行列の反復解法

１．点Jacobi法

数値解法の重要な概念の一つである反復法を取り上げ、連立一次方程式Au=bの反復解法を調べる。

行列のスペクトル半径と収束行列の定義を与える。

行列のスペクトル半径
行列Aの固有値の絶対値の最大値でもって、行列Aのスペクトル半径r(A)を与える。

$$\rho(A)=\max\left|\lambda_i(A)\right|$$

収束行列
Bが正方行列で、

$$\lim_{k\to\infty} B^k = 0$$

のとき、Bを収束行列と呼ぶ。

定理
収束行列のスペクトル半径は、

$$\rho(B)<1$$

である。

簡単な証明
もし、ρ(B)≥1ならば、固有ベクトルp≠0で
Bp=λp, $|\lambda| \geq 1$
となるpがある。このとき、B^kp=λ^kpだから、ベクトル列{B^kp}は0に収束しない。

行列の分離と反復解法

行列Aを分離(regular splitting)することを考える。分離方法には様々な方法が考えられるが、今、A=S-Tと分離する。

Au=b
(S-T)u=b
Su=Tu+b

そこで、初期値u0として、反復m回目の計算を構成できる。

$$Su_m = Tu_{m-1} + b$$

$$u_m = S^{-1}Tu_{m-1} + S^{-1}b$$

列 $\{u_m\}$が反復計算となる条件を満たし、反復解法のアルゴリズムを構成するためには、

条件１　新しいu_mが容易に計算される。したがって、S^{-1}が簡単に計算できる。
条件２　列 $\{u_m\}$は連立一次方程式Au＝bの真の解uに収束しなければならない。

条件２を満たすためには、B=$S^{-1}T$が収束行列でなければならない。
実際、
$\epsilon_m = u_m - u$とおくと、

$$B\epsilon_{m-1} = Bu_{m-1} - Bu$$

$$= u_m - S^{-1}b - Bu$$

$$= u_m - u$$
$$= \epsilon_m$$

よって、

$$\epsilon_m = B\epsilon_{m-1} = \cdots = B^m \epsilon_0$$

m−>∞、

$B^m \to 0$ 、すなわち、Bが収束行列ならば、列 $\{u_m\}$は連立一次方程式Au＝bの真の解uに収束する。

分離の方法

点Jacobi行列

A=D−E−Fとし、

D:対角行列
E:狭義下三角行列
F:狭義上三角行列

とAを分離することができる。
このとき、

$$u_m = \mathrm{D}^{-1}(E+F)u_{m-1} + \mathrm{D}^{-1}b$$
$$\mathrm{B} = \mathrm{D}^{-1}(E+F)$$

Bを点Jacobi行列と呼ぶ。

条件１　D^{-1}は簡単に計算できる。
条件２　Bは収束行列

点Jacobi法のコード

```
> n:=10:
> A:=Matrix(1..n,1..n):
> b:=Vector([1,1,1,1,1,1,1,1,1,1]):
> for i from 1 to n-1 do:
    A[i,i+1]:=-1:
    A[i+1,i]:=-1:
  end do:
> for i from 1 to n do:
    A[i,i]:=2:
  end do:
> A;
```

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \quad (1)$$

```
> E:=Matrix(n,n,shape=identity):
> DD:=2*E:
> IDD:=(1/2)*E:
```

Aの対角行列DDの逆行列は容易に計算できて、IDDと定義できる。
このとき、点Jacobi行列は

```
> B:=IDD.(DD-A);
```

$$B := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \quad (2)$$

と計算できる。

<u>点Jacobi法のコード</u>

```
> N:=200:
> m:=1:
```

```
> eps:=0.001:
> err:=1.0:
> u:=Vector([1,1,1,1,1,1,1,1,1,1]):
> ER:=Array(1..N-1):
> x:=Array(1..N-1):
> while ( err>eps and m<N ) do:
    um:=u:
    u:=B.u+IDD.b:
    err:=(u-um).(u-um):
    err:=evalf(sqrt(err),7):
    ER[m]:=err:
    x[m]:=m:
    m:=m+1:
  end do:
> evalf(u,7);
```

$$\begin{bmatrix} 4.99720800000000 \\ 8.99464300000000 \\ 11.9925100000000 \\ 13.9909900000000 \\ 14.9901900000000 \\ 14.9901900000000 \\ 13.9909900000000 \\ 11.9925100000000 \\ 8.99464300000000 \\ 4.99720800000000 \end{bmatrix} \tag{3}$$

```
> evalf(err,7);
```

