & & & \vdots & 1 & & & \\
 & & & \vdots & & & & \vdots & & \ddots & & \\
 & & & \vdots & & & & \vdots & & & 1 &
\end{array}\right]
\end{array}
\tag{2.22}
$$

ここで，部分軸選択付き LU 分解の前進消去は，M_k と P_k を使って

$$
M_{n-1}P_{n-1} \cdots M_kP_k \cdots M_1P_1A\boldsymbol{x} = M_{n-1}P_{n-1} \cdots M_1P_1\boldsymbol{b} \tag{2.23}
$$

と表される $(k = 1, 2, \cdots, n-1)$.

$$
M_{n-1}P_{n-1} \cdots M_1P_1A = U
$$

とおいて

$$
A = (M_{n-1}P_{n-1} \cdots M_1P_1)^{-1}U
$$

これを

$$
\begin{aligned}
L'U &= A \\
L' &= (M_{n-1}P_{n-1}\cdots M_1P_1)^{-1} \\
&= P_1^{-1}M_1^{-1}\cdots P_{n-1}^{-1}M_{n-1}^{-1}
\end{aligned} \tag{2.24}
$$

と表して，これも A の **LU 分解** と呼ぶ．プログラミングの際には，LU 分解と求解（前進代入，後退代入）とで，別々の関数にするなど，プログラムを分けておくのが普通である．そのとき，行交換のインデックスを記憶したベクトル（プログラミングでは配列）に，交換した情報を格納しておく．

ところで，$P_1 = P_2 = \cdots = P_{n-1} = I$ のとき（行交換が不要な場合）には，L' は下三角行列 L と等しくなる．部分軸選択付き LU 分解では，L' は下三角行列にはならないが，一般にこのような場合でも，下三角行列と呼ぶ．理由は次のとおり．

いま，必要な行交換をあらかじめ A に施しておくと考えると，結果的に行交換は不要なので，

$$
PA = LU \tag{2.25}
$$

となる（実際には，枢軸の値は分解の過程で変化するため，前もって必要な行交換を施すようなことはしないが）．ここで，

$$P = P_{n-1} \cdots P_2 P_1$$

である．したがって，

$$PL' = L \tag{2.26}$$

となり，つまり，L' に行交換をすれば下三角行列 L になるので，L' も広義に解釈して下三角行列と考える．

実際の求解では，$A\boldsymbol{x} = \boldsymbol{b}$ に対して，

$$PA\boldsymbol{x} \quad = \quad LU\boldsymbol{x} = P\boldsymbol{b} \tag{2.27}$$

と変換されている．つまり，右辺項ベクトル $\boldsymbol{b}$ に対し入換えを行なったものについて連立一次方程式を解く．

部分軸選択 (2)

ガウス消去法の説明で，ピボッティングについて説明した (p.9)．そこでは，枢軸が 0 となることを回避するための手法として説明したが，ピボッティングの重要性はこれだけではなく，有限桁のコンピュータにおける演算の安定性 (誤差の影響具合など) にも関係する．枢軸が 0 ではなくとも，小さい値のまま用いた場合について説明する．

$$\begin{bmatrix} 1.00 & -1.00 & 0.00 \\ -1.00 & 1.02 & 5.00 \\ 0.00 & -0.60 & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 5.02 \\ 0.40 \end{bmatrix} \tag{2.28}$$

を 10 進 3 桁切捨てで計算する（真の解は，$x_1 = x_2 = x_3 = 1.00$ である）．

● ピボッティングを行なわずに消去した場合（LU 分解を用いても同じ）：
第 1 段：

$$\begin{bmatrix} 1.00 & -1.00 & 0.00 \\ 0.00 & \underline{0.02} & 5.00 \\ 0.00 & \underline{-0.60} & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 5.02 \\ 0.40 \end{bmatrix} \tag{2.29}$$

ここで，乗数は $m_{3,2} = \frac{-0.60}{0.02} = -30.0$ であるから，

$$\begin{aligned} a_{3,3}^{(3)} &= 1.00 - (-30.0) \times 5.00 \ = \ 151 \\ b_3^{(3)} &= 0.40 - (-30.0) \times 5.02 \\ &= 0.40 + 150.\underline{6} \ \approx \ 0.40 + 150 \qquad (*) \\ &= 150.\underline{4} \ \approx \ 150 \qquad (**) \end{aligned} \tag{2.30}$$

となる（記号 $\approx$ は，“近似” を意味する）．ここでは，(*) の演算で “丸め誤差” [3] ，(**) にて “情報落ち” [4] による誤差が生じている．

第２段：

$$\begin{bmatrix} 1.00 & -1.00 & 0.00 \\ 0.00 & \underline{0.02} & 5.00 \\ 0.00 & 0.00 & 151 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 5.02 \\ 150 \end{bmatrix} \tag{2.31}$$

これを後退代入すると，

$$x_3 = 0.993, \quad x_2 = 3.00, \quad x_1 = 3.00$$

と，x_3 以外は誤差が非常に大きい．理由は，式 (2.31) から後退代入で，x_2 を計算する際に，

$$\begin{aligned} 0.02 \times x_2 &= 5.02 - 5.00 \times 0.993 \\ &= 5.02 - 4.96\underline{5} \qquad (*) \\ &\approx 5.02 - 4.96 \qquad (***) \\ &= 0.06 \end{aligned}$$

となり，(*) の演算で丸め誤差が生じ，最終的には，(***) の段階で “桁落ち” [5] が生じている．

この例では，第２段において第２式の枢軸 0.02 の絶対値が，第３式の２番目の値 -0.60 の絶対値と比べて小さいのにも関わらず，ピボッティングを行わなかったために，特に，“桁落ち” が生じた．その結果，右辺に混入した誤差が拡大されて x_2, x_1 と精度が悪くなっていったのである．

● ピボッティングを行なった場合（LU 分解を用いても同じ）：

第１段：

ピボッティング無しの場合と同じである．

第２段：

ピボッティングを行った結果は，次の通りである．

$$\begin{bmatrix} 1.00 & -1.00 & 0.00 \\ 0.00 & \underline{-0.60} & 1.00 \\ 0.00 & \underline{0.02} & 5.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 0.40 \\ 5.02 \end{bmatrix} \tag{2.32}$$

ここで，乗数は $m_{3,2} = \frac{0.02}{-0.60} = -0.0333\underline{333\cdots} \approx -0.0333$ であるから，

$$\begin{aligned} a_{3,3}^{(3)} &= 5.00 - (-0.0333) \times 1.00 \\ &= 5.00 + 0.03\underline{33} \approx 5.03 \qquad (*) \end{aligned}$$

[3] **丸め誤差**：数値の桁数が計算機の有効桁数を上回った場合，有効桁に近似せざるを得ない．ここで生じた誤差のこと．この近似を “丸める” と呼び，計算機の処理に応じて「切り上げ」「切り捨て」「四捨五入」などがある．

[4] **情報落ち**：絶対値が大きく異なる数値の加減算では，計算機が扱える桁数の制限により小さい方の数値の下位桁が失われてしまう現象である．

[5] **桁落ち**：絶対値が近い数値どうしの減算を行うと，計算結果の有効数字の上位桁が 0 になってしまい，結局，有効桁数が少なくなったために誤差の影響が大きくなる．

$$
\begin{aligned}
b_3^{(3)} &= 5.02 - (-0.0333) \times 0.40 \\
&= 5.02 + 0.01332 \qquad (*) \\
&\approx 5.0333 \qquad (*) \\
&\approx 5.03
\end{aligned}
$$

となる．ここでは，(*) の丸め誤差は随所に生じているものの，計算結果には大きな影響をもたらさず，情報落ちも桁落ちも生じていないことに注目して欲しい．

$$
\begin{bmatrix} 1.00 & -1.00 & 0.00 \\ 0.00 & -0.60 & 1.00 \\ 0.00 & 0.00 & 5.03 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 0.40 \\ 5.03 \end{bmatrix} \tag{2.33}
$$

ピボッティングを行なった場合には，正しい結果が得られる．

ガウス消去法と LU 分解についてのまとめ

ガウス消去法と LU 分解との間に本質的な差はない．これらの解法には，扱う行列の特徴，計算環境などによって色々とバラエティに富んだアルゴリズムが開発されている．線形代数の教科書などでは，ガウス消去法をさらに発展させたかのごとく，最終的に係数行列を単位行列にしてしまう “ガウス-ジョルダンの消去法 (掃き出し法)” という計算法が紹介されていることが多いが，この方法は無駄な演算を伴い，コンピュータ向けのアルゴリズムとしてはあまり用いられない．

2.3 コレスキー法

係数行列の性質に着目して，LU 分解（ガウス消去法）を改良したアルゴリズムが提案されている．

ここでは，行列 A が対称かつ正定値 (SPD:Symmetric Positive Definite) 行列のときに，LU 分解の代わりに

$$A \quad = \quad LL^T, \quad L : \text{下三角行列} \tag{2.34}$$

と分解（コレスキー分解）し，下三角行列 L のみを用いて計算する**コレスキー法** (Cholesky method) について説明する．ここで L^T は，行列 L に対する転置行列を表す．

この方法のポイントは，行列の対称性に着目し，下三角行列 L のみを作成すれば上三角行列は L の転置を用いるだけで良く，あらたな計算は不要となるということである．

正定値行列： 任意の非零ベクトル$\boldsymbol{x}$に対して，

$$(\boldsymbol{x}, A\boldsymbol{x}) > 0 \tag{2.35}$$

を満たす行列を正定値行列 (positive definite matrix) という．ここで，$(\boldsymbol{x}, \boldsymbol{y})$ は，$\boldsymbol{x}$と$\boldsymbol{y}$の内積を表す．

コレスキー法 1

いま，A の要素を a_{ij}，L の要素を l_{ij} とすると，式 (2.34) は

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & \mathbf{0} & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & \cdots & l_{n1} \\ & l_{22} & l_{32} & \cdots & l_{n2} \\ & & l_{33} & \cdots & l_{n3} \\ & \mathbf{0} & & \ddots & \vdots \\ & & & & l_{nn} \end{bmatrix} \tag{2.36}$$

である．ただし，行列 A は対称行列なので，$a_{ij} = a_{ji}$ である．

ここで，(2.36) を 1 列ごとに計算していくと，

第 1 列目の計算：

$$\begin{cases} a_{11} &= \underline{l_{11}} \times \underline{l_{11}} \\ a_{21} &= \underline{l_{21}} \times l_{11} \\ a_{31} &= \underline{l_{31}} \times l_{11} \\ \vdots & \vdots \\ a_{n1} &= \underline{l_{n1}} \times l_{11} \end{cases}$$

第 2 列目の計算：

$$\begin{cases} (a_{12} &= l_{11} \times l_{21}) \\ a_{22} &= l_{21} \times l_{21} + \underline{l_{22}} \times \underline{l_{22}} \\ a_{32} &= l_{31} \times l_{21} + \underline{l_{32}} \times l_{22} \\ \vdots & \vdots \\ a_{n2} &= l_{n1} \times l_{21} + \underline{l_{n2}} \times l_{22} \end{cases}$$

第 3 列目の計算：

$$\begin{cases} (a_{13} &= l_{11} \times l_{31}) \\ (a_{23} &= l_{21} \times l_{31} + l_{22} \times l_{32}) \\ a_{33} &= l_{31} \times l_{31} + l_{32} \times l_{32} + \underline{l_{33}} \times \underline{l_{33}} \\ \vdots & \vdots \\ a_{n3} &= l_{n1} \times l_{31} + l_{n2} \times l_{32} + \underline{l_{n3}} \times l_{33} \end{cases}$$

$$\vdots$$

第 n 列目の計算：

$$\begin{cases} (a_{1n} &= l_{11} \times l_{n1}) \\ (a_{2n} &= l_{21} \times l_{n1} + l_{22} \times l_{n2}) \\ (a_{3n} &= l_{31} \times l_{n1} + l_{32} \times l_{n2} + l_{33} \times l_{n3}) \\ (\vdots & \vdots \qquad\qquad) \\ a_{nn} &= l_{n1} \times l_{n1} + l_{n2} \times l_{n2} + \cdots + \underline{l_{nn}} \times \underline{l_{nn}} \end{cases}$$

であり，$a_{ij} = a_{ji}$ なので () 内は計算不要となり，各式のアンダーラインの項が，各々の計算式で求める値となる．

これを一般性のあるように表現すると，() で囲まれた式も含めて，

$$\sum_{k=1}^{\min(i,j)} l_{ik} l_{jk} \quad = \quad a_{ij}, \qquad (i > j) \tag{2.37}$$

と表わされるが，ここでは A は対称であり，下三角の要素のみ考えれば十分なので，() の式を除いて，

$$\sum_{k=1}^{j} l_{ik} l_{jk} \quad = \quad a_{ij} \tag{2.38}$$

である．したがって，L の要素とは，上記のアンダーラインの項に対応するので，(2.38) を用いて，

$$\begin{aligned} i \neq j \text{ のとき}, \sum_{k=1}^{j} l_{ik} l_{jk} &= a_{ij} \ \text{ から}, l_{ij} l_{jj} + \sum_{k=1}^{j-1} l_{ik} l_{jk} = a_{ij} \text{として}, \\ l_{ij} &= \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right), \qquad j = 1, 2, \cdots, i-1 \ (i > j) \end{aligned} \tag{2.39}$$

$$\begin{aligned} i = j \text{ のとき}, \sum_{k=1}^{i} l_{ik} l_{ik} &= a_{ii} \ \text{ から}, l_{ii}^2 + \sum_{k=1}^{i-1} l_{ik}^2 = a_{ii} \text{として}, \\ l_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \qquad i = 1, 2, \cdots, n \end{aligned} \tag{2.40}$$

によって計算できる[6]．

このように分解した係数行列を用いると，求解では，

$$A\boldsymbol{x} = \boldsymbol{b} \rightarrow LL^T\boldsymbol{x} = \boldsymbol{b} \quad \begin{cases} L\boldsymbol{y} = \boldsymbol{b}, \\ L^T\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

として解けば良い．具体的な手順は，LU 分解を参照.

ところが，この第１の方法では平方根の計算を含んでおり，計算コストがかかることや，平方根計算による精度の低下などのデメリットを伴う[7]．これに対する修正アルゴリズムが開発されたこともあって，この方法はあまり用いられない．次に，その修正されたアルゴリズムについて説明する．

コレスキー法 2

コレスキー法の計算に含まれる平方根の計算を無くしたものは，修正コレスキー法 (modified Cholesky method) とも呼ばれる．その１つは[8]，

$$L = \tilde{L}\mathrm{diag}(l_{ii}), \quad (i = 1, 2, \cdots, n) \tag{2.41}$$

とするものである．ここで，$\mathrm{diag}(l_{ii})$ は，$l_{11}, l_{22}, \cdots, l_{nn}$ を対角要素とする対角行列を表し，$\tilde{L}$ は単位下三角行列（対角要素がすべて 1）である．そうすると，

$$\begin{aligned} A &= LL^T = \tilde{L}\mathrm{diag}(l_{ii})\mathrm{diag}(l_{ii})\tilde{L}^T \\ &= \tilde{L}\mathrm{diag}(l_{ii}^2)\tilde{L}^T = \tilde{L}\tilde{D}\tilde{L}^T \end{aligned} \tag{2.42}$$

となる．$\tilde{D}$ は正の対角行列である．$\tilde{D}$ の要素を $\tilde{d}_i$，$\tilde{L}$ の要素を $\tilde{l}_{ij}$ とし，$\tilde{l}_{ii}^2 = 1$ を上手く利用して，

$$\begin{aligned} \sum_{k=1}^{j} \tilde{l}_{ik}\tilde{d}_k\tilde{l}_{jk} &= a_{ij} \ \text{から} \\ \tilde{l}_{ij} &= \left(a_{ij} - \sum_{k=1}^{j-1} \tilde{l}_{ik}\tilde{d}_k\tilde{l}_{jk}\right)/\tilde{d}_j, \qquad j = 1, 2, \cdots, i-1 \ (i > j) \end{aligned} \tag{2.43}$$

$$\begin{aligned} \sum_{k=1}^{i} \tilde{l}_{ik}\tilde{d}_k\tilde{l}_{ik} &= a_{ii} \ \text{から} \\ \tilde{d}_i &= a_{ii} - \sum_{k=1}^{i-1} \tilde{l}_{ik}^2\tilde{d}_k, \qquad i = 1, 2, \cdots, n \end{aligned} \tag{2.44}$$

と分解できる．このように分解した係数行列を用いると，求解では，

$$A\boldsymbol{x} = \boldsymbol{b} \rightarrow \tilde{L}\tilde{D}\tilde{L}^T\boldsymbol{x} = \boldsymbol{b}$$

[6] A が正定値であれば，その対角要素は $a_{ii} > 0$ である（式 (2.35) のベクトル $\boldsymbol{x}$ に単位ベクトルを代入して確かめてみよう）．したがって，(2.40) では，l_{ii} の計算式の根号の中がつねに正になるので l_{ii} は実数になる．また，正定値性があると，コレスキー分解を行うときの枢軸要素は正であることが理論的に保証されているので，この観点からのピボッティングは不要である．さらに，正定値行列になるようなシステムにおいては，安定に分解できる（情報落ちや桁落ちなどが生じにくい）場合が多いため，一般にコレスキー分解に対しては，ピボッティングを考えないことが多い.

