工学基礎実験実習 C 言語の基礎
— ニュートン法に関する実験 —

氏名: 池田 海斗 (IKEDA, Kaito)
工学部 情報系学科
学生番号: 09501502

出題日: 2019 年 07 月 09 日
提出日: 2019 年 07 月 23 日
締切日: 2019 年 07 月 23 日

1 概要

本レポートでは，以下の 2 つの課題について実験および考察を行う．

課題 1 次の方程式の解をニュートン法を用いて求めよ．方程式を $f(x) = 0$ とおき，収束の様子を調べよ．また，収束の早さを調べ，グラフ等を用いてわかりやすく示せ．

1. $\sin e^x = 0$
2. $x^3 - 3x - 2 = 0$
3. 任意の方程式 $(x^3 - 3x^2 - x + 3 = 0)$

課題 2 方程式 $x^3 - 2x - 5 = 0$ の実数解を初期値 $x_0 = 0$ として，ニュートン法で求めよ．このとき，収束の様子を調べるため，k，x_k，$f(x_k)$，および $f'(x_k)$ の値を表示せよ．

2 はじめに

2.1 ニュートン法とは

方程式の解を求める方法として，ニュートン法がある．ニュートン法とは，曲線を直線に近似して計算することで，$f(x) = 0$ の解 x の近似を求めることが出来る計算法である．

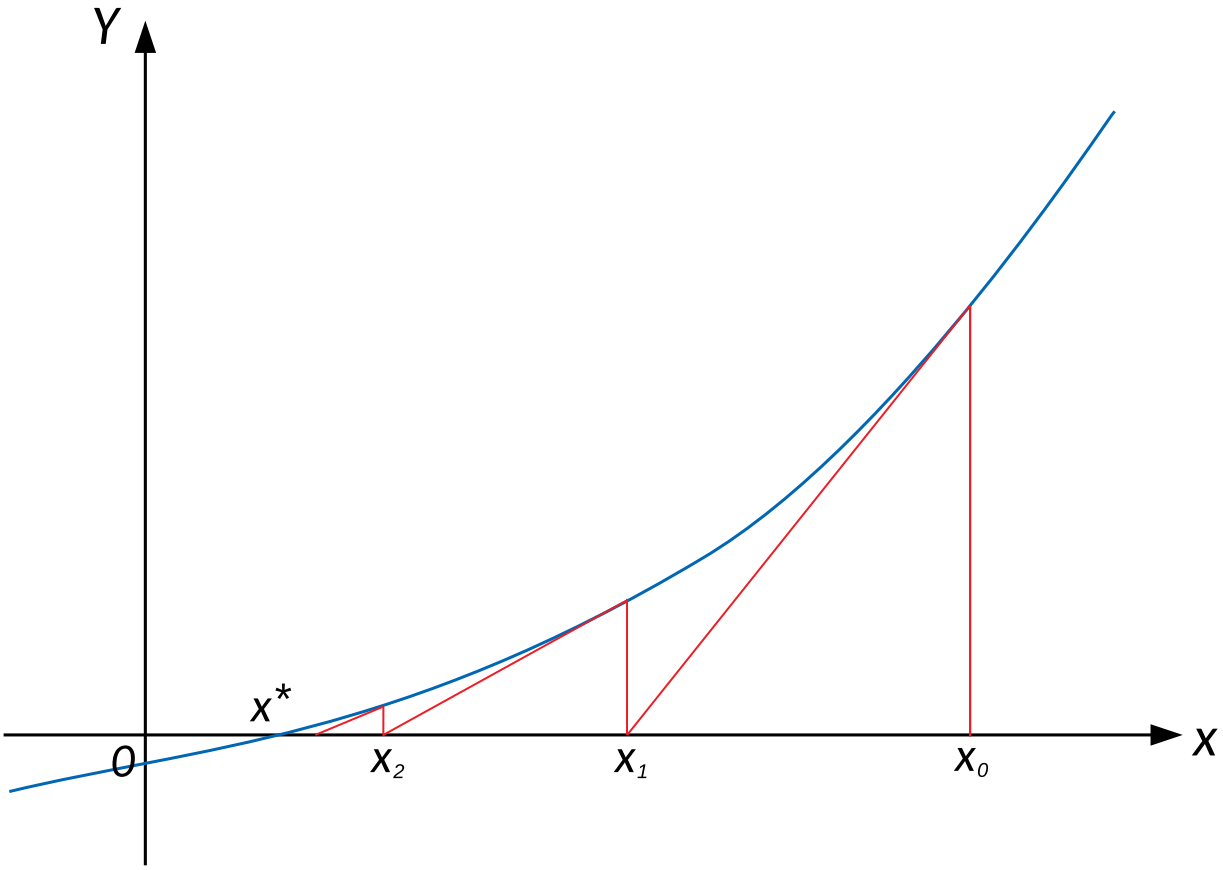

図 1: ニュートン法のグラフ

ニュートン法での導出手順は以下の通りである．

1. $y = f(x)$ のグラフで，初期値 x_0 に適当な値を入れる．
2. $y = f(x)$ のグラフで，$(x_0, f(x_0))$ における接線を求める．
3. 求めた接線と x 軸との交点 $(x_1, 0)$ とする．
4. 同様に繰り返し x の値を更新していき，解を導出する．

