1.4 連立 1 次方程式

連立 1 次方程式は

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{1.13}$$

と書ける. 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に注意しよう. 式 (1.13) 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

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

と書ける. このとき

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

である.

$A\boldsymbol{x} = \boldsymbol{b}$ が解を持つ条件は

$$\det(A) \neq 0$$
$$\mathrm{rank}(A) = n$$

である.

この方程式を実際に解くにはガウスの消去法を使う. コンピュータに解かせることを前提にするなら, 単純かつ汎用性のある手順を作り上げることが重要である.

1.4.1 直接法 — ガウスの消去法 —

ガウス (Gauss) 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 (elimination) する. それには第 1 番方程式を何倍かして加え第 2 番方程式から第 n 番方程式までの 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.

名取亮編
数値計算法, オーム社, 1998

る. つまり第 i 番方程式用の乗数を α_{i1} とすると

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

である. $\alpha_{i1}a_{11} + a_{i1} = 0$ とするには

$$\alpha_{i1} = -\frac{a_{i1}}{a_{11}} \quad (i = 2, \cdots, n)$$

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 α_{21}, α_{31}, $\cdots, \alpha_{n1}$ が求まる. 第 1 番方程式に α_{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, 連立方程式 (1.13) は同値な連立方程式

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad \vdots \\ \qquad a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases}$$

に変換される. ここで $a_{22}^{(1)}$, $b_2^{(1)}$ などとあるのは

$$\begin{cases} \alpha_{i1} = -\dfrac{a_{i1}}{a_{11}} \\ a_{ij}^{(1)} = a_{ij} + \alpha_{i1}a_{1j} \\ b_i^{(1)} = b_i + \alpha_{i1}b_1 \end{cases} \quad (i = 2, \cdots, n)$$

によって更新された値である. この操作を第 $(k-1)$ 番方程式を用いた消去まで繰り返すと

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1k}x_k + \cdots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + \cdots + a_{2k}^{(1)}x_k + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad \ddots \\ \qquad\qquad\qquad a_{kk}^{(k-1)}x_k + \cdots + a_{kn}^{(k-1)}x_n = b_k^{(k-1)} \\ \qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad a_{nk}^{(k-1)}x_k + \cdots + a_{nn}^{(k-1)}x_n = b_n^{(k-1)} \end{cases}$$

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 $a_{kk}^{(k-1)} \neq 0$ のとき

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

を用いて

$$\begin{cases} a_{ij}^{(k)} &= a_{ij}^{(k-1)} &+ \alpha_{ik} a_{kj}^{(k-1)} \\ b_i^{(k)} &= b_i^{(k-1)} &+ \alpha_{ik} b_k^{(k-1)} \end{cases}$$

という操作を $i = k+1, \cdots, n;\ j = k+1, \cdots, n$ に施せばよい. このようにして第 $(n-1)$ 番方程式による消去までを繰り返すと, 連立方程式は

$$\begin{cases} a_{11}x_1 + \cdots + a_{1k}x_k + \cdots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{2k}^{(1)}x_k + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\quad \ddots \\ \qquad\qquad a_{kk}^{(k-1)}x_k + \cdots + a_{kn}^{(k-1)}x_n = b_k^{(k-1)} \\ \qquad\qquad\quad \ddots \\ \qquad\qquad\qquad a_{n-1,n-1}^{(n-2)}x_{n-1} + a_{n-1,n}^{(n-2)}x_n = b_{n-1}^{(n-2)} \\ \qquad\qquad\qquad\qquad a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

となる. ここまでの過程を**前進消去過程** (forward elimination process) , または**上三角化過程**という. いちばん下の第 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, $a_{nn}^{(n-1)} \neq 0$ なら

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

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

$$x_{n-1} = \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)}x_n}{a_{n-1,n-1}^{(n-2)}}$$

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

$$x_k = \frac{b_k^{(k-1)} - \sum_{j=k+1}^{n} a_{kj}^{(k-1)}x_j}{a_{kk}^{(k-1)}}$$

