

[〆切] 2020/10/22 19:00

[III. 求根アルゴリズム (1)]

III-B. (1) $f(x) = 2 - e^x$ が与えられた時、 $x \in [0, 1]$ の範囲にある $f(x)$ の根を 2 分法を用いて求める場合を考える。解の存在する範囲を 10^{-8} 以下に限定するには 2 分法の繰り返し操作を最低何回行えば良いか答えよ。

(2) 2 分法により $f(x)$ の根を求めるプログラムを作成して問 1 で求めた回数繰り返し、繰り返し操作後の解の存在する x の区間の両端の値を 10 進 11 桁以降を切り捨てて求めよ。またその区間の両端の値の平均値を求め根の近似値として 10 進 11 桁以降を切り捨てて求めよ。作成したプログラムも提出すること。プログラミング言語は問わない。

(1) 2 分法では 1 回の操作で解が存在する

区間の幅は $\frac{1}{2}$ にならない。従って 1 回操作を繰り返す。

$$\inf\left\{n \in \mathbb{N} \mid \frac{1}{2^n} \leq 10^{-8}\right\} \dots \textcircled{1}$$

$\leftarrow \inf_{\text{下限}}(\text{TPR}) \Leftrightarrow \sup_{\text{上限}}(\text{TPR})$

対数関数は単調増加の関数であるため、不等式の両辺の対数をとると不等式の向きは変化しない。

$$\log_2(\text{左込}) = \log_2\left(\frac{1}{2^n}\right) = -n$$

$$\log_2(\text{右込}) = \log_2 10^{-8} = -8 \log_2 10$$

より

$$-n \leq -8 \log_2 10 \Leftrightarrow n \geq 8 \log_2 10$$

$$\text{また } 8 \log_2 10 = 26.59 \dots \text{より}$$

$$\textcircled{1} = \inf\left\{n \in \mathbb{N} \mid n \geq 8 \log_2 10\right\}$$