[7] 最近のコンピュータでは改善されていることだが，当時としては，大問題だったと思われる.

[8] 通常，“修正コレスキー法” というと，この方法を指す.

$$\begin{cases} \tilde{L}\boldsymbol{y} = \boldsymbol{b} \\ \tilde{D}\tilde{L}^T\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

として解けば良い．具体的な計算手順は，次のとおりである．

【前進代入】

$$y_i = b_i - \sum_{k=1}^{i-1} \tilde{l}_{ik} y_k, \qquad i = 1, 2, \cdots, n, \tag{2.45}$$

【後退代入】

$$x_i = \tilde{d}_i^{-1} y_i - \sum_{k=i+1}^{n} \tilde{l}_{ki} x_k, \quad i = n, n-1, \cdots, 1 \tag{2.46}$$

アルゴリズム: 2.2 (修正コレスキー法)

$A\boldsymbol{x} = \boldsymbol{b}$ を修正コレスキー法（コレスキー法 2）を用いて解く．

/* 分解過程開始 */
for $i = 1, 2, \cdots, n$
　　for $j = 1, 2, \cdots, i-1$
　　$w_j = a_{ij} - \sum_{k=1}^{j-1} w_k l_{jk}$
　　$l_{ij} = w_j d_j^{-1}$
　　end
　　$d_i = a_{ii} - \sum_{k=1}^{i-1} w_k l_{ik}$
end
/* 分解過程終了 */
/* 前進代入 */
for $i = 1, 2, \cdots, n$
　　$y_i = b_i - \sum_{k=1}^{i-1} \tilde{l}_{ik} y_k$
end
/* 後退代入 */
for $i = n, n-1, \cdots, 1$
　　$x_i = \tilde{d}_i^{-1} y_i - \sum_{k=i+1}^{n} \tilde{l}_{ki} x_k$
end

コレスキー法 3

もう 1 つは,

$$L = \bar{L}\mathrm{diag}(l_{ii}^{-1}), \quad (i = 1, 2, \cdots, n) \tag{2.47}$$

とするものである．この第 3 の方法は，厳密な求解（特に分解）においてはあまり効果はないが，後に出てくる共役勾配法 (CG 法) の前処理という技法として応用されると，これらコレスキー分解の中では，非常に効果のあるものである．ここでは，第 3 の方法におけるアイデアの紹介という程度で確認して欲しい．

式 (2.47) のとき，

$$\begin{aligned} A &= LL^T = \bar{L}\mathrm{diag}(l_{ii}^{-1})\mathrm{diag}(l_{ii}^{-1})\bar{L}^T \\ &= \bar{L}\mathrm{diag}(l_{ii}^{-2})\bar{L}^T = \bar{L}\bar{D}\bar{L}^T \end{aligned} \tag{2.48}$$

となる．$\bar{D}$ の要素を $\bar{d}_i$，$\bar{L}$ の要素を $\bar{l}_{ij}$ とし，

$$\bar{d}_i = \frac{1}{\bar{l}_{ii}} \tag{2.49}$$

という条件を課すと，

$$\sum_{k=1}^{j} \bar{l}_{ik}\bar{d}_k\bar{l}_{jk} = a_{ij} \quad \text{から}$$

$$\bar{l}_{ij} = a_{ij} - \sum_{k=1}^{j-1} \bar{l}_{ik}\bar{d}_k\bar{l}_{jk}, \qquad j = 1, 2, \cdots, i-1 \ (i > j) \tag{2.50}$$

$$\sum_{k=1}^{i} \bar{l}_{ik}\bar{d}_k\bar{l}_{ik} = a_{ii} \quad \text{から}$$

$$\bar{l}_{ii} = a_{ii} - \sum_{k=1}^{i-1} \bar{l}_{ik}^2\bar{d}_k, \quad \bar{d}_i = \frac{1}{\bar{l}_{ii}}, \quad i = 1, 2, \cdots, n \tag{2.51}$$

によって分解できる．

このように分解した係数行列を用いると，求解では，

$$A\boldsymbol{x} = \boldsymbol{b} \rightarrow \bar{L}\bar{D}\bar{L}^T\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{cases} \bar{L}\boldsymbol{y} = \boldsymbol{b} \\ \bar{D}\bar{L}^T\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

として解けば良い．具体的な計算手順は，次のとおりである (コレスキー法 2 とは計算式が異なることに注意)．

【前進代入】

$$y_i = \left(b_i - \sum_{k=1}^{i-1} \bar{l}_{ik}y_k \right) d_i, \qquad i = 1, 2, \cdots, n, \tag{2.52}$$

【後退代入】

$$x_i = y_i - \left(\sum_{k=i+1}^{n} \bar{l}_{ki}x_k \right) d_i, \quad i = n, n-1, \cdots, 1 \tag{2.53}$$

この第 3 の方法を，後述する共役勾配法 (CG 法) の前処理として応用するときには，この分解方法特有の面白い結果を確認できる．

コレスキー分解の具体的な計算例

次の対称正定値行列 A をコレスキー分解 1,2,3 を用いて分解する．

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \tag{2.54}$$

● コレスキー分解 1

$$\begin{aligned} A &= LL^T \\ &= \begin{bmatrix} \sqrt{2} & & 0 \\ \frac{-1}{\sqrt{2}} & \sqrt{\frac{3}{2}} & \\ 0 & -\sqrt{\frac{2}{3}} & \sqrt{\frac{4}{3}} \end{bmatrix} \begin{bmatrix} \sqrt{2} & \frac{-1}{\sqrt{2}} & 0 \\ & \sqrt{\frac{3}{2}} & -\sqrt{\frac{2}{3}} \\ 0 & & \sqrt{\frac{4}{3}} \end{bmatrix} \end{aligned} \tag{2.55}$$

※コレスキー分解 1 では，平方根が出てくることが特徴であり，“平方根法” とも呼ばれることもある．

分解計算について

$$\begin{aligned} i=1 \quad & l_{11} = \sqrt{a_{11}} = \sqrt{2} \\ i=2 \quad & l_{21} = \frac{1}{l_{11}} a_{21} = \frac{-1}{\sqrt{2}} \\ & l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{2 - \frac{1}{2}} = \sqrt{\frac{3}{2}} \\ i=3 \quad & l_{31} = \frac{1}{l_{11}} a_{31} = 0 \\ & l_{32} = \frac{1}{l_{22}} \left(a_{32} - l_{31} l_{21} \right) = \sqrt{\frac{2}{3}} \left(-1 - 0 \right) = -\sqrt{\frac{2}{3}} \\ & l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{2 - \left(0 + \frac{2}{3} \right)} = \sqrt{\frac{4}{3}} \end{aligned}$$

● コレスキー分解 2

$$\begin{aligned} A &= \tilde{L} \tilde{D} \tilde{L}^T \\ &= \begin{bmatrix} 1 & & 0 \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & & 0 \\ & \frac{3}{2} & \\ 0 & & \frac{4}{3} \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ & 1 & -\frac{2}{3} \\ 0 & & 1 \end{bmatrix} \end{aligned} \tag{2.56}$$

※コレスキー分解 2 では，行列 $\tilde{L}$ の対角要素を 1 とすることが特徴である．

分解計算について

$$\begin{aligned} i=1 \quad & \tilde{d}_{11} = a_{11} = 2 \\ i=2 \quad & \tilde{l}_{21} = \frac{1}{\tilde{d}_1} a_{21} = \frac{-1}{2} \\ & \tilde{d}_2 = a_{22} - \tilde{l}_{21}^2 \tilde{d}_1 = 2 - \frac{1}{4} \times 2 = \frac{3}{2} \end{aligned}$$

$$
\begin{aligned}
i=3 \quad \tilde{l}_{31} &= \frac{1}{\tilde{d}_1} a_{31} = 0 \\
\tilde{l}_{32} &= \frac{1}{\tilde{d}_2}\left(a_{32} - \tilde{l}_{31}\tilde{d}_1\tilde{l}_{21}\right) = \frac{2}{3}(-1-0) = -\frac{2}{3} \\
\tilde{d}_3 &= a_{33} - \left(\tilde{l}_{31}^2\tilde{d}_1 + \tilde{l}_{32}^2\tilde{d}_2\right) = 2 - \left(0 + \frac{4}{9}\times\frac{3}{2}\right) = \frac{4}{3}
\end{aligned}
$$

● コレスキー分解 3

$$
\begin{aligned}
A &= \bar{L}\bar{D}\bar{L}^T \\
&= \begin{bmatrix} 2 & & 0 \\ -1 & \frac{3}{2} & \\ 0 & -1 & \frac{4}{3} \end{bmatrix}
\begin{bmatrix} \frac{1}{2} & & 0 \\ & \frac{2}{3} & \\ 0 & & \frac{3}{4} \end{bmatrix}
\begin{bmatrix} 2 & -1 & 0 \\ & \frac{3}{2} & -1 \\ 0 & & \frac{4}{3} \end{bmatrix}
\end{aligned} \tag{2.57}
$$

※コレスキー分解 3 では，行列 $\bar{L}$ の対角要素に対応する $\bar{D}$ の要素を掛けた値を 1 とする ($\bar{l}_{ii}\bar{d}_i = 1$) ことが特徴である．

分解計算について

$$
\begin{aligned}
i=1 \quad \bar{l}_{11} &= a_{11} = 2 \\
\bar{d}_1 &= \frac{1}{\bar{l}_{11}} = \frac{1}{2} \\
i=2 \quad \bar{l}_{21} &= a_{21} = -1 \\
\bar{l}_{22} &= a_{22} - \bar{l}_{21}^2\bar{d}_1 = 2 - (-1)^2 \times \frac{1}{2} = \frac{3}{2} \\
\bar{d}_2 &= \frac{1}{\bar{l}_{22}} = \frac{2}{3} \\
i=3 \quad \bar{l}_{31} &= a_{31} = 0 \\
\bar{l}_{32} &= a_{32} - \bar{l}_{31}\bar{d}_1\bar{l}_{21} = -1 \\
\bar{l}_{33} &= a_{33} - \left(\bar{l}_{31}^2\bar{d}_1 + \bar{l}_{32}^2\bar{d}_2\right) = 2 - \left\{0 + (-1)^2 \times \frac{2}{3}\right\} = \frac{4}{3} \\
\bar{d}_3 &= \frac{1}{\bar{l}_{33}} = \frac{3}{4}
\end{aligned}
$$

2.4 反復改良法

連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を，LU 分解 (ガウス消去法) を用いて求めたときの解を，以下の手順を用いることにより，真の解に近付けることができる．これを，**反復改良法 (iterative improvement method)** という．

1) $A\boldsymbol{x} = \boldsymbol{b}$ をガウス消去法で解き，近似解を $\tilde{\boldsymbol{x}}$ とする．

2) 残差 $\boldsymbol{r} = \boldsymbol{b} - A\tilde{\boldsymbol{x}}$ を高精度で計算する．

3) $A\boldsymbol{e} = \boldsymbol{r}$ を解いて近似解を $\tilde{\boldsymbol{e}}$ とする．ここで，$\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{e}}$ が改良解となる．

4) $\tilde{\boldsymbol{e}}$ が許容量より小さければ終了，そうでなければ $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + \tilde{\boldsymbol{e}}$ として 2) の手順へ戻って繰返す．

第3章　数値解(コンピュータで計算した解)の評価

連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{3.1}$$

などを計算機で解いて得られた**数値解** $\hat{\boldsymbol{x}}$ と**厳密解** $\boldsymbol{x}$ との誤差を評価する際，ベクトルの各要素ごとに評価するのは大変である．このような場合に，どれだけ離れているかを**距離** $(\hat{\boldsymbol{x}} - \boldsymbol{x})$ として測り，１つの値で評価できるような指標として**ベクトルノルム** (vector norm) という物差しを用いる．また，計算によって求まった解の正確さを表すのに，**誤差**や**残差**を測る方法がある．行列に対しても**行列ノルム** (matrix norm) を定義することで，方程式の解き易さなどを評価することが可能であり，その際の指標として**条件数**という数値を考える．

ここでは，n 次実ベクトル $\boldsymbol{x}$ と n 次実正方行列 A, B を例に取り説明する．

3.1　距離を表す数値 (ノルム)

1) ベクトルノルム

ベクトルノルムとはベクトル $\boldsymbol{x}$ の「大きさ」を示す量であり，$\|\boldsymbol{x}\|$ と表され，次の性質を満足する．

(1) $\|\boldsymbol{x}\| \geq 0$．$\boldsymbol{x} = \boldsymbol{0}$ のときに限り $\|\boldsymbol{x}\| = 0$.

(2) $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$，($\alpha$: スカラー).

(3) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$.

具体的には，よく用いられるものとして次のようなノルムがある．

1–ノルム

$$\|\boldsymbol{x}\|_1 = \sum_{k=1}^{n} |x_k|$$

2–ノルム

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2} \quad \left(= \sqrt{(\boldsymbol{x}, \boldsymbol{x})}\right)$$

∞–ノルム

$$\|\boldsymbol{x}\|_\infty = \max_{1 \leq k \leq n} |x_k|$$

※ベクトルの2–ノルムは，ユークリッド・ノルムとも呼ばれ，∞–ノルムは，最大ノルムとも呼ばれる．

2) 行列ノルム

(1) $\|A\| \geq 0$.　　$A = O$ (O:零行列) のときに限り $\|A\| = 0$.

(2) $\|\alpha A\| = |\alpha|\|A\|$,　($\alpha$: スカラー).

(3) $\|A + B\| \leq \|A\| + \|B\|$.

(4) $\|AB\| \leq \|A\|\|B\|$.

具体的には，よく用いられるものとして次のようなノルムがある．

1–ノルム

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \quad \text{（最大列和 : 列ごとに絶対値の和を取った中での最大値）}$$

2–ノルム

$$\begin{aligned} \|A\|_2 &= \sqrt{A^T A \text{ の最大固有値}}, \\ \|A\|_2 &= A \text{ の絶対値最大の固有値, （} A \text{ が対称のとき）} \end{aligned}$$

∞–ノルム

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \quad \text{（最大行和 : 行ごとに絶対値の和を取った中での最大値）}$$

※行列の 2–ノルムは，スペクトル・ノルムとも呼ばれる．

さらに，ベクトルのノルムとの間には，次の関係が成り立つ．

(5) $\|A\boldsymbol{x}\| \leq \|A\|\|\boldsymbol{x}\|$.

3) ノルムの同値性 (どのノルムを用いても良い)

ベクトルや行列が，あるノルムで小さいならば，他のノルムでも小さいことが言える．これをノルムの同値性（あるいは，ノルムの等価性）という[1] ．ノルムを求める際には，計算の都合の良いものを用いれば良い．

2 種類のベクトルノルム $\|\boldsymbol{x}\|_a, \|\boldsymbol{x}\|_b$ に対して，すべての $\boldsymbol{x}$ について

$$m\|\boldsymbol{x}\|_a \leq \|\boldsymbol{x}\|_b \leq M\|\boldsymbol{x}\|_a \tag{3.2}$$

が成り立つような正定数 m, M が存在する．例えば，

$$\begin{aligned} \|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\|\boldsymbol{x}\|_\infty, \\ \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq n\|\boldsymbol{x}\|_2 \end{aligned}$$

[1] 各ノルムの値が同じという意味では無い点に注意.

という関係が成り立つ.

2つの行列ノルム $\|A\|_a, \|A\|_b$ に対して，すべての A について

$$m\|A\|_a \leq \|A\|_b \leq M\|A\|_a \tag{3.3}$$

が成り立つような正定数 m, M が存在する．例えば,

$$\|A\|_\infty \leq \|A\|_2 \leq \sqrt{n}\|A\|_\infty,$$
$$\|A\|_2 \leq \|A\|_1 \leq \sqrt{n}\|A\|_2$$

という関係が成り立つ.

