反復法による連立一次方程式の解法

次元の呪縛

連立一次方程式の標準的な数値解法であるガウスの消去法では，計算量は未知数の個数の 3 乗に比例する。したがって，空間が一次元の問題を N 等分して解けば，計算量は N^3 に比例して増えていくことになる。これに対して，二次元問題，三次元問題では，各次元を N 分割すれば，それぞれ N^6 と N^9 に比例して計算量が増えていくことになる。

いま，分割を細かくして精度を 10 進で 1 桁向上させることを考える。通常の差分法では誤差は Δx^2 に比例して小さくなっていくので，Δx を $1/\sqrt{10}$ 倍しなければならない。すなわち一つの次元あたり分割数を $\sqrt{10}$ 倍しなければならない。そうすると，一次元のときは $(\sqrt{10})^3 \approx 31.6$ 倍，二次元のときは $(\sqrt{10})^6 = 1000$ 倍，三次元のときは $(\sqrt{10})^9 \approx 31600$ 倍の計算量を要することになり，二次元，三次元の問題は手に負えなくなってくる。これを **次元の呪縛** と呼んでいる。一方，記憶容量の方は未知数の個数の 2 乗に比例して増えていくので，10 進で 1 桁の精度を向上させるには，一次元では 10 倍，二次元では 100 倍，三次元では 1000 倍になり，それほど悲観的でもない。

この次元の呪縛を回避するため，計算量および記憶容量が，未知数の個数を m としたとき，$\mathrm{O}(m^2)$ や $\mathrm{O}(m^3)$ 以下の速さで増えていくアルゴリズムが望まれる。それらの初歩的なものを紹介する。

連立一次方程式の反復解法

まず，次の連立一次方程式を考える：

$$\begin{aligned} 3\,x + 2\,y + z &= 4 \\ x + 3\,y - 2\,z &= 6 \\ 2\,x - y + 4\,z &= -3 \end{aligned} \tag{1}$$

この方程式に対して以下の 3 つの反復法を適用すると次のようになる。

1. ヤコビ法

$$\begin{aligned} x^{(p+1)} &= \frac{1}{3}\,(4 - 2y^{(p)} - z^{(p)}), \\ y^{(p+1)} &= \frac{1}{3}\,(6 - x^{(p)} + 2z^{(p)}), \qquad p = 0, 1, \ldots, \\ z^{(p+1)} &= \frac{1}{4}\,(-3 - 2x^{(p)} + y^{(p)}) \end{aligned} \tag{2}$$

2. ガウス・ザイデル法

$$\begin{aligned} x^{(p+1)} &= \frac{1}{3}\,(4 - 2y^{(p)} - z^{(p)}), \\ y^{(p+1)} &= \frac{1}{3}\,(6 - x^{(p+1)} + 2z^{(p)}), \qquad p = 0, 1, \ldots, \\ z^{(p+1)} &= \frac{1}{4}\,(-3 - 2x^{(p+1)} + y^{(p+1)}), \end{aligned} \tag{3}$$

3. SOR 法

$p = 0, 1, \ldots$ について

$$\begin{aligned}
\bar{x} &= \frac{1}{3}\,(4 - 2\,y^{(p)} - z^{(p)}),\\
\bar{y} &= \frac{1}{3}\,(6 - \bar{x} + 2\,z^{(p)}),\\
\bar{z} &= \frac{1}{4}\,(-3 - 2\,\bar{x} + \bar{y})
\end{aligned}$$

$$\begin{aligned}
x^{(p+1)} &= x^{(p)} + \omega\,(\bar{x} - x^{(p)}),\\
y^{(p+1)} &= y^{(p)} + \omega\,(\bar{y} - y^{(p)}),\\
z^{(p+1)} &= z^{(p)} + \omega\,(\bar{z} - z^{(p)})
\end{aligned} \tag{4}$$