ここで，$(x_k, f(x_k))$ における接線を求める式として，以下の式 (1) を使用する．

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

また，x_{k+1} の値を求める式として，以下の式 (2) を使用する．

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2}$$

2.2　ニュートン法の証明

次に，上記のニュートン法が成り立つ証明をテイラー展開を用いて行う．$f(x) = 0$ の収束値を x^* とすると，ニュートン法の第 k ステップでの誤差 r_k は，

$$\begin{aligned} r_k &= x_k - x^* && (3) \\ \therefore r_{k+1} &= x_{k+1} - x^* && (4) \end{aligned}$$

とおける．r_k を用いて，$f(x)$ を $x = x^*$ の周りでテイラー展開すると，以下のようになる．

$$\begin{aligned} f(x_k) &= f(x^* + r_k) \\ &= f(x^*) + f'(x^*)r_k + \frac{1}{2}f''(x^*)r_k^2 + O(r_k^3) \\ &\approx f'(x^*)r_k + \frac{1}{2}f''(x^*)r_k^2 && (5) \end{aligned}$$

最終行については，x^* は $f(x)$ の**解であるから** $f(x^*) = 0$ となり，また r_k は**十分に小さいため**，それを 3 乗した r_k^3 についても近似して 0 とみなす．同様にして，$f'(x)$ もテイラー展開すると，

$$\begin{aligned} f'(x_k) &= f'(x^* + r_k) \\ &= f'(x^*) + f''(x^*)r_k + \frac{1}{2}f'''(x^*)r_k^2 + O(r_k^3) \\ &\approx f'(x^*) + f''(x^*)r_k && (6) \end{aligned}$$

ここで，ニュートン法の反復式

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{7}$$

の両辺から x^* を引くと，式 (3)(4) より，

$$r_{k+1} = r_k - \frac{f(x_k)}{f'(x_k)} \tag{8}$$

となり，これに式 (5)(6) を代入すると，以下のようになる．

$$\begin{aligned} r_{k+1} &= r_k - \frac{f'(x^*)r_k + \frac{1}{2}f''(x_k)r_k^2}{f'(x^*) + f''(x^*)r_k} \\ &= \frac{r_k f'(x^*) + r_k f''(x^*)r_k^2 - f'(x^*)r_k - \frac{1}{2}f''(x_k)r_k^2}{f'(x^*) + f''(x^*)r_k} && (9) \end{aligned}$$

ここで，$f'(x^*) \gg f''(x^*)r_k$ より，また，今回の課題では**重解を持たない方程式**であり，$f'(x^*)$ が 0 になることはないため，$f''(x^*)r_k$ を 0 と近似し，消すことが出来る．よって，

$$r_{k+1} = \frac{1}{2}\frac{f''(x^*)}{f'(x^*)}r_k^2 \tag{10}$$

となる．よって，ニュートン法は 2 次収束する解法である．

3　プログラム及びその説明

3.1　ユーザ定義関数

今回使うユーザ定義の関数は主に 4 つある．

1. *f()*　　x の値を代入することで，$f(x)$ の値を返す関数
2. *f1()*　　x の値を代入することで，$f'(x)$ の値を返す関数
3. *newton()*　　x_k の値を代入することで，x_{k+1} の値を返す関数
4. *main()*　　今回のプログラムの主な処理を行う関数

また，ヘッダーファイルには，stdio.h と math.h を使用している．

3.2　使用した変数

1. n　　繰り返し回数を記録する．
2. *ans*　　解を格納しておき，終了条件に使用する．
3. *delta*　　許容誤差値を格納しておき，解との差を使って終了条件に使用する．
4. *xk*　　k 回目の x の値である x_k を記録する．

3.3　終了条件

求めた値 *xk* と実際の解 *ans* との差の絶対値が，許容誤差値 *delta* よりも小さければ終了する．

3.4　実際に使用したプログラム

実際に使用したプログラムのコードは，レポートの最後に掲載する．(図 17, 18, 19, 20)

4　プログラムの使用法

実際にプログラムを動かす際には，以下の三点を行う必要がある．

- 求めたい方程式の解を求め，許容誤差値とともに入力する．
- 求める方程式，その微分した式を関数内に記載する．
- 初期値，終了条件について考察をし，適切かどうかを判断して訂正する．

4.1　課題 1-(a)