$$= \lceil 8 \log_2 10 \rceil = 27$$

従って 27 回

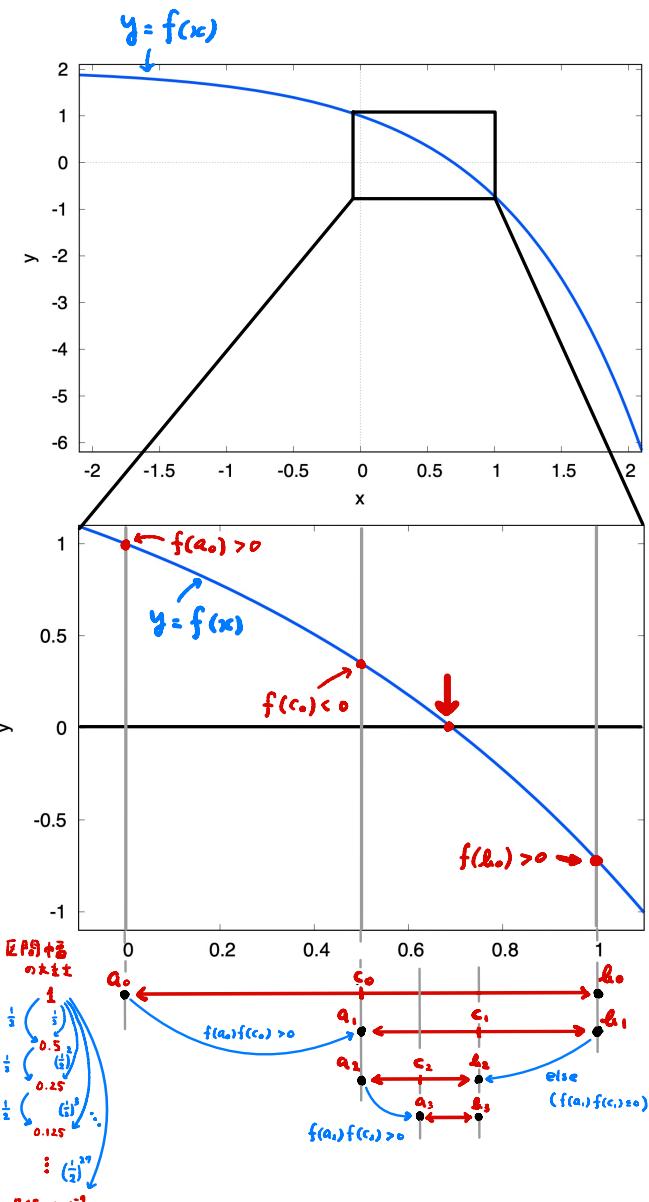
(2) 作成したプログラムの出力より

解が存在する区間幅

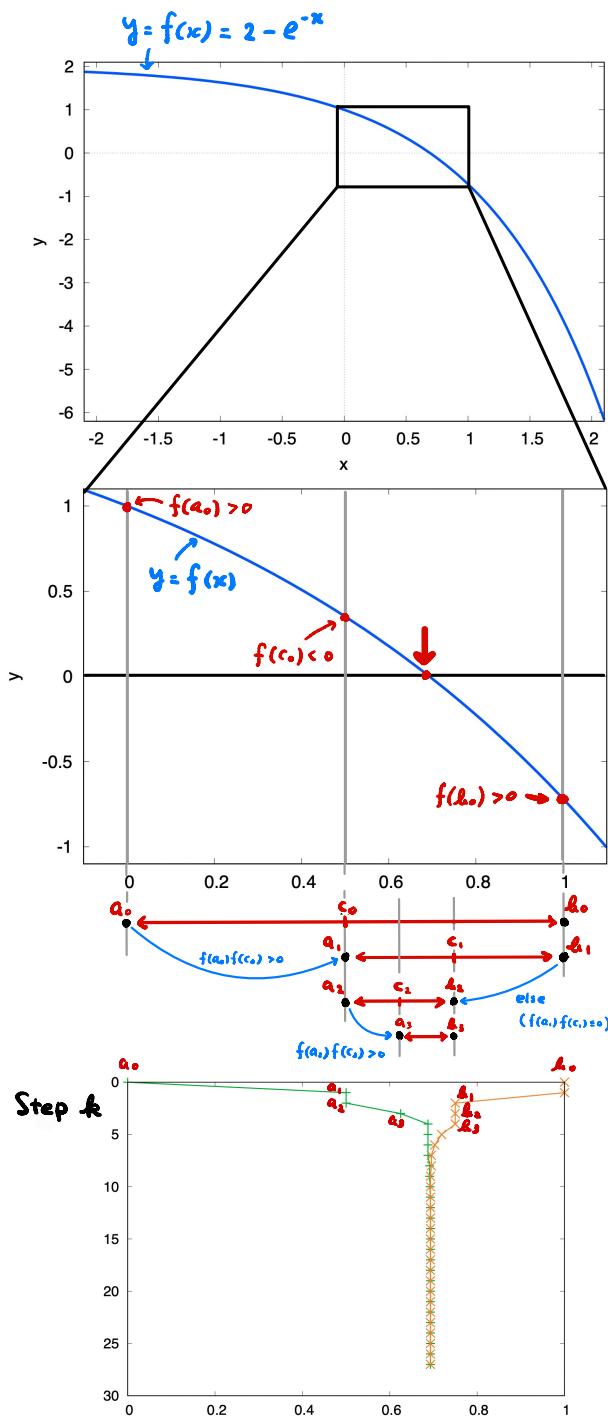
[0.6931471750, 0.6931471824]

