連立非線形方程式に対するニュートン法

アギレ・エルナン

信州大学　電子情報システム工学部

数値計算

2023 年 5 月 8 日

LU分解による
連立一次方程式の解

LU 分解による連立一次方程式の解法

やるべきこと

- A 行列を LU 分解する $\rightarrow$ 既に書いたソースコードを *1* つの関数にせよ (ludecomp)
- LU による連立一次方程式の解法を *1* つの関数にせよ (lu_solve)
  - ludecomp を呼び出し，L,U を得る
  - 前進代入による $Ly = B$ を解く
  - 後退代入による $Ux = y$ を解く

ludecomp 関数

```
int ludecomp(int n, double a[N][N], double l[N][N], double u[N][N]) {
    int i, j, k;
    double p, q;

    for(i=1; i<=n; i++) {
        p = a[i][i];
        if (fabs(p) < 1.0e-6) {
            printf("この行列はLU分解出来ません. \n");
            return -1;
        }

        for(j=i; j<=n+1; j++) {
            l[j][i] = a[j][i];
            a[i][j] = a[i][j] / p;
        }

        for(k=i+1; k<=n; k++) {
            q = a[k][i];
            for(j=1; j<=n+1; j++) {
                a[k][j] = a[k][j] - a[i][j] * q;
            }
        }

        for(j=i; j<=n; j++) {
            u[i][j]= a[i][j];
        }
    }

    return 0;
}
```

lu_solve 関数

```
int lu_solve(int n, double a[N][N], double b[N], double x[N]) {
    int i,j;
    double l[N][N] = {0}, u[N][N] = {0};
    double y[N] = {0};

    /*LU分解*/
    int ret = ludecomp(n, a, l, u);
    if (ret != 0) {
        return ret;
    }
    /*前進代入*/
    for(i=1; i<= n; i++){
        double py = b[i];
        for(j=1; j < i; j++) {
            py -= l[i][j]*y[j];
        }
        y[i] = py/l[i][i];
    }
    /*後退代入*/
    for (i = n; i >= 1; i--) {
        double px = y[i];
        for(int j = i+1; j <= n; j++) {
            px -= u[i][j] * x[j];
        }
        x[i] = px/u[i][i];
    }
    return 0;
}
```

連立非線形方程式に対するニュートン法

連立非線形方程式の例

$$
\begin{aligned}
f(x,y,z) &= x^2 + y^2 + xz - x - y - 1 \\
g(x,y,z) &= x^3 + z^3 + 3x^2 - z^2 + 2yz - 2z - 4 \\
h(x,y,z) &= 3xy + 2xz + 4yz - 3y - 4z - 2x
\end{aligned}
$$

連立非線形方程式の定義

n 元連立非線形方程式

$$
\begin{aligned}
f_1(x_1, x_2, \cdots, x_n) &= 0 \\
f_2(x_1, x_2, \cdots, x_n) &= 0 \\
&\vdots \\
f_n(x_1, x_2, \cdots, x_n) &= 0
\end{aligned}
$$

に対する反復法を考えます.

反復法による連立非線形方程式の解 1/5

$$
\begin{aligned}
x =& (x_1, x_2, \cdots, x_n) \\
f_i(x) =& f_i(x_1, x_2, \cdots, x_n) \\
\mathbf{f}(x) =& [f_1(x), f_2(x), \cdots, f_n(x)]^t
\end{aligned}
$$

とおくと

$$\mathbf{f}(x) = 0$$

を求めよう

反復法による連立非線形方程式の解 2/5

普通のニュートン法

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)}$$

真の解 α に収束するまで反復する

ニュートン法の拡張

$$\begin{aligned} x^{(k+1)} =& g(x^{(k)}), \qquad k = 0, 1, 2, \cdots \\ g(x) =& x - A(x) f(x) \end{aligned}$$

解 α の近くの点 $x^{(0)}$ でテイラー展開すると,

$$0 = f_k(\alpha) \approx f_k(x^{(0)}) + \boxed{\sum_{i=1}^{n} \frac{\partial f_k}{\partial x_i}(x^{(0)})(\alpha_i - x_i^{(0)})}$$ $= 0$ となります...