課題 1-(a) について，解は $\log \pi$ である．今回解が $1.14\cdots$ であるので，初期値を 1 に設定し，許容誤差は 3.0×10^{-16} とする．グラフと実行結果については図 2, 4 に記載する．

ここで，収束の速さについても調査する．収束の速さは図 3 の通りであった．

4.2　課題 1-(b)

課題 1-(b) について，解は -1, 2 である．しかし -1 は重解であり，**上記のニュートン法の証明の前提条件**として，解で $f'(x) \neq 0$ としているため，今回は解を 2 とする．初期値を 3 とし，許容誤差は同様に 3.0×10^{-16} とする．グラフと実行結果については図 5, 7 に記載する．また，収束の速さについては図 6 の通りであった．

4.3　課題 1-(c)

課題 1-(c) について，解は -1, 1, 3 である．今回解を 3 とし，解に近づけるよう初期値を 5 に設定した．他の値に設定しても動作するようにしたいので，**後ほど考察にて深く調査していきたい**．同様に許容誤差を 3.0×10^{-16} とし，グラフと収束の速さ，実行結果については図 8, 9, 10 に記載する．

4.4 課題 2

課題 2 について，解は $2.09\cdots$ となった．課題 1 と同様に許容誤差を 3.0×10^{-16} とすると，いつまで経っても解が求まらなかった．そこで，許容誤差の値をもう少し大きくしてみると，解が一定値に収まった．この可否の境界についても，**後ほど考察にて調査したい**．グラフと実行結果は図 11, 12 に記載する．

5 プログラムの考察

疑問を持ったところやプログラムの改善すべき点について，ここでは追求していきたい．上記に記載した通り，ここでは以下の二点について深層調査を行う．

1. 課題 1 の (c) など，解が複数ある方程式において，すべての解を求められるようにしたい．
2. 課題 2 の解が求まるギリギリの許容誤差の値はいくらなのか．

5.1 すべての解を求めるプログラム

課題 1 の (a) のプログラムでは，解を一つしか求めることができなかった．勿論，重解の場合などはニュートン法では求められない．課題 1 の発展として，解をすべて求められるプログラムを作りたい．

図 13 のように，細分化してたくさんの区間を作り，$f(x_k) \times f(x_{k+1}) \leqq 0$ となる所を探索していくプログラムである．この区間の間隔は，とりあえず 1.0×10^{-3} とする．この探索は 0 から正に向かって探索を行い，次に負を行う．範囲は絶対値 10 の間とする．プログラムは図 21 に記載する．

- 課題 1(c) の解は，$-1, 1, 3$ であり，このプログラムで実行してみた．(図 14)
- 課題 1(a) の $\sin e^x = 0$ についても，きちんと解が出ることが確認できた．(図 15)

プログラムを見ると分かると思うが，$f'(x_k)$ の積についても条件分岐をしている．今回の式にはあまり意味は無いが，$f'(x_k) \times f'(x_{k+1}) \leqq 0$ となる場合で条件分岐することで，重解の場合も対処できると考えた．時間が足りず実装できなかったが，時間がある時にチャレンジしてみたい．

5.2　許容誤差の値選定

課題 2 について，許容誤差の値を 1.0×10^{-15} にすると上手くいったが，1.0×10^{-16} にすると上手くいかなかった．誰しもが，その境界について気になったであろう．...少なくとも私は気になったので，探求していきたいと思う．

プログラムの概要としては，1.0×10^{-15} から 1.0×10^{-16} の間を，小数第 16 位から 17,18... と順に値を当てはめていく．もし 30 回以内に収束しなかった場合，一つ前の数字に戻り，位の 1 つ小さい所の探索を行う．double 型とはいえ，やはり位が小さすぎると演算誤差が出るので，小数第 30 位で丸めることとする．

プログラムは図 22 に，実行結果は図 16 に記載する．実行結果より，$8.88178419700126 \times 10^{-16}$ (小数第 30 位まで) であることが分かった．

6　まとめ

今回のレポートでは，方程式の解をニュートン法で求めるプログラムを作成した．ニュートン法の証明についても右田先生にご教授いただき，重解の場合などに式が成り立たないことも理解できた．またオリジナルのプログラムを 2 つ作成することで，疑問に思っていたところを深く追求していくことができた．

それと同時に，まだまだ調査を行いたいところが出てきた．例えば，考察で述べた方法で重解を持つ場合に場合分けを行い，課題 1(b) でも動作するようにしたり，その 1.0×10^{-3} 単位区間においても，考察 1 と考察 2 のプログラムを組み合わせ，単位区間内でも同様に探索プログラムを実行させ，小数第 30 位まで正確に解を求められるようなプログラムも作成したかった．

また，今回 30 桁以下では桁落ちが発生していたが，プログラムを作成し終えた後で long double 型というものがあることを知った．この変数を使っていれば，もっと精密な値が出ていたかもしれない．