次に $x^{(0)} = y^{(0)} = z^{(0)} = 0$ から出発したときの計算結果を示す。

1. ヤコビ法

```
 1: Program Jacobi
 2:   implicit none
 3:   real (8) :: x0=0.,y0=0.,z0=0.,x1,y1,z1
 4:   integer :: p=0
 5:
 6:   write (*,'(i3,1pd13.5,1pd13.5,1pd13.5)') p,x0,y0,z0
 7:   do p=1,30
 8:      x1=(4.d0-2.d0*y0-z0)/3.d0
 9:      y1=(6.d0-x0+2.d0*z0)/3.d0
10:      z1=(-3.d0-2.d0*x0+y0)/4.d0
11:      write (*,'(i3,1pd13.5,1pd13.5,1pd13.5)') p,x1,y1,z1
12:      x0=x1
13:      y0=y1
14:      z0=z1
15:   enddo
16:
17: end Program Jacobi
```

計算結果

p	$x^{(p)}$	$y^{(p)}$	$z^{(p)}$
0	0.00000d+00	0.00000d+00	0.00000d+00
1	1.33333d+00	2.00000d+00	-7.50000d-01
2	2.50000d-01	1.05556d+00	-9.16667d-01
	...	...	...
8	8.32240d-01	1.17504d+00	-8.53188d-01
9	8.34372d-01	1.15379d+00	-8.72361d-01
10	8.54924d-01	1.14030d+00	-8.78738d-01
	...	...	...
18	9.30372d-01	1.06688d+00	-9.43441d-01
19	9.36558d-01	1.06092d+00	-9.48465d-01
20	9.42211d-01	1.05550d+00	-9.53050d-01
	...	...	...
28	9.72577d-01	1.02634d+00	-9.77722d-01
29	9.75016d-01	1.02399d+00	-9.79704d-01
30	9.77239d-01	1.02186d+00	-9.81510d-01

2. ガウス・ザイデル法

```
 1: Program Gauss_Seidel
 2:   implicit none
 3:   real (8) :: x=0.,y=0.,z=0.
 4:   integer :: p=0
 5:
 6:   write (*,'(i3,1pd13.5,1pd13.5,1pd13.5)') p,x,y,z
 7:   do p=1,30
 8:      x=(4.d0-2.d0*y-z)/3.d0
 9:      y=(6.d0-x+2.d0*z)/3.d0
10:      z=(-3.d0-2.d0*x+y)/4.d0
11:      write (*,'(i3,1pd13.5,1pd13.5,1pd13.5)') p,x,y,z
12:   enddo
13:
14: end Program Gauss_Seidel
```