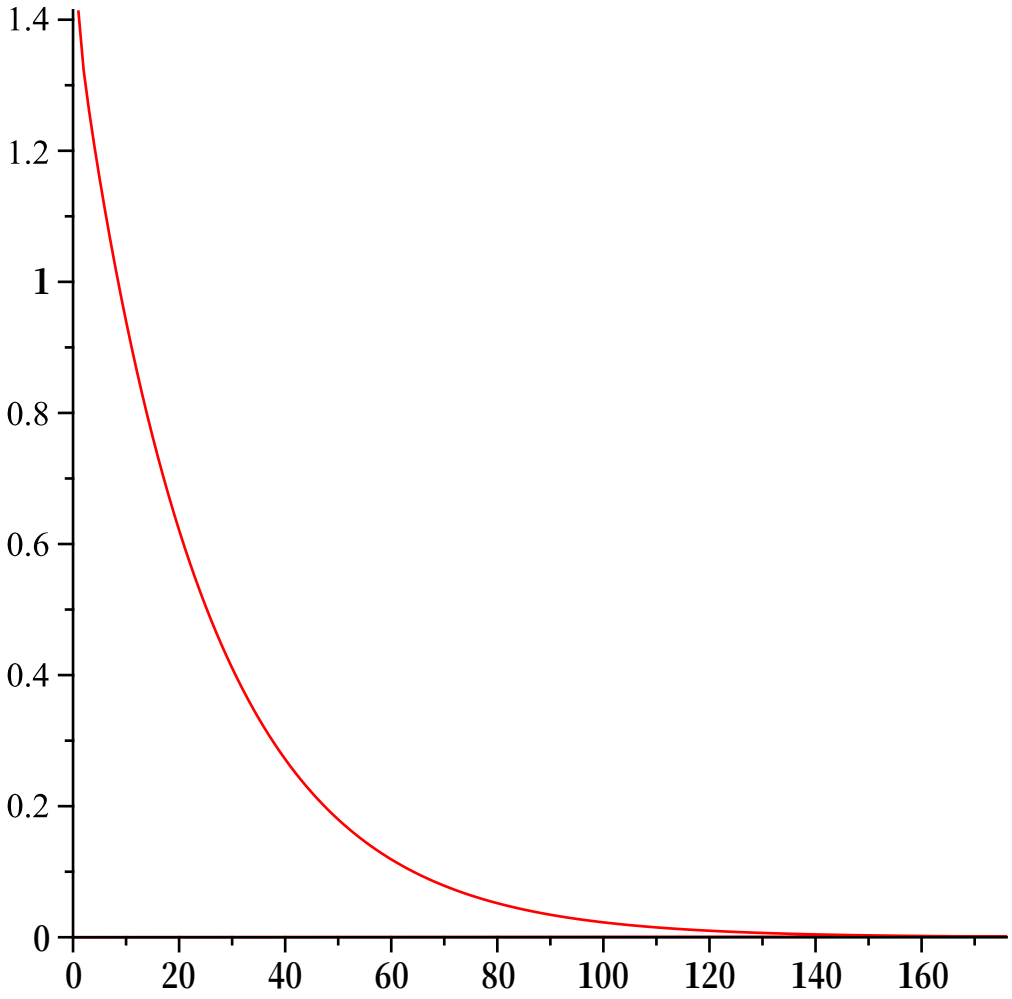
$$0.0009811129 \tag{4}$$

```
> m-1;
```