参考文献

[1] http://www.swlab.cs.okayama-u.ac.jp/ gotoh/lect/p1/c_prog/c3/newton_proof.pdf

[2] http://www.swlab.cs.okayama-u.ac.jp/ gotoh/lect/p1/c_prog/c3/newton_repetition.pdf

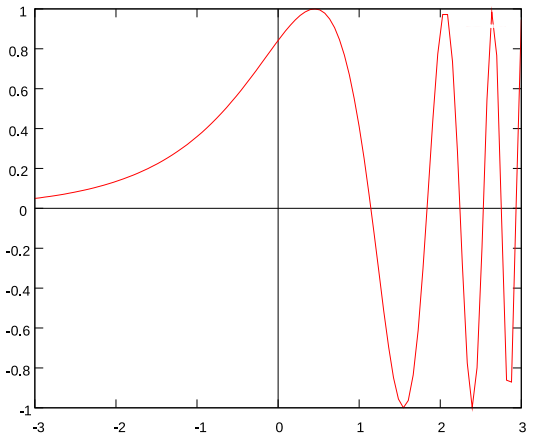

図 2: $f(x) = \sin e^x$

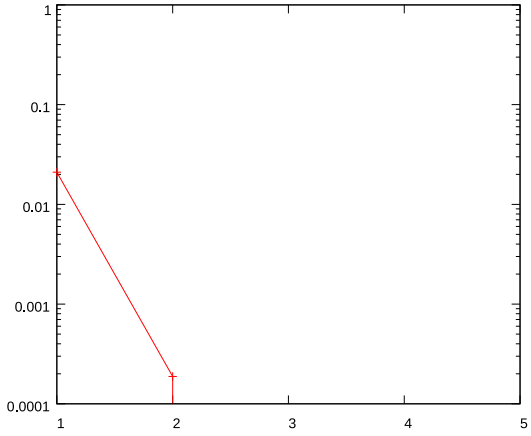

図 3: 収束の様子 1 - (a)

```
1 Newton method program start.
2 n:1 xk: 1.165748 f(xk):-0.066679 ans-xk:-0.021018
3 n:2 xk: 1.144918 f(xk):-0.000592 ans-xk:-0.000188
4 n:3 xk: 1.144730 f(xk):-0.000000 ans-xk:-0.000000
5 n:4 xk: 1.144730 f(xk):-0.000000 ans-xk:-0.000000
6 n:5 xk: 1.144730 f(xk): 0.000000 ans-xk: 0.000000
7 done.
```

図 4: 実行結果 1 - (a)

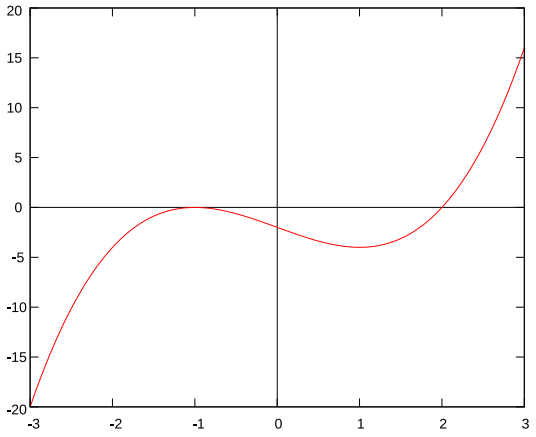

図 5: $f(x) = x^3 - 3x - 2$

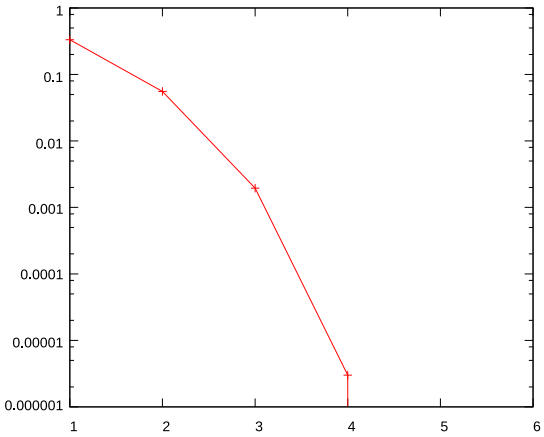

図 6: 収束の様子 1 - (b)

```
1 Newton method program start.
2 n:1 xk: 2.333333 f(xk): 3.703704 ans-xk:-0.333333
3 n:2 xk: 2.055556 f(xk): 0.518690 ans-xk:-0.055556
4 n:3 xk: 2.001949 f(xk): 0.017567 ans-xk:-0.001949
5 n:4 xk: 2.000003 f(xk): 0.000023 ans-xk:-0.000003
6 n:5 xk: 2.000000 f(xk): 0.000000 ans-xk:-0.000000
7 n:6 xk: 2.000000 f(xk): 0.000000 ans-xk: 0.000000
8 done.
```