計算結果

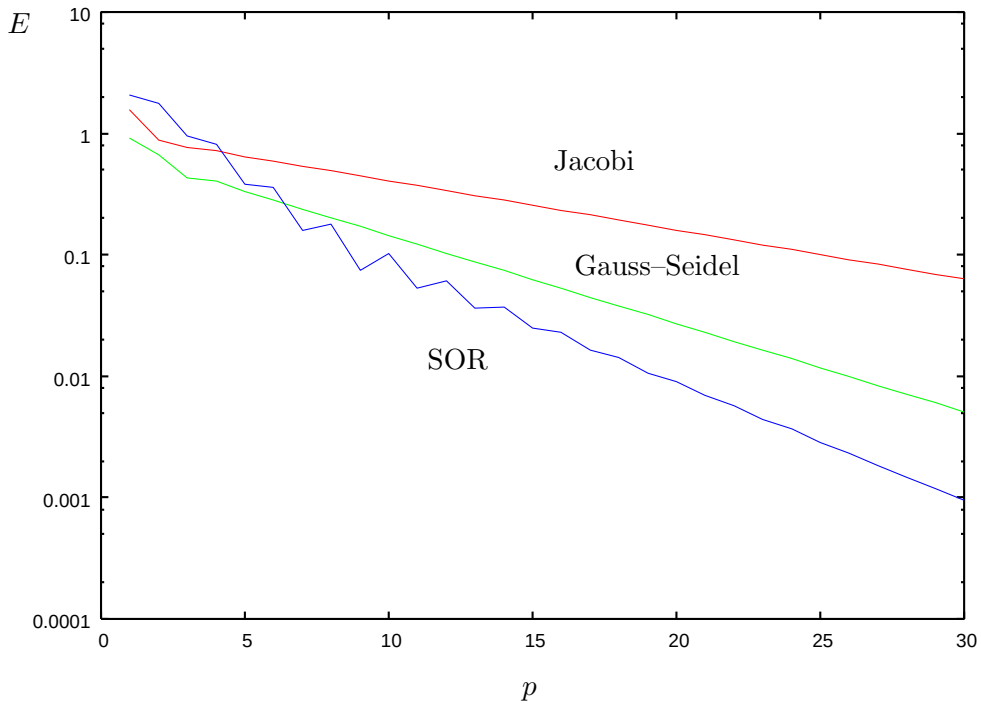
p	$x^{(p)}$	$y^{(p)}$	$z^{(p)}$
0	0.00000d+00	0.00000d+00	0.00000d+00
1	1.33333d+00	1.55556d+00	-1.02778d+00
2	6.38889d-01	1.10185d+00	-7.93981d-01
	...	...	...
8	9.23417d-01	1.06929d+00	-9.44385d-01
9	9.35266d-01	1.05865d+00	-9.52969d-01
10	9.45220d-01	1.04961d+00	-9.60206d-01
	...	...	...
18	9.85626d-01	1.01302d+00	-9.89558d-01
19	9.87839d-01	1.01101d+00	-9.91166d-01
20	9.89712d-01	1.00932d+00	-9.92526d-01
	...	...	...
28	9.97300d-01	1.00245d+00	-9.98039d-01
29	9.97716d-01	1.00207d+00	-9.98341d-01
30	9.98068d-01	1.00175d+00	-9.98596d-01

3. SOR 法 ($\omega = 1.3$)

```
 1: Program SOR
 2:   implicit none
 3:   real (8) :: x0=0.,y0=0.,z0=0.,x1,y1,z1,omega=1.3
 4:   integer :: p=0
 5:
 6:   write (*,'(i3,1pd13.5,1pd13.5,1pd13.5,)') p,x0,y0,z0
 7:
 8:   do p=1,30
 9:      x1=(4.d0-2.d0*y0-z0)/3.d0
10:      y1=(6.d0-x1+2.d0*z0)/3.d0
11:      z1=(-3.d0-2.d0*x1+y1)/4.d0
12:
13:      x0=(1.d0-omega)*x0+omega*x1
14:      y0=(1.d0-omega)*y0+omega*y1
15:      z0=(1.d0-omega)*z0+omega*z1
16:      write (*,'(i3,1pd13.5,1pd13.5,1pd13.5,)') p,x0,y0,z0
17:   enddo
18:
19: end Program SOR
```

計算結果

p	$x^{(p)}$	$y^{(p)}$	$z^{(p)}$
0	0.00000d+00	0.00000d+00	0.00000d+00
1	1.73333d+00	1.84889d+00	-1.50078d+00
2	2.61300d-01	6.31429d-01	-4.89397d-01
	...	...	...
8	9.16188d-01	9.67318d-01	-9.42370d-01
9	1.02849d+00	1.04740d+00	-1.02040d+00
10	9.59211d-01	9.85770d-01	-9.71991d-01
	...	...	...
18	9.97630d-01	9.99619d-01	-9.98382d-01
19	1.00034d+00	1.00137d+00	-1.00026d+00
20	9.98825d-01	9.99872d-01	-9.99199d-01
	...	...	...
28	9.99926d-01	1.00001d+00	-9.99950d-01
29	9.99994d-01	1.00004d+00	-9.99997d-01
30	9.99963d-01	1.00001d+00	-9.99975d-01



