ニュートン法

中島 崇晴

提出日 2016 年 12 月 7 日

未来ロボティクス・1526084

1　目的

ニュートン法のアルゴリズムを実装し，最低 5 種類の非線形方程式について解を求める。
解を求めるだけでなく，以下の要素を含める。

- 解の収束の過程が分かるように途中経過を出力する。
- その結果をもとに，収束する様子をグラフで表されるようにする。
- 解が複数存在する場合もすべての解が求まるようにプログラムを拡張する。

2　理論

数値解析の分野において、ニュートン法またはニュートン・ラフソン法は、方程式系を数値掲載によって解くための反復法による求根アルゴリズムの一つである。

ニュートン法では、以下の考え方に基づいて計算が行われる。

「$f(x) = 0$ になるような値 x を探す時、ある値 $x1$ における接線の切片 $x2$ は、元の値 $x1$ より真の値 x に近くなる」

具体的にはまず、$f(x) = 0$ の解 α に近い値 x_1 を 1 つとり、 関数 $y = f(x)$ のグラフ上の点 $(x_1, f(x_1))$ における接線が、x 軸と交わる点の x 座標 を x_2 とする。 このとき接線の方程式は、

$$y = f'(x_1)(x - x_1) + f(x_1) \tag{1}$$

となります。 ここで $y = 0$ とおいて、 x について解くと x_2 が求まる。
なので、

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{2}$$

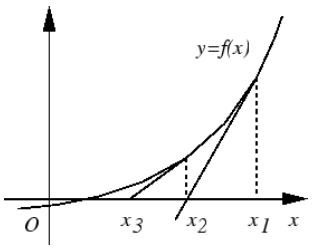

図 1　ニュートン法

次に, 点 $(x_2, f(x_2))$ における曲線の接線と x 軸との交点の x 座標を x_3 とし, この操作を繰り返すと, 以下の数列が出てくる

$$x_1, x_2, x_3, ..., x_n, ... \tag{3}$$

よって、以下の漸化式が出てくる。

$$x_n + 1 = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

式 (4) を使うことにより解が出てくる。

3　内容および方法

まず、ニュートン法で近似する 5 つの非線形方程式を準備する。

$$x^3 + x + 1 \tag{5}$$
$$3x^3 - 6x^2 - 3x - 6 \tag{6}$$
$$2sin(x) - x \tag{7}$$
$$cos(x) - x^2 \tag{8}$$
$$e^x + x \tag{9}$$

式 (5) を式 (4) に代入し計算する工程ををコードにおろしたのが以下のである。

```
double formula1(double ans)
{
    return (ans-(pow(ans,3.0)+ans+1.0)/(3.0*pow(ans,2.0)+1.0));
}
```

```
ans = 0.0;
ans_new = formula1(ans);
while(fabs(ans-ans_new) > EPS){
    ans_new = ans;
    ans = formula1(ans_new);
}
```

更に、複数解を求めるため初期値を－ 10 から 10 までの間を 0.01 ずつ変えていき解を出すことにした。そのコードが以下である。

```
for(n=0; n<2000; n++){
    ans = -10.0 + n * 0.01;
    ans_new = formula1(ans);
    while(fabs(ans-ans_new) > EPS){
```

```
        ans_new = ans;
        ans = formula1(ans_new);
    }

    for(int i=0; i<=num_values; i++){
        if(fabs(ans-ans_old[i]) > EPS){
            discrimination = 1;
        } else {
            discrimination = 0;
            break;
        }
    }

    if(discrimination == 1){
        ans_old[num_values] = ans;
        num_values++;
        cout << ans << endl;
    }
}
```

4 結果

式 (5) から式 (9) を実際に作成したプログラムで実行した結果が以下の通りである。

初期値	0
1	-1
2	-0.75
3	-0.686047
4	-0.68234
5	-0.682328
6	-0.682328

表 1　式 (5)

初期値	0	0	0
1	-3.86638	1	6.93547
2	-2.48674	1	4.92385
3	-1.64092	1	3.61781
4	-1.18916	1	2.7959
5	-1.02406	1	2.31437
6	-1.00047	1	2.07871
7	-1	1	2.00706
8	-1	1	2.00007
9	-1	1	2
10	-1	1	2

表 2　式 (6)

初期値	-3	-10
1	-2.088	-5.8598
2	-1.91223	-13.9742
3	-1.89565	3.78916
4	-1.89565	1.86413
5	-1.89565	1.89609
6	-1.89565	1.89549
7	-1.89565	1.89549

表 3　式 (7)