図 7: 実行結果 1 - (b)

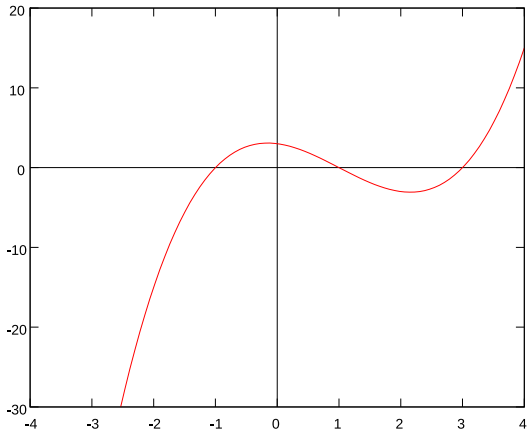

図 8: $f(x) = x^3 - 3x^2 - x + 3$

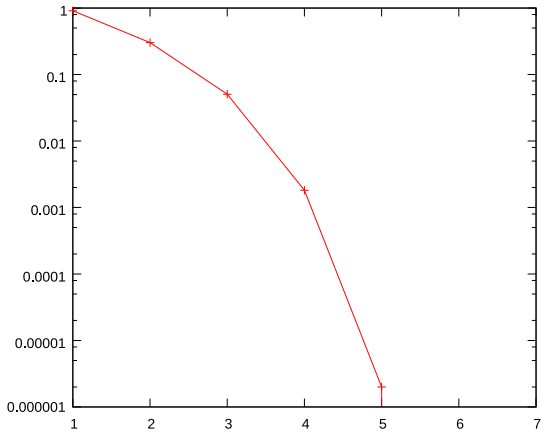

図 9: 収束の様子 1 - (c)

```
1 Newton method program start.
2 n:1 xk:3.909091 f(xk):12.982720 ans-xk:-0.909091
3 n:2 xk:3.302094 f(xk):2.991881 ans-xk:-0.302094
4 n:3 xk:3.050652 f(xk):0.420738 ans-xk:-0.050652
5 n:4 xk:3.001817 f(xk):0.014555 ans-xk:-0.001817
6 n:5 xk:3.000002 f(xk):0.000020 ans-xk:-0.000002
7 n:6 xk:3.000000 f(xk):0.000000 ans-xk:-0.000000
8 n:7 xk:3.000000 f(xk):0.000000 ans-xk:0.000000
9 done.
```

図 10: 実行結果 1 - (c)

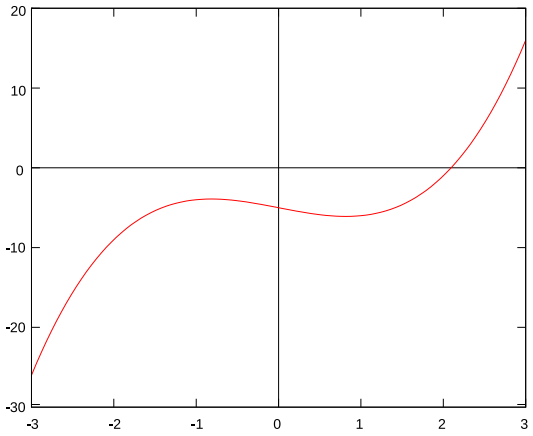

図 11: $f(x) = x^3 - 3x^2 - x + 3$

```
 1 Newton method program start.
 2 k: 1 xk:-2.500000 f(xk):-15.625000 f'(xk):16.750000
 3 k: 2 xk:-1.567164 f(xk): -5.714632 f'(xk): 5.368011
 4 k: 3 xk:-0.502592 f(xk): -4.121770 f'(xk):-1.242203
 5 k: 4 xk:-3.820706 f(xk):-53.132488 f'(xk):41.793394
 6 k: 5 xk:-2.549393 f(xk):-16.470758 f'(xk):17.498220
 7 k: 6 xk:-1.608111 f(xk): -5.942390 f'(xk): 5.758068
 8 k: 7 xk:-0.576100 f(xk): -4.039002 f'(xk):-1.004325
 9 k: 8 xk:-4.597710 f(xk):-92.995258 f'(xk):61.416800
10 k: 9 xk:-3.083543 f(xk):-28.151977 f'(xk):26.524715
11 k:10 xk:-2.022194 f(xk): -9.224909 f'(xk):10.267809
12 k:11 xk:-1.123764 f(xk): -4.171613 f'(xk): 1.788537
13 k:12 xk: 1.208652 f(xk): -5.651658 f'(xk): 2.382516
14 k:13 xk: 3.580790 f(xk): 33.751515 f'(xk):36.466172
15 k:14 xk: 2.655233 f(xk):  8.409627 f'(xk):19.150790
16 k:15 xk: 2.216106 f(xk):  1.451367 f'(xk):12.733381
17 k:16 xk: 2.102125 f(xk):  0.084892 f'(xk):11.256789
18 k:17 xk: 2.094584 f(xk):  0.000358 f'(xk):11.161841
19 k:18 xk: 2.094551 f(xk):  0.000000 f'(xk):11.161438
20 k:19 xk: 2.094551 f(xk): -0.000000 f'(xk):11.161438
21 done.
```