$$176 \tag{5}$$

点Jacobi法の収束のふるまい

```
> plot(x,ER);
```



2. 点Gauss-Siedel法

Au=bの反復解法をさらに調べ、そのアルゴリズムの改善をはかる。

<u>行列の分離と反復解法</u>

行列Aを分離(regular splitting)することを考える。分離方法には様々な方法が考えられるが、今、A=S-Tと分離する。

Au=b
(S-T)u=b
Su=Tu+b

そこで、初期値u0として、反復m回目の計算を構成できる。

$$Su_m = Tu_{m-1} + b$$

$$u_m = S^{-1}Tu_{m-1} + S^{-1}b$$

列 $\{u_m\}$ が反復計算となる条件を満たし、反復解法のアルゴリズムを構成するためには、

条件１　新しいu_mが容易に計算される。したがって、S^{-1}が簡単に計算できる。
条件２　列 $\{u_m\}$ は連立一次方程式Au＝bの真の解uに収束しなければならない。

分離の方法

点Gauus-Siedel行列

A=D−E−Fとし、
D:対角行列
E:狭義下三角行列
F:狭義上三角行列
とAを分離することができる。
このとき、

$$(\mathrm{D}-E)u_m = Fu_{m-1} + b$$

$$u_m = (\mathrm{D}-E)^{-1}Fu_{m-1} + (\mathrm{D}-E)^{-1}b$$

$$\mathrm{C} = (\mathrm{D}-E)^{-1}F$$

Cを点Gauss-Siedel行列と呼ぶ。

条件１　$(\mathrm{D}-E)^{-1}$は簡単に計算できる。
条件２　Cは収束行列

アルゴリズムの改善点
4×4行列を例にとって、　$Du_m = Eu_m + Fu_{m-1} + b$を成分ごとに書くと、

$$\begin{aligned}
a_{11}u1^m &= -a_{12}u2^{m-1} - a_{13}u3^{m-1} - a_{14}u4^{m-1} + b_1 \\
a_{22}u2^m &= -a_{21}u1^m - a_{23}u3^{m-1} - a_{24}u4^{m-1} + b_2 \\
a_{33}u3^m &= -a_{31}u1^m - a_{32}u2^m - a_{34}u4^{m-1} + b_3 \\
a_{44}u4^m &= -a_{41}u1^m - a_{42}u2^m - a_{43}u3^m + b_4
\end{aligned}$$

であり、成分
u_iの新しい推定値が求まると、その後の計算ではすべてそれを用いていることに注意する。

行列　A=D−E−Fの設定

```
> n:=10:
> A:=Matrix(1..n,1..n):
> AE:=Matrix(1..n,1..n):
> AF:=Matrix(1..n,1..n):
> b:=Vector([1,1,1,1,1,1,1,1,1,1]):
> for i from 1 to n-1 do:
    A[i,i+1]:=-1:
    AF[i,i+1]:=1:
    A[i+1,i]:=-1:
    AE[i+1,i]:=1:
  end do:
> for i from 1 to n do:
    A[i,i]:=2:
  end do:
> A;
```

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{6}$$

```
> AF;AE;
```

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{7}$$

```
>
```

点Gauss-Siedel法のコード

```
> EE:=Matrix(n,n,shape=identity):
> DD:=2*EE:
```

DD-AEは下三角行列だから、逆行列は容易に計算できるが、ここでは、MatrixInverseコマ

ンドを使う。

```
> with(LinearAlgebra):
> IDE:=MatrixInverse(DD-AE);
```

$$IDE := \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ \frac{1}{512} & \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ \frac{1}{1024} & \frac{1}{512} & \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \quad (8)$$

このとき、点Gauss-Siedel行列は

```
> C:=IDE.AF;
```

(9)

$$C := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{512} & \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{1024} & \frac{1}{512} & \frac{1}{256} & \frac{1}{128} & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} \end{bmatrix} \quad (9)$$

