数値計算の理論と実際　最終レポート

16T1601W 杉山隼太

最終レポートの課題は,

1. これまで講義で扱った数値計算法を 2 つ以上組み合わせて数値解を求める問題を設定する．コードを実装し，数値解を計算する．

を選択した．

非線型方程式の解法には，2 分法やニュートン法といった様々な解法がある．ニュートン法は解に収束するのは速いが，問題と初期値によっては収束しないこともある．例えば，

$$3\tan^{-1}(x-1)+\frac{x}{4}=0 \tag{1}$$

という方程式をニュートン法で解くとき，初期値が$x_0=2$のときは収束するが，$x_0=4$のときは振動してしまう．こういった問題と初期値の時も，収束が速いという長所をできるだけ残しながら正しく解を求めることができるように，ニュートン法と 2 分法を組み合わせて数値解を求めることを考える．アルゴリズムとしては，次のxの値x_{next}を決めるときに，2 分法による区間の中点の値

$$x_{next}=\frac{a+b}{2}$$

と，ニュートン法による値

$$x_{next}=x-\frac{f(x)}{f'(x)}$$

の 2 つを求める．これらのx_{next}を問題の方程式に代入し，0 に近いほうのx_{next}を次のxとする．これを繰り返すことでxは方程式の解に収束すると考えた．

次に，ニュートン法と 2 分法を組み合わせて数値解を得るプログラムのソースコードを示す．

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

using namespace std;

double func(const double x) {
```

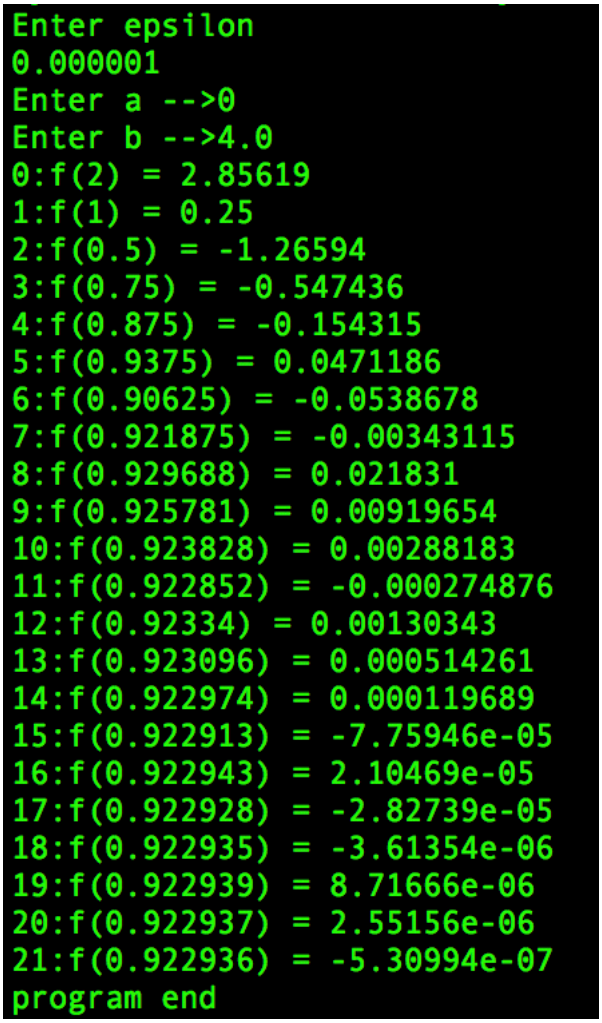
```
	//関数の値を返す
	//return pow(x, 3) - 3 * pow(x, 2) + 9 * x - 8;
	return 3 * atan(x - 1) + x / 4;
}

double func_d(const double x) {
	//微分値を返す
	double h = 0.0001;
	return (func(x + h) - func(x)) / h;
}

