数値計算論　前期期末レポート
非線形方程式の解法・定積分の解法

人間情報システム工学科 4 年 36 号
松山　京介

2022 年 6 月 6 日

1　演習課題

課題 1　$\sqrt{7}$ の近似値をニュートン法で求めよ．(解：$\sqrt{7} = 2.64575$)

アルゴリズム

非線形方程式

$$f(x) = 0 \tag{1}$$

の解を初期値 x_0 を適当に与えて順次 $x_1, x_2, \ldots, x_n$ を計算して近似解を求める方法で，テーラー展開の 1 次近似式をもとにした方法がニュートン法である (図 1 参照)．いま $f(x)$ を点 x_{k-1} でテーラー展開すると

$$f(x) = f(x_{k-1}) + f'(x_{k-1})(x - x_{k-1}) + \frac{f^{(2)}(x_{k-1})}{2!}(x - x_{k-1})^2 + \cdots \tag{2}$$

となり，2 次以降の項を無視すると

$$f(x) \approx f(x_{k-1}) + f'(x_{k-1})(x - x_{k-1}) \tag{3}$$

$f(x) = 0$ の解を直接求める代わりに近似的に 3= 0 の解を x_k として求めると，

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} \tag{4}$$

である．これを反復使用するのがニュートン法である．

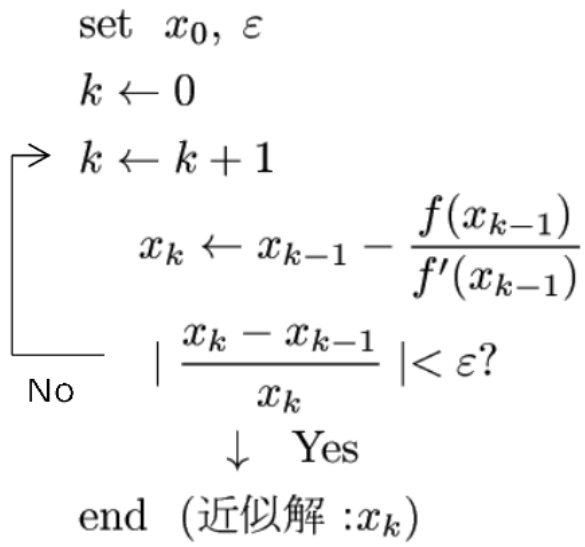

図 1: ニュートン法のアルゴリズム

ニュートン法は初期値 x_0 の取り方により必ずしも収束するとは限らない，または求めたい値が得られるとは限らない．2 は初期値 $x_0 = a$ にしたとき α に $x_0 = b$ にしたとき β に収束する例である．3 は初期値 $x_0 = a$ にしても a と b を繰り返して振動し，収束しない場合である．4 は，初期値 $x_0 = a$ で出発し $b, c, d \ldots$

と収束しない場合であり，特に関数の極近くでは $f'(x)$ が 0 に近くなり発散する場合もある．これらの問題を回避するために，繰り返し中に条件：

$$|f'(x_k)| < \epsilon_1? \text{ または } |x_{k+1} - x_k| > |x_k - x_{k-1}| \tag{5}$$

が成り立てば計算を終了し，初期値をもっとも求めたい解近くにして再計算する．

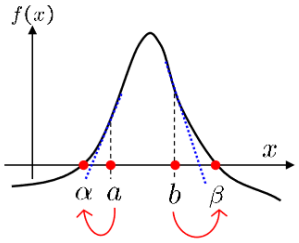

図 2: 初期値の違いによる収束

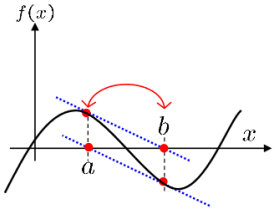

図 3: 振動する場合

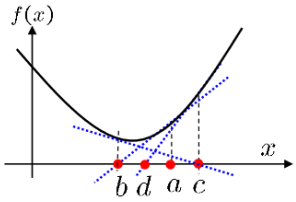

図 4: 振動または発散

プログラムリスト

ソースコード 1: ニュートン法のプログラム

```
1  x=[3]#初期値
2  k=0
3  e=10**-5
4  while True:
5      k = k+1
6      X = x[k-1]-((x[k-1]**2-7)/(2*x[k-1]))
7      x.append(X)
8      print((x[k]))
9      if (abs((x[k]-x[k-1])/x[k]) < e):
10         break
11
12 print("ニュートン法:"+str(x[k]))
```