図 12: 実行結果 2

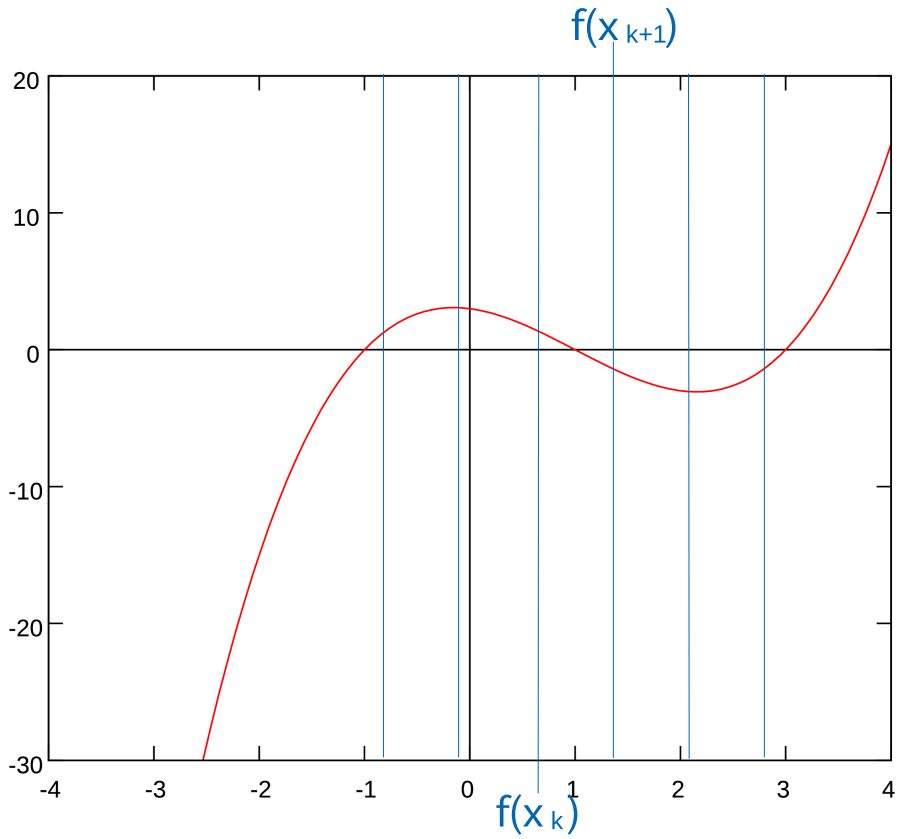

図 13: 考察 1 解説

```
1 解は，1.00 3.00 -1.00 です.
```

図 14: 考察 1 $x^3 - 3x^2 - x + 3 = 0$ での実行結果

```
1    解は，1.14 1.84 2.24 2.53 2.75 2.94 3.09 3.22 3.34 3.45 3.54 3.63 3.71 3.78
   3.85 3.92 3.98 4.03 4.09 4.14 4.19 4.23 4.28 4.32 4.36 4.40 4.44 4.48 4.51
   4.54 4.58 4.61 4.64 4.67 4.70 4.73 4.75 4.78 4.81 4.83 4.86 4.88 4.90 4.93
   4.95 4.97 4.99 5.02 5.04 5.06 5.08 5.10 5.12 5.13 5.15 5.17 5.19 5.21 5.22
   5.24 5.26 5.27 5.29 5.30 5.32 5.33 5.35 5.36 5.38 5.39 5.41 5.42 5.44 5.45
   5.46 5.48 5.49 5.50 5.51 5.53 5.54 5.55 5.56 5.58 5.59 5.60 5.61 5.62 5.63
   5.64 5.66 5.67 5.68 5.69 5.70 5.71 5.72 5.73 5.74 5.75 です.
```