と計算できる。

```
>
> N:=100:
> m:=1:
> eps:=0.001:
> err:=1.0:
> u:=Vector([1,1,1,1,1,1,1,1,1,1]):
> ER:=Array(1..N-1):
> x:=Array(1..N-1):
> while ( err>eps and m<N ) do:
    um:=u:
    u:=C.u+IDE.b:
    err:=(u-um).(u-um):
    err:=evalf(sqrt(err),7):
    ER[m]:=err:
    x[m]:=m:
    m:=m+1:
  end do:
> evalf(u,7);
```

(10)

$$\begin{bmatrix} 4.998363 \\ 8.996987 \\ 11.99596 \\ 13.99533 \\ 14.99513 \\ 14.99532 \\ 13.99588 \\ 11.99671 \\ 8.997744 \\ 4.998872 \end{bmatrix} \quad (10)$$

```
> evalf(err,7);
```

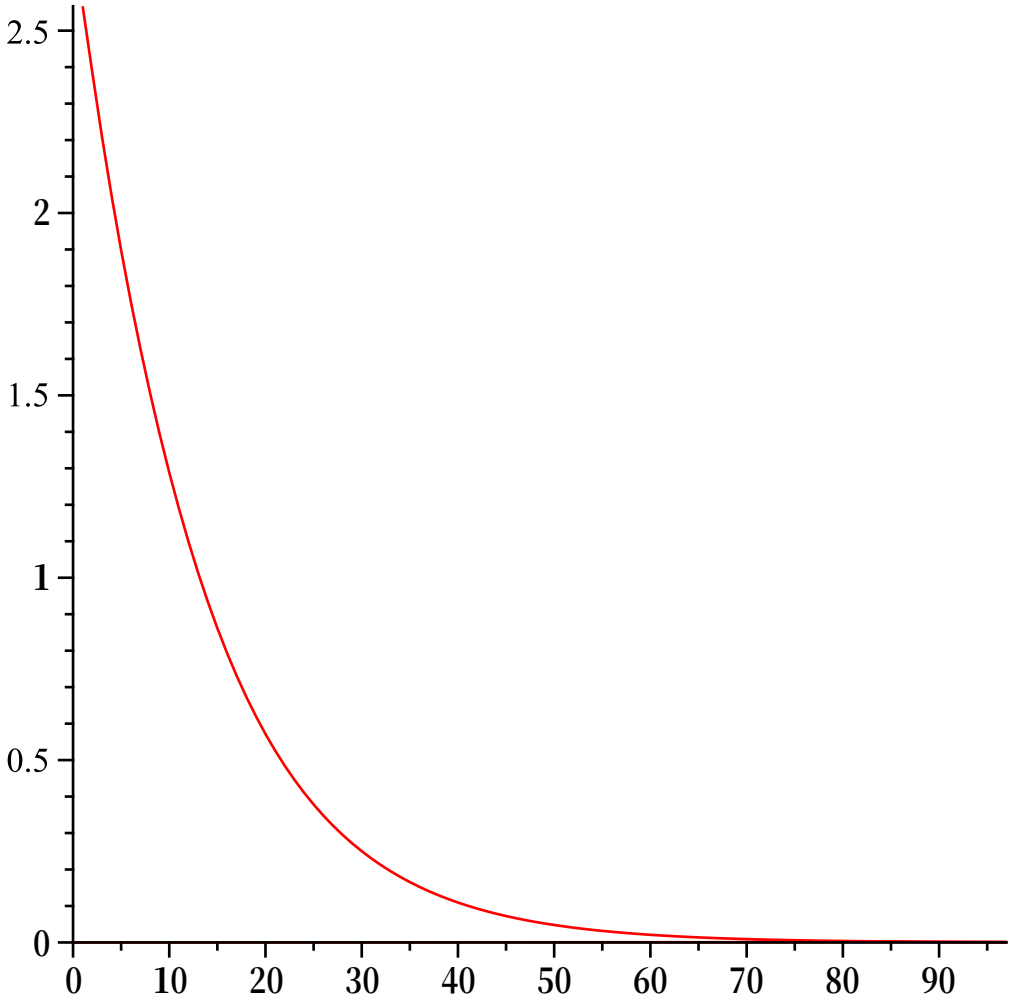
$$0.0009816868 \quad (11)$$

```
> m-1;
```

$$97 \quad (12)$$

点Gauss-Siedel法の収束のふるまい

```
>
> plot(x,ER);
```



3. SOR法

連立一次方程式Au=bの反復解法を構成した。その中で、点Gauss-Siedel法は点Jacobi法よりも収束性が向上していることがわかった。その結果を基に、アルゴリズムをさらに改善してみる。