実行結果

```
2.6666666666666665
2.645833333333333
2.645751312335958
2.6457513110645907

ニュートン法:2.6457513110645907
```

考察

実行結果より，解とほぼ同じ値が計算できていることが確認できる．図 5 に計算値の収束過程を示す．

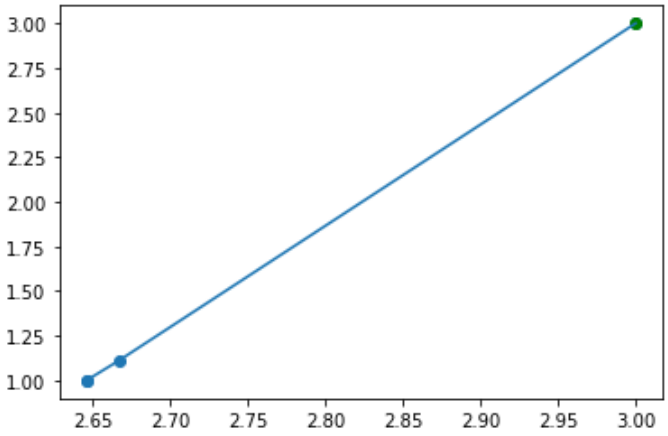

図 5: ニュートン法の収束過程

図 5 は初期値を緑の点で示している．図 5 と実行結果より，4 回の反復でほぼ収束できていることが確認できる．

課題 2　$\sqrt{7}$ の近似値を 2 分法で求めよ．(解：$\sqrt{7} = 2.64575$)

アルゴリズム

非線形方程式 $f(x) = 0$ の解が存在すると思われる区間で $f(x)$ の符号を調べながら次第に区間の幅を半分ずつ狭めて，方程式の解を求める方法である．6 のような $f(x_1)f(x_1+h) < 0, (h > 0)$ を満たす区間 $[x_1, x_1+h]$ をとると，この区間内に必ず $f(x) = 0$ の解が 1 つは存在する．この区間 $[x_1, x_1 + h]$ の中点 $x_1 + (1/2)h$ をとり，$f(x_1)f(x_1 + (1/2)h) > 0$ なら $[x_1 + (1/2)h, x_1 + h]$ 内に解が存在するので，$x_3 = x_2$ とする．この操作を繰り返し近似解を得る方法が 2 分法である．n 回目で得られた近似解の誤差 ϵ_n は $\epsilon_n < \frac{h}{2^n}$ となる．