三つの反復法と誤差 $E = |x^{(p)} - 1| + |y^{(p)} - 1| + |z^{(p)} + 1|$ の変化

問題 Laplace 方程式

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad (x, y) \in (0,1) \times (0,1),$$
$$u(x,0) = \sin(\pi x), \qquad u(x,1) = 0,$$
$$u(0,y) = \sin(\pi y), \qquad u(1,y) = 0,$$

を差分化し得られた連立一次方程式を SOR 法で解き，解を図示せよ (下図参照)。

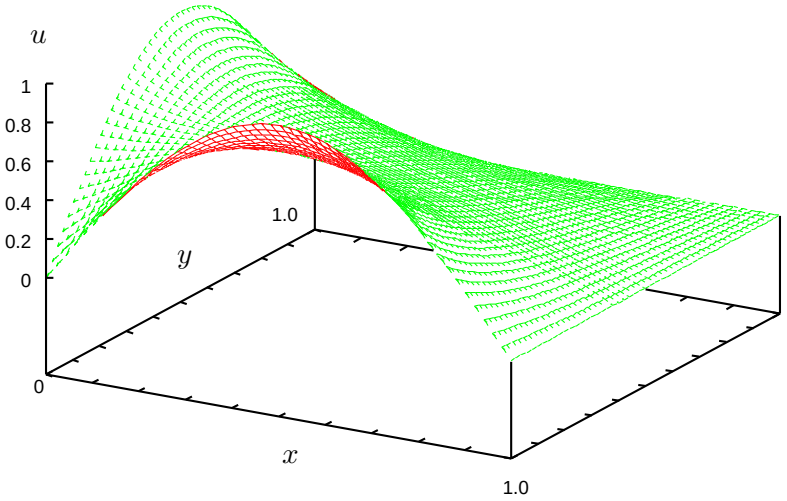


図 1. x, y 方向それぞれ 50 等分した場合の数値解

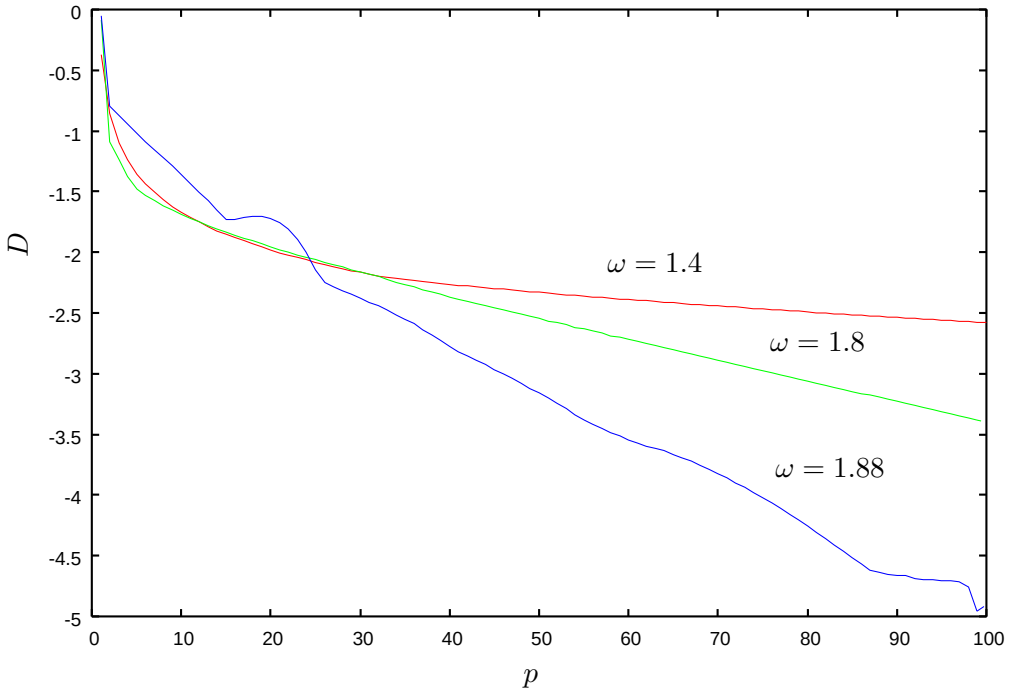


図 2. SOR 法の収束 (ただし，$D = \log_{10} \max_{i,j} |u_{i,j}^{(p)} - u_{i,j}^{(p+1)}|$)