反復法による連立非線形方程式の解 3/5

ニュートン法の拡張

$$
\begin{bmatrix} f_1(x^{(0)}) \\ f_2(x^{(0)}) \\ \vdots \\ f_n(x^{(0)}) \end{bmatrix}
+
\begin{bmatrix}
\dfrac{\partial f_1}{\partial x_1}(x^{(0)}) & \dfrac{\partial f_1}{\partial x_2}(x^{(0)}) & \cdots & \dfrac{\partial f_1}{\partial x_n}(x^{(0)}) \\
\dfrac{\partial f_2}{\partial x_1}(x^{(0)}) & \dfrac{\partial f_2}{\partial x_2}(x^{(0)}) & \cdots & \dfrac{\partial f_2}{\partial x_n}(x^{(0)}) \\
\vdots & & & \vdots \\
\dfrac{\partial f_n}{\partial x_1}(x^{(0)}) & \dfrac{\partial f_n}{\partial x_2}(x^{(0)}) & \cdots & \dfrac{\partial f_n}{\partial x_n}(x^{(0)})
\end{bmatrix}
\begin{bmatrix} \alpha_1 - x_1^{(0)} \\ \alpha_2 - x_2^{(0)} \\ \vdots \\ \alpha_n - x_n^{(0)} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

反復法による連立非線形方程式の解 4/5

ニュートン法の拡張

$$J(x^{(0)}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x^{(0)}) & \frac{\partial f_1}{\partial x_2}(x^{(0)}) & \cdots & \frac{\partial f_1}{\partial x_n}(x^{(0)}) \\ \frac{\partial f_2}{\partial x_1}(x^{(0)}) & \frac{\partial f_2}{\partial x_2}(x^{(0)}) & \cdots & \frac{\partial f_2}{\partial x_n}(x^{(0)}) \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x^{(0)}) & \frac{\partial f_n}{\partial x_2}(x^{(0)}) & \cdots & \frac{\partial f_n}{\partial x_n}(x^{(0)}) \end{bmatrix}$$ を $x^{(0)}$ における

ヤコビ行列といいます. $\Delta x = \alpha - x^{(0)}$ とすると

$$f(x^{(0)}) + J(x^{(0)})\Delta x = 0 \tag{1}$$

$$\Delta x = -[J(x^{(0)})]^{-1} f(x^{(0)}) \tag{2}$$

$$x^{k+1} = x^k - [J(x^{(k)})]^{-1} f(x^{(k)}) \tag{3}$$

反復法による連立非線形方程式の解 5/5

ニュートン法の拡張

$$J(x^k)d = -f(x^{(k)}) \tag{4}$$

を $Ax = b$ の形で解いて d を求め,

$$x^{(k+1)} = x^{(k)} + d \tag{5}$$

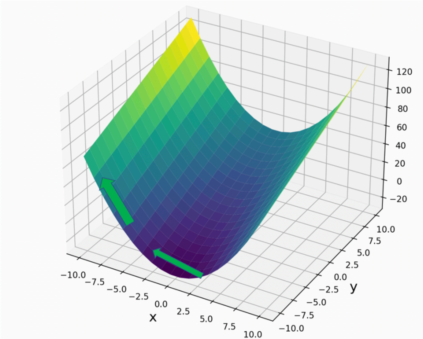
を計算します

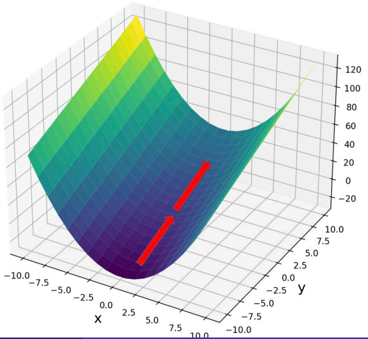
ヤコビ行列は

関数の 1 次偏微分を含む行列です。これにより、複数の次元に関数の傾きがわかります。

$$f(x,y) \quad = x^2 + 3y$$

$$\frac{\partial f(x,y)}{\partial(x)} = 2x \qquad \frac{\partial f(x,y)}{\partial(y)} = 3$$





問題

連立非線形方程式に対するニュートン法のアルゴリズムによる
次の方程式を解きましょう

$$f(x,y,z) = x^2 + y^2 + xz - x - y - 1$$
$$g(x,y,z) = x^3 + z^3 + 3x^2 - z^2 + 2yz - 2z - 4$$
$$h(x,y,z) = 3xy + 2xz + 4yz - 3y - 4z - 2x$$