根の近似値

0.6931471787

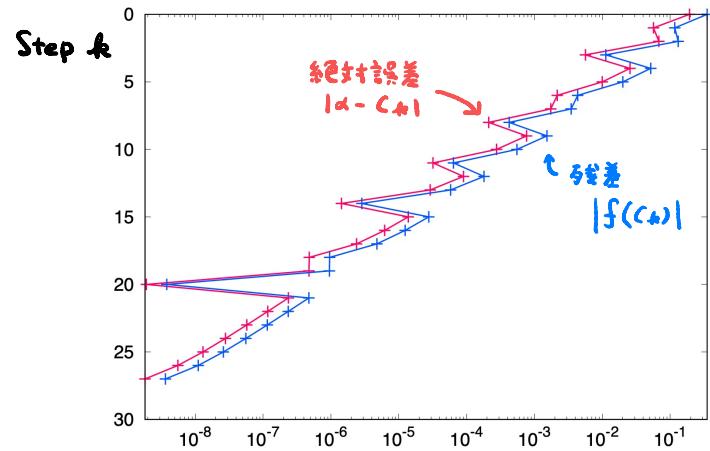


○ 絶対誤差と残差をプロット (Ⅲ-B)



```
(i = 0) [0.000000000000000, 1.000000000000000] 残差 0.351278729299872
(i = 1) [0.500000000000000, 1.000000000000000] 残差 0.117000016612675
(i = 2) [0.500000000000000, 0.750000000000000] 残差 0.131754042567778
(i = 3) [0.625000000000000, 0.750000000000000] 残差 0.011262530417708
(i = 4) [0.687500000000000, 0.750000000000000] 残差 0.051866773487977
(i = 5) [0.687500000000000, 0.718750000000000] 残差 0.020055527708696
(i = 6) [0.687500000000000, 0.703125000000000] 残差 0.004335330874331
(i = 7) [0.687500000000000, 0.695312500000000] 残差 0.003478832038738
(i = 8) [0.691406250000000, 0.695312500000000] 残差 0.000424433909775
(i = 9) [0.691406250000000, 0.693359375000000] 残差 0.001528152010194
(i = 10) [0.692382812500000, 0.693359375000000] 残差 0.000552097402978
(i = 11) [0.692871093750000, 0.693359375000000] 残差 0.000063891349343
(i = 12) [0.693115234375000, 0.693359375000000] 残差 0.000180256377712
(i = 13) [0.693115234375000, 0.693237304687500] 残差 0.000058178788786
(i = 14) [0.693115234375000, 0.693176269531250] 残差 0.000002857211600
(i = 15) [0.693145751953125, 0.693176269531250] 残差 0.000027660555759
(i = 16) [0.693145751953125, 0.693161010742188] 残差 0.000012401613872
(i = 17) [0.693145751953125, 0.693153381347656] 残差 0.000004772186584
(i = 18) [0.693145751953125, 0.693149566650391] 残差 0.000000957483854
(i = 19) [0.693145751953125, 0.693147659301758] 残差 0.000000949864782
(i = 20) [0.693146705627441, 0.693147659301758] 残差 0.00000003809308
(i = 21) [0.693146705627441, 0.693147182464600] 残差 0.000000473027794
(i = 22) [0.693146944046021, 0.693147182464600] 残差 0.000000234609257
(i = 23) [0.693147063255310, 0.693147182464600] 残差 0.000000115399978
(i = 24) [0.693147122859955, 0.693147182464600] 残差 0.000000055795335
(i = 25) [0.693147152662277, 0.693147182464600] 残差 0.000000025993014
(i = 26) [0.693147167563438, 0.693147182464600] 残差 0.000000011091853
(i = 27) [0.693147175014019, 0.693147182464600] 残差 0.000000003641272
```

解の存在する区間幅 [0.693147175014019, 0.693147182464600]
根の近似値 0.693147178739309
残差 0.000000003641272



- ・毎ステップ必ず絶対誤差・残差が減少するわけではない。

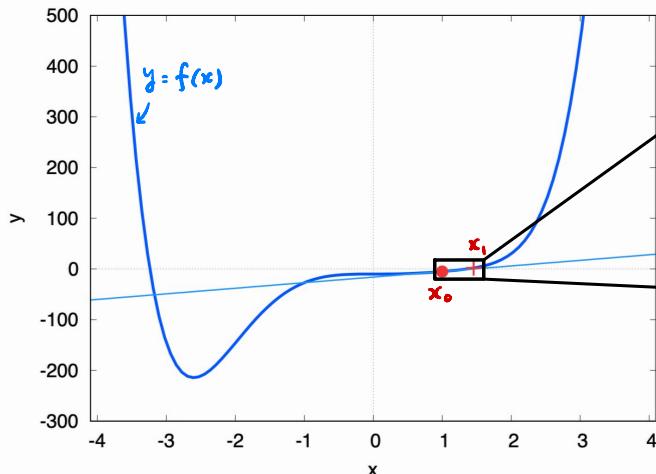
- ・このケースでは絶対誤差と残差が似た動きをしている

→ 異常な動き 小さければ絶対誤差も小さい

[〆切] 2020/10/22 19:00

[IV. 求根アルゴリズム (2)]

IV-A. ニュートン法により x の多項式 $f(x) = x^6 - 7x^4 + 11x^3 - 10$ の根を求めるプログラムを初期値を $x = 1$ として作成し、根の真値を 10 進 16 桁まで示した $\alpha = 1.357271472605337$ と比べて絶対誤差が 10^{-8} 以下となる最低の反復回数 n を求めよ。また n 回反復した時の根の近似値と絶対誤差も求めよ。近似値は 10 進 11 桁以降を切り捨てて求め、絶対誤差は 10 進 4 桁以降を切り捨ててよ。作成したプログラムも提出すること。プログラミング言語は問わない。