として求められる. この過程を**後退代入過程** (backward substitution process) という. 前進消去が問題なく終了していれば $a_{kk}^{(k-1)} \neq 0$ 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方程式は解けるが, 次のようなとき

$$\begin{cases} 0x_1 + 1x_2 + 2x_3 = 2 \\ 1x_1 + 0x_2 + 3x_3 = 2 \\ 3x_1 + 1x_2 + 0x_3 = -3 \end{cases} \qquad (1.14)$$

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 $a_{k,k}^{(k-1)}, \cdots, a_{n,k}^{(k-1)}$ 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. 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 $|\alpha_{ik}| \leq 1$ 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, 式 (1.14) を部分軸選択つきガウスの消去法で解いてみよう.

前進消去過程

$k = 1$ 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.

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

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.

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

$k = 2$ 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.

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

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.

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

$k = 3$ 段:

第 3 番方程式の x_3 の係数はゼロではない. 前進消去過程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.

後退代入過程

$k = 3$ 段: $\frac{1}{3}x_3 = \frac{1}{3}$ 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 $x_1 = \frac{-3-x_2}{3}$, $x_2 = 0$ を代入して $x_1 = -1$.

部分軸選択つきガウスの消去法のアルゴリズムをプログラムふうにまとめたのが例 1.1 である. 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 k を 1 から n まで変えた繰り返し

を意味する. ε 以下をゼロとみなす. ir にはゼロとみなされた回数が入る. 例 1.1 のアルゴリズムに必要な浮動小数演算の概数を見積もると

前進消去過程	左辺	$\frac{2}{3}n^3$ 回	(*)
	右辺	n^2 回	(**)
後退代入過程	右辺	n^2 回	(***)

〈例 1.1〉 $Ax = b$ を部分軸選択つきガウスの消去法で解く.

[アルゴリズム]

```
ir=0
for k = 1, 2, ..., n
    amax =| a_kk | ; ip = k
    for i = k+1, k+2, ..., n
        if | a_ik |> amax then
            amax =| a_ik | ; ip = k
        end if
    if amax > ε then
        for j = k+1, k+2, ..., n
            a_ip,j ⇌ a_kj
        for i = k+1, k+2, ..., n
            a_ik = -a_ik / a_kk
            for j = k+1, k+2, ..., n
                a_ij = a_ij + a_ik · a_kj          (*)
            b_i = b_i + a_ik · b_k                 (**)
    else
        ir = ir + 1
    end if
if ir ≠ 0 then
    for k = n, n-1, ..., 1
        x_k = ( b_k - Σ_{j=k+1}^{n} a_kj · x_j ) / a_kk   (***)
end if
```

である. 左辺の前進消去と右辺の前進消去・後退代入は分離可能で, しかも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 n 元連立 1 次方程式を p

回解くための浮動小数演算回数はおよそ

$$\frac{2}{3}n^3 + p(2n^2) \text{ 回}$$

である．$p = n$ 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．式 (1.8) 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

$$\frac{8}{3}n^3 + 2n^2 \text{ 回}$$

の浮動小数演算が必要となる．ガウスの消去法で直接計算する場合の $\frac{2}{3}n^3 + 2n^2$ と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，逆行列の要素の値が知りたいとき以外，**数値計算では逆行列を使わない**のが常識である．

ガウスの消去法 k 段の操作

P_k ： k 番方程式と ip 番方程式を入れ換える

G_k ： k 番方程式をα_{ik}倍して i 番方程式 $(i = k+1, \cdots, n)$ に加える

は次のような行列で表せる．

$$P_k = \begin{array}{c} \\ \\ \\ k \text{ 行} \\ \\ ip \text{ 行} \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & k \text{ 列} & & ip \text{ 列} & & & \end{array} \\ \begin{pmatrix} 1 & & & \vdots & & \vdots & & & \\ & \ddots & & \vdots & & \vdots & & & \\ & & 1 & \vdots & & \vdots & & & \\ \cdots & \cdots & \cdots & 0 & \cdots & 1 & \cdots & \cdots & \cdots \\ & & & \vdots & \ddots & \vdots & & & \\ \cdots & \cdots & \cdots & 1 & \cdots & 0 & \cdots & \cdots & \cdots \\ & & & \vdots & & \vdots & 1 & & \\ & & & \vdots & & \vdots & & \ddots & \\ & & & \vdots & & \vdots & & & 1 \end{pmatrix} \end{array}$$