行列の分離と反復解法

反復解法は、行列Aを分離(regular splitting)して構成した。分離方法には様々な方法が考えられるが、今、A=S-Tと分離する。

$$Au=b$$
$$(S-T)u=b$$
$$Su=Tu+b$$

そこで、初期値u0として、反復m回目の計算を構成できる。

$$Su_m = Tu_{m-1} + b$$
$$u_m = S^{-1}Tu_{m-1} + S^{-1}b$$

列 $\{u_m\}$が反復計算となる条件を満たし、反復解法のアルゴリズムを構成するためには**以下の条件を満たすことが必要である。**

条件１　**新しい**u_m**が容易に計算される。したがって、**S^{-1}が簡単に計算できる。
条件２　**列** $\{u_m\}$は連立一次方程式Au＝bの真の解uに収束しなければならない。

分離の方法

点Gauus-Siedel行列

A=D−E−Fとし、
D:対角行列
E:狭義下三角行列
F:狭義上三角行列
とAを分離することができる。
このとき、

$$(D-E)u_m = Fu_{m-1} + b$$
$$u_m = (D-E)^{-1}Fu_{m-1} + (D-E)^{-1}b$$
$$C=(D-E)^{-1}F$$

Cを点Gauss-Siedel行列と呼ぶ。

条件１　$(D-E)^{-1}$は簡単に計算できる。
条件２　Cは収束行列

アルゴリズムの改善点

4×4行列を例にとって、 $Du_m = Eu_m + Fu_{m-1} + b$を成分ごとに書くと、

$$a_{11}u1^m = -a_{12}u2^{m-1} - a_{13}u3^{m-1} - a_{14}u4^{m-1} + b_1$$
$$a_{22}u2^m = -a_{21}u1^m - a_{23}u3^{m-1} - a_{24}u4^{m-1} + b_2$$
$$a_{33}u3^m = -a_{31}u1^m - a_{32}u2^m - a_{34}u4^{m-1} + b_3$$
$$a_{44}u4^m = -a_{41}u1^m - a_{42}u2^m - a_{43}u3^m + b_4$$

であり、

u_iの新しい推定値が求まると、その後の計算ではすべてそれを用いることによって
点Jacobi法よりも収束性の向上が図られて**いる**。

SOR反復法(Successive Overrelaxation Iterative Method)

点Gauss-Siedel法で新しく計算された成分に加速パラメータwを乗じて、修正量の効果を大きく補正し、新しい反復解を構成する。すなわち、反復ベクトルを

$$u_m = u_{m-1} + \omega\left(\bar{u}_m - u_{m-1}\right)$$
$$= (1-\omega)u_{m-1} + \omega\cdot\bar{u}_m$$

として求める。
ω＞1 のとき、新しく計算された成分の修正量の効果が大きく、反復を加速するという。
ω＜1 **のとき、新しく**計算された成分の修正量の効果が小さく、反復を減速するという。
Aを4×4行列とし成分ごとに書いて、このことを説明する。

$$a_{11}\bar{u1}^m = -a_{12}u2^{m-1} - a_{13}u3^{m-1} - a_{14}u4^{m-1} + b_1$$
$$u1^m = (1-\omega)u1^{m-1} + \omega\cdot\bar{u1}^m$$
$$a_{22}\bar{u2}^m = -a_{21}u1^m - a_{23}u3^{m-1} - a_{24}u4^{m-1} + b_2$$
$$u2^m = (1-\omega)u2^{m-1} + \omega\cdot\bar{u2}^m$$
$$a_{33}\bar{u3}^m = -a_{31}u1^m - a_{32}u2^m - a_{34}u4^{m-1} + b_3$$
$$u3^m = (1-\omega)u3^{m-1} + \omega\cdot\bar{u3}^m$$
$$a_{44}\bar{u4}^m = -a_{41}u1^m - a_{42}u2^m - a_{43}u3^m + b_4$$
$$u4^m = (1-\omega)u4^{m-1} + \omega\cdot\bar{u4}^m$$