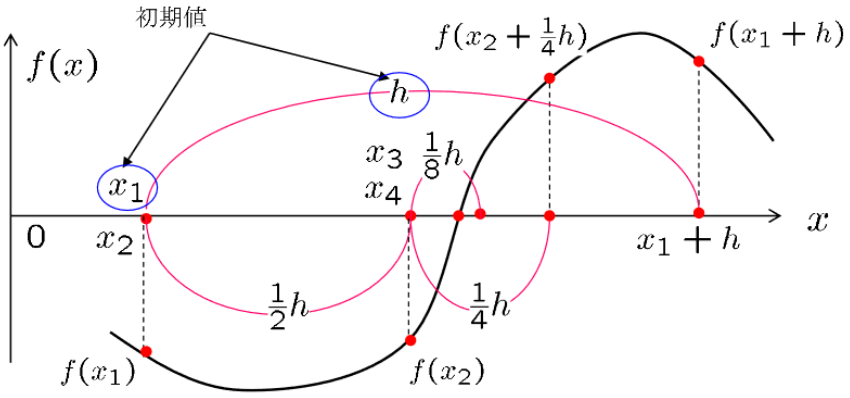

図 6: 2 分法

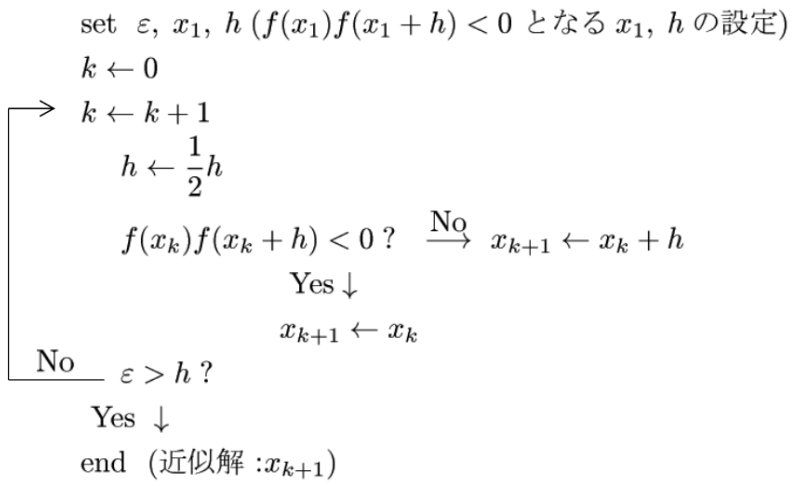

図 7: 2 分法のアルゴリズム

2 分法のアルゴリズム (図 7 参照) は確実に近似解を求めることができるが，反復回数が多くなる．
一方ニュートン法は初期値により近似解を得られないこともあるが，適切な初期値を定めることにより，2 分法よりも少ない反復回数で近似解を得られる．

プログラムリスト

ソースコード 2: 2 分法のプログラム

```
import matplotlib.pyplot as plt
import sympy as sym
import math

e=1e-6
x=[]
x1=[]
y=[4]
y1=[]
x.append(2)
k=0
h=1
x1.append(x[0])
y1.append(y[0])

while (1):
    k=k+1
    h=h/2
    print(x[k-1])
    if((x[k-1]**2-7)*((x[k-1]+h)**2-7)<0):
        X=x[k-1]
        x.append(X)
```

```
23      else:
24          X=x[k-1]+h
25          x.append(X)
26      if(e>h):
27          break
28
29  print("\n 二分法:"+str(x[k]))
```

実行結果

```
2
2.5
2.5
2.625
2.625
2.625
2.640625
2.640625
2.64453125
2.64453125
2.6455078125
2.6455078125
2.6455078125
2.6456298828125
2.64569091796875
2.645721435546875
2.6457366943359375
2.6457443237304688
2.6457481384277344
2.645750045776367

二分法:2.6457509994506836
```