// 2 分法
double dichotomy(double a, double b) {
	return (a + b) / 2;
}

double regulaFalsi(double a, double b) {
	return (a * func(b) - b * func(a)) / (func(b) - func(a));
}

void newtonAndDual(double epsilon) {
	int step = 0;
	double x = 0.0;
	double xa = 0.0;
	double xb = 0.0;

	do {
		//f(xa) * f(xb) < 0 になるまで a,b の入力を繰り返す
		cerr << "Enter xa -->";
		cin >> xa;

		cerr << "Enter xb -->";
```

```
                cin >> xb;
        }   while (func(xa) * func(xb) >= 0);

        x = xa;
        double nextX = x;

        do {
                step++;
                x = nextX;

                if (abs(func(x - (func(x) / func_d(x)))) <= abs(func(dichotomy(xa, xb)))) {
                        nextX = x - (func(x) / func_d(x));
                } else if (abs(func(x - (func(x) / func_d(x)))) > abs(func(dichotomy(xa, xb)))) {
                        nextX = dichotomy(xa, xb);
                }

                //nextX = dichotomy(xa, xb);

                double fc = func(nextX);

                cerr << step << endl;
                cerr << "x" << " = " << nextX << ", func = " << func(nextX) << endl;

                if (fc == 0) break;
                else if (fc > 0) {
                        if (func(xa) > 0) xa = nextX;
                        else if (func(xb) > 0) xb = nextX;
                } else if (fc < 0) {
                        if (func(xa) < 0) xa = nextX;
```

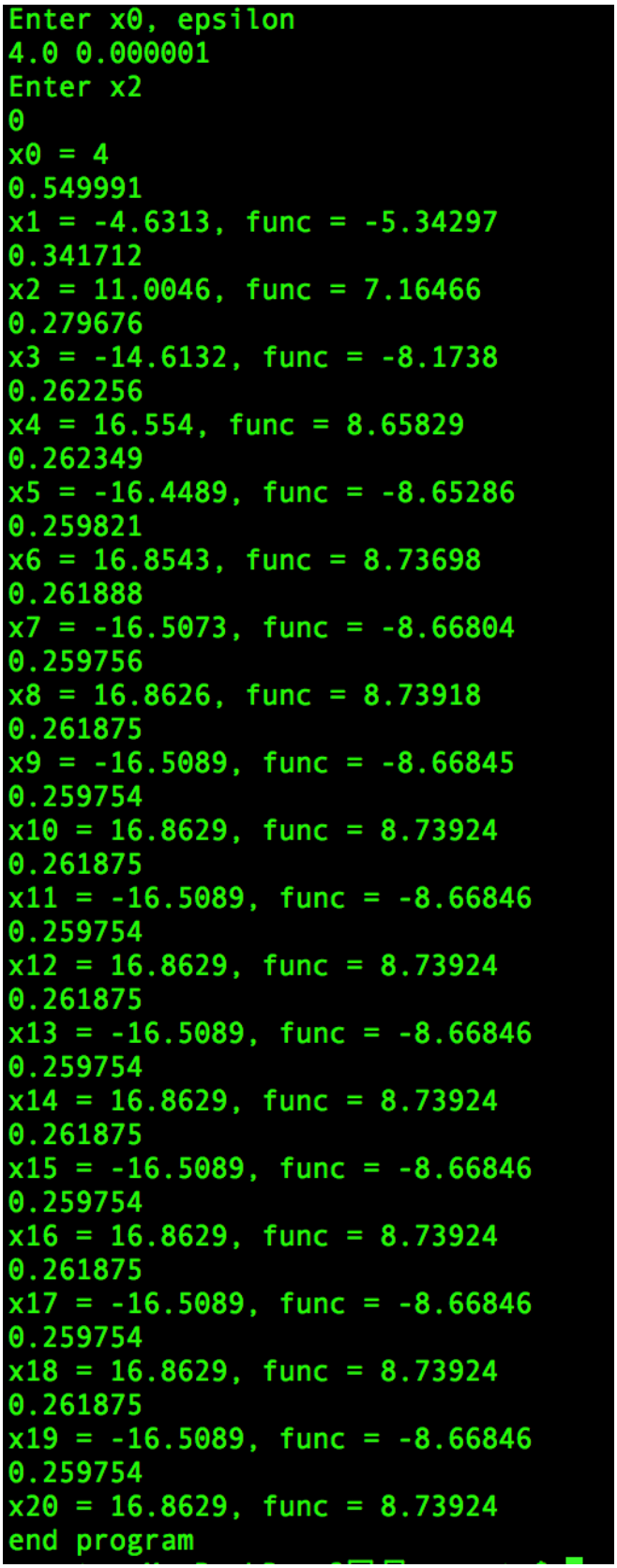
```
                                else if (func(xb) < 0) xb = nextX;
                        }
        } while (abs(xa - xb) > epsilon && abs(nextX - x) > epsilon);
}