Python 3.6 のやり方

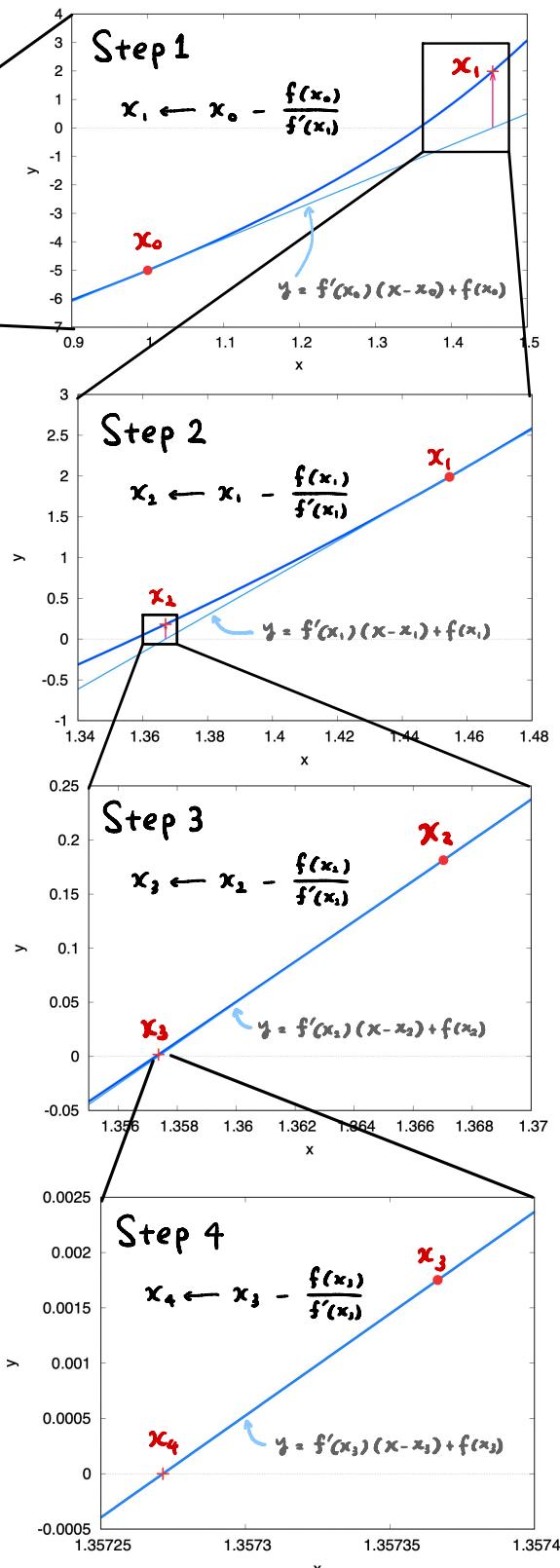
```
#def f(x):
#    x2 = x*x; x3 = x2*x
#    return ((x2-7)*x+11)*x3-10 } 見やすい

#define dfdx(x):
#    x2 = x*x
#    return ((6*x2-28)*x+33)*x2 } 違い

def f_and_dfdx(x):
    x2 = x*x; x3 = x2*x
    return ((x2-7)*x+11)*x3-10, ((6*x2-28)*x+33)*x2
```

```
x = 1
alpha = 1.357271472605337
i = 0
for i in range(100):
    f_0, f_1 = f_and_dfdx(x)
    if abs(alpha - x) <= 10**-8:
        break
    else:
        x = x - f_0 / f_1
    i += 1
```

```
(i=0) 1.000000000000000 誤差 0.357271472605337 残差 5.000000000000000
(i=1) 1.454545454545455 誤差 0.097273981940118 残差 1.988161852738914
(i=2) 1.367024392736051 誤差 0.009752920130714 残差 0.181403933828856
(i=3) 1.357366563718892 誤差 0.000095091113555 残差 0.001751653873868
(i=4) 1.357271481600892 誤差 0.000000008995555 残差 0.000000165689585
```