初期値	-10	5
1	-4.81707	2.26622
2	-2.14337	1.17637
3	-1.1417	0.871247
4	-0.863747	0.825308
5	-0.824971	0.824133
6	-0.824133	0.824132
7	-0.824132	0.824132
8	-0.824132	0.824132
9	-0.824132	0.824132

表 4　式 (8)

初期値	-10
1	-0.000499377
2	-0.500125
3	-0.566314
4	-0.567143
5	-0.567143
6	-0.567143

表 5　式 (9)

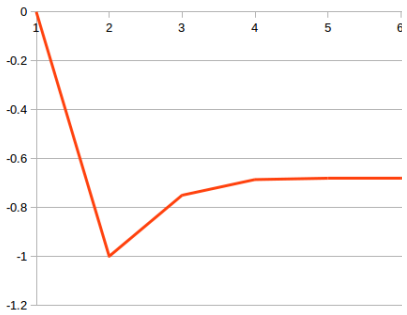

図 2　式 (5)

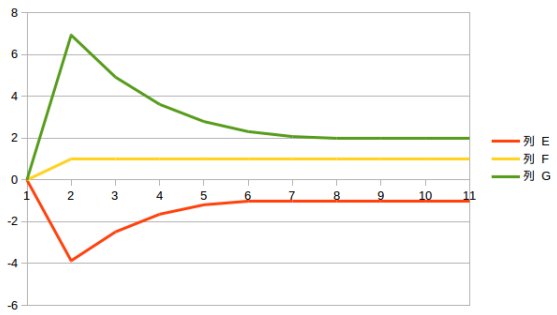

図 3　式 (6)

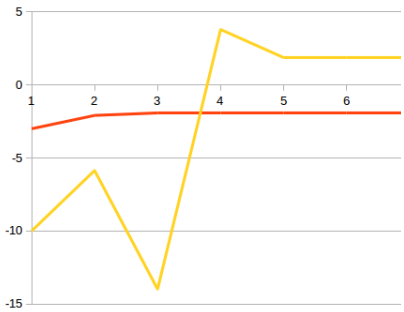

図 4　式 (7)

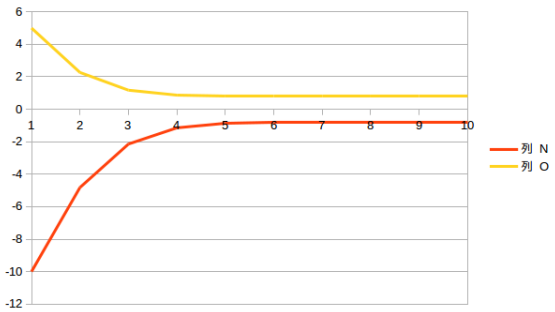

図 5　式 (8)

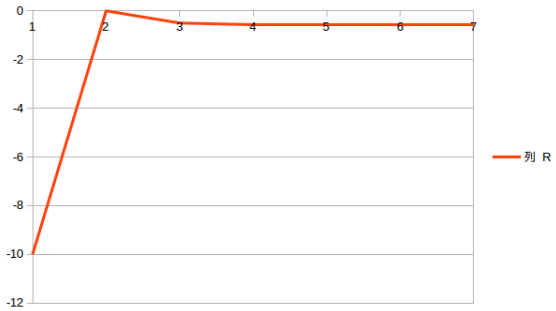

図 6　式 (9)

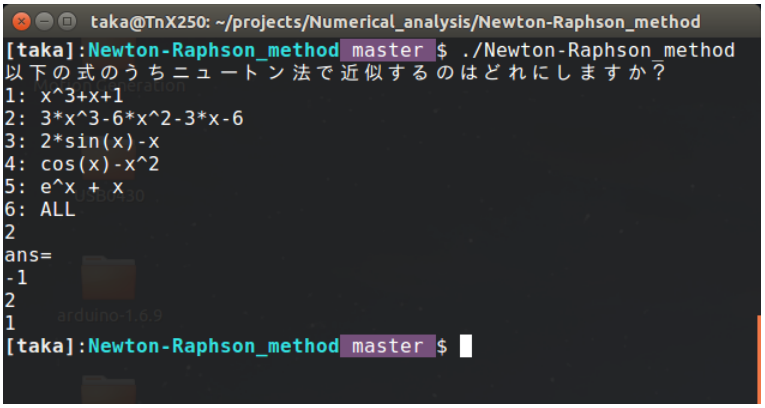

図 7　式 (6) を実行した場合

5 考察

今回、ニュートン法のアルゴリズムを実装したことによって非線形方程式の近似解を求めることができるので、マニュピレータの制御などが可能になった。

また、今回のレポートで LaTeX を初めて使用しました。まだ使い方が慣れていないところがありますが、卒業論文を書くまでにはなれるようにこれからも使っていきたいと思う。