3.2 誤差と残差

さて，(3.1) 式，すなわち，元の連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の厳密解を $\boldsymbol{x}$，コンピュータで解いて得られた数値解を $\hat{\boldsymbol{x}}$ で表わすと，数を有限桁の浮動小数で表現し，有限桁で計算を実行したことにより一般には $\hat{\boldsymbol{x}} \neq \boldsymbol{x}$ となる．このとき

$$\boldsymbol{e} \quad = \quad \hat{\boldsymbol{x}} - \boldsymbol{x} \tag{3.4}$$

を**誤差** (error) あるいは誤差ベクトルという．誤差の程度は $\boldsymbol{x}$ の各要素の値の大きさによっても変化するため，$\boldsymbol{x}$ のノルムで割って，**相対誤差** (relative error) あるいは相対誤差ベクトル

$$\frac{\hat{\boldsymbol{x}} - \boldsymbol{x}}{\|\boldsymbol{x}\|} \tag{3.5}$$

で評価することもある．誤差の大きさだけが興味の対象なら，スカラー量である**相対誤差ノルム**

$$\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \tag{3.6}$$

で評価すればよい.

実際に厳密解 $\boldsymbol{x}$ が分かっていることは稀なので（分かっているなら計算しない！），別の基準を用いて計算解の正しさを評価することになる．よく使われるのは

$$\boldsymbol{r} \quad = \quad \boldsymbol{b} - A\hat{\boldsymbol{x}} \tag{3.7}$$

で**残差** (residual) または残差ベクトルという.

残差に対しても用途に応じて，**相対残差**や**相対残差ノルム**を考える．これは，残差の計算における各項の中で絶対値最大の値と残差との比を指すが，これら計算中の値を厳密に評価するのは大変である．よく用いられるのは,

$$\frac{\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\|}{\|\boldsymbol{b}\|} \tag{3.8}$$

である.

一般には，ベクトルに対するノルムは，2–ノルムがよく用いられる.

3.3 条件数 (方程式の解き易さや誤差の影響の受け易さを示す数値)

連立一次方程式 (3.1) をコンピュータで計算したとき，A の要素に誤差は混入せず，$\boldsymbol{b}$ に関する演算において何らかの誤差 $\Delta\boldsymbol{b}$ が入って，その結果，計算解が $\boldsymbol{x}+\Delta\boldsymbol{x}$ になったとしよう（微積分と同じく “わずかな量” を表すときの記号として Δ(デルタ) を用いている）．式で書くと

$$A(\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{b}+\Delta\boldsymbol{b} \tag{3.9}$$

である．式 (3.1),(3.9) から

$$\Delta\boldsymbol{x} = A^{-1}(\Delta\boldsymbol{b})$$

となり, 両辺のノルムをとれば

$$\|\Delta\boldsymbol{x}\| \le \|A^{-1}\|\cdot\|\Delta\boldsymbol{b}\| \tag{3.10}$$

である（行列ノルムの条件 (5) を用いて求めた）．式 (3.1) の両辺のノルムをとれば

$$\|A\|\cdot\|\boldsymbol{x}\| \ge \|\boldsymbol{b}\| \tag{3.11}$$

なので，(3.10),(3.11) を合わせると

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \|A\|\cdot\|A^{-1}\|\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{3.12}$$

となる．$\|A\|\cdot\|A^{-1}\|$ のことを行列 A の**条件数** (condition number) といい，このテキストでは $\mathrm{cond}(A)$ と表わす．特に，どのノルムを用いたかを明示するときは，$\mathrm{cond}_p(A)$ などと記述する（この場合には，p-ノルムを用いたという）．$AA^{-1}=I$ より,

$$\mathrm{cond}(A) = \|A\|\cdot\|A^{-1}\| \ge 1 \tag{3.13}$$

である.

(3.12) は (3.9) の右辺に入った誤差の割合 $\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$ が，左辺 $\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$ でどれくらいの倍率になりうるかを示している．(3.12) はコンピュータ上の数値表現 (浮動小数点の内部表現など) や演算方式に関係していないことに注意しよう．これは純粋に数学的な関係であり，行列 A と右辺 $\boldsymbol{b},\Delta\boldsymbol{b}$ によって決まる．

実際, コンピュータで連立一次方程式を解けば

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \alpha\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \le \mathrm{cond}(A)\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{3.14}$$

満足する α が求められる．α は $\mathrm{cond}(A)$ に近くなることもある．条件数 $\mathrm{cond}(A)$ が大きいときには，**悪条件の問題**や**悪条件方程式**，あるいは，**性質の悪い問題**などという．反対に条件数が小さいときには，**良条件の問題**や**性質の良い問題**などという．悪条件方程式は誤差が入りやすいので，コンピュータで扱うときにはよりいっそうの注意がいる．

係数行列 A が ΔA だけ変わったとき，解が $\Delta\boldsymbol{x}$ だけ変わるとすると,

$$(A+\Delta A)(\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{b} \tag{3.15}$$

となり，これより

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}+\Delta\boldsymbol{x}\|} \le \|A\|\cdot\|A^{-1}\|\frac{\|\Delta A\|}{\|A\|} \tag{3.16}$$

が得られる．ここでも，条件数

$$\mathrm{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

との関係が確認できる．

例題

連立一次方程式について，

悪条件の問題の例：

$$\begin{cases} 7.6x_1 + 9.3x_2 &= 7.6 \\ 3.1x_1 + 3.8x_2 &= 3.1 \end{cases} \tag{3.17}$$

および，**良条件の問題の例：**

$$\begin{cases} 7.6x_1 - 9.3x_2 &= 7.6 \\ 3.1x_1 + 3.8x_2 &= 3.1 \end{cases} \tag{3.18}$$

を，10 進 3 桁切捨てで計算して数値解を求める．これらの厳密解は，ともに，

$$\boldsymbol{x} \;=\; [x_1,\ x_2]^T = [1.0,\ 0.0]^T \tag{3.19}$$

である．

また，各々の方程式の右辺をわずかに，

$$\boldsymbol{\Delta b} \;=\; [0.0,\ 0.01]^T \tag{3.20}$$

だけ変化させたときの数値解について考える．

● まず，連立一次方程式 (3.17) についてガウス消去法で計算すると，第 1 段の消去では，

$$\begin{aligned} m_{21} &= \frac{3.1}{7.6} = 0.407 \\ a_{22}^{(2)} &= 3.8 - (0.407 \times 9.3) = 3.8 - 3.78 = 0.02 \\ b_2^{(2)} &= 3.1 - (0.407 \times 7.6) = 3.1 - 3.09 = 0.01 \end{aligned}$$

となり，前進消去した結果を行列とベクトルを用いて表すと，

$$\begin{bmatrix} 7.6 & 9.3 \\ 0.0 & 0.02 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7.6 \\ 0.01 \end{bmatrix} \tag{3.21}$$

となる．後退代入を行なうと，数値解 ($\hat{\boldsymbol{x}} = [\hat{x}_1,\ \hat{x}_2]^T$)

$$\hat{x}_2 = \frac{0.01}{0.02} = 0.5 \tag{3.22}$$

$$\begin{aligned} \hat{x}_1 &= \frac{1}{7.6}(7.6 - 9.3 \times \hat{x}_2) = \frac{1}{7.6}(7.6 - 4.65) \\ &= \frac{2.95}{7.6} = 0.388 \end{aligned} \tag{3.23}$$

が得られる．厳密解 (3.19) と比較すると，数値解は正確に求められていないことが分かる．

このような悪条件の問題の場合，右辺ベクトルが僅かに変化しただけで，数値解が全く異なる値になってしまう．

● 次に，右辺ベクトルに (3.20) を加えて，わずかに変化させると，

$$\begin{bmatrix} 7.6 \\ 3.1\underline{1} \end{bmatrix} \tag{3.24}$$

となる．前進消去した結果は，

$$\begin{bmatrix} 7.6 & 9.3 \\ 0.0 & 0.02 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7.6 \\ 0.02 \end{bmatrix} \tag{3.25}$$

となる．後退代入すると，数値解は，

$$\hat{x}_2 = 1.00, \quad \hat{x}_1 = -0.223 \tag{3.26}$$

となって，先程の数値解 (3.22)(3.23) とは大幅に異なる．これは，行列の条件が悪いために計算の途中で混入した誤差や右辺のわずかな変化のために解が大きく変化してしまったためである．

● 今度は，良条件の連立一次方程式 (3.18) について，ガウス消去法で計算すると，第 1 段の消去における乗数 m_{21} は，

$$m_{21} \;=\; \frac{a_{21}}{a_{11}} \;=\; \frac{3.1}{7.6} \;=\; 0.407$$

となり，前進消去した結果は，

$$\begin{bmatrix} 7.6 & -9.3 \\ 0.0 & 7.58 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7.6 \\ 0.01 \end{bmatrix} \tag{3.27}$$

となる．後退代入を行なうと，数値解 ($\hat{\boldsymbol{x}} = [\hat{x}_1,\ \hat{x}_2]^T$)

$$\hat{x}_2 \;=\; \frac{0.01}{7.58} \;=\; 0.00131 \tag{3.28}$$

$$\hat{x}_1 \;=\; \frac{1}{7.6}\{7.6 - (-9.3) \times x_2\} \;=\; 1.00 \tag{3.29}$$

が得られる．厳密解 (3.19) と比較しても，数値解はほぼ正確な値が算出されていることが分かる．

● 次に，右辺ベクトルに (3.20) を加えた問題について考える．

前進消去した結果は，

$$\begin{bmatrix} 7.6 & -9.3 \\ 0.0 & 7.58 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7.6 \\ 0.02 \end{bmatrix} \tag{3.30}$$

となる．後退代入すると，数値解は，

$$\hat{x}_2 = 0.00263, \quad \hat{x}_1 = 1.00 \tag{3.31}$$

となって，先程の数値解 (3.28)(3.29) とほとんど同じ値が得られている．

第4章　どのようにして連立一次方程式が作られるのか

本章では，どのようにして連立一次方程式が作られるのかについて，数値シミュレーションなどでは典型的な物理問題を例に取り上げて説明する．ここでは，物理問題そのものよりも連立一次方程式を作り出す過程に注目して欲しい．

4.1　１次元の問題について

例題１（１次元の問題）：　いま，図 4.1(左) のように，長さ l の糸に重さ m の n 個の質点（大きさ，形を考えないおもりのこと）を等間隔につけて，糸の両端を水平に固定したときの重力の影響による位置 x における糸のたわみ量 $u(x)$ を考える．各質点は，両隣からの張力（一様に T とする）の影響も受けている．

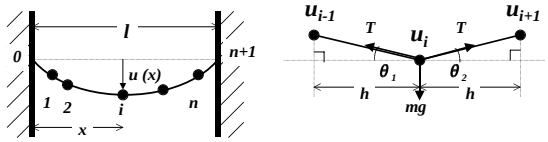

図 4.1: 糸のたわみの例題 (左) と力のつりあい (右).

この例題は，第１章でも例３として示したが，ここで扱っている現象は，

$$-\frac{d^2u}{dx^2} = f \tag{4.1}$$

という２階の常微分方程式として表される．

4.1.1　方程式の離散化

図 4.1(左) のように糸が張られている領域を $n+1$ 等分すると，この時のきざみ幅 h は，$h = 1/(n+1)$ である．原点 0 からの長さは，$x = ih$ と表される．また，点 i における u の値 $u(ih)$ を u_i と表す．f も同様に f_i と表す．

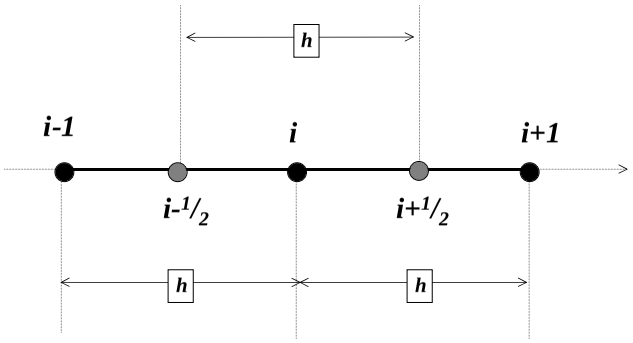

図 4.2: 点 i における差分.

図 4.2 において，仮想の点 $i-\frac{1}{2}$ と $i+\frac{1}{2}$ を考えて，点 i における**中心差分**を行うと，左辺は,

$$\left(\frac{d^2u}{dx^2}\right)_i \approx \frac{1}{h}\left\{\left(\frac{du}{dx}\right)_{i+\frac{1}{2}} - \left(\frac{du}{dx}\right)_{i-\frac{1}{2}}\right\} \tag{4.2}$$

と表せる（要するに $i-\frac{1}{2}$ と $i+\frac{1}{2}$ の値の差分）．式 (4.1) から，点 i において,

$$-\left[\frac{1}{h}\left\{\left(\frac{du}{dx}\right)_{i+\frac{1}{2}} - \left(\frac{du}{dx}\right)_{i-\frac{1}{2}}\right\}\right] = f_i \tag{4.3}$$

である．ここで,

$$\left(\frac{du}{dx}\right)_{i-\frac{1}{2}} \approx \frac{u_i - u_{i-1}}{h}, \tag{4.4}$$

$$\left(\frac{du}{dx}\right)_{i+\frac{1}{2}} \approx \frac{u_{i+1} - u_i}{h} \tag{4.5}$$

であり，これらも，点 $i-\frac{1}{2}$ と $i+\frac{1}{2}$ での各々の中心差分を行ったものである．

結局，式 (4.3)〜(4.5) から，離散化後の点 i における式 (4.1) は,

$$\begin{aligned} -\frac{1}{h^2}\left\{(u_{i+1}-u_i)-(u_i-u_{i-1})\right\} &= f_i, \\ -u_{i-1}+2u_i-u_{i+1} &= h^2 f_i \end{aligned} \tag{4.6}$$

という漸化式となる．これは，第 1 章の式 (1.7) において $f_i = -\frac{mg}{hT}$ ということである．

$i = 0, n+1$ の点における条件（ここでは，糸の両端にあたる境界部分の条件を指す．一般に**境界条件**と呼び，g_0, g_{n+1} などと記述する）は,

$$u_0 = g_0 = 0, \quad u_{n+1} = g_{n+1} = 0 \tag{4.7}$$

である（つまり，これら 2 箇所には，たわみが無い）．

4.1.2　係数行列と右辺項が持つ情報

式 (4.6) をすべての質点 $(i = 1, 2, 3, \cdots, n)$ について書き並べると，連立一次方程式が得られる．各質点のたわみを表現したベクトル $\boldsymbol{u}$ を

$$\boldsymbol{u} \quad = \quad (u_1, u_2, u_3, \cdots, u_{n-1}, u_n)^T$$

とすると，連立一次方程式 $A\boldsymbol{u} = \boldsymbol{b}$ は，

$$\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & \mathbf{0} & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} h^2 f_1 + g_0 \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{n-1} \\ h^2 f_n + g_{n+1} \end{bmatrix} \tag{4.8}$$

と表すことができる．

4.2　２次元の問題について

例題２（２次元の問題）：　各辺が長さ 1 である正方形の領域を考える．この領域 (領域 Ω と名付ける) の内部で，関数 $u(x, y)$ についての 2 次元の偏微分方程式

$$\frac{\partial}{\partial x}\left(-k\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k\frac{\partial u}{\partial y}\right) = f \tag{4.9}$$

を満たし，境界上 $(x = 0, x = 1, y = 0, y = 1)$ で

$$u(x, y) \quad = \quad g(x, y) \quad g \text{ は境界上に与えた値や関数などで表される条件} \tag{4.10}$$

となる関数 $u(x, y)$ を求める[1] ．

このような方程式で表現される代表的なものに，熱が伝わった後の温度分布，圧力，電荷分布など，定常状態（時間変化しない状態）でのポテンシャルエネルギーと呼ばれるものがある．係数 k は，伝導率を意味し，例えば，Ω 内に温度変化し易い物体や，しにくい物体などがあるときには，この係数の値が異なる．物理現象としては，熱の伝わり方が違ってくる．

また，先ほどの**例題** 1 の２次元版として考えると，この問題は,「各辺のサイズが $1m$ である正方形テーブルクロス（材質は少々重さのある布が理想的!）の各辺を持ち高さ（つまり境界条件）を変えたときの布の高さを計算している」ということである．

[1] 詳しく述べると，2 次元正方領域 $\Omega = (0, 1) \times (0, 1)$ におけるポアソン方程式の全周ディリクレ問題

$$\begin{aligned} -\Delta u \quad &= \quad f \quad \text{in } \Omega, \\ u \quad &= \quad g \quad \text{on } \partial\Omega \end{aligned}$$

と表現される．ここで，$\Delta = \frac{\partial}{\partial x}\left(-k\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k\frac{\partial u}{\partial y}\right)$ を表す．x, y についての 2 変数関数 $u(x, y), f(x, y), g(x, y), k(x, y)$ を，単に u, f, g, k と略記している．また，g は境界条件と呼ばれる．

4.2.1 方程式の離散化

領域 Ω を x, y 方向にそれぞれ $(m+1)$ 等分した格子を作る．このとき，きざみ幅 (格子の間隔)h は $h = 1/(m+1)$ である．この格子において，$x = ih, y = jh$ の格子点 (i, j) における u の値 $u(ih, jh)$ を u_{ij} と表す．境界上の値は式 (4.10) の境界条件で与えられているので，Ω の内部の格子点上の値を求めればよい．図 4.3 に $m = 3$ のときの例を示す.