$$G_k = \begin{array}{c} \\ \\ \\ \\ \\ k \text{ 列} \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & & & & \end{array} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & \alpha_{k+1,k} & 1 & & & \\ & & \vdots & & 1 & & \\ & & \alpha_{i,k} & & & 1 & \\ & & \vdots & & & & \ddots \\ & & \alpha_{n,k} & & & & 1 \end{pmatrix} \end{array}$$

ガウスの消去法の前進消去過程は P_k, G_k を使って

$$G_{n-1}P_{n-1} \cdots G_kP_k \cdots G_1P_1A\boldsymbol{x} = G_{n-1}P_{n-1} \cdots G_1P_1\boldsymbol{b}$$

と書ける．$G_{n-1}P_{n-1} \cdots G_kP_k \cdots G_1P_1A$ は**上三角行列** (upper triangular matrix)

$$U = \begin{pmatrix} a_{11} & \cdots & \cdots\cdots & \cdots & a_{1n} \\ & a_{22}^{(1)} & \cdots\cdots & \cdots & a_{2n}^{(1)} \\ & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ & & & & a_{nn}^{(n-1)} \end{pmatrix}$$

である．

$$G_{n-1}P_{n-1} \cdots G_1P_1A = U$$

より

$$(G_{n-1}P_{n-1} \cdots G_1P_1)^{-1}U = A$$

と書いて A の ***LU*分解**(LU decomposition または LU factorization) ということもある．狭義の LU 分解では A を**下三角行列** (lower triangular matrix) 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

$$L\boldsymbol{y} = \boldsymbol{b}$$
$$U\boldsymbol{x} = \boldsymbol{y}$$

の 2 段階，すなわち L に対する前進代入と U に対する後退代入に分けて行う．LU 分解とガウスの消去法に本質的な差はない．またガウスの消去法には，扱う行列の特徴，計算環境などによっていろいろな手法がある．

1.3 ノルム，誤差，残差，条件数

$n \times n$ の行列 A と n 次元ベクトル $\boldsymbol{x}, \boldsymbol{b}$ で表された連立 1 次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1.8}$$

を考える. ベクトル $\boldsymbol{x}$ を 1 つの数値で代表させたものに**ノルム** (norm) があり, $\|\boldsymbol{x}\|$ と書く. ベクトルのノルムには

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$$

$$\|\boldsymbol{x}\|_\infty = \max_i |x_i|$$

などがあり, $\|\boldsymbol{x}\|_1$ を **1–ノルム**, $\|\boldsymbol{x}\|_2$ を **2–ノルム** (ユークリッドノルム) と呼ぶ.

同様に行列のノルム $\|A\|$ には

$$\|A\|_1 = \sup_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_1}{\|\boldsymbol{x}\|_1}$$

$$\|A\|_2 = \sup_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}$$

$$\|A\|_\infty = \sup_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}$$

などがある. 行列の 1–ノルム $\|A\|_1$ は

$$\|A\|_1 = \max_j \|\boldsymbol{a}_j\|_1 = \max_j \left(\sum_{i=1}^{n} |a_{ij}|\right)$$

が成り立つので, 簡単に計算できる. 2–ノルム $\|A\|_2$ に対しては

$$\|A\|_2 = (\lambda \max(A^T A))^{\frac{1}{2}}$$

という式が成り立つ. $\lambda \max(A^T A)$ は $A^T A$ の最大固有値であり，A が対称なら

$$(\lambda \max(A^T A))^{\frac{1}{2}} = |\lambda \max(A)|$$

である. 一般に A の絶対値最大の固有値を $\rho(A)$ とすると

$$\rho(A) \leq \|A\|_2, \quad \rho(A) \leq \|A\|_1$$

↖ 問題あり!