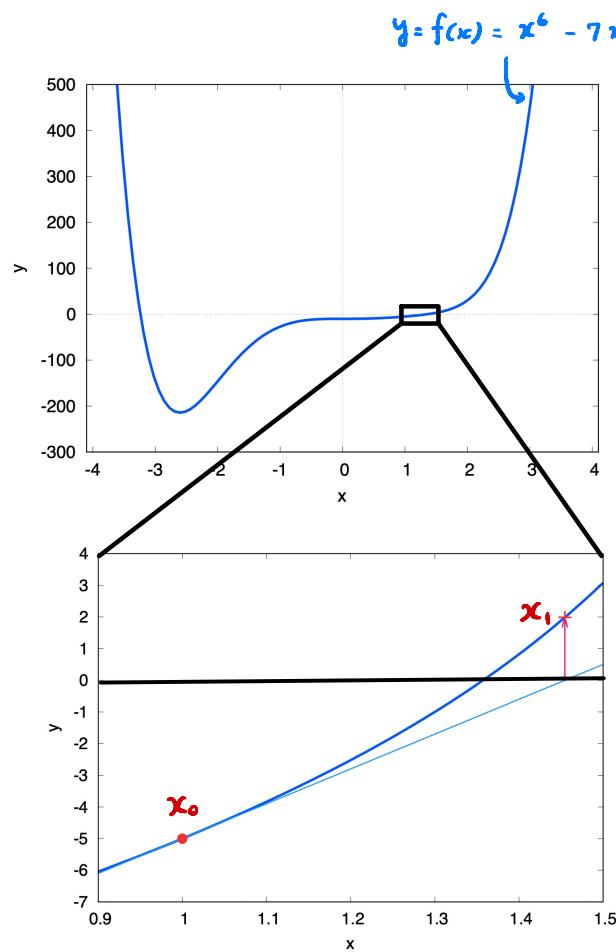


反復回数 4

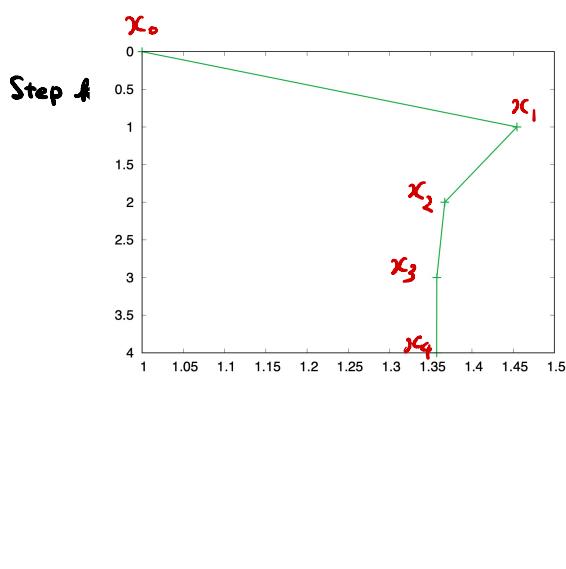
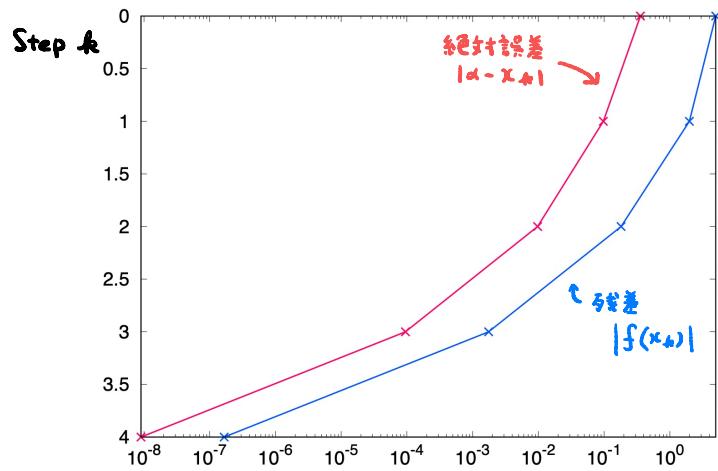
根の近似値 1.357271481

誤差 8.99×10^{-9}

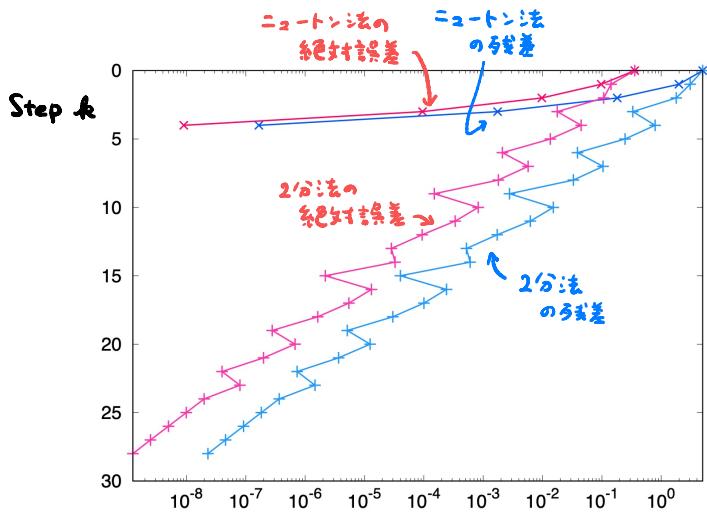
④ 絶対誤差と残差をプロット (IV-A)



```
(i=0) 1.000000000000000 誤差 0.357271472605337 残差 5.000000000000000
(i=1) 1.454545454545455 誤差 0.097273981940118 残差 1.988161852738914
(i=2) 1.367024392736051 誤差 0.009752920130714 残差 0.181403933828856
(i=3) 1.357366563718892 誤差 0.000095091113555 残差 0.001751653873868
(i=4) 1.357271481600892 誤差 0.000000089955555 残差 0.000000165689585
```