考察

実行結果より，解に限りなく近い値を計算できていることが確認できる．図 8 に 2 分法を使ったときの収束過程を示す．

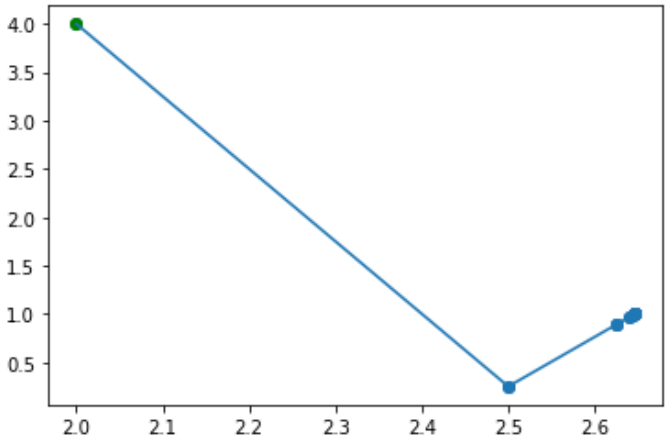

図 8: 2 分法の収束過程

図 8 と図 5 より，どちらも解に収束しているが，2 分法は 20 回，ニュートン法は 4 回の計算で収束していることが確認できる．

課題 3　次の積分を区分求積法，台形公式，シンプソンの公式の各手法を使って計算し，解析解 π と比較し各手法の精度の違いを考察せよ．ただし，分割数 n は適当に定めよ．

$$S = \int_0^1 \frac{4}{1+x^2} dx$$

アルゴリズム

- 区分求積法

  $f(x)$ を閉区間 $[a, b]$ で定義された有界関数とし，その定積分：

  $$S = \int_a^b f(x)dx \tag{6}$$

  を考える．積分区間 $[a, b]$ を n 等分し，各区間を

  $$x_k = a + k \cdot h, \text{ ただし}, h = \frac{b-a}{n} \tag{7}$$

  とする (h は刻み幅)．この区間ごとの面積の総和：

  $$S_n = \sum_{k=0}^{n-1} f(x_k)h \tag{8}$$

  で近似値を求める方法が区間求積法 (式 9 参照) である．n を十分大きくとると，8 式 S_n の近似値が式 6 の S の真値に近づく．ただし，実際に計算機で行う際には n がある程度以上になると丸め誤差が累積するため精度には限界がある．

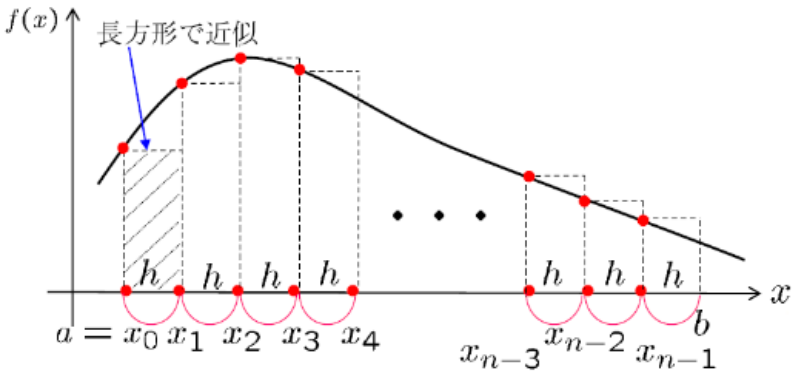

図 9: 区分求積法

- 台形公式

  台形公式は，区間 $[a, b]$ を n 等分し各小区間ごとの第系の面積の総和で定積分の近似値を求める方法で，

$$S_n = \sum_{k=0}^{n-1} \frac{h}{2}(f(x_k) + f(x_{k+1})),\ \text{ただし}, x_k = a + kh = a + k\frac{b-a}{n} \tag{9}$$

  となる (図 10)．図 9 と図 10 からわかるように，同一の刻み幅 h では，区分求積法と比較して台形公式の方法が精度が高いことがわかる．

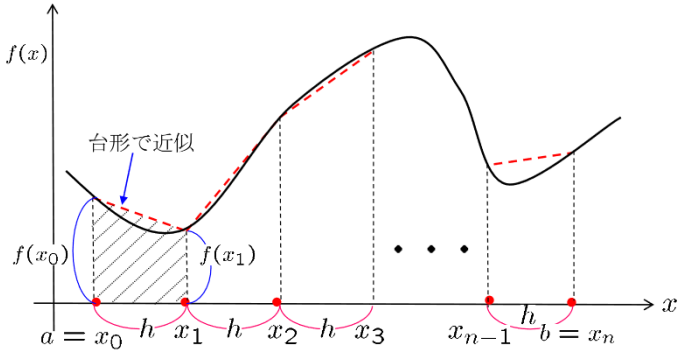

図 10: 台形公式

- シンプソンの公式

  シンプソンの (1/3) 公式は，関数の 3 点を通る 2 次の補間多項式で近似した式を基にしており，区間 $[a, b]$ を n 等分 (ただし，n は偶数) し，

$$S_n = \sum_{k=0}^{\frac{n}{2}-1} \frac{h}{3}(f(x_{2k}) + 4f(x_{2k+1}) + f(x_{2k+2}))\ \text{ただし}, x_k = a + kh = a + k\frac{b-a}{n} \tag{10}$$

