Chapter 3 連立1次方程式解法:反復法

Section 1: Jacobi & Gauss-Seidel法

- 反復法は、高次元と行列の零成分が多いときに効率である。
  Iterative methods are efficient when the dimension of the system is not too small and when there are many zero entries.
- 偏微分方程式解放において、零成分の多い行列に作り出されて、よく使かわれている。
  Discretization methods for partial differential equations produce such matrices so iterative methods are used a lot.
- Principle 原理：$Ax = b$. Initial value 初期値　$x^{(0)}$から作り出す数列　$\{x^{(k)}\}_{k=1}^{\infty}$, $\lim_{k\to\infty} |x^{(k)} - x| = 0$.

Jacobi's method ヤコビ法　(Easy !)

- $Ax = b$　から

  $$\sum_j a_{ij}x_j = b_i \Rightarrow x_i = \frac{b_i}{a_{ii}} - \sum_{j \neq i} \frac{a_{ij}x_j}{a_{ii}}, \qquad \text{if } a_{ii} \neq 0.$$

- From the (k-1)-th iteration $\boldsymbol{x}^{(k-1)}$ compute the k-th iteration $\boldsymbol{x}^{(k)}$ as follows:
  第k-1反復$\boldsymbol{x}^{(k-1)}$から第k反復$\boldsymbol{x}^{(k)}$を以下の公式に従い計算する：

$$x_i^{(k)} = \frac{b_i}{a_{ii}} - \sum_{j \neq i} \frac{a_{ij}x_j^{(k-1)}}{a_{ii}}$$

Jacobi's method in matrix form
$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}$　という行列の形のヤコビ法

- 反復法を記述するように、また解析するように、より一般な表現が使われていて便利である。
  Iterative methods are conveniently described by a more general description.
- 連立1次方程式 $A\,\boldsymbol{x} = \boldsymbol{b}$ を同値の方程式 $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$ に変換することである：(T行列、**c**ベクトル）。
  We transform the system Ax=b to an equivalent one x=Tx+c　with T a matrix and **c** a vector.
- 初期ベクトル $\boldsymbol{x}^{(0)}$ を選択したあと、反復を簡単に記述できる：
  $$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}$$
  Then the iteration consists of the above formula.

Jacobi's method in vector form
$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}$ というベクトル形のヤコビ法

- Let $D, -L, -U$ be the diagonal, the lower part and upper part of the matrix A.
  $D, -L, -U$を行列Aの対角分、下三角部分、上三角部分とする。

$$A = D - U - L$$

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix}$$

- $(D - U - L)\boldsymbol{x} = \boldsymbol{b}$ $\qquad D\boldsymbol{x} = (L + U)\boldsymbol{x} + \mathbf{b}$
- $\boldsymbol{x} = D^{-1}(L + U)\boldsymbol{x} + D^{-1}\boldsymbol{b}$ $\qquad T = D^{-1}(L + U)$

$\boldsymbol{c} = D^{-1}\mathbf{b}$, $\qquad \boxed{\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}}$

Jacobi法のアルゴリズム

- INPUT: n, a_{ij} Aの成分　　**b**の成分　b_i
  精度ϵ　　最大の反復回数　N.
  初期ベクトル　$\boldsymbol{x}^{(0)} = (y_1, y_2, \dots, y_n)$
- OUTPUT 解の近似ベクトル $x_1, x_2, \dots, x_n$
  or　「最大反復回数を超えた.」

➢ Step 1: $k = 1$

➢ Step 2: While ($k \le N$) do Steps 3-5

　➢ Step 3: For $i = 1, \dots, n$　do　　$x_i = \frac{b_i}{a_{ii}} - \sum_{j \ne i} \frac{a_{ij} y_j}{a_{ii}}$

　➢ Step 4: If $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty < \epsilon$ then OUTPUT $\boldsymbol{x}$; SUCCEED.