④ 2分法とニュートン法の比較



① ニュートンの誤差公式の証明

テイラーの定理より $f(x)$ は $x = x_k$ において

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(\xi_k)}{2!} (x - x_k)^2$$

を満たす ξ_k を $x = x_k$ の附近に持つ。ここで $x = \alpha$ とおくと
 $f(\alpha) = 0$ より

$$0 = f(x_k) + f'(x_k)(\alpha - x_k) + \frac{f''(\xi_k)}{2!} (\alpha - x_k)^2$$

\Leftrightarrow

\Leftrightarrow

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

が成立する。

さらに 左辺を $(\alpha - x_k)^2$ で割り、 $k \rightarrow \infty$ の極限を考えると

$$\lim_{k \rightarrow \infty} \frac{\alpha - x_{k+1}}{(\alpha - x_k)^2} = \lim_{k \rightarrow \infty} \left\{ -\frac{1}{2} \frac{f''(\xi_k)}{f'(x_k)} \right\} = -\frac{f''(\alpha)}{2f'(\alpha)}$$

[〆切] 2020/10/22 19:00

[III. 求根アルゴリズム (1)]

III-B. (1) $f(x) = 2 - e^x$ が与えられた時、 $x \in [0, 1]$ の範囲にある $f(x)$ の根を 2 分法を用いて求める場合を考える。解の存在する範囲を 10^{-8} 以下に限定するには 2 分法の繰り返し操作を最低何回行えば良いか答えよ。

(2) 2 分法により $f(x)$ の根を求めるプログラムを作成して問 1 で求めた回数繰り返し、繰り返し操作後の解の存在する x の区間の両端の値を 10 進 11 桁以降を切り捨てて求めよ。またその区間の両端の値の平均値を求め根の近似値として 10 進 11 桁以降を切り捨てて求めよ。作成したプログラムも提出すること。プログラミング言語は問わない。

(1) 2 分法では 1 回の操作で解が存在する

区間の幅は $\frac{1}{2}$ にならない。従って 1 回操作を繰り返す。

$$\inf\left\{n \in \mathbb{N} \mid \frac{1}{2^n} \leq 10^{-8}\right\} \dots \textcircled{1}$$

$\leftarrow \inf_{\text{下限}}(\text{TPR}) \Leftrightarrow \sup_{\text{上限}}(\text{TPR})$

対数関数は単調増加の関数であるため、不等式の両辺の対数をとると不等式の向きは変化しない。

$$\log_2(\text{左込}) = \log_2\left(\frac{1}{2^n}\right) = -n$$

$$\log_2(\text{右込}) = \log_2 10^{-8} = -8 \log_2 10$$

より

$$-n \leq -8 \log_2 10 \Leftrightarrow n \geq 8 \log_2 10$$

$$\text{また } 8 \log_2 10 = 26.59 \dots \text{より}$$

$$\textcircled{1} = \inf\left\{n \in \mathbb{N} \mid n \geq 8 \log_2 10\right\}$$