  となる．図 11 からこの式を説明する．定積分は少区間 $[x_{2k}, x_{2k+2}]$ ごとの面積 S'_k で以下で表せる．

$$\int_a^b f(x)dx = \sum_{k=0}^{\frac{n}{2}-1} S'_k \tag{11}$$

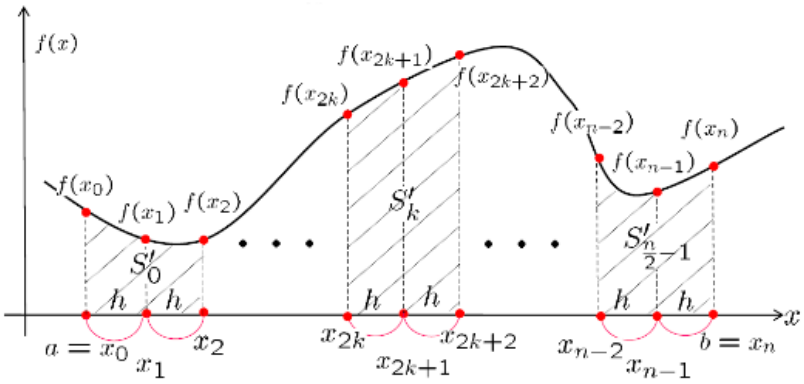

図 11: シンプソンの公式

まず図 11 の区間 $[x_{2k}, x_{2k+2}]$ の面積 S'_k：

$$S'_k = \int_{x_{2k}}^{x_{2k+2}} f(x)dx \tag{12}$$

の近似値を，関数 $f(s)$ の 2 次近似補間多項式：

$$f(x) \approx g(x) = a_0 + a_1(x - x_{2k}) + a_2(x - x_{2k})(x - x_{2k+1}) \tag{13}$$

といわれるこの $g(x)$ を用いて近似することを考える．$f(x)$ がこの 2 次近似式 $g(x)$ と 3 点 $(x_{2k}, x_{2k+1}, x_{2k+2})$ で $f(x) = g(x)$ となるためには

$$\left\{ \begin{array}{ll} f(x_{2k}) & = g(x_{2k}) = a_0 \\ f(x_{2k+1}) & = g(x_{2k+1}) = a_0 + a_1(x_{2k+1} - x_{2k}) = a_0 + a_1 h \\ f(x_{2k+2}) & = g(x_{2k+2}) = a_0 + a_1(x_{2k+2} - x_{2k}) \\ & \quad + a_2(x_{2k+2} - x_{2k})(x_{2k+2} - x_{2k+1}) = a_0 + a_1 2h + a_2 2h^2 \end{array} \right. \tag{14}$$

となり，13 式の近似式の係数 a_0, a_1, a_2 は

$$\left\{ \begin{array}{l} a_0 = f(x_{2k}) \\ a_1 = \frac{1}{h}(f(x_{2k+1}) - f(x_{2k})) \\ a_2 = \frac{1}{2h^2}(f(x_{2k}) - 2f(x_{2k+1}) + f(x_{2k+2})) \end{array} \right. \tag{15}$$