である. $\rho(A)$ をスペクトル半径という.

ベクトルと行列のノルムは性質

$$\|\alpha \boldsymbol{x}\| = |\alpha| \cdot \|\boldsymbol{x}\|$$
$$\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$$
$$\|A\boldsymbol{x}\| \leq \|A\| \cdot \|\boldsymbol{x}\|$$

を満足する. また 1–ノルムと 2–ノルムは

$$\frac{1}{\sqrt{n}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$$

という関係を満足する.

さて, 式 (1.8) をコンピュータで解いて得られた数値解を $\hat{\boldsymbol{x}}$ とすると, 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

$$\hat{\boldsymbol{x}} - \boldsymbol{x}$$

を**誤差** (error) あるいは誤差ベクトルという. 誤差の程度は $\boldsymbol{x}$ 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

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

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

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

で評価すればよい.

実際に解 $\boldsymbol{x}$ がわかっていることは稀なので (わ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. もっともよく使われるのは

$$\boldsymbol{b} - A\hat{\boldsymbol{x}}$$

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, 相対残差ノルムを考える. 通常, ベクトルに対するノルムは 2–ノルムが使われる.

式 (1.8) をコンピュータで計算したとき, A は正しく表されたが $\boldsymbol{b}$ に何らかの誤差 $\delta\boldsymbol{b}$ 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. 式で書くと

$$A(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{b} + \delta\boldsymbol{b} \tag{1.9}$$

である.

式 (1.8),(1.9) から

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

となり, 両辺のノルムをとれば

$$\|\delta\boldsymbol{x}\| \leq \|A^{-1}\| \cdot \|\delta\boldsymbol{b}\| \tag{1.10}$$

である. 式 (1.8) の両辺のノルムをとれば

$$\|A\| \cdot \|\boldsymbol{x}\| \geq \|\boldsymbol{b}\| \tag{1.11}$$

なので, 式 (1.10),(1.11) を合わせると

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{1.12}$$

となる. $\|A\| \cdot \|A^{-1}\|$ のことを行列 A の**条件数** (condition number) といい, $\mathrm{cond}(A)$ と書く. $AA^{-1} = I$ より

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

である.

式 (1.12) は式 (1.9) 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. 式 (1.12) はコンピュータ上の数値表現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 1 次方程式を解けば

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \alpha\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \mathrm{cond}(A)\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

を満足する α が求められる. α は $\mathrm{cond}(A)$ に近くなることもある. $\mathrm{cond}(A)$ が大きいとき**悪条件方程式**, あるいは方程式の性質がよくないという. 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.

定常反復法

○ Jacobi法　　n本の方程式を独立に

$$x_i^{(k)} = \left\{ b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)} \right\} / a_{i,i}$$

○ Gauss-Seidel法　　iより前の $x_l^{(k)}$ を利用する

$$x_i^{(k)} = \left\{ b_i - \underbrace{\sum_{j<i} a_{ij} x_j^{(k)}}_{\text{この部分がより新しい値}} - \sum_{j>i} a_{ij} x_j^{(k-1)} \right\} / a_{i,i}$$

○ Successive Over-Relaxation法（逐次的過剰緩和法）

$$x_i^{(k)} = \omega \tilde{x}_i^{(k)} + (1-\omega) x_i^{(k-1)}$$

$$\tilde{x}_i = \left\{ b_i - \sum_{j<i} a_{ij} \tilde{x}_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)} \right\} / a_{i,i}$$

：Gauss-Seidel法の反復解

$x_i^{(k)}$ を直接更新するのでなく ω をパラメータとして更新量を調節

$0<\omega<1$：過小　　$\omega=1$：Gauss-Seidel法　　$1<\omega<2$ 過大