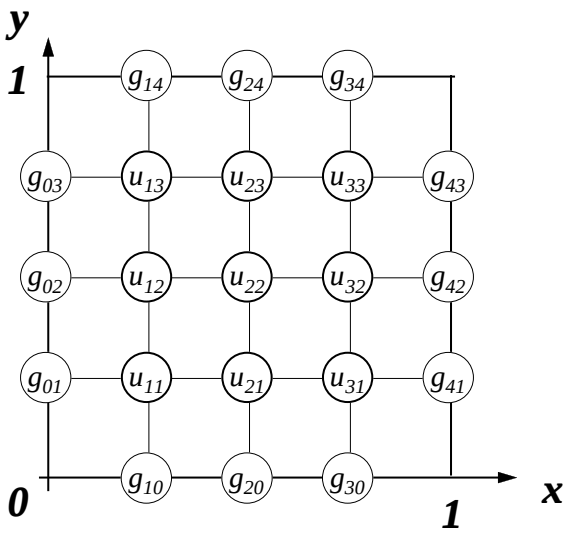

図 4.3: $m = 3$ のときの例.

微分を差分で近似する方法を差分法 (FDM: finite difference method) という．方程式 (4.9) を 5 点中心差分 (x, y 方向に対して差分するので，x 方向の ± 1，y 方向の ± 1 と基準点を含めて 5 点である) により近似すると，

$$\left[\frac{\partial}{\partial x}\left(-k\frac{\partial u}{\partial x}\right)\right]_{ij} \approx \frac{1}{h}\left\{\left(-k\frac{\partial u}{\partial x}\right)_{i+1/2,j} - \left(-k\frac{\partial u}{\partial x}\right)_{i-1/2,j}\right\} \tag{4.11}$$

となる (記号「≈」は,「近似」を表す)．ここで,

$$\left(-k\frac{\partial u}{\partial x}\right)_{i+1/2,j} \approx -k_{i+1/2,j}\frac{u_{i+1,j} - u_{ij}}{h}, \tag{4.12}$$

$$\left(-k\frac{\partial u}{\partial x}\right)_{i-1/2,j} \approx -k_{i-1/2,j}\frac{u_{ij} - u_{i-1,j}}{h} \tag{4.13}$$

である[2]．これらを式 (4.11) に代入すると,

$$\left[\frac{\partial}{\partial x}\left(-k\frac{\partial u}{\partial x}\right)\right]_{ij} \approx \frac{1}{h^2}\left\{-k_{i+1/2,j}u_{i+1,j} + \left(k_{i+1/2,j} + k_{i-1/2,j}\right)u_{ij} - k_{i-1/2,j}u_{i-1,j}\right\} \tag{4.14}$$

を得る．y 方向についても同様にして,

$$\left[\frac{\partial}{\partial y}\left(-k\frac{\partial u}{\partial y}\right)\right]_{ij} \approx \frac{1}{h^2}\left\{-k_{i,j+1/2}u_{i,j+1} + \left(k_{i,j+1/2} + k_{i,j-1/2}\right)u_{ij} - k_{i,j-1/2}u_{i,j-1}\right\} \tag{4.15}$$

[2] 図 4.3 中には，$i + 1/2$，$i - 1/2$ という点は存在しないが，式 (4.11) のとおり，x 方向に対する平均を表す上での便宜的な表記である．j 方向 (y 方向) に対しても同様である.

となる．よって，格子点 (i,j) における方程式は (u の添え字にしたがってまとめると),

$$\begin{aligned} -\ k_{i,j-1/2}u_{i,j-1}\ -\ k_{i-1/2,j}u_{i-1,j} & \\ +\ (k_{i+1/2,j}\ +\ k_{i-1/2,j}\ \ +\ \ k_{i,j+1/2}\ +\ k_{i,j-1/2})\ u_{ij} & \\ -\ \ k_{i+1/2,j}u_{i+1,j}\ -\ k_{i,j+1/2}u_{i,j+1}\ =\ h^2 f_{ij} & \end{aligned} \tag{4.16}$$

となる．ここで,

$$k_{ij} = k(ih, jh), \tag{4.17}$$

$$f_{ij} = f(ih, jh) \tag{4.18}$$

である．話を簡単にするために，以下では $k = 1$(一定) として説明する．格子点 (i,j) における方程式は

$$-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1} = h^2 f_{i,j} \tag{4.19}$$

という i,j に関する漸化式となる.

4.2.2 連立一次方程式を用いた表現

式 (4.19) をすべての内部格子点について書くと，連立一次方程式が得られる．ベクトル $\boldsymbol{u}$ を

$$\boldsymbol{u}\ \ =\ \ (u_{11}, u_{21}, \cdots, u_{m1}, u_{12}, u_{22}, \cdots, u_{m2}, \cdots, u_{1m}, u_{2m}, \cdots, u_{mm})^T \tag{4.20}$$

とすると，連立一次方程式は

$$A\boldsymbol{u}\ \ =\ \ \boldsymbol{b} \tag{4.21}$$

と表すことができる．$m = 3$ のとき，係数行列 A (9×9 行列) と右辺項ベクトル $\boldsymbol{b}$ (RHS: right hand side vector，9 次ベクトル) は,

$$A\ \ =\ \ \left[\begin{array}{ccc|ccc|ccc} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & \mathbf{0} & \\ & -1 & 4 & & & -1 & & & \\ \hline -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ \hline & & & -1 & & & 4 & -1 & \\ & \mathbf{0} & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{array}\right], \quad \boldsymbol{b} = \left[\begin{array}{l} h^2 f_{11} + g_{10} + g_{01} \\ h^2 f_{21} + g_{20} \\ h^2 f_{31} + g_{30} + g_{41} \\ h^2 f_{12} + g_{02} \\ h^2 f_{22} \\ h^2 f_{32} + g_{42} \\ h^2 f_{13} + g_{03} + g_{14} \\ h^2 f_{23} + g_{24} \\ h^2 f_{33} + g_{43} + g_{34} \end{array}\right] \tag{4.22}$$

である（行列内の縦と横線は構造を見易くするために記載したものである．また，空白の要素は全て 0 である）.

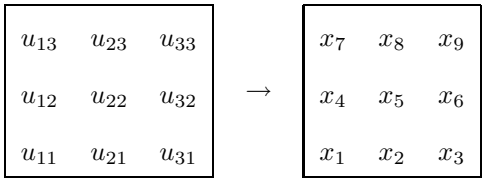

図 4.4: $\boldsymbol{u}$ と $\boldsymbol{x}$ に対する順序づけ.

このように，格子点 $u(ih, jh)$ の値を表すベクトル $\boldsymbol{u}$ は，$m \times m \equiv n$ (n と置くという意味) 項からなる列ベクトルであり，あらためて，

$$\boldsymbol{x} \quad = \quad (x_1, x_2, \cdots, x_n)^T \tag{4.23}$$

として，

$$A\boldsymbol{x} \quad = \quad \boldsymbol{b} \tag{4.24}$$

と表すことにする．以後，特にことわりの無い限り，係数行列 A は正則 (regular, nonsingular) な n 次正方行列，解ベクトル $\boldsymbol{x}$ ，右辺項ベクトル $\boldsymbol{b}$ は n 次ベクトルである．

したがって，図 4.3 における $\boldsymbol{u}$ の点を $\boldsymbol{x}$ で表すと，図 4.4 の対応関係となる．この一般的な順序付けは，ナチュラルオーダリング (natural ordering)，あるいは，辞書式オーダリング (dictionaly ordering) などと呼ばれる．

4.3 係数行列の特徴とデータ格納

係数行列 A は，その要素のほとんどが 0 である．m を大きくすると，0 要素の割合はさらに大きくなる．このように，ほとんどの要素が 0 である行列は疎行列 (sparse matrix) と呼ばれる．反対に，0 要素が少ない行列は密行列 (dense matrix) と呼ばれる．

また，(4.9) の方程式を 5 点中心差分により離散化すると，その行列は対称である．係数 k を任意の値に設定しても，やはり離散化後の行列 A は対称である．

一般に，自然界に現れる方程式を離散化して得られる連立一次方程式の係数行列 $A = (a_{ij})$ は正則で，$a_{ii} > 0, a_{ij} \leq 0 \ \ (j \neq i)$ かつ逆行列 A^{-1} の全ての成分が正となる性質を持っていることが多く[3] ．数値解法にとってメリットがある．一例として，ガウス消去法における軸選択が生じない．また，後に説明する反復法の前処理における不完全分解が可能である．

[3] 専門的な話ではあるが，このような性質を持つ行列は，M 行列 (M-matrix) と呼ばれ，特徴ある行列として考えられている．

第5章　共役勾配法

LU 分解 (ガウス消去法) の計算手順に従えば，計算機による誤差の影響を除くと，解が得られるので頑健 (robust) ではあるが，計算量が n^3 に比例する ($O(n^3)$ と表す) ので n が 10 倍になれば計算時間は 1000 倍になる．これは 1 分で解けたものが 16 時間になるということである．現実の問題では，第 4 章で見たような，連立一次方程式の係数行列に非ゼロ要素がごく一部にしかない巨大な行列を解くことが多い．このような非ゼロ要素が一部にしかないという行列を，一般には疎行列 (sparse matrix) と呼ぶ．このような係数行列を持つ問題を解くには，共役勾配法を用いると良い．

5.1　共役勾配法の性質

連立一次方程式の係数行列が対称正定値な行列 A に対しては

$$A\boldsymbol{x} \quad = \quad \boldsymbol{b} \tag{5.1}$$

の解が

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}, A\boldsymbol{x}) - (\boldsymbol{x}, \boldsymbol{b}) \tag{5.2}$$

という関数を最小にするという事実に基づいて作られたのが**共役勾配法** (Conjugate Gradient method; CG 法) である．CG 法では，任意の初期値 $\boldsymbol{x}^{(0)}$ から始めて，$f(\boldsymbol{x})$ が最小となる $\boldsymbol{x}$ を逐次探索していく．

共役勾配法：
連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ (A : $n \times n$ 対称正定値行列，$\boldsymbol{x}, \boldsymbol{b}$: n 次ベクトル) に対して，次の手順に従い計算すると，**for ～ end** の間は，高々(at most) n 回の繰返しで解 $\boldsymbol{x}$ が求まる[1] .

[1] ここで α_k や $\boldsymbol{p}^{(k)}$ などの表記は，第 k 回目の繰返しステップにおけるスカラー α，ベクトル $\boldsymbol{p}$ ということを表している．ガウス消去法のときには，右下の添字は行列のインデックスを表していたが，それとは指しているものが異なることに注意．例えば，$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}$ とは，第 k 回目のステップでは，ベクトル $\boldsymbol{x}^{(k)}$ は $\alpha_k \boldsymbol{p}^{(k)}$ を加えられて $\boldsymbol{x}^{(k+1)}$ として更新される．プログラミングの際には，同一の配列を更新すれば良い．$\boldsymbol{r}^{(k)}$,$\boldsymbol{p}^{(k)}$ についても同様である．

$\boldsymbol{x}^{(0)}$: 初期ベクトル, $\quad \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$,
$\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$,
for $\quad k = 0, 1, \cdots, n-1$

$$
\begin{aligned}
\alpha_k &= \frac{(\boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})}, \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, \\
\boldsymbol{r}^{(k+1)} &= \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)}, \\
\beta_k &= -\frac{(\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})}, \\
\boldsymbol{p}^{(k+1)} &= \boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)},
\end{aligned}
$$

end

このとき，残差を表すベクトル $\boldsymbol{r}^{(k)}$ と探索方向と呼ばれるベクトル $\boldsymbol{p}^{(k)}$ は,

$$\textbf{直交性}: \quad (\boldsymbol{r}^{(i)}, \boldsymbol{r}^{(j)}) = 0, \quad (i \neq j) \tag{5.3}$$

と

$$\textbf{共役性}: \quad (\boldsymbol{p}^{(i)}, A\boldsymbol{p}^{(j)}) = 0, \quad (i \neq j) \tag{5.4}$$

を満たす.

共役勾配法では，任意の初期値 $\boldsymbol{x}_0$ から始めて，$f(\boldsymbol{x})$ の最小値を繰返し探索する.

5.2 共役勾配法の導出

では，このような計算手順はどのようにして導出されたのであろうか？また，係数行列 A が対称正定値であることとは，どのように結び付いているのであろうか.

5.2.1 すじの良いベクトルを結合するといづれは解になる

一次独立な n 個のベクトル $\boldsymbol{p}^{(0)}, \boldsymbol{p}^{(1)}, \cdots, \boldsymbol{p}^{(n-1)}$ を用いると，n 元の連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の解はそれらの一次結合によって表すことができる．あるいは，任意のベクトル $\boldsymbol{x}^{(0)}$ を用意したとき，これ自体は，$A\boldsymbol{x} = \boldsymbol{b}$ を満足する解ではないものの，これに対して適当な修正を加えて

$$\boldsymbol{x} = \boldsymbol{x}^{(0)} + \alpha_0 \boldsymbol{p}^{(0)} + \alpha_1 \boldsymbol{p}^{(1)} + \cdots + \alpha_{n-1} \boldsymbol{p}^{(n-1)} \tag{5.5}$$

とすれば，これは解になり得る（$\boldsymbol{p}^{(k)}(k = 0, 1, \cdots, n-1)$ が，基本ベクトルの場合を考えてみよう）．$\boldsymbol{p}^{(k)}(k = 0, 1, \cdots, n-1)$ が一次独立な組として,

$$(\boldsymbol{p}^{(i)}, A\boldsymbol{p}^{(j)}) = 0, \quad (i \neq j) \tag{5.6}$$

という条件を満足するものを考えることにする．これは,

「$\boldsymbol{p}^{(i)}$ と $\boldsymbol{p}^{(j)}$ は，行列 A に関して内積を取ったものが 0 である」

ことを表し，このことを，行列 A に関して**共役** (conjugate) である，あるいは，A に関して**直交**すると表現する．または単に，A 共役，A 直交とも言う.

5.2.2 アルゴリズムを作ってみる

共役勾配法では $(k+1)$ 回目の反復における近似解を $\boldsymbol{x}^{(k+1)}$ とし，$\boldsymbol{x}^{(k)}$ の探索方向を表すベクトルを $\boldsymbol{p}^{(k)}$ として

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)} \tag{5.7}$$

とする．また，一般に第 k 回目の反復における近似解 $\boldsymbol{x}^{(k)}$ に対する残差は，

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)} \tag{5.8}$$

である．

ここで，(5.7) を (5.2) に代入すると，

$$\begin{aligned}
f(\boldsymbol{x}^{(k+1)}) &= f(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}) \\
&= \frac{1}{2}(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, A(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)})) - (\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, \boldsymbol{b}) \\
&= \frac{1}{2}\left((\boldsymbol{x}^{(k)}, A\boldsymbol{x}^{(k)}) + \alpha_k(\boldsymbol{x}^{(k)}, A\boldsymbol{p}^{(k)}) + \alpha_k(\boldsymbol{p}^{(k)}, A\boldsymbol{x}^{(k)}) + \alpha_k^2(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})\right) - (\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, \boldsymbol{b}) \\
&= \frac{1}{2}(\boldsymbol{x}^{(k)}, A\boldsymbol{x}^{(k)}) - (\boldsymbol{x}^{(k)}, \boldsymbol{b}) + \alpha_k(\boldsymbol{p}^{(k)}, A\boldsymbol{x}^{(k)}) + \frac{1}{2}\alpha_k^2(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) - \alpha_k(\boldsymbol{p}^{(k)}, \boldsymbol{b}) \\
&= f(\boldsymbol{x}^{(k)}) - \alpha_k(\boldsymbol{p}^{(k)}, \boldsymbol{b} - A\boldsymbol{x}^{(k)}) + \frac{1}{2}\alpha_k^2(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) \\
&= f(\boldsymbol{x}^{(k)}) - \alpha_k(\boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)}) + \frac{1}{2}\alpha_k^2(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})
\end{aligned} \tag{5.9}$$

を得る．(5.9) を α_k の関数として最小化するには

$$\alpha_k = \frac{(\boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})} \tag{5.10}$$

とすればよい．探索方向 $\boldsymbol{p}^{(k+1)}$ は

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)} \tag{5.11}$$

のように決める[2]．ただし，

$$\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)} \tag{5.12}$$

とする．β_k は，$\boldsymbol{p}^{(k+1)}$ と $A\boldsymbol{p}^{(k)}$ とが直交する（A 共役）ように選ぶ．すなわち，

$$\begin{aligned}
(\boldsymbol{p}^{(k+1)}, A\boldsymbol{p}^{(k)}) &= (\boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) \\
&= (\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)}) + \beta_k(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) \;=\; 0
\end{aligned} \tag{5.13}$$