Thomas のアルゴリズム

次のような n 元連立一次方程式を考える：

$$\begin{pmatrix} b_1 & -c_1 & & & & \\ -a_2 & b_2 & -c_2 & & & \\ & -a_3 & b_3 & -c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & & & -a_n & b_n \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ \vdots \\ U_{n-1} \\ U_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix} \tag{5}$$

この方程式を少ない計算量で解くアルゴリズムとして，トーマスのアルゴリズム (Thomas' algorithm) が有名である。以下，このアルゴリズムを簡単に解説する。

まず，式 (5) の一番上の式は

$$U_1 - e_1 U_2 = f_1 \tag{6}$$

と変形できる。ここで

$$e_1 = \frac{c_1}{b_1}, \qquad f_1 = \frac{d_1}{b_1}$$

である。これを二番目の式に代入すると

$$-a_2 (e_1 U_2 + f_1) + b_2 U_2 - c_2 U_3 = d_2$$

となる。これをさらに変形すると

$$U_2 - e_2 U_3 = f_2 \tag{7}$$

を得る。ここで

$$e_2 = \frac{c_2}{b_2 - a_2 e_1}, \qquad f_2 = \frac{d_2 + a_2 f_1}{b_2 - a_2 e_1}$$

である。次に，式 (7) を式 (5) の三番目に代入して

$$U_3 - e_3 U_4 = f_3 \tag{8}$$

を得る。ここで

$$e_3 = \frac{c_3}{b_3 - a_3 e_2}, \qquad f_3 = \frac{d_3 + a_3 f_2}{b_3 - a_3 e_2}$$

である。以下，同様の式変形を繰り返し，$(n-1)$ 番目の式まで変形して行く。そうして得られた $(n-1)$ 番目の式

$$U_{n-1} - e_{n-1} U_n = f_{n-1} \tag{9}$$

と，式 (5) の一番下の式

$$-a_n U_{n-1} + b_n U_n = d_n \tag{10}$$

より U_n を求めると

$$U_n = \frac{d_n + a_n f_{n-1}}{b_n - a_n e_{n-1}} \tag{11}$$

となる。

これを式 (9) に代入すると U_{n-1} が求まり，その値を変形後のすぐ上の式に代入すると U_{n-2} が求まり，$\cdots$，という順に計算していけば，最後に U_1 までが求まる。

以上をアルゴリズムとしてまとめると次のようになる：

Thomas のアルゴリズム

```
begin
  e_0 := 0;  f_0 := 0;
  for  k :=1  to n do
    begin
      e_k := c_k/(b_k − a_k e_{k−1});
      f_k := (d_k + a_k f_{k−1})/(b_k − a_k e_{k−1})
    end;
  U_n := f_n;
  for  k := n − 1 downto 1 do
    U_k := e_k U_{k+1} + f_k
end.
```

問 式 (1) において，$n = 100$〜1000 の範囲で

$$a_i = c_i = 1, \quad b_i = 4$$
$$d_1 = d_n = 3, \quad d_i = 2 \ (i = 2, \ldots, n-1)$$

とおいた連立一次方程式を Thomas のアルゴリズムで解け (正確な答は $U_i = 1 \, (i = 1, 2, \ldots, n)$)。

参考文献

[1] K.W. Morton & D.F. Mayers, Numerical Solution of Partial Differential Equations, Cambridg University Press, 1994.