図 15: 考察 1 $\sin e^x = 0$ での実行結果

```
 1 Newton method program start.
 2 0.00000000000000010000000000000000 の時は収束しました.
 3 0.00000000000000009000000000000000 の時は収束しました.
 4 0.00000000000000008000000000000000 の時は収束しませんでした.
 5 0.00000000000000008900000000000000 の時は収束しました.
 6 0.00000000000000008800000000000000 の時は収束しませんでした.
 7 0.00000000000000008890000000000000 の時は収束しました.
 8 0.00000000000000008880000000000000 の時は収束しませんでした.
 9 0.00000000000000008889000000000000 の時は収束しました.
10 0.00000000000000008888000000000000 の時は収束しました.
11 0.00000000000000008887000000000000 の時は収束しました.
12 0.00000000000000008886000000000000 の時は収束しました.
13 0.00000000000000008885000000000000 の時は収束しました.
14 0.00000000000000008884000000000000 の時は収束しました.
15 0.00000000000000008883000000000000 の時は収束しました.
16 0.00000000000000008882000000000001 の時は収束しました.
17 0.00000000000000008881000000000000 の時は収束しませんでした.
18 0.00000000000000008881900000000001 の時は収束しました.
19 0.00000000000000008881800000000001 の時は収束しました.
20 0.00000000000000008881700000000001 の時は収束しませんでした.
21 0.00000000000000008881790000000001 の時は収束しました.
22 0.00000000000000008881780000000001 の時は収束しませんでした.
23 0.00000000000000008881789000000001 の時は収束しました.
24 0.00000000000000008881788000000001 の時は収束しました.
25 0.00000000000000008881787000000001 の時は収束しました.
26 0.00000000000000008881786000000001 の時は収束しました.
27 0.00000000000000008881785000000001 の時は収束しました.
28 0.00000000000000008881784000000001 の時は収束しませんでした.
29 0.00000000000000008881784900000001 の時は収束しました.
30 0.00000000000000008881784800000001 の時は収束しました.
31 0.00000000000000008881784700000001 の時は収束しました.
32 0.00000000000000008881784600000001 の時は収束しました.
33 0.00000000000000008881784500000001 の時は収束しました.
34 0.00000000000000008881784400000001 の時は収束しました.
35 0.00000000000000008881784300000001 の時は収束しました.
36 0.00000000000000008881784200000001 の時は収束しました.
37 0.00000000000000008881784100000001 の時は収束しませんでした.
38 0.00000000000000008881784190000001 の時は収束しませんでした.
39 0.00000000000000008881784199000001 の時は収束しました.
40 0.00000000000000008881784198000001 の時は収束しました.
41 0.00000000000000008881784197000001 の時は収束しませんでした.
42 0.00000000000000008881784197900001 の時は収束しました.
43 0.00000000000000008881784197800001 の時は収束しました.
44 0.00000000000000008881784197700001 の時は収束しました.
45 0.00000000000000008881784197600001 の時は収束しました.
```

```
46 0.00000000000000008881784197500001 の時は収束しました.
47 0.00000000000000008881784197400001 の時は収束しました.
48 0.00000000000000008881784197300001 の時は収束しました.
49 0.00000000000000008881784197200001 の時は収束しました.
50 0.00000000000000008881784197100001 の時は収束しました.
51 0.00000000000000008881784197000001 の時は収束しませんでした.
52 0.00000000000000008881784197090001 の時は収束しました.
53 0.00000000000000008881784197080001 の時は収束しました.
54 0.00000000000000008881784197070001 の時は収束しました.
55 0.00000000000000008881784197060000 の時は収束しました.
56 0.00000000000000008881784197050000 の時は収束しました.
57 0.00000000000000008881784197040000 の時は収束しました.
58 0.00000000000000008881784197030000 の時は収束しました.
59 0.00000000000000008881784197020000 の時は収束しました.
60 0.00000000000000008881784197010000 の時は収束しました.
61 0.00000000000000008881784197000000 の時は収束しませんでした.
62 0.00000000000000008881784197009000 の時は収束しました.
63 0.00000000000000008881784197008000 の時は収束しました.
64 0.00000000000000008881784197007000 の時は収束しました.
65 0.00000000000000008881784197006000 の時は収束しました.
66 0.00000000000000008881784197005000 の時は収束しました.
67 0.00000000000000008881784197004000 の時は収束しました.
68 0.00000000000000008881784197003000 の時は収束しました.
69 0.00000000000000008881784197002000 の時は収束しました.
70 0.00000000000000008881784197001000 の時は収束しませんでした.
71 0.00000000000000008881784197001900 の時は収束しました.
72 0.00000000000000008881784197001800 の時は収束しました.
73 0.00000000000000008881784197001700 の時は収束しました.
74 0.00000000000000008881784197001600 の時は収束しました.
75 0.00000000000000008881784197001500 の時は収束しました.
76 0.00000000000000008881784197001400 の時は収束しました.
77 0.00000000000000008881784197001300 の時は収束しました.
78 0.00000000000000008881784197001200 の時は収束しませんでした.
79 0.00000000000000008881784197001290 の時は収束しました.
80 0.00000000000000008881784197001280 の時は収束しました.
81 0.00000000000000008881784197001270 の時は収束しました.
82 0.00000000000000008881784197001260 の時は収束しました.
83 0.00000000000000008881784197001250 の時は収束しませんでした.
84 0.00000000000000008881784197001260 の時は収束しました.
85 done.
```