$$= \lceil 8 \log_2 10 \rceil = 27$$

従って 27 回

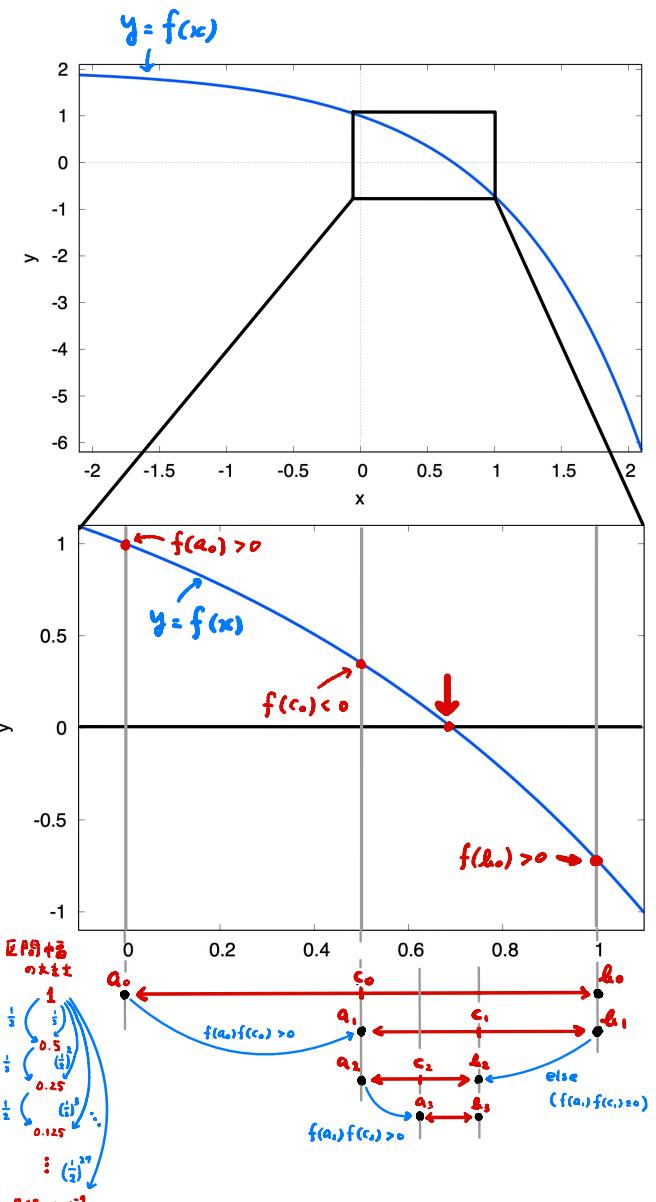
(2) 作成したプログラムの出力より

解が存在する区間幅

[0.6931471750, 0.6931471824]