6 付録・プログラム

```
#include <iostream>
#include <cmath>

using namespace std;

#define EPS 1e-7

double formula1(double ans)
{
    return (ans-(pow(ans,3.0)+ans+1.0)/(3.0*pow(ans,2.0)+1.0));
}

double formula2(double ans)
{
    return (ans-(3*pow(ans,3.0)-6*pow(ans,2.0)-3*ans+6)/(9*pow(ans,2.0)-12*ans-3));
}

double formula3(double ans)
{
    return (ans-(2*sin(ans)-ans)/(2*cos(ans)-1));
}

double formula4(double ans)
{
    return (ans-(cos(ans)-pow(ans,2.0))/(-sin(ans)-2*ans));
}

double formula5(double ans)
{
    return (ans-(exp(ans)+ans)/(exp(ans)+1));
}

//double divergence(double ans){


    int main(void)
    {
        int n=0, select, num_values=0, discrimination;
        double ans, ans_new, ans_old[200];

        for(int i=0; i<200; i++) ans_old[i] = 1e-6;

        celect;
```

```
switch(select){
    case 1:
    cout << "ans= " << endl;
    for(n=0; n<2000; n++){
        ans = -10.0 + n * 0.01;
        ans_new = formula1(ans);
        while(fabs(ans-ans_new) > EPS){
            ans_new = ans;
            ans = formula1(ans_new);
        }

        for(int i=0; i<=num_values; i++){
            if(fabs(ans-ans_old[i]) > EPS){
                discrimination = 1;
            } else {
                discrimination = 0;
                break;
            }
        }

        if(discrimination == 1){
            ans_old[num_values] = ans;
            num_values++;
            cout << ans << endl;
        }
    }
    break;
    case 2:
    cout << "ans= " << endl;
    for(n=0; n<2000; n++){
        ans = -10.0 + n * 0.01;
        ans_new = formula2(ans);
        while(fabs(ans-ans_new) > EPS){
            ans_new = ans;
            ans = formula2(ans_new);
        }

        for(int i=0; i<=num_values; i++){
            if(fabs(ans-ans_old[i]) > EPS){
                discrimination = 1;
            } else {
                discrimination = 0;
                break;
            }
        }

        if(discrimination == 1){
            ans_old[num_values] = ans;
            num_values++;
            cout << ans << endl;
        }
    }
    break;
    case 3:
    cout << "ans= " << endl;
    for(n=0; n<2000; n++){
```

```
        ans = -10.0 + n * 0.01;
        ans_new = formula3(ans);
        while(fabs(ans-ans_new) > EPS){
            ans_new = ans;
            ans = formula3(ans_new);
        }

        for(int i=0; i<=num_values; i++){
            if(fabs(ans-ans_old[i]) > EPS){
                discrimination = 1;
            } else {
                discrimination = 0;
                break;
            }
        }

        if(discrimination == 1){
            ans_old[num_values] = ans;
            num_values++;
            cout << ans << endl;
        }
    }
    break;
    case 4:
    cout << "ans= " << endl;
    for(n=0; n<2000; n++){
        ans = -10.0 + n * 0.01;
        ans_new = formula4(ans);
        while(fabs(ans-ans_new) > EPS){
            ans_new = ans;
            ans = formula4(ans_new);
        }

        for(int i=0; i<=num_values; i++){
            if(fabs(ans-ans_old[i]) > EPS){
                discrimination = 1;
            } else {
                discrimination = 0;
                break;
            }
        }

        if(discrimination == 1){
            ans_old[num_values] = ans;
            num_values++;
            cout << ans << endl;
        }
    }
    break;
    case 5:
    cout << "ans= " << endl;
    for(n=0; n<2000; n++){
        ans = -10.0 + n * 0.01;
        ans_new = formula5(ans);
        while(fabs(ans-ans_new) > EPS){
            ans_new = ans;
            ans = formula5(ans_new);
```

```
                }

                for(int i=0; i<=num_values; i++){
                    if(fabs(ans-ans_old[i]) > EPS){
                        discrimination = 1;
                    } else {
                        discrimination = 0;
                        break;
                    }
                }

                if(discrimination == 1){
                    ans_old[num_values] = ans;
                    num_values++;
                    cout << ans << endl;
                }
            }
            break;
            case 6:

            break;
            default:
            break;
        }
    }
```