初期値 (x,y,z) は $(1.1, 1.2, 1.3)$, $(1.2, 1.2, 1.2)$, $(1.3, 1.4, 1.5)$

問題の定義 1/4

メイン関数: f,g,h

```
double f(double x, double y, double z) {
    return(-1.0 - x + x*x - y + y*y + x*z);
}

double g(double x, double y, double z) {
    return(-4.0 + 3.0*x*x + x*x*x - 2.0*z + 2.0*y*z - z*z + z*z*z );
}

double h(double x, double y, double z) {
    return(-2.0*x - 3.0*y + 3.0*x*y - 4.0*z + 2.0*x*z + 4.0*y*z);
}
```

問題の定義 2/4

f の偏微分

```
/*f関数の偏微分*/
double f_x(double x, double y, double z) {
    return(-1.0 + 2.0*x + z);
}
double f_y(double x, double y, double z) {
    return(-1.0 + 2.0*y);
}
double f_z(double x, double y, double z) {
    return(x);
}
```

問題の定義 3/4

g の偏微分

```
/*g関数の偏微分*/
double g_x(double x, double y, double z) {
    return(6.0*x + 3.0*x*x);
}
double g_y(double x, double y, double z) {
    return(2.0*z);
}
double g_z(double x, double y, double z) {
    return(-2.0 + 2.0*y - 2.0*z + 3.0*z*z);
}
```

問題の定義 4/4

h の偏微分

```
/*h関数の偏微分*/
double h_x(double x, double y, double z) {
    return(-2.0 + 3.0*y + 2.0*z);
}
double h_y(double x, double y, double z) {
    return(-3.0 + 3.0*x + 4.0*z);
}
double h_z(double x, double y, double z) {
    return(-4.0 + 2.0*x + 4.0*y);
}
```

プログラムの前準備

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


#define EPS pow(10.0, -8.0) /* epsilonの設定*/
#define KMAX 100 /* 最大反復回数*/
#define N  4

#include "my_library_v3.h"
#include "nonlinear_system.h"
```

プログラム 1/4

初期値の入力

```
int main(void) {
    double x, y, z;

    printf("初期値x0, y0, z0を入力してください---->  x0 y0 z0\n");
    scanf("%lf %lf %lf", &x, &y, &z);

    int i, k=0;
    double xk[N], d[N], J[N][N];
    double Jx[N]; /* ヤコビ行列の解 */

    xk[1] = x; xk[2] = y; xk[3] = z;
```

プログラム 2/4

反復

```
do {
    /*右辺ベクトルの作成*/
    ...

    /*ヤコビ行列の作成*/
    ...

    lu_solve(3, J, d, Jx);
    for(i=1; i <= 3; i++) {
        xk[i] += Jx[i];
    }
    k++;

} while(vector_norm1(Jx) > EPS && k < KMAX);
```

プログラム 3/4

右辺ベクトル d の作成

```
d[1] = -f(xk[1], xk[2], xk[3]);
d[2] = -g(xk[1], xk[2], xk[3]);
d[3] = -h(xk[1], xk[2], xk[3]);
```

プログラム 4/4

ヤコビ行列 J の作成

```
/*ヤコビ行列の作成*/
J[1][1] = f_x(xk[1], xk[2], xk[3]);
J[1][2] = f_y(xk[1], xk[2], xk[3]);
J[1][3] = f_z(xk[1], xk[2], xk[3]);

J[2][1] = g_x(xk[1], xk[2], xk[3]);
J[2][2] = g_y(xk[1], xk[2], xk[3]);
J[2][3] = g_z(xk[1], xk[2], xk[3]);

J[3][1] = h_x(xk[1], xk[2], xk[3]);
J[3][2] = h_y(xk[1], xk[2], xk[3]);
J[3][3] = h_z(xk[1], xk[2], xk[3]);
```

解

問題

$$f(x, y, z) = x^2 + y^2 + xz - x - y - 1$$
$$g(x, y, z) = x^3 + z^3 + 3x^2 - z^2 + 2yz - 2z - 4$$
$$h(x, y, z) = 3xy + 2xz + 4yz - 3y - 4z - 2x$$

初期値 (x, y, z) は $(1.1, 1.2, 1.3)$, $(1.2, 1.2, 1.2)$, $(1.3, 1.4, 1.5)$

解

- $x = 0.972127$, $y = 1.302878$, $z = 0.650619$
- $x = 1.000000$, $y = 1.000000$, $z = 1.000000$
- $x = 0.710074$, $y = 0.970596$, $z = 1.738422$