int main() {
        double epsilon = 0.0;
        cerr << "Enter epsilon" << endl;
        cin >> epsilon;

        newtonAndDual(epsilon);

        cerr << "end program" << endl;
}
```

図 1 ニュートン法と 2 分法を組み合わせて数値解を得るプログラムのソースコード

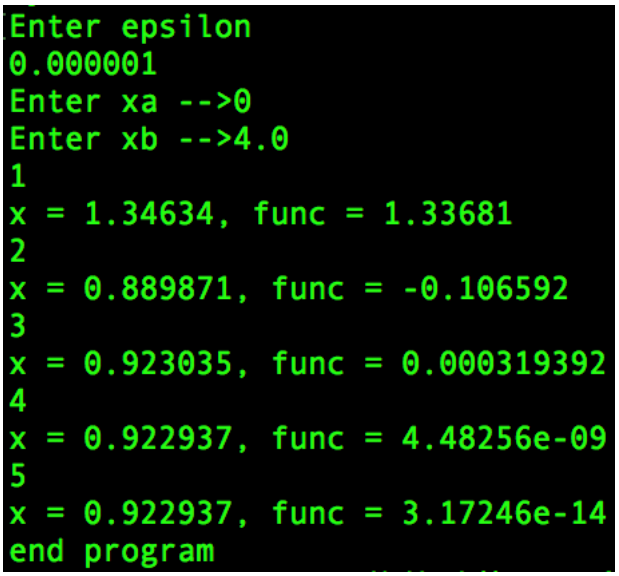
次に式(1)を区間[0, 4]の 2 分法で解いた結果，初期値$x_0 = 4$でニュートン法で解いた結果，同じ初期値で図 1 のプログラムで解いた結果を示す.


Enter epsilon
0.000001
Enter a -->0
Enter b -->4.0
0:f(2) = 2.85619
1:f(1) = 0.25
2:f(0.5) = -1.26594
3:f(0.75) = -0.547436
4:f(0.875) = -0.154315
5:f(0.9375) = 0.0471186
6:f(0.90625) = -0.0538678
7:f(0.921875) = -0.00343115
8:f(0.929688) = 0.021831
9:f(0.925781) = 0.00919654
10:f(0.923828) = 0.00288183
11:f(0.922852) = -0.000274876
12:f(0.92334) = 0.00130343
13:f(0.923096) = 0.000514261
14:f(0.922974) = 0.000119689
15:f(0.922913) = -7.75946e-05
16:f(0.922943) = 2.10469e-05
17:f(0.922928) = -2.82739e-05
18:f(0.922935) = -3.61354e-06
19:f(0.922939) = 8.71666e-06
20:f(0.922937) = 2.55156e-06
21:f(0.922936) = -5.30994e-07
program end

(a)2 分法


Enter x0, epsilon
4.0 0.000001
Enter x2
0
x0 = 4
0.549991
x1 = -4.6313, func = -5.34297
0.341712
x2 = 11.0046, func = 7.16466