であり、補正量をwによって調整する。
これを行列で書くと、

$$D\bar{u}_m = Eu_m + Fu_{m-1} + b$$
$$u_m = (1-\omega)u_{m-1} + \omega\cdot\bar{u}_m$$

である。
$\bar{u}_m$**を消去すれば、**

$$Du_m = (1-\omega)Du_{m-1} + \omega\cdot D\bar{u}_m$$
$$= (1-\omega)Du_{m-1} + \omega\cdot\left(Eu_m + Fu_{m-1} + b\right)$$

よって、

$$(D - \omega E)u_m = ((1-\omega)D + \omega F)u_{m-1} + \omega\cdot b$$
$$u_m = (D - \omega E)^{-1}((1-\omega)D + \omega F)u_{m-1} + \omega\cdot(D - \omega E)^{-1}b$$

行列

$$SOR = (D - \omega E)^{-1}((1-\omega)D + \omega F)$$

が得られ、この行列を点SOR行列と呼ぶ。点SOR行列において
ω＝1 のとすれば、点Gauss－Siedel行列となる。 **以上をまとめ、SOR法のコードを作成する。**

SOR法のコード

```
> n:=10:
> A:=Matrix(1..n,1..n):
> AE:=Matrix(1..n,1..n):
> AF:=Matrix(1..n,1..n):
> b:=Vector(1..n):
> for i from 1 to n-1 do:
```

```
      A[i,i+1]:=-1.0:
      AF[i,i+1]:=1.0:
      A[i+1,i]:=-1.0:
      AE[i+1,i]:=1.0:
   end do:
> for i from 1 to n do:
      A[i,i]:=2.0:
      b[i]:=1.0:
   end do:
> AF:AE:
> EE:=Matrix(n,n,shape=identity):
> DD:=2.0*EE:
```

(13)

DD-ω·AEは下三角行列だから、逆行列は容易に計算でき、

```
> OMEGA:=1.5:
> with(LinearAlgebra):
> IDE:=MatrixInverse(DD-OMEGA*AE):
```

このとき、点SOR行列は

```
> SOR:=IDE.((1.0-OMEGA)*DD+OMEGA*AF):
```

と計算できる。

課題9.1

行列SORは、0<ω<2.0のとき収束行列であることを数値実験で調べよ。

コード

```
> N:=100:
> m:=1:
> eps:=0.001:
> err:=1.0:
> u:=Vector(1..n):
> for i from 1 to n do:
      u[i]:=1.0:
   end do:
> ER:=Array(1..N-1):
> x:=Array(1..N-1):
> while ( err>eps and m<N ) do:
    um:=u:
    u:=SOR.u+OMEGA*IDE.b:
    err:=(u-um).(u-um):
    err:=evalf(sqrt(err),7):
    ER[m]:=err:
    x[m]:=m:
    m:=m+1:
   end do:
> evalf(u,7);
```

(14)

$$\begin{bmatrix} 4.99951900000000 \\ 8.99921200000000 \\ 11.9990600000000 \\ 13.9990300000000 \\ 14.9991000000000 \\ 14.9992400000000 \\ 13.9994000000000 \\ 11.9995700000000 \\ 8.99974100000000 \\ 4.99988500000000 \end{bmatrix} \quad (14)$$

```
> evalf(err,7);
```

$$0.0008080718 \quad (15)$$

```
> m-1;
```

$$32 \quad (16)$$

```
> plot(x,ER);
```