より，

$$\beta_k = -\frac{(\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})} \tag{5.14}$$

となる．

[2] 探索方向 $\boldsymbol{p}^{(k+1)}$ の決め方は，色々と工夫をこらすところである．$\boldsymbol{p}^{(k)} = \boldsymbol{r}^{(k)}$ としたものは最急降下法と呼ばれ，CG 法の前身にあたる解法である．CG 法では $\boldsymbol{p}^{(k)}$ に対して A 共役の条件を課しているので，式 (5.11) となった．

ここで，行列 A とベクトルとの積に着目すると，残差ベクトルの計算式 (5.8) では $A\boldsymbol{x}^{(k)}$ だが，これ以外は全て $A\boldsymbol{p}^{(k)}$ である．(5.8) においても，$A\boldsymbol{p}^{(k)}$ の演算となれば，計算コストを減少させられる．導出手順は，次のとおりである．

$$\begin{aligned}\boldsymbol{r}^{(k+1)} &= \boldsymbol{b} - A\boldsymbol{x}^{(k+1)} \\ &= \boldsymbol{b} - A\boldsymbol{x}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)} \qquad \because \text{式 (5.7)} \\ &= \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)} \end{aligned} \tag{5.15}$$

すなわち，残差ベクトル $\boldsymbol{r}^{(k+1)}$ に対しても，式 (5.15) のとおり，$\boldsymbol{r}^{(k)}$ の漸化式で表される．

以上で，共役勾配法の反復計算のアルゴリズムが導出された．

5.2.3 共役性と直交性の確認

次に，$i \neq j$ のとき，

$$(\boldsymbol{r}^{(i)}, \boldsymbol{r}^{(j)}) = 0, \quad \text{(直交性)}, \tag{5.16}$$

$$(\boldsymbol{p}^{(i)}, A\boldsymbol{p}^{(j)}) = 0, \quad \text{(共役性)} \tag{5.17}$$

が成り立つことを示す．

これらを帰納法を用いて証明する．$i \neq j$ かつ，$i \leq k$ と $j \leq k$ に対し，式 (5.16) と (5.17) が成り立つと仮定する．その時，$i = k+1$，$j \leq k$ に対して，

$$(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(j)}) = 0 \tag{5.18}$$

と

$$(\boldsymbol{p}^{(k+1)}, A\boldsymbol{p}^{(j)}) = 0 \tag{5.19}$$

が成り立つことを示せば良い．

● 直交性について：　(5.18) を示す

i) $j < k$ のとき

$$\begin{aligned}(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(j)}) &= (\boldsymbol{r}^{(j)}, \boldsymbol{r}^{(k+1)}) \\ &= (\boldsymbol{r}^{(j)}, \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)}) \\ &= -\alpha_k (\boldsymbol{r}^{(j)}, A\boldsymbol{p}^{(k)}) \qquad \because \text{式 (5.16) の仮定} \\ &= -\alpha_k (\boldsymbol{p}^{(j)} - \beta_{j-1}\boldsymbol{p}^{(j-1)}, A\boldsymbol{p}^{(k)}) \qquad \because \text{式 (5.11)} \\ &= 0 \qquad \because \text{式 (5.17) の仮定} \end{aligned}$$

ii) $j = k$ のとき

$$\begin{aligned}(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k)}) &= (\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}) - \alpha_k (\boldsymbol{r}^{(k)}, A\boldsymbol{p}^{(k)}) \\ &= (\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}) - \alpha_k (\boldsymbol{p}^{(k)} - \beta_{k-1}\boldsymbol{p}^{(k-1)}, A\boldsymbol{p}^{(k)}) \qquad \because \text{式 (5.11)} \\ &= (\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}) - \alpha_k (\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) \qquad \because \text{式 (5.17) の仮定} \end{aligned}$$

$$
\begin{aligned}
&= (\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}) - (\boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)}) \\
&= -\beta_{k-1}(\boldsymbol{p}^{(k-1)}, \boldsymbol{r}^{(k)}) && \because \text{式 (5.11)} \\
&= -\beta_{k-1}(\boldsymbol{p}^{(k-1)}, \boldsymbol{r}^{(k-1)} - \alpha_{k-1}A\boldsymbol{p}^{(k-1)}) \\
&= 0
\end{aligned}
$$

以上から，式 (5.18) が成り立ち，式 (5.16) が示された．

● 共役性について：　(5.19) を示す

i) $j < k$ のとき

$$
\begin{aligned}
(\boldsymbol{p}^{(k+1)}, A\boldsymbol{p}^{(j)}) &= (\boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(j)}) \\
&= (\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(j)}) && \because \text{式 (5.17) の仮定} \\
&= \frac{1}{\alpha_j}(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(j)} - \boldsymbol{r}^{(j+1)}) && \because \text{式 (5.15)} \\
&= 0 \qquad \because (\boldsymbol{r}^{(k)}\text{の直交性}) && \because \text{式 (5.16)}
\end{aligned}
$$

ii) $j = k$ のとき

$$
\begin{aligned}
(\boldsymbol{p}^{(k+1)}, A\boldsymbol{p}^{(k)}) &= (\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)}) + \beta_k(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) \\
&= (\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)}) - (\boldsymbol{r}^{(k+1)}, A\boldsymbol{p}^{(k)}) && \because \text{式 (5.14)} \\
&= 0
\end{aligned}
$$

以上から，式 (5.19) が成り立ち，式 (5.17) が示された．

これで，5.1 節に示した共役勾配法と直交性，共役性が示された．

しかし，5.1 節の計算手順には，改良の余地がある．直交性と共役性を用いると，

$$
(\boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)}) = (\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}), \tag{5.20}
$$

$$
(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)}) = (\boldsymbol{r}^{(k)}, A\boldsymbol{p}^{(k)}) \tag{5.21}
$$

の関係を導くことができる．式 (5.20) を (5.10) の α_k へ代入して，

$$
\alpha_k = \frac{(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})} \tag{5.22}
$$

である．次に，式 (5.21) を (5.14) の β_k に代入し，その式に (5.15)，および，(5.18) から求まる関係 $(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k)}) = 0$ を用いると，

$$
\beta_k = \frac{(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)})}{(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})} \tag{5.23}
$$

と表すこともできる．確かめてみよう．

このように変形するメリットとして，第 $k+1$ 回目の反復において，$(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})$ は，すでに第 k 回目の計算で算出済みなので，新たに計算する必要が無く，式 (5.14) よりも計算量は少なくなる．

このようにして，CG 法のアルゴリズムとなる (**アルゴリズム 5.1**).

アルゴリズム: 5.1 (共役勾配法)

対称正定値行列 A を係数とする連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を CG 法で解く.

初期ベクトル $\boldsymbol{x}^{(0)}$ を用意　(たいていは，$\boldsymbol{x}^{(0)} = \boldsymbol{0}$)

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}, \quad \boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$

for $k = 0, 1, \cdots$

$$\alpha_k = \frac{(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})}$$

$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}$

$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)}$

収束判定

$$\beta_k = \frac{(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)})}{(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})}$$

$\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}$

end

共役勾配法のアルゴリズムの特徴として,「行列とベクトルの積」および「内積」が主な演算である.

残差ベクトル $\boldsymbol{r}^{(k)}$ の直交性より，数値計算による誤差がなければ高々 n 回の反復で $\boldsymbol{r}^{(k)} = \boldsymbol{0}$ となる．したがって CG 法は理論的には，n 回までの反復で必ず厳密解が得られる解法であるが，実際には，計算機による誤差の影響により理論どおりとはならない.

このようなとき,「数値解がどれだけ厳密解を近似しているか」と考えて，相対残差がどれだけ小さくなったのか（収束しているか）を評価し，許容値以内ならば，近似解が得られたと考える．その収束判定では,

$$\begin{aligned} \frac{\|\boldsymbol{r}^{(k+1)}\|}{\|\boldsymbol{r}^{(0)}\|} &\leq \text{許容値}, \\ \boldsymbol{r}^{(0)} &= \boldsymbol{b} - A\boldsymbol{x}^{(0)} \end{aligned} \tag{5.24}$$

のように，相対残差ノルムに対する評価などを用いる．ここで出てきた $\|\boldsymbol{r}^{(k+1)}\|$ について，2-ノルムを用いると，その計算は $\sqrt{(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)})}$ であり，この内積演算自体は，アルゴリズム 5.1 中の β_k の計算に現れているので，収束判定のための余分な計算は不要である.

5.3　前処理付き共役勾配法 (PCG 法)

反復法の収束は係数行列 A と右辺項 $\boldsymbol{b}$ の性質に依存するため, 反復法では $A\boldsymbol{x} = \boldsymbol{b}$ を，同値で性質の良い方程式に変換してから解くと，速く収束することが多い．この操作を**前処理** (preconditioning) という．前処理行列 (preconditioner) として A を近似する行列 $K(K \approx A)$ を用意し, $K^{-1}A\boldsymbol{x} = K^{-1}\boldsymbol{b}$ とすれば,$K^{-1}A$ は単位行列に近くなって早く収束することが期待される（$K = A$ なら解けたことになる）.

ここで注意することは，K が対称であっても，一般には $K^{-1}A$ は対称とはならない．しかし， A が対称正定値行列であるとき，前処理行列を $K = LL^T$ とコレスキー分解すれば，$L^{-1}AL^{-T}$ も対称正定値行列である.

適当な前処理行列 $K = LL^T$ を作り，この下三角行列 L を用いて，元の連立一次方程式 (5.1) を

$$\left(L^{-1}AL^{-T}\right)\left(L^T\boldsymbol{x}\right) \quad = \quad \left(L^{-1}\boldsymbol{b}\right) \tag{5.25}$$

と変換する．実際には，あらかじめ変換された方程式

$$\tilde{A}\tilde{\boldsymbol{x}} \quad = \quad \tilde{\boldsymbol{b}}, \tag{5.26}$$

$$\tilde{A} \quad = \quad L^{-1}AL^{-T}, \quad \tilde{\boldsymbol{x}} = L^T\boldsymbol{x}, \quad \tilde{\boldsymbol{b}} = L^{-1}\boldsymbol{b} \tag{5.27}$$

を作るのではなく，式 (5.26) を満たすようにアルゴリズムの方を書き換える．これを**前処理付き共役勾配法** (Preconditioned Conjugate Gradient method; PCG 法) と呼ぶ.

式 (5.27) の変換以外にも，残差ベクトルは,

$$\tilde{\boldsymbol{r}} \quad = \quad \tilde{\boldsymbol{b}} - \tilde{A}\tilde{\boldsymbol{x}} \ \ (= L^{-1}\boldsymbol{r}) \tag{5.28}$$

と変換される．探索方向は，$\tilde{A}$ と共役になるよう変換すると,

$$\tilde{\boldsymbol{p}} \quad = \quad L^T\boldsymbol{p} \tag{5.29}$$

である.

これらから，前処理付き共役勾配法のアルゴリズムは，**アルゴリズム 5.2** のとおりである.

アルゴリズム: 5.2 (前処理付き共役勾配法)

前処理付き共役勾配法.

初期ベクトル $\boldsymbol{x}^{(0)}$ を用意　(たいていは，$\boldsymbol{x}^{(0)} = \boldsymbol{0}$)

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}, \quad \boldsymbol{p}^{(0)} = K^{-1}\boldsymbol{r}^{(0)}$

for $k = 0, 1, \cdots$

$$\alpha_k = \frac{(K^{-1}\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})}$$

$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k\boldsymbol{p}^{(k)}$

$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)}$

収束判定

$$\beta_k = \frac{(K^{-1}\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)})}{(K^{-1}\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})}$$

$\boldsymbol{p}^{(k+1)} = K^{-1}\boldsymbol{r}^{(k+1)} + \beta_k\boldsymbol{p}^{(k)}$

end

つまり，前処理の演算とは,

$$\boldsymbol{q}^{(k)} \quad = \quad K^{-1}\boldsymbol{r}^{(k)} \tag{5.30}$$

を解くことに帰着される．この式自体も連立一次方程式となっているが，この計算に時間がかかるようでは本末転倒である．この計算には，計算時間をかけずに効率良く解きたい．効率の良い前処理とは,

(1) K が A をよく近似していること ($K \approx A$)

(2) $K = LL^T$ という L が容易に計算できること

(3) L が A と同程度に疎であること

(4) $K^{-1}\boldsymbol{r}$ の計算が容易なこと

などの条件を満足するものである．(1) と (2) だけを考えてみても，前処理された係数行列 $L^{-1}AL^{-T}$ は，単位行列に近いことが分かる．

具体的に前処理行列 K を作る方法の一つに**不完全コレスキー分解** (Incomplete Cholesky factorization) がある．これは，既出の修正コレスキー分解に基づくものであるが，前処理の計算においては，厳密に計算しなくとも良いだろうという観点から，“不完全な” 修正コレスキー分解を行なう．一般的に，この方法を不完全コレスキー分解と呼び，この前処理を適用した CG 法のことを特に，**不完全コレスキー分解前処理付き共役勾配法** (Incomplete Cholesky Conjugate Gradient method; ICCG 法) と呼ぶ.

5.3.1 前処理としての不完全なコレスキー分解

対称正定値な係数行列 A に対する, 不完全コレスキー分解では，2.3 節の “第 3 のコレスキー法”(p.23) に基づく分解を用いて,

$$\begin{aligned} A &= LL^T - R \quad (L \text{ は下三角行列}) \\ &= \bar{L}\bar{D}\bar{L}^T - R \\ &\equiv K - R \end{aligned} \tag{5.31}$$

のように前処理行列 K を作り，R は無視して $K \approx A$ と考える（ここでは，A に対しては不完全な分解，K に対して完全な分解であることに注意）.

具体的な例として，第 4 章でも取上げた，２次元ポアソン方程式などに基づく偏微分方程式を５点中心差分で離散化したときに得られる係数行列

$$A = \left[\begin{array}{ccc|ccc|ccc} a_1 & b_1 & & c_1 & & & & & \\ b_1 & a_2 & b_2 & & c_2 & & & \multicolumn{2}{c}{0} & \\ & b_2 & a_3 & & & c_3 & & & \\ \hline c_1 & & & a_4 & b_4 & & c_4 & & \\ & c_2 & & b_4 & a_5 & b_5 & & c_5 & \\ & & c_3 & & b_5 & a_6 & & & c_6 \\ \hline & & & c_4 & & & a_7 & b_7 & \\ & \multicolumn{2}{c|}{0} & & c_5 & & b_7 & a_8 & b_8 \\ & & & & & c_6 & & b_8 & a_9 \end{array}\right] \tag{5.32}$$

$m = 3, \quad n = m \times m = 9$

を用いて説明する．

ここで b_i は，A の副対角要素を指しており，右辺項ベクトル $\boldsymbol{b}$ の要素とは異なるので注意．本節における説明では，行列 A の要素の記述を簡明にするために，式 (5.32) のように，A の非零要素を a_i, b_i, c_i を用いて表すことにする．インデックス i は，A の下三角要素の列番号であり，

$$a_1 = a_{11},\quad b_1 = a_{21} = a_{12},\quad c_1 = a_{41} = a_{14},\quad \ldots$$

ということである（各々，左辺は a_i, b_i, c_i に基づき，右辺は a_{ij} に基づくものである）．

不完全コレスキー分解では，元の係数行列 A（A の要素のインデックスを (i,j) と表す）が疎行列であることを有効に利用する．

$$\text{非零要素のインデックス集合}\quad G_A = \{(i,j);\ a_{ij} \neq 0\} \tag{5.33}$$

式 (5.32) の係数行列 A を不完全コレスキー分解するときには，L の要素中

$$G_A \quad = \quad \{(i,j);\ j = i,\ i \pm 1,\ i \pm m\} \tag{5.34}$$

に属するインデックスを持つものだけ計算し，他は 0 のままにする．

実際に計算すると，次式のとおりである．行列中の空白箇所は零要素である．

不完全コレスキー分解 (コレスキー分解 3 に基づく方法)