0.279676
x3 = -14.6132, func = -8.1738
0.262256
x4 = 16.554, func = 8.65829
0.262349
x5 = -16.4489, func = -8.65286
0.259821
x6 = 16.8543, func = 8.73698
0.261888
x7 = -16.5073, func = -8.66804
0.259756
x8 = 16.8626, func = 8.73918
0.261875
x9 = -16.5089, func = -8.66845
0.259754
x10 = 16.8629, func = 8.73924
0.261875
x11 = -16.5089, func = -8.66846
0.259754
x12 = 16.8629, func = 8.73924
0.261875
x13 = -16.5089, func = -8.66846
0.259754
x14 = 16.8629, func = 8.73924
0.261875
x15 = -16.5089, func = -8.66846
0.259754
x16 = 16.8629, func = 8.73924
0.261875
x17 = -16.5089, func = -8.66846
0.259754
x18 = 16.8629, func = 8.73924
0.261875
x19 = -16.5089, func = -8.66846
0.259754
x20 = 16.8629, func = 8.73924
end program

(b)ニュートン法



(c)ニュートン法と 2 分法の組み合わせ

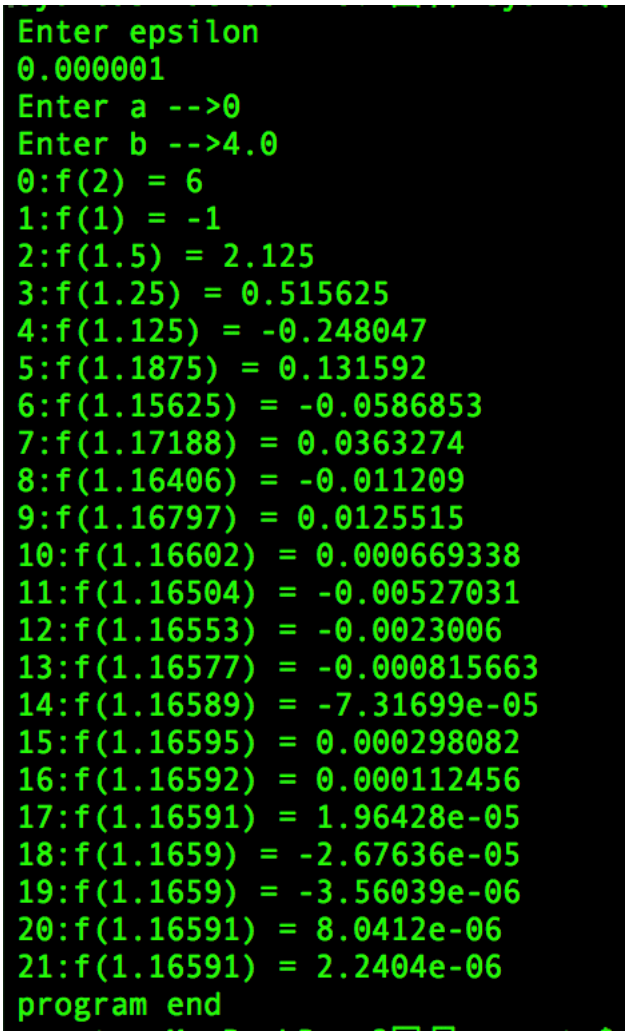
図 2 式(1)をそれぞれの解法で解いた結果

図 2(a)を見ると，2 分法は 21 回目で解に収束していることがわかる．(b)を見ると，ニュートン法では何回繰り返しても正しい解に収束することなく，振動してしまっている．(c)を見ると，ニュートン法と 2 分法の組み合わせでは 5 回目で解に収束していることがわかる．この結果より，ニュートン法では収束しない問題も，2 分法よりも速く解を得ることができるということがわかる．

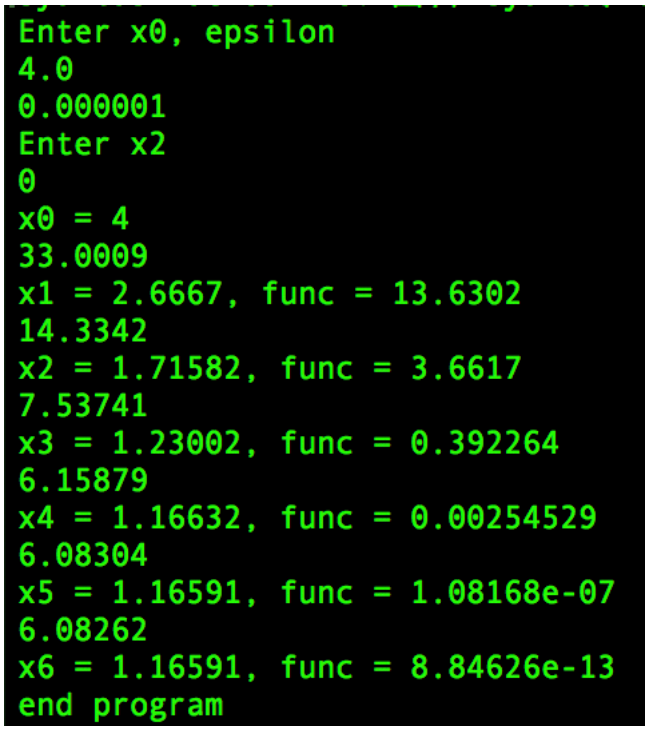
次に，

$$x^3 - 3x^2 + 9x - 8 = 0 \tag{2}$$