となる．13 式を 12 式の右辺に代入して積分すると，

$$\begin{aligned} S'_k &= \int_{x_{2k}}^{x_{2k+2}} f(x)dx \approx \int_{x_{2k}}^{x_{2k+2}} g(x)dx \\ &= \int_{x_{2k}}^{x_{2k+2}} \left\{ f(x_{2k}) + \frac{1}{h}(f(x_{2k+1}) - f(x_{2k}))(x - x_{2k}) \right. \qquad (16) \\ &\quad \left. + \frac{1}{2h^2}(f(x_{2k}) - 2f(x_{2k+1}) + f(x_{2k+2}))(x - x_{2k})(x - x_{2k+1}) \right\} dx \\ &= \frac{h}{3}(f(x_{2k}) + 4f(x_{2k+1}) + f(x_{2k+2})) \qquad (17) \end{aligned}$$

となり，11 式よりシンプソンの公式：

$$\int_a^b f(x)dx \approx \sum_{k=0}^{\frac{n}{2}-1} S'_k = \sum_{k=0}^{\frac{n}{2}-1} \frac{h}{3}(f(x_{2k}) + 4f(x_{2k+1}) + f(x_{2k+2})) \tag{18}$$

が導かれる.

シンプソンの公式による定積分は，図 11 からわかるように 2 次曲線で近似した関数から求めており，一般に台形公式による定積分と比較して近似精度がよい.

プログラムリスト

```
a = 0
b = 1
k = 0
n = 100
h = (b - a)/n
x = [0]*(n+1)
S = 0

def kubun():
    S1=0
    for k in range(n):
        x[k] = a + k * h
        S1 += 4 / (1 + x[k]**2) * h
    return S1

def daikei():
    S2=0
    for k in range(n):
        x[k] = a + k * h
        x[k + 1] = a + (k + 1) * h
        S2+= h * 0.5 * ((4 / (1 + x[k]**2)) + (4 / (1 + x[k + 1]**2)))
    return S2

def sinp():
    S3=0
    for k in range(int(n/2)):
        x[2*k] = a + (2*k)*h
        x[2*k+2] = a + (2*k+2) * h
        S3 += h * 1/3 * ((4 / (1 + x[2*k]**2)) + (4 * 4 / (1 + x[2*k+1]**2)) + (4 /
    (1 + x[2*k+2]**2)))
    return S3

print("区分求積法:"+str(kubun()))
print("\n 台形公式による解法:" +str(daikei()))
print("\n シンプソンの公式による解法:" + str(sinp()))
```

実行結果

```
区分求積法:3.151575986923127
```

台形公式による解法:3.141575986923131

シンプソンの公式による解法:3.1415926535897527

考察

実行結果より，どの手法も解析解 π に限りなく近い値が計算できていることが確認できる．

まず，解析解の導出を以下に示す．

$$
\begin{aligned}
S &= \int_0^1 \frac{4}{1+x^2} dx \\
&= 4\int_0^1 \frac{1}{1+x^2} dx \\
&= 4\Big[\arctan x\Big]_0^1 \\
&= 4\frac{\pi}{4} - 0 \\
&= \pi.
\end{aligned}
$$

よって解は π になることが確認できる．

区分求積法, 台形公式による解法, シンプソンの公式による解法それぞれを比較すると区分求積法が一番誤差が大きく, シンプソンの公式が一番誤差が小さいことがわかる．区分求積法では長方形で近似するためどうしても値の誤差が大きく出てしまう．今回は分割数を 100 としたためより誤差が顕著に現れている．

課題 4　$C-R$ の直列回路 (図 12) に突然直流電圧 E を加えたとき，加電の瞬時から測って t の時刻の電流 $i(t)$ は次の式で与えられる．

$$
i(t) = \frac{E}{R}\exp -\frac{1}{CR}t[\mathrm{A}] \tag{19}
$$

電流 i がその最大値 $i_m = E/R$ の $1/2$ になる時間 T(図 13) を，ニュートン法で求めよ．ただし，$E = 1[\mathrm{V}], R = 100[\mathrm{k\Omega}], C = 80[\mu\mathrm{F}]$ とする．(解：$T = 5.5452\mathrm{sec}$)