$\bar{L}\bar{D}\bar{L}^T$

$$= \left[\begin{array}{ccc|ccc|ccc} \bar{a}_1 & & & & & & & & \\ \bar{b}_1 & \bar{a}_2 & & & & & & & \\ & \bar{b}_2 & \bar{a}_3 & & & & & & \\ \hline \bar{c}_1 & & & \bar{a}_4 & & & & & \\ & \bar{c}_2 & & \bar{b}_4 & \bar{a}_5 & & & & \\ & & \bar{c}_3 & & \bar{b}_5 & \bar{a}_6 & & & \\ \hline & & & \bar{c}_4 & & & \bar{a}_7 & & \\ & & & & \bar{c}_5 & & \bar{b}_7 & \bar{a}_8 & \\ & & & & & \bar{c}_6 & & \bar{b}_8 & \bar{a}_9 \end{array}\right] \left[\begin{array}{ccc|ccc|ccc} \bar{d}_1 & & & & & & & & \\ & \bar{d}_2 & & & & & & & \\ & & \bar{d}_3 & & & & & & \\ \hline & & & \bar{d}_4 & & & & & \\ & & & & \bar{d}_5 & & & & \\ & & & & & \bar{d}_6 & & & \\ \hline & & & & & & \bar{d}_7 & & \\ & & & & & & & \bar{d}_8 & \\ & & & & & & & & \bar{d}_9 \end{array}\right] \left[\begin{array}{ccc|ccc|ccc} 1 & \bar{b}_1/\bar{a}_1 & & \bar{c}_1/\bar{a}_1 & & & & & \\ & 1 & \bar{b}_2/\bar{a}_2 & & \bar{c}_2/\bar{a}_2 & & & & \\ & & 1 & & & \bar{c}_3/\bar{a}_3 & & & \\ \hline & & & 1 & \bar{b}_4/\bar{a}_4 & & \bar{c}_4/\bar{a}_4 & & \\ & & & & 1 & \bar{b}_5/\bar{a}_5 & & \bar{c}_5/\bar{a}_5 & \\ & & & & & 1 & & & \bar{c}_6/\bar{a}_6 \\ \hline & & & & & & 1 & \bar{b}_7/\bar{a}_7 & \\ & & & & & & & 1 & \bar{b}_8/\bar{a}_8 \\ & & & & & & & & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc|ccc} \bar{a}_1 & \bar{b}_1 & 0 & \bar{c}_1 & & & & & \\ \bar{b}_1 & \bar{d}_1\bar{b}_1^2 + \bar{a}_2 & \bar{b}_2 & \bar{d}_1\bar{b}_1\bar{c}_1 & \bar{c}_2 & & & & \\ 0 & \bar{b}_2 & \bar{d}_2\bar{b}_2^2 + \bar{a}_3 & 0 & \bar{d}_2\bar{b}_2\bar{c}_2 & \bar{c}_3 & & & \\ \hline \bar{c}_1 & \bar{d}_1\bar{b}_1\bar{c}_1 & 0 & \bar{d}_1\bar{c}_1^2 + \bar{a}_4 & \bar{b}_4 & 0 & \bar{c}_4 & & \\ & \bar{c}_2 & \bar{d}_2\bar{b}_2\bar{c}_2 & \bar{b}_4 & \bar{d}_2\bar{c}_2^2 + \bar{d}_4\bar{b}_4^2 + \bar{a}_5 & \bar{b}_5 & \bar{d}_4\bar{b}_4\bar{c}_4 & \bar{c}_5 & \\ & & \bar{c}_3 & 0 & \bar{b}_5 & \bar{d}_3\bar{c}_3^2 + \bar{d}_5\bar{b}_5^2 + \bar{a}_6 & 0 & \bar{d}_5\bar{b}_5\bar{c}_5 & \bar{c}_6 \\ \hline & & & \bar{c}_4 & \bar{d}_4\bar{b}_4\bar{c}_4 & 0 & \bar{d}_4\bar{c}_4^2 + \bar{a}_7 & \bar{b}_7 & 0 \\ & & & & \bar{c}_5 & \bar{d}_5\bar{b}_5\bar{c}_5 & \bar{b}_7 & \bar{d}_5\bar{c}_5^2 + \bar{d}_7\bar{b}_7^2 + \bar{a}_8 & \bar{b}_8 \\ & & & & & \bar{c}_6 & 0 & \bar{b}_8 & \bar{d}_6\bar{c}_6^2 + \bar{d}_8\bar{b}_8^2 + \bar{a}_9 \end{array}\right]$$

$$
= \left[\begin{array}{ccc|ccc|ccc}
a_1 & b_1 & & c_1 & & & & & \\
b_1 & a_2 & b_2 & & c_2 & & & & \\
 & b_2 & a_3 & & & c_3 & & & \\
\hline
c_1 & & & a_4 & b_4 & & c_4 & & \\
 & c_2 & & b_4 & a_5 & b_5 & & c_5 & \\
 & & c_3 & & b_5 & a_6 & & & c_6 \\
\hline
 & & & c_4 & & & a_7 & b_7 & \\
 & & & & c_5 & & b_7 & a_8 & b_8 \\
 & & & & & c_6 & & b_8 & a_9
\end{array}\right]
+ \left[\begin{array}{ccc|ccc|ccc}
 & & & & 0 & & & & \\
 & & & \bar{d}_1\bar{b}_1\bar{c}_1 & 0 & & & & \\
 & & & 0 & \bar{d}_2\bar{b}_2\bar{c}_2 & 0 & & & \\
\hline
 & & & & & & & 0 & \\
0 & \bar{d}_1\bar{b}_1\bar{c}_1 & 0 & & & & \bar{d}_4\bar{b}_4\bar{c}_4 & 0 & \\
 & 0 & \bar{d}_2\bar{b}_2\bar{c}_2 & & & & 0 & \bar{d}_5\bar{b}_5\bar{c}_5 & 0 \\
\hline
 & & & 0 & \bar{d}_4\bar{b}_4\bar{c}_4 & 0 & & & \\
 & & & & 0 & \bar{d}_5\bar{b}_5\bar{c}_5 & & & \\
 & & & & & 0 & & &
\end{array}\right]
$$

$$
\equiv \quad A + R \tag{5.35}
$$

※ 途中で，$\bar{d}_i\bar{a}_i = 1$ の関係 (コレスキー分解 3 の特徴) を用いている．

したがって，

$$
\bar{L}\bar{D}\bar{L}^T \quad = \quad A + R
$$

と表せる．元の係数行列 A をコレスキー分解した場合には，

$$
A \quad = \quad \bar{L}\bar{D}\bar{L}^T - R \;\equiv\; K - R \tag{5.36}
$$

であり，行列 R の要素とは，元々零であった要素（インデックス集合 G_A に含まれていない要素）が，分解により非零 (non-zero) 要素になったもの (これを特に，*fill* 要素と呼ぶ) である[3]．不完全コレスキー分解では，厳格な分解を行なわず，これらの *fill* 要素，すなわち行列 R を無視する．

この分解過程の計算手順は，

$$
\bar{a}_i \quad = \quad \bar{d}_i^{-1} = a_i - b_{i-1}^2\bar{d}_{i-1} - c_{i-m}^2\bar{d}_{i-m}, \quad (1 \le i \le n), \tag{5.37}
$$

$$
\bar{b}_i \quad = \quad b_i, \qquad\qquad (1 \le i \le n-1), \tag{5.38}
$$

$$
\bar{c}_i \quad = \quad c_i, \qquad\qquad (1 \le i \le n-m). \tag{5.39}
$$

したがって，分解過程では対角要素 ($\bar{d}_i$) のみを計算すれば良い．これが，前処理でコレスキー法 3 に基づく分解を用いる大きなメリットである．具体的な不完全コレスキー分解の例を示す．

不完全コレスキー分解の具体的な計算例

[3] この現象を *fill-in* と呼ぶ.

次の対称正定値行列 A を不完全コレスキー分解する．

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \tag{5.40}$$

この行列は，2次元ポアソン方程式において，$m = 2, n(= m \times m) = 4$ に基づくものである．

● 不完全コレスキー分解

$$\begin{aligned} A &= \bar{L}\bar{D}\bar{L}^T - R \\ &= \begin{bmatrix} 4 & & & 0 \\ -1 & \frac{15}{4} & & \\ -1 & 0 & \frac{15}{4} & \\ 0 & -1 & -1 & \frac{52}{15} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & & & 0 \\ & \frac{4}{15} & & \\ & & \frac{4}{15} & \\ 0 & & & \frac{15}{52} \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 & 0 \\ & \frac{15}{4} & 0 & -1 \\ & 0 & \frac{15}{4} & -1 \\ 0 & & & \frac{52}{15} \end{bmatrix} - \begin{bmatrix} 0 & & & 0 \\ & 0 & \frac{1}{4} & \\ & \frac{1}{4} & 0 & \\ 0 & & & 0 \end{bmatrix} \end{aligned} \tag{5.41}$$

※不完全コレスキー分解では，行列 $\bar{L}$ の対角要素に対応する $\bar{D}$ の要素を掛けた値を 1 とする ($\bar{l}_{ii}\bar{d}_i = 1$，ここで $\bar{l}_{ii} = \bar{a}_i$ である) ことが特徴である．

分解計算について

式 (5.37)〜(5.39) を用いて計算し，a_i, b_i, c_i のインデックス i は，式 (5.32) の表記に基づく．

$$\begin{aligned} \bar{a}_1 &= \bar{d}_1^{-1} = a_1 - b_0^2\bar{d}_0 - c_{-1}^2\bar{d}_{-1} = 4 - 0 - 0 = 4, \\ \bar{a}_2 &= \bar{d}_2^{-1} = a_2 - b_1^2\bar{d}_1 - c_0^2\bar{d}_0 = 4 - (-1)^2 \times \frac{1}{4} - 0 = \frac{15}{4}, \\ \bar{a}_3 &= \bar{d}_3^{-1} = a_3 - b_2^2\bar{d}_2 - c_1^2\bar{d}_1 = 4 - 0 - (-1)^2 \times \frac{1}{4} = \frac{15}{4}, \\ \bar{a}_4 &= \bar{d}_4^{-1} = a_4 - b_3^2\bar{d}_3 - c_2^2\bar{d}_2 = 4 - (-1)^2 \times \frac{1}{15} - (-1)^2 \times \frac{1}{15} = \frac{52}{15}, \\ \bar{b}_1 &= b_1 = -1, \\ \bar{b}_3 &= b_3 = -1, \\ \bar{b}_2 &= 0, \\ \bar{c}_1 &= c_1 = -1, \\ \bar{c}_2 &= c_2 = -1. \end{aligned}$$

参考までに，この行列 A をコレスキー分解 3 を用いて完全に分解すると，

$$\begin{aligned} A &= \check{L}\check{D}\check{L}^T \\ &= \begin{bmatrix} 4 & & & 0 \\ -1 & \frac{15}{4} & & \\ -1 & -\frac{1}{4} & \frac{56}{15} & \\ 0 & -1 & -\frac{16}{15} & \frac{24}{7} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & & & 0 \\ & \frac{4}{15} & & \\ & & \frac{15}{56} & \\ 0 & & & \frac{7}{24} \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 & 0 \\ & \frac{15}{4} & -\frac{1}{4} & -1 \\ & & \frac{56}{15} & -\frac{16}{15} \\ 0 & & & \frac{24}{7} \end{bmatrix} \end{aligned} \tag{5.42}$$

である．インデックス集合 G_A に含まれていない $\check{l}_{32}$ 要素が，*fill* 要素であることに注目しよう．

以上が不完全コレスキー分解の手順であるが，実際の前処理の計算では，(5.30) のように，連立一次方程式

$$K\boldsymbol{q} = \boldsymbol{r} \quad \rightarrow \quad (\bar{L}\bar{D}\bar{L}^T)\boldsymbol{q} = \boldsymbol{r} \tag{5.43}$$

を解く．つまり，

$$\begin{cases} \bar{L}\boldsymbol{y} &= \boldsymbol{r}, & \text{(前進代入)}, \\ \bar{D}\bar{L}^T\boldsymbol{q} &= \boldsymbol{y}, & \text{(後退代入)} \end{cases}$$

の 2 つの連立一次方程式を解くことに帰着される．ここで，実際の前処理計算で必要な行列は，$\bar{L}$ と $\bar{D}$ のみであり，前処理行列 K は，議論の上で便宜的に用いているだけであることにも注目しよう．

ここで，$\bar{a}_i\bar{d}_i = 1$ の関係を用いると[4]，上記の計算手順は以下のとおりである．

$$y_i = \bar{d}_i(r_i - b_{i-1}y_{i-1} - c_{i-m}y_{i-m}), \tag{5.44}$$
$$i = 1, 2, \cdots, n, \qquad \text{(前進代入)},$$
$$q_i = y_i - \bar{d}_i(b_iq_{i+1} + c_iq_{i+m}), \tag{5.45}$$
$$i = n, n-1, \cdots, 1, \quad \text{(後退代入)}.$$

※ $\bar{b}_i = b_i$，$\bar{c}_i = c_i$ であることに注意．

一般に，効率的な前処理を作るには問題に対する知識，例えば解こうとしている方程式がどのような物理問題から作られているか，などの情報があると良い．反復法の場合，様々な解法や前処理に対して，相性の良い問題（連立一次方程式）とそうでない問題があり，すべての問題に対して最良である解法は，未だに提案されていない．

5.3.2 CG 法の収束について ～ 前処理によって何が起きたのだろう? ～

CG 法のアルゴリズムでは，同じ計算手順を繰返し，残差の収束を評価して，解に近付いたと判断した．また，その収束を速めるために，前処理という技術も併用した．なぜ,「前処理をほどこす」と「残差が速く収束する」のだろうか？その仕組みについて説明する．

CG 法のアルゴリズム中で，次のとおり，残差ベクトルと探索方向ベクトルの計算を行なっている．

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)},$$
$$\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \beta_k\boldsymbol{p}^{(k)}.$$

これらの 2 式は，互いのベクトルを参照しており，特に $\boldsymbol{r}$ と $\boldsymbol{p}$ の**交代漸化式**と呼ばれる．両式から，

$$\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(0)} + \sum_{l=1}^{k} a_l^{(k)} A^l \boldsymbol{r}^{(0)} \tag{5.46}$$

[4] コレスキー法 3 では，$\bar{d}_i = 1/\bar{l}_{ii}$ と表した（$\bar{a}_i = \bar{l}_{ii}$ である）．

が導かれる．この第２項は，行列 A と初期残差ベクトル $\boldsymbol{r}^{(0)}$ との積であることに着目すると，共役勾配法の反復では，$A^k\boldsymbol{r}^{(0)}$（A の k 乗 $\times\boldsymbol{r}^{(0)}$ ベクトル）で表される部分空間を張っている (span) と考えてよい．これを，

$$\mathcal{K}_k \quad = \quad \mathrm{span}\left\{A\boldsymbol{r}^{(0)}, A^2\boldsymbol{r}^{(0)}, \cdots, A^k\boldsymbol{r}^{(0)}\right\} \tag{5.47}$$

と表現し，この空間は**クリロフ部分空間 (Krylov subspace)** と呼ばれる．

$k = 0, 1, \cdots$ と代入したときの $\boldsymbol{r}^{(k)}$ と $\boldsymbol{p}^{(k)}$ を計算すると，両者とも $\boldsymbol{r}^{(0)}$ と多項式の積で表される．すなわち，

$$\boldsymbol{r}^{(k)} \quad = \quad R_k(A)\boldsymbol{r}^{(0)}, \tag{5.48}$$

$$\boldsymbol{p}^{(k)} \quad = \quad P_k(A)\boldsymbol{r}^{(0)} \tag{5.49}$$

であり，ここでは，$R_k(A)$ を**残差多項式**，$P_k(A)$ を**方向多項式**と呼ぶことにする．各々は k 次多項式である．

ここで，係数行列 A ($n \times n$ 行列) の固有値に着目して，λ_i を A の固有値，$\boldsymbol{v}_i$ を λ_i に対する固有ベクトルとすると，

$$A\boldsymbol{v}_i \quad = \quad \lambda_i\boldsymbol{v}_i \tag{5.50}$$

である．

全ての固有値が異なるときには，固有ベクトルは，互いに一次独立であるので，初期残差ベクトル $\boldsymbol{r}^{(0)}$ を $\boldsymbol{v}_i$ を用いて展開し，

$$\boldsymbol{r}^{(0)} \quad = \quad \sum_{i=1}^{n} c_i\boldsymbol{v}_i \tag{5.51}$$

と表される．(5.48) に対して (5.50) と (5.51) を適用すると，

$$\begin{aligned} \boldsymbol{r}^{(k)} \quad &= \quad R_k(A)\boldsymbol{r}^{(0)} \\ &= \quad \sum_{i=1}^{n} c_i R_k(\lambda_i)\boldsymbol{v}_i \end{aligned} \tag{5.52}$$

と表すことができる．

一方，$\lambda_1, \lambda_2, \cdots, \lambda_p$ $(p < n)$ までは異なる値で，$\lambda_p, \lambda_{p+1}, \cdots, \lambda_{p+s}$ $(p + s \leq n)$ が重複している時には，

$$R_k(\lambda_p) = R_k(\lambda_{p+1}) = \cdots = R_k(\lambda_{p+s}) \tag{5.53}$$

であり，$\boldsymbol{r}^{(k)}$ に含まれている $\boldsymbol{v}_p, \boldsymbol{v}_{p+1}, \cdots, \boldsymbol{v}_{p+s}$ の成分比

$$c_pR_k(\lambda_p) : c_{p+1}R_k(\lambda_{p+1}) : \cdots : c_{p+s}R_k(\lambda_{p+s}) \tag{5.54}$$

が元の $\boldsymbol{r}^{(0)}$ における成分比

$$c_p : c_{p+1} : \cdots : c_{p+s} \tag{5.55}$$

と同じである．このとき，$\boldsymbol{r}^{(k)}$ の動きうる空間の次元が s だけ少なくなる．したがって，高々$n - s$ 回の反復で収束する．固有値が完全に一致していなくても，ある値付近に密集している場合には，上記のことが近似的に成立するため，速く収束することが期待できる．

関連図書

[1] 名取亮 (編), 長谷川秀彦, 他 (著). 数値計算法 (新コンピュータサイエンス講座). 東京, オーム社, 1998.