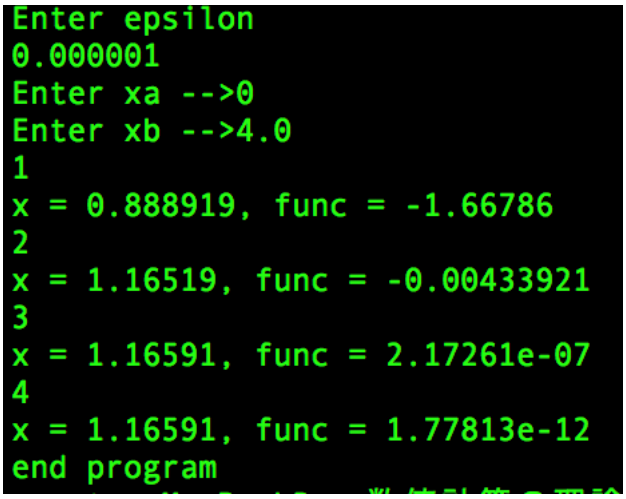
を 2 分法，ニュートン法，ニュートン法と 2 分法の組み合わせで解いた結果を示す．初期値は全て先ほどと同じである．


Enter epsilon
0.000001
Enter a -->0
Enter b -->4.0
0:f(2) = 6
1:f(1) = -1
2:f(1.5) = 2.125
3:f(1.25) = 0.515625
4:f(1.125) = -0.248047
5:f(1.1875) = 0.131592
6:f(1.15625) = -0.0586853
7:f(1.17188) = 0.0363274
8:f(1.16406) = -0.011209
9:f(1.16797) = 0.0125515
10:f(1.16602) = 0.000669338
11:f(1.16504) = -0.00527031
12:f(1.16553) = -0.0023006
13:f(1.16577) = -0.000815663
14:f(1.16589) = -7.31699e-05
15:f(1.16595) = 0.000298082
16:f(1.16592) = 0.000112456
17:f(1.16591) = 1.96428e-05
18:f(1.1659) = -2.67636e-05
19:f(1.1659) = -3.56039e-06
20:f(1.16591) = 8.0412e-06
21:f(1.16591) = 2.2404e-06
program end

(a)2 分法



(b)ニュートン法



(c)ニュートン法と 2 分法の組み合わせ

図 3 式(2)をそれぞれの解法で解いた結果

図 3 より，どの解法も同じ解に収束している．また，(c)のニュートン法と 2 分法の組み合わせが一番速く収束していることがわかる．

これらの結果より，ニュートン法と 2 分法の組み合わせは，ニュートン法では収束しない問題もニュートン法と同等の速度で解を求めることができる．ニュートン法では振動して

しまう状態になっても，2 分法によって求められる次の値を利用することでループを抜けることができるからだと思う．この方法は，ニュートン法で収束するかどうかの判定を事前に行わなくて良いという点，ニュートン法と同等の速度で解を得ることができるという点が優れていると考える．また，改善することで，さらに速く解を得ることも可能であると思われる．

参考文献

・数値計算の理論と実際　講義資料 2