　➢ Step 5: $k = k + 1$;　For $i = 1, \dots, n$　do $y_i = x_i$

➢ Step 6: OUTPUT最大反復回数を超えた (FAIL)

Comments on the algorithm

- $\forall i,\ a_{ii} \neq 0$と想定した。
  もしAは非特異だったら　および　$a_{ii} = 0$を満たすiがあったら、行の置換を行うことによって$a_{ii} \neq 0$になる。
  If the matrix is nonsingular and if $a_{ii} = 0$ for some i, then a reordering of the lines permits to assume that there is no $a_{ii} = 0$.

- 精度に関するSTOP標準については、
  $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty < \epsilon$ の代わりに　$\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_\infty}{\|x\|_\infty} < \epsilon$も使われている。
  There is another STOP criterion for the tolerance ϵ which commonly used as well.

Gauss-Seidel 法の原理

- ヤコビ反復においての公式：

$$x_i^{(k)} = \frac{b_i}{a_{ii}} - \sum_{j \neq i} \frac{a_{ij} x_j^{(k-1)}}{a_{ii}}$$

- 手続きによると、$x_i^{(k)}$ を計算するときに、もう $x_j^{(k)}, j = 1, \ldots, i-1$ の成分を計算されているが、$x_j^{(k-1)}$ に比べて、より明確な近似である。
- According to the process, when we compute $x_i^{(k)}$ with the above formula, values $x_j^{(k)}, j = 1, \ldots, i-1$ have already been computed,
- and they are more precise than $x_j^{(k-1)}$

Gauss-Seidel法の反復

$$x_i^{(k)} = \frac{b_i}{a_{ii}} - \underbrace{\sum_{j>i} \frac{a_{ij} x_j^{(k-1)}}{a_{ii}}}_{} - \underbrace{\sum_{j<i} \frac{a_{ij} x_j^{(k)}}{a_{ii}}}_{}$$

Same as in
Jacobi's method
ヤコビ法と同じ

Use more recent
update
より新鮮な値を使う

- Jacobiアルゴリズム(ページ5)を整形する：
- Step3だけを変わる。
  - Step 3: For $i = 1, \dots, n$ do

$$x_i = \frac{b_i}{a_{ii}} - \sum_{j>i} \frac{a_{ij} y_j}{a_{ii}} - \sum_{j<i} \frac{x_j}{a_{ii}}$$

Example: comparison Jacobi vs Gauss-Seidel

- Test

$$\begin{aligned}
10x_1 - x_2 + 2x_3 &= 6 \\
-x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\
2x_1 - x_2 + 10x_3 - x_4 &= -11 \\
3x_2 - x_3 + 8x_4 &= 15
\end{aligned}$$

- $x^{(0)} = (0,0,0,0)^T$

- Jacobi

k	0	1	2	3	4		10
$x_1^{(k)}$	0.0	0.6000	1.0473	0.9326	1.0152	$\cdots$	1.0001
$x_2^{(k)}$	0.0	2.2727	1.7159	2.053	1.9537	$\cdots$	1.9998
$x_3^{(k)}$	0.0	−1.1000	−0.8052	−1.0493	−0.9681	$\cdots$	−0.9998
$x_4^{(k)}$	0.0	1.8750	0.8852	1.1309	0.9739	$\cdots$	0.9998

- Gauss-Seidel

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	−0.9873	−1.014	−1.0025	−1.0003	−1.0000
$x_4^{(k)}$	0.0000	0.8789	0.984	0.9983	0.9999	1.0000

Gauss-Seidel's method in matrix form
$\boldsymbol{x}^{(k)} = T_g \boldsymbol{x}^{(k-1)} + \boldsymbol{c}_g$　という行列の形のヤコビ法

$$A = D - U - L$$

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix}$$

Gauss-Seidel: $(D - L)\boldsymbol{x}^{(k)} = U\boldsymbol{x}^{(k-1)} + \boldsymbol{b}$ が成り立つ。

- $\boldsymbol{x}^{(k)} = (D - L)^{-1} U \, \boldsymbol{x}^{(k-1)} + (D - L)^{-1} \boldsymbol{b}$
- $T_g = (D - L)^{-1} U$ and $\boldsymbol{c}_g = (D - L)^{-1} \boldsymbol{b}$ とおくと；

$$\boldsymbol{x}^{(k)} = T_g \boldsymbol{x}^{(k-1)} + \boldsymbol{c}_g$$