[2] 名取亮. 数値計算とその応用 (コンピュータ数学シリーズ 15). 東京, コロナ社, 1990.

[3] 名取亮. 線形計算 (すうがくぶっくす 12). 東京, 朝倉書店, 1993.

[4] 森正武. 数値解析 (共立数学講座 12) 第 2 版. 東京, 共立出版, 2002.

[5] 小国力 (編), 長谷川秀彦, 他 (著). 行列計算ソフトウェア - WS, スーパーコン, 並列計算機 -. 東京, 丸善, 1991.

[6] G. H. Golub and C. F. Van Loan. Matrix Computations, 3rd ed. Baltimore, Johns Hopkins University Press, 1996.

[7] Richard Barrett et. al. (長谷川里美 他訳). 反復法 Templates (応用数値計算ライブラリ). 東京, 朝倉書店, 1996. (「`http://www.netlib.org/linalg/html_templates/Templates.html`」から，原書のオンライン版にアクセスできる．)

付 録A　アルゴリズムに現れる基本演算
– MATLAB編 –

連立一次方程式をはじめとする線形計算のアルゴリズムを考えるとき，計算に必要な情報は，ベクトルや行列として表現する．ここでは，これらの表現を用いた基本演算である,「ベクトルどうしの演算」,「行列とベクトルの積」，および,「行列積（行列どうしの積)」のアルゴリズム記述方法とプログラム表現方法について説明する．

また，ガウス消去法やコレスキー分解などの直接法でよく見かける，$\sum$ 記号を含む典型的な演算 (ガウス消去法の後退代入の例) のプログラム表現についても説明する．

さらに，MATLAB を用いた演算記述では，ある程度自由な記述方法が許されているので，それらの中から，理解しておくと便利な記述方法を幾つか紹介する．

A.1　ベクトルどうしの演算

異なる２種類の演算について説明する．

● ベクトル和

$$\boldsymbol{r} \quad = \quad \alpha\boldsymbol{p} + \boldsymbol{q}, \quad \alpha : \text{スカラー定数}, \ \boldsymbol{r}, \boldsymbol{p}, \boldsymbol{q} : n \text{ 次ベクトル} \tag{A.1}$$

これを MATLAB で実行するときには,

【実行例】

```
>> alpha = 3.0          % α の値自体は問題では無い
>> p = [1;2;3;4]        % p の要素の値自体は問題では無い
>> q = [1;1;1;1]        % q の要素の値自体は問題では無い
>> r = alpha*p + q
```

この手続きを MATLAB のスクリプトとして記述するときには,「ex1.m」などと名付けたファイル中に

【記述例１】

```
alpha = 3.0          % α の値自体は問題では無い
p = [1;2;3;4]        % p の要素の値自体は問題では無い
q = [1;1;1;1]        % q の要素の値自体は問題では無い
r = alpha*p + q
```

と記述すれば良い．

ところで，(A.1) 式は，ベクトルの要素を用いて表すと，

$$r_i \quad = \quad \alpha p_i + q_i, \quad i = 1, 2, \cdots, n \tag{A.2}$$

と表現できる．これに基づいて MATLAB のスクリプトファイルを記述する（alpha，p，q の値は定義済みとする）と，
【記述例２】

```
n=4;          % n の値は，[n,n]=size(p) などとしても設定できる．
rr(1:n) = alpha*p(1:n) + q(1:n)
```

あるいは，
【記述例３】

```
n=4;          % n の値は，[n,n]=size(p) などとしても設定できる．
for i = 1 : n
  r(i) = alpha*p(i) + q(i)
end
```

などと表せる (n への代入文の最後の「;」は，結果表示を抑止する命令である)．ただし，変数やベクトルへの値の代入文は省略した．通常，MATLAB では【記述例１】を用いるが，ベクトルの要素の一部を用いる演算 (例えば第 A.4 節) などでは【記述例２，３】のような記述を行う場面もある．

本章の以後の説明では，必要に応じて【記述例１】〜【記述例３】に基づいた記述方法について説明する．

● 内積演算

$$s \quad = \quad (\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{p}^T \boldsymbol{q}, \quad s : \text{スカラー}, \ \boldsymbol{p}, \boldsymbol{q} : n \text{ 次ベクトル} \tag{A.3}$$

これを MATLAB で【記述例１】に基づいて記述すると，

```
p = [1;2;3;4]
q = [4;3;1;2]
s = p'*q          % p の右にある「'」は (共役) 転置を表す．
```

である．

さらに，(A.3) 式を，ベクトルの要素を用いて記述すると，

$$\begin{aligned} s \quad &= \quad p_1 * q_1 + p_2 * q_2 + \cdots + p_n * q_n \\ &= \quad \sum_{i=1}^{n} p_i * q_i \end{aligned} \tag{A.4}$$

である．このような演算はプログラミングにおいては**内積型**や**総和演算**と呼ばれ,【記述例３】に基づいて次のような記述もできる．

```
s = 0;
for i = 1 : n
  s = s + p(i)*q(i)
end
```

このとき，内積の値を順次加えていく変数 s に対しては，計算前に 0 を代入して初期化する必要があることも忘れてはいけない．また，内積演算を表すプログラムを記述するときには注意が必要である．単純に

```
s = p(i)*q(i)
```

と記述すると，これはベクトル $\boldsymbol{p}, \boldsymbol{q}$ の最後の要素だけが変数s に格納されることになる．

A.2　行列とベクトルの積

$$\boldsymbol{r} \quad = \quad A\boldsymbol{p}, \quad A : n \times n \text{ 行列}, \quad \boldsymbol{r}, \boldsymbol{p} : n \text{ 次ベクトル} \tag{A.5}$$

※ 同じ表記であっても，ベクトル $\boldsymbol{p}$ が求めるべき未知数であるときは，連立一次方程式を解くことになるが，ここでは，行列 A にベクトル $\boldsymbol{p}$ をかけた結果を $\boldsymbol{r}$ に格納しているだけである．

多くの場合，行列ベクトル積では，全ての要素に対して同じ演算を施すことになるので，MATLAB の場合には,【記述例１】に基づく記述方法を理解しておけば十分である．

```
A = [1 2 3;4 5 6]
p = [3;2;1]
r = A*p
```

参考として,【記述例２， ３】に基づく記述も以下に記しておく．

```
n=3;
for k = 1 : 2
  r(k) = A(k,1:n) * p(1:n);
end
disp(r)
```

```
n=3;
r(1:2) = 0;
for k = 1 : 2
  for j = 1 : n
    r(k) = r(k) + A(k,j)*p(j);
  end
end
disp(r)
```

A.3　行列積

$$C \quad = \quad AB, \quad A : l \times m \text{ 行列}, \quad B : m \times n \text{ 行列}, \quad C : l \times n \text{ 行列} \tag{A.6}$$

多くの場合，行列積では，全ての要素に対して同じ演算を施すことになるので，MATLAB の場合には，【記述例１】に基づく記述方法を理解しておけば十分である．

```
A = [1 2;3 4;5 6]
B = [4 2;3 1]
C = A*B
```

混乱を避けるために，ここでは【記述例２，３】に基づく参考記述は示さない．

A.4　∑（総和）を含む演算

ガウス消去法の後退代入を例として取上げる．

$$\begin{aligned} &\text{for} && k = n-1, \cdots, 1 \\ &&& x_k \quad = (b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j)/a_{kk}^{(k)} \\ &\text{end} \end{aligned} \tag{A.7}$$

この記述は,【記述例３】に従って,

```
for k = n-1 : -1 : 1    % k=n-1 から k=1 まで，-1 づつ繰り返す
  work = 0.0 ;
  for j = k+1 : n
    work = work + A(k,j) * x(j);   % まず，Σの計算のみを行っている
  end
  x(k) = (b(k) - work) / A(k,k);
end
```

である．

ここで注意するのは，(A.7) 式中の $\sum_{j=k+1}^{n} a_{kj}^{(k)} x_j$ の演算は，一見，行列 A とベクトル $\boldsymbol{x}$ の行列ベクトル積と同じ演算形態に思えるが，これらの全要素に対する演算ではないので，行列ベクトル積とは異なる点である．

また，$\sum$ の部分で１つのループ構造を作っておくと，理解し易いプログラムになる．MATLAB 以外のプログラミング言語でも，このような方法が一般的である．

付 録B　アルゴリズムに現れる基本演算
– C言語編 –

連立一次方程式をはじめとする線形計算のアルゴリズムを考えるとき，計算に必要な情報は，ベクトルや行列として表現する．ここでは，これらの表現を用いた基本演算である,「ベクトルどうしの演算」,「行列とベクトルの積」，および,「行列積（行列どうしの積)」のアルゴリズム記述方法とプログラム表現方法について説明する．

また，ガウス消去法やコレスキー分解などの直接法でよく見かける，$\sum$ 記号を含む典型的な演算 (ガウス消去法の後退代入の例) のプログラム表現についても説明する．

各々の演算に対して，計算の型がいくつかある（行列積では6とおりもある）が，これは，使用するプログラミング言語，計算機のアーキテクチャの特性（例えば，並列計算の効率化などの観点）に応じて適したものは異なる．

少々厄介なのが，ベクトルや行列のインデックスに関する数学的記述と，それを表現するための C 言語プログラミングにおいて通常使われる配列要素のアロケーション方法およびインデックス表現との間に生ずる違いである．これは，ちょっと工夫することで表面上の解決が可能であり，第 B.1 節においていくつかの記述例について説明している．

B.1　ベクトルどうしの演算

異なる2種類の演算について説明する．

● ベクトル和

$$\boldsymbol{r} \quad = \quad \alpha\boldsymbol{p} + \boldsymbol{q}, \quad \alpha : \text{スカラー定数}, \ \boldsymbol{r}, \boldsymbol{p}, \boldsymbol{q} : n \text{ 次ベクトル} \tag{B.1}$$

これは，ベクトルの要素を用いて表すと，

$$r_i \quad = \quad \alpha p_i + q_i, \quad i = 1, 2, \cdots, n \tag{B.2}$$

ということである．C 言語のプログラミングでは,

【記述例1】

```
int    i;
double p[n],q[n],r[n];     /* 配列の値は定義済とする */
double alpha;              /* alpha の値は定義済とする */
for ( i=0 ; i<n ; ++i ){
     r[i] = alpha * p[i] + q[i];
}
```

上記の書き方は，C 言語としては普通の記述だが，数学の観点ではベクトル要素のインデックスが「$1, 2, \cdots, n$」なのに対し，配列要素の表現では，インデックスが「0，1,⋯,n-1」となるため，慣れないと紛らわしい．そのような場合，次のような工夫もできる．

【記述例２】

```
int    I;
double p[n],q[n],r[n];     /* 配列の値は定義済とする */
double alpha;              /* alpha の値は定義済とする */
for ( I=1 ; I<=n ; ++I ){
     i = I-1;
     r[i] = alpha * p[i] + q[i];
}
```

あるいは，配列 p,q の第 0 要素を使わずに，

【記述例３】

```
int    i;
double p[n+1],q[n+1],r[n+1];      /* 配列の値は定義済とする */
double alpha;                     /* alpha の値は定義済とする */
for ( i=1 ; i<=n ; ++i ){
     r[i] = alpha * p[i] + q[i];
}
```

と記述するなど．

本章の以後の説明では，数学的記述と C 言語プログラミングでの整合性の良い【記述例３】の記述方法にしたがって説明する．

● 内積演算

$$s \quad = \quad (\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{p}^T \boldsymbol{q}, \quad s : \text{スカラー}, \ \boldsymbol{p}, \boldsymbol{q} : n \text{ 次ベクトル} \tag{B.3}$$

これは，ベクトルの要素を用いて表すと，

$$\begin{aligned} s &= p_1 * q_1 + p_2 * q_2 + \cdots + p_n * q_n \\ &= \sum_{i=1}^{n} p_i * q_i \end{aligned} \tag{B.4}$$

ということである．このような演算は**内積型**や**総和演算**と呼ばれ，

```
     s = s + p[i]*q[i];
```

あるいは

```
     s += p[i]*q[i];
```

と記述する．変数 s に対しては，計算前に 0 を代入して初期化する必要があることも忘れてはいけない．

【記述例】

```
int    i, n;
double p[n+1],q[n+1];       /* 配列の値は定義済とする */
double s;
s = 0.0;                    /* 0 クリアによる初期化 */
for ( i=1 ; i<=n ; ++i ){
     s += p[i]*q[i];
}
```

ここで，内積演算を表すプログラムを記述するときには注意が必要である．単純に

```
s = p[i]*q[i];
```

と記述すると，これはベクトル $\boldsymbol{p}, \boldsymbol{q}$ の最後の要素しか変数 s に格納されない．

B.2 行列とベクトルの積

$$\boldsymbol{r} \quad = \quad A\boldsymbol{p}, \quad A : n \times n \text{ 行列}, \quad \boldsymbol{r}, \boldsymbol{p} : n \text{ 次ベクトル} \tag{B.5}$$

※ 同じ表記であっても，ベクトル $\boldsymbol{p}$ が求めるべき未知数であるときは，連立一次方程式を解くことになるが，ここでは，行列 A にベクトル $\boldsymbol{p}$ をかけた結果を $\boldsymbol{r}$ に格納しているだけである．

行列とベクトルの積には 2 とおりの計算方法がある．

● 行方式（内積型）

$$\begin{aligned} \boldsymbol{r} &= A\boldsymbol{p} \\ &= \left[\begin{array}{c} \boldsymbol{a}^1 \\ \hline \boldsymbol{a}^2 \\ \hline \vdots \\ \hline \boldsymbol{a}^n \end{array}\right] \left[\begin{array}{c} \\ \boldsymbol{p} \\ \\ \end{array}\right] = \left[\begin{array}{c} \boldsymbol{a}^1\boldsymbol{p} \\ \hline \boldsymbol{a}^2\boldsymbol{p} \\ \hline \vdots \\ \hline \boldsymbol{a}^n\boldsymbol{p} \end{array}\right] \end{aligned} \tag{B.6}$$

ここで，$\boldsymbol{a}^i$ は，行列 A の第 i 行を表す行ベクトルである．これは，各々の要素を用いて表現すると

$$r_i \quad = \quad \sum_{j=1}^{n} a_{ij} p_j, \quad i = 1, 2, \cdots, n \tag{B.7}$$

である．a_{ij} は，行列 A の (i, j) 要素（行ベクトル a^i の j 番目の要素と同じ）を表す．つまり，前節の内積計算と同じ演算である．C 言語のプログラミング例は，以下のとおり．

【記述例】

```
int    i, j, n;
double a[n+1][n+1],p[n+1],r[n+1];   /* 配列の値は定義済とする */
double s;
for ( i=1 ; i<=n ; ++i ){
   s = 0.0;                          /* 0 クリアによる初期化 */
   for ( j=1 ; j<=n ; ++j ){
        s += a[i][j]*p[j];
   }
   r[i] = s;
}
```

● 列方式（三項演算型）

$$
\begin{aligned}
\boldsymbol{r} &= A\boldsymbol{p} \\
&= \left[\begin{array}{c|c|c|c} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{array}\right] \left[\begin{array}{c} p_1 \\ \hline p_2 \\ \hline \vdots \\ \hline p_n \end{array}\right] \\
&= p_1\boldsymbol{a}_1 + p_2\boldsymbol{a}_2 + \cdots + p_n\boldsymbol{a}_n
\end{aligned} \tag{B.8}
$$

ここで，$\boldsymbol{a}_j$ は，行列 A の第 j 列を表す列ベクトルである．これは，各々の要素を用いて表現すると

$$
r_i = r_i + a_{ij}p_j, \quad j = 1, 2, \cdots, n \tag{B.9}
$$

の演算を，$i = 1, 2, \cdots, n$ まで行なう．つまり，前節のベクトル和と同じ計算である（ある i において，p_j がスカラー変数，a_{ij} が第 j 番の列ベクトルということ）．C 言語のプログラミング例は，以下のとおり．