図 16: 考察 2 実行結果

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return sin(exp(x));
 7 }
 8
 9 double f1(double x)
10 {
11     return exp(x) * cos(exp(x));
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21     int n = 0;
22     double ans = log(3.14159265358979323846);
23     double delta = 3E-16;
24
25     double xk = 1;
26
27     printf("Newton method program start.\n");
28
29     while (fabs(f(xk)) > delta){
30         n = n + 1;
31         xk = newton(xk);
32         printf("n:%d xk:%f f(xk):%f ans-xk:%f\n", n, xk, f(xk), ans - xk);
33     }
34     printf("done.\n");
35 }
```

図 17: 課題 1(a) プログラム

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return x*x*x-3*x-2;
 7 }
 8
 9 double f1(double x)
10 {
11     return 3*x*x-3;
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21     int n = 0;
22     double ans = 2;
23     double delta = 3E-16;
24
25     double xk = 3;
26
27     printf("Newton method program start.\n");
28
29     while (fabs(f(xk)) > delta){
30         n = n + 1;
31         xk = newton(xk);
32         printf("n:%d xk:%f f(xk):%f ans-xk:%f\n", n, xk, f(xk), ans - xk);
33     }
34     printf("done.\n");
35 }
```

図 18: 課題 1(b) プログラム

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return x*x*x-3*x*x-x+3;
 7 }
 8
 9 double f1(double x)
10 {
11     return 3*x*x-6*x-1;
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21     int n = 0;
22     double ans = 3;
23     double delta = 3E-16;
24
25     double xk = 5;
26
27     printf("Newton method program start.\n");
28
29     while (fabs(f(xk)) > delta){
30         n = n + 1;
31         xk = newton(xk);
32         printf("n:%d xk:%f f(xk):%f ans-xk:%f\n", n, xk, f(xk), ans - xk);
33     }
34     printf("done.\n");
35 }
```

図 19: 課題 1(c) プログラム

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return x*x*x-2*x-5;
 7 }
 8
 9 double f1(double x)
10 {
11     return 3*x*x-2;
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21     int k = 0;
22     double delta = 1E-15;
23
24     double xk = 0;
25
26     printf("Newton method program start.\n");
27
28     while (fabs(f(xk)) > delta){
29         k = k + 1;
30         xk = newton(xk);
31         printf("k:%2d xk:%2.6f f(xk):%2f f'(xk):%2f\n", k, xk, f(xk), f1(xk));
32     }
33     printf("done.\n");
34 }
```

図 20: 課題 2 プログラム

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return x*x*x-3*x*x-x+3;
 7 }
 8
 9 double f1(double x)
10 {
11     return 3*x*x-6*x-1;
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21
22     double delta = 1E-3; /* 間隔 */
23     double x = 0; /* 探索の現在地 */
24     int a = 0; /* 配列の要素番号 */
25     int i; /* printf に使う変数 */
26     double array[99]; /* 配列 (Max99) */
27
28     while (fabs(x) <= 10){
29         if (f(x + delta) * f(x) <= 0 && f1(x + delta) * f1(x) >= 0){
30             array[a] = x;
31             a = a + 1;
32         }
33         x = x + delta;
34     }
35
36     x = 0;
37
38     while (fabs(x) <= 10){
39         if (f(x + delta) * f(x) <= 0 && f1(x + delta) * f1(x) >= 0){
40             array[a] = x;
41             a = a + 1;
42         }
43         x = x - delta;
44     }
45
46     printf("解は, ");
47     for (i = 0; i <= a-1; ++i){
48         printf("%.2f ",array[i]);
49     }
50     printf("です. \n");
51
52 }
```


図 21: 考察 1 プログラム

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 double f(double x)
 5 {
 6     return x*x*x-2*x-5;
 7 }
 8
 9 double f1(double x)
10 {
11     return 3*x*x-2;
12 }
13
14 double newton(double xk)
15 {
16     return xk - (f(xk) / f1(xk));
17 }
18
19 main()
20 {
21     printf("Newton method program start.\n");
22     double delta = 1E-15;
23     double delta2 = 1E-16;
24     int a = 0;
25
26     while (a < 3){
27
28         int n = 0;
29
30         while (n < 10){
31             int k = 0;
32             double xk = 0;
33
34             while (fabs(f(xk)) > delta && k <50){
35                 k = k + 1;
36                 xk = newton(xk);
37             }
38             if (k != 50){
39                 printf("%2.30f の時は収束しました. \n", delta);
40                 delta = delta - delta2;
41             }else{
42                 printf("%2.30f の時は収束しませんでした. \n", delta);
43                 delta2 = delta2 / 10;
44                 delta = delta2 * 9 + delta;
45                 a = a + 1;
46                 n = -1;
47             }
48             n = n + 1;
49         }
50     }
51     printf("done.\n");
52 }
```


図 22: 考察 2 プログラム