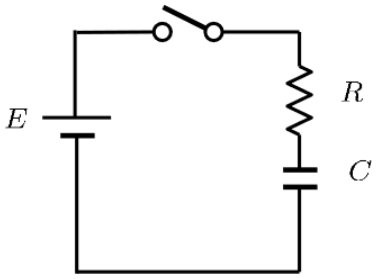

図 12: CR 回路

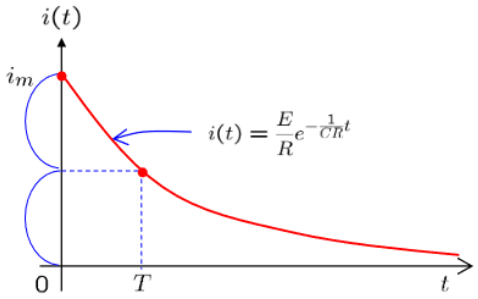

図 13: CR 回路の過渡応答

アルゴリズム

ニュートン法のアルゴリズムは課題 1 に同じ.

プログラムリスト

```
import math
import matplotlib.pyplot as plt
x =[5]
e= 10**-6
k = 0
E = 1
R = 100000
C = 0.00008
I = E/R

def f(x):
    X = I*pow(math.e,-x/(C*R))-I/2
    return X

def df(y):
    h = 1e-5
    Y = (f(y+h)-f(y))/h
    return Y

while(True):
    k += 1
    x.append(x[k-1]-(f(x[k-1])/df(x[k-1])))
    plt.plot(x)
```

```
    print(x[k])
    if (math.fabs((x[k]-x[k-1])/x[k])) < e:
        break

print("T="+str(x[k]))
plt.show()
```

実行結果

```
5.527016499697729
5.545156857663651
5.545177444465939
5.545177444479562
T=5.545177444479562
```

考察

実行結果より，ニュートン法で解と有効桁数 5 桁で等しい値が求められていることが確認できる．

以下にニュートン法を適用する上で定めた t についての方程式の導出を示す．

$$i(t) = \frac{E}{R}\exp -\frac{1}{CR}t[\mathrm{A}], \quad i_m = \frac{E}{R}$$

求める時間 T は，最大値 $i_m = E/R$ の $1/2$ になるときなので，

$$\begin{aligned} \frac{1}{2}i_m &= \frac{1}{2}\frac{E}{R} \\ \exp -\frac{1}{CR}t &= \frac{1}{2} \\ \therefore f(t) &= \exp -\frac{1}{CR}t - \frac{1}{2} = 0. \end{aligned}$$

収束過程を図 14 に示す

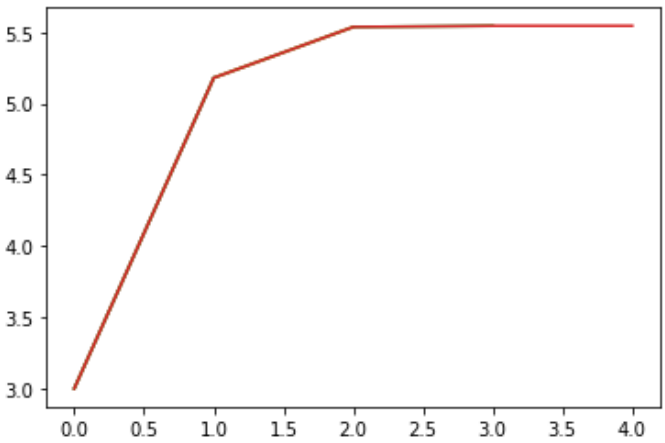

図 14: 課題 4 の収束過程

初期値 $x_0 = 5$ と定めたとき，反復回数 4 回で収束していることが確認できる．

感想

今回は問題数は少なかったが, 思ったよりプログラム作成に時間がかかってしまい, 応用課題ができなかった. 次回は応用課題まで完成させられるようにしたい