【記述例】

```
int    i,j;
double a[n+1][n+1],p[n+1],r[n+1];    /* 配列の値は定義済とする */
double s;
for ( j=1 ; j<=n ; ++j ){
   for ( i=1 ; i<=n ; ++i ){
      r[i] = 0.0;
   }
   for ( i=1 ; i<=n ; ++i ){
      r[i] += p[j] * a[i][j];
   }
}
```

プログラムの比較では分かりにくいが，C コンパイラなどのように行方向 (配列を a[i][j] と定義したときの j インデックスの方向) に連続アクセスする場合には，前者の行方式が良く，Fortran コンパイラなどのように列方向 (配列を a(i,j) と定義したときの i インデックスの方向) に連続アクセスする場合には，後者の列方式が良い．

また，異なるアーキテクチャの計算機にて計算効率を議論するときに，上記のように検討することもある．

B.3 行列積

$$C \quad = \quad AB, \quad A : l \times m \text{ 行列}, \quad B : m \times n \text{ 行列}, \quad C : l \times n \text{ 行列} \tag{B.10}$$

このとき，行列 C の (i, j) 要素 c_{ij} は，

$$c_{ij} \quad = \quad \sum_{k=1}^{m} a_{ik} b_{kj}, \quad i = 1, \cdots, l, \quad j = 1, \cdots, n \tag{B.11}$$

である．行列 C の計算では，i，j，k のインデックスを用いるので，３重ループ構造が必要になり，インデックスの並べ方に応じて６とおりの記述ができる．ここでは，内積型演算に基づく ijk 型とベクトル和型演算に基づく ikj 型を紹介する．

● ijk 型

【記述例】

```
int    i,j,k,l,m,n;
double a[l+1][m+1],b[m+1][n+1],c[l+1][n+1];  /* 配列の値は定義済とする */
double s;
for ( i=1 ; i<=l ; ++i ){
   for ( j=1 ; j<=n ; ++j ){
      s = 0.0;                    /* 0 クリアによる初期化 */
      for ( k=1 ; k<=m ; ++k ){
         s += a[i][k]*b[k][j];    /* メモリアクセスを減らすために，配列 c */
      }                           /* に直接代入せず，スカラー変数に代入する */
      c[i][j] = s;
   }
}
```

● ikj 型

【記述例】

```
int    i,j,k,l,m,n;
double a[l+1][m+1],b[m+1][n+1],c[l+1][n+1];    /* 配列の値は定義済とする */
for ( i=1 ; i<=l ; ++i ){
   for ( j=1 ; j<=n ; ++j ){
      c[i][j] = 0.0;
   }
   for ( k=1 ; k<=m ; ++k ){
      for ( j=1 ; j<=n ; ++j ){
         c[i][j] += a[i][k]*b[k][j];
      }
   }
}
```

B.4 ∑（総和）を含む演算

ガウス消去法の後退代入を例として取上げる.

$$\begin{aligned} &\text{for} && k = n-1, \cdots, 1 \\ &&& x_k = (b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j)/a_{kk}^{(k)} \\ &\text{end} \end{aligned} \tag{B.12}$$

C 言語のプログラミングでは, $\sum$ の部分で 1 つのループ構造ができる.

```
int    i,j,k;
double a[n+1][n+1],b[n+1];    /* 配列の値は定義済とする */
double x[n+1];                /* 配列 x の値は初期化済とする */
double work;
  for(k=n-1; k>=1; --k){
    work = 0.0e0;             /* 総和演算用の work 変数を初期化する */
    for(j=k+1; j<=n; ++j){
      work += a[k][j] * x[j];       /* Σ_{j=k+1}^{n} a_{kj}^{(k)} x_j の演算だけ取出して計算しておく */
    }
    x[k] = ( b[k] - work ) / a[k][k];
  }
```

数値計算法　第10回（補足資料-2）

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

この $A\boldsymbol{x} = \boldsymbol{b}$ を，初期ベクトル $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ として，共役勾配法を用いて解く．
ただし，ここでは，理論通りに解が求められることを確認するために，丸め誤差の生じない厳密な計算を行い，計算途中では分数表現を用いて良い．

初期ベクトルの計算

$$\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}$$

$k = 0$ について

$$A\boldsymbol{p}^{(0)} = \begin{bmatrix} 8 \\ -4 \\ 0 \end{bmatrix}$$

$$\alpha_0 = \frac{(\boldsymbol{r}^{(0)}, \boldsymbol{r}^{(0)})}{\boldsymbol{p}^{(0)}, A\boldsymbol{p}^{(0)})} = \frac{16}{32} = \frac{1}{2}$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \alpha_0 \boldsymbol{p}^{(0)} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{r}^{(1)} = \boldsymbol{r}^{(0)} - \alpha_0 A\boldsymbol{p}^{(0)} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 4 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

$$\beta_0 = \frac{(\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(1)})}{(\boldsymbol{r}^{(0)}, \boldsymbol{r}^{(0)})} = \frac{4}{16} = \frac{1}{4},$$

$$\boldsymbol{p}^{(1)} = \boldsymbol{r}^{(1)} + \beta_0 \boldsymbol{p}^{(0)} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

$k = 1$ について

$$A\boldsymbol{p}_1 = \begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix}, \quad \left(\equiv \begin{bmatrix} 0 & 3 & -2 \end{bmatrix}^T, \right)$$

$$\alpha_1 = \frac{(\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(1)})}{(\boldsymbol{p}^{(1)}, A\boldsymbol{p}^{(1)})} = \frac{4}{6} = \frac{2}{3},$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \alpha_1 \boldsymbol{p}^{(1)} = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T + \frac{2}{3} \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T = \begin{bmatrix} 8/3 & 4/3 & 0 \end{bmatrix}^T,$$

$$\boldsymbol{r}^{(2)} = \boldsymbol{r}^{(1)} - \alpha_1 A\boldsymbol{p}^{(1)} = \begin{bmatrix} 0 & 2 & 0 \end{bmatrix}^T - \frac{2}{3} \begin{bmatrix} 0 & 3 & -2 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 4/3 \end{bmatrix}^T,$$

$$\beta_1 = \frac{(\boldsymbol{r}^{(2)}, \boldsymbol{r}^{(2)})}{(\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(1)})} = \frac{16/9}{4} = \frac{4}{9},$$

$$\boldsymbol{p}^{(2)} = \boldsymbol{r}^{(2)} + \beta_1 \boldsymbol{p}^{(1)} = \begin{bmatrix} 0 & 0 & 4/3 \end{bmatrix}^T + \frac{4}{9} \begin{bmatrix} 1 & 9 & 0 \end{bmatrix}^T = \begin{bmatrix} 4/9 & 8/9 & 4/3 \end{bmatrix}^T$$

$k = 2$ について

$$A\boldsymbol{p}^{(2)} = \begin{bmatrix} 0 & 0 & \frac{16}{9} \end{bmatrix}^T,$$

$$\alpha_2 = \frac{3}{4},$$

$$\boldsymbol{x}^{(3)} = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}^T,$$

$$\boldsymbol{r}^{(3)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

ここで，残差ベクトルの全要素が 0 となっていることに注目．

※ 通常，共役勾配法は計算機で実行させるものであり，この例題のように一つ一つの計算過程について検討するようなことはしない．

数値計算法　第8回（補足資料-1）

次の連立一次方程式を LU 分解して解きたい．(a)〜(c) の手順に従って求解せよ．

$$\begin{bmatrix} 3 & 4 & 1 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 14 \\ 14 \\ 20 \end{bmatrix} \tag{1}$$

(a) 上三角行列 U を求めよ．

(b) 下三角行列 L を求めよ．

(c) 前進代入と後退代入の手順により，解 $\boldsymbol{x}$ を求めよ．

$$A = \begin{bmatrix} 3 & 4 & 1 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 14 \\ 14 \\ 20 \end{bmatrix}$$

(a) 上三角行列 U を求めよ．

第 1 段： $m_{21} = \frac{1}{3}$, $m_{31} = \frac{2}{3}$

$$M_1 A = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{2}{3} & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & 1 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 1 \\ 0 & \frac{2}{3} & \frac{8}{3} \\ 0 & \frac{1}{3} & \frac{10}{3} \end{bmatrix} \equiv A^{(2)} \tag{2}$$

第 2 段： $m_{32} = \left(\frac{1}{3}/\frac{2}{3}\right) = \frac{1}{2}$

$$M_2 A^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & 1 \\ 0 & \frac{2}{3} & \frac{8}{3} \\ 0 & \frac{1}{3} & \frac{10}{3} \end{bmatrix} = \begin{bmatrix} 3 & 4 & 1 \\ 0 & \frac{2}{3} & \frac{8}{3} \\ 0 & 0 & 2 \end{bmatrix} \equiv U \tag{3}$$

したがって，

$$U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix} = \begin{bmatrix} 3 & 4 & 1 \\ 0 & \frac{2}{3} & \frac{8}{3} \\ 0 & 0 & 2 \end{bmatrix} \tag{4}$$

(b) 下三角行列 L を求めよ．

$M_2 M_1 A = U$ より，$A = M_1^{-1} M_2^{-1} U$ である．したがって，$L = M_1^{-1} M_2^{-1}$ である．

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{2}{3} & 0 & 1 \end{bmatrix}, \quad M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} \tag{5}$$

これより,

$$L = M_1^{-1}M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{2}{3} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{2}{3} & \frac{1}{2} & 1 \end{bmatrix} \tag{6}$$

したがって,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{2}{3} & \frac{1}{2} & 1 \end{bmatrix} \tag{7}$$

(c) 前進代入と後退代入の手順により，解 $\boldsymbol{x}$ を求めよ.

LU 分解された後の前進代入と後退代入の一般的な式を行列とベクトルにより表してから,
実際に解 $\boldsymbol{x} = [x_1\ x_2\ x_3]^T$ を求めることにする.

$$A\boldsymbol{x} = \boldsymbol{b} \text{ より } LU\boldsymbol{x} = \boldsymbol{b} \tag{8}$$

このとき,

$$L\boldsymbol{y} = \boldsymbol{b}, \quad \text{(前進代入)}, \tag{9}$$

$$U\boldsymbol{x} = \boldsymbol{y}, \quad \text{(後退代入)}. \tag{10}$$

前進代入について：

$$\begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{11}$$

$$\begin{cases} y_1 = b_1 \\ y_2 = b_2 - m_{21}y_1 \\ y_3 = b_3 - \sum_{j=1}^{2} m_{3j}y_j = b_3 - m_{31}y_1 - m_{32}y_2 \end{cases} \tag{12}$$

であり，各値を実際に代入すると，$\boldsymbol{y} = \left[14, \frac{28}{3}, 6\right]^T$ が求まる.

後退代入について：

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{13}$$

$$\begin{cases} x_3 = \frac{1}{a_{33}^{(3)}} y_3 \\ x_2 = \frac{1}{a_{22}^{(2)}} \left(y_2 - a_{23}^{(2)} x_3\right) \\ x_1 = \frac{1}{a_{11}^{(1)}} \left(y_1 - \sum_{j=2}^{3} a_{1j}^{(1)} x_j\right) = \frac{1}{a_{11}^{(1)}} \left(y_1 - a_{12}^{(1)} x_2 - a_{13}^{(1)} x_3\right) \end{cases} \tag{14}$$

であり，各値を実際に代入すると，解は $\boldsymbol{x} = [1, 2, 3]^T$ と求まる.

数値計算法　第10回（補足資料-1）

1. 一次結合，一次独立，一次従属

一次結合（線形結合）　線形空間 V のベクトル $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k$ を，それぞれスカラー倍したものの和

$$t_1\boldsymbol{a}_1 + t_2\boldsymbol{a}_2 + \cdots + t_k\boldsymbol{a}_k \tag{1}$$

を $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k$ の一次結合（線形結合）という．

一次独立（線形独立）と一次従属（線形従属）
$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k$ を，線形空間 V のベクトルとする．

$$t_1\boldsymbol{a}_1 + t_2\boldsymbol{a}_2 + \cdots + t_k\boldsymbol{a}_k = \boldsymbol{0} \tag{2}$$
$$\text{ならば} \quad t_1 = t_2 = \cdots = t_k = 0$$

のとき，$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k$ は，**一次独立（線形独立）**であるという．

一方，少なくとも一つは 0 でない $t_1, t_2, \cdots, t_k$ に対して，(2) が成り立つことがあるとき，$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k$ は，一次従属（線形従属）であるという．

2. 内積

内積空間の定義　線形空間 V において，V の任意のベクトル $\boldsymbol{a}$，$\boldsymbol{b}$ に対して実数値 $(\boldsymbol{a}, \boldsymbol{b})$ が定まり，次の４条件をみたすとき，$(\boldsymbol{a}, \boldsymbol{b})$ を $\boldsymbol{a}$ と $\boldsymbol{b}$ の内積という．

(i) $(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{b}, \boldsymbol{a})$
(ii) $(\boldsymbol{a}, \boldsymbol{b}+\boldsymbol{c}) = (\boldsymbol{a}, \boldsymbol{b}) + (\boldsymbol{a}, \boldsymbol{c})$
(iii) $(\lambda\boldsymbol{a}, \boldsymbol{b}) = \lambda\,(\boldsymbol{a}, \boldsymbol{b})$
(iv) $\boldsymbol{a} \neq \boldsymbol{0}$のとき $(\boldsymbol{a}, \boldsymbol{a}) > 0$

(ii)(iii) を線形性，(iv) を正値性という．内積の ( ) を展開すると，

$$(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b}. \tag{3}$$
$$(\quad = \quad {}^t\boldsymbol{a}\boldsymbol{b} \quad \text{とも記述される})$$

また，$(\boldsymbol{a}, \boldsymbol{b}) = 0$ のとき，ベクトル $\boldsymbol{a}$ と $\boldsymbol{b}$ は直交するという．

3. 対称行列

対称行列　n 次正方行列 $A = [a_{ij}]$（a_{ij} は実数）が，

$$A = A^T \tag{4}$$
$$\text{すなわち,} \quad a_{ij} = a_{ji} \quad (i, j = 1, \cdots, n)$$

をみたすとき，A を対称行列という．

対称行列における関係式，

$$\begin{aligned}(\boldsymbol{x}, A\boldsymbol{y}) &= \boldsymbol{x}^T A\boldsymbol{y} = \boldsymbol{x}^T A^T\boldsymbol{y} \\ &= (A\boldsymbol{x})^T\boldsymbol{y} = (A\boldsymbol{x}, \boldsymbol{y}) \\ &= (\boldsymbol{y}, A\boldsymbol{x}).\end{aligned} \tag{5}$$

4. 正定値 (正値) 行列

任意のベクトル $\boldsymbol{x} \neq \boldsymbol{0}$ に対して，

$$(\boldsymbol{x}, A\boldsymbol{x}) > 0 \tag{6}$$

を満たす行列を正定値 (正値) 行列 (positive definite matrix) という．ここで，$(\boldsymbol{x}, \boldsymbol{y})$ は，$\boldsymbol{x}$と$\boldsymbol{y}$の内積を表す．

5. 数学的帰納法

自然数について述べられている命題が $n \leq n_0$（n_0 は自然数）について成り立つことを証明するのに，

(a) その命題は $n = n_0$ のとき成り立つ．
(b) その命題が $n = k$（k は $k \leq n_0$ の自然数）のとき成り立つことを仮定すれば，$n = k+1$ のときもまた成り立つ．

の２段階を証明して，与えられた命題が成り立つことを示す方法．

$n_0 = 1$ とすれば，全ての自然数について成り立つことを示せる．また，(b) の仮定を「$n \leq k$ のとき成り立つ」としても良い．

例題

$$\begin{aligned}&1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots - \frac{1}{2n} \\ &= \frac{1}{n+1} + \frac{1}{n+2} + \cdots + \frac{1}{2n}\end{aligned} \tag{7}$$

証明

(a) $n = 1$ のとき，$1 - \frac{1}{2} = \frac{1}{2}$ であり，(7) は成り立つ．
(b) $n = k$ のとき，(7) は成り立つと仮定すると，

$$\begin{aligned}&\left(1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots - \frac{1}{2k}\right) + \frac{1}{2k+1} - \frac{1}{2(k+1)} \\ =\ &\left(\frac{1}{k+1} + \frac{1}{k+2} + \cdots + \frac{1}{2k}\right) + \frac{1}{2k+1} - \frac{1}{2(k+1)} \\ =\ &\frac{1}{k+2} + \cdots + \frac{1}{2k} + \frac{1}{2k+1} + \frac{1}{2(k+1)}\end{aligned}$$

であり，(7) は $n = k+1$ に対しても成り立つ．

以上，(a)(b) より，(7) は全ての自然数 n に対して成り立つ．