根の近似値

0.6931471787

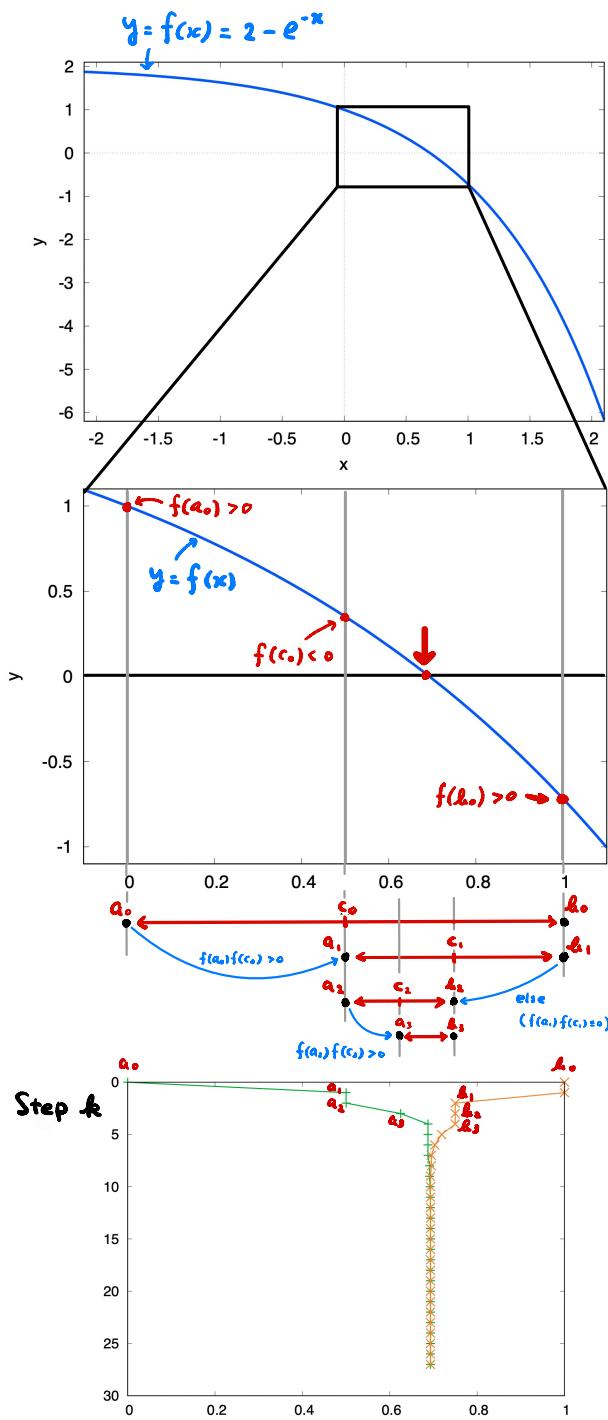


Python 3.6 の例

```
import math
def f(x):
    return 2 - math.exp(x)
```

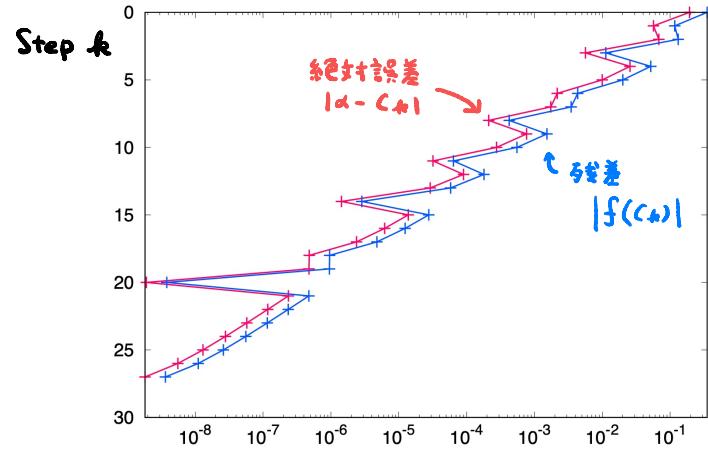
```
a = 0; b = 1
for i in range(1, 27):
    #c = (a+b)/2
    c = a + (b-a)/2 # 平均値の計算を旨とする
    if f(a) * f(c) > 0:
        a = c
    else:
        b = c
```

○ 絶対誤差と残差をプロット (Ⅲ-B)



```
(i = 0) [0.000000000000000, 1.000000000000000] 残差 0.351278729299872
(i = 1) [0.500000000000000, 1.000000000000000] 残差 0.117000016612675
(i = 2) [0.500000000000000, 0.750000000000000] 残差 0.131754042567778
(i = 3) [0.625000000000000, 0.750000000000000] 残差 0.011262530417708
(i = 4) [0.687500000000000, 0.750000000000000] 残差 0.051866773487977
(i = 5) [0.687500000000000, 0.718750000000000] 残差 0.020055527708696
(i = 6) [0.687500000000000, 0.703125000000000] 残差 0.004335330874331
(i = 7) [0.687500000000000, 0.695312500000000] 残差 0.003478832038738
(i = 8) [0.691406250000000, 0.695312500000000] 残差 0.000424433909775
(i = 9) [0.691406250000000, 0.693359375000000] 残差 0.001528152010194
(i = 10) [0.692382812500000, 0.693359375000000] 残差 0.000552097402978
(i = 11) [0.692871093750000, 0.693359375000000] 残差 0.000063891349343
(i = 12) [0.693115234375000, 0.693359375000000] 残差 0.000180256377712
(i = 13) [0.693115234375000, 0.693237304687500] 残差 0.000058178788786
(i = 14) [0.693115234375000, 0.693176269531250] 残差 0.000002857211600
(i = 15) [0.693145751953125, 0.693176269531250] 残差 0.000027660555759
(i = 16) [0.693145751953125, 0.693161010742188] 残差 0.000012401613872
(i = 17) [0.693145751953125, 0.693153381347656] 残差 0.000004772186584
(i = 18) [0.693145751953125, 0.693149566650391] 残差 0.000000957483854
(i = 19) [0.693145751953125, 0.693147659301758] 残差 0.000000949864782
(i = 20) [0.693146705627441, 0.693147659301758] 残差 0.00000003809308
(i = 21) [0.693146705627441, 0.693147182464600] 残差 0.000000473027794
(i = 22) [0.693146944046021, 0.693147182464600] 残差 0.000000234609257
(i = 23) [0.693147063255310, 0.693147182464600] 残差 0.000000115399978
(i = 24) [0.693147122859955, 0.693147182464600] 残差 0.000000055795335
(i = 25) [0.693147152662277, 0.693147182464600] 残差 0.000000025993014
(i = 26) [0.693147167563438, 0.693147182464600] 残差 0.000000011091853
(i = 27) [0.693147175014019, 0.693147182464600] 残差 0.000000003641272
```

解の存在する区間幅 [0.693147175014019, 0.693147182464600]
根の近似値 0.693147178739309
残差 0.000000003641272



- ・毎ステップ必ず絶対誤差・残差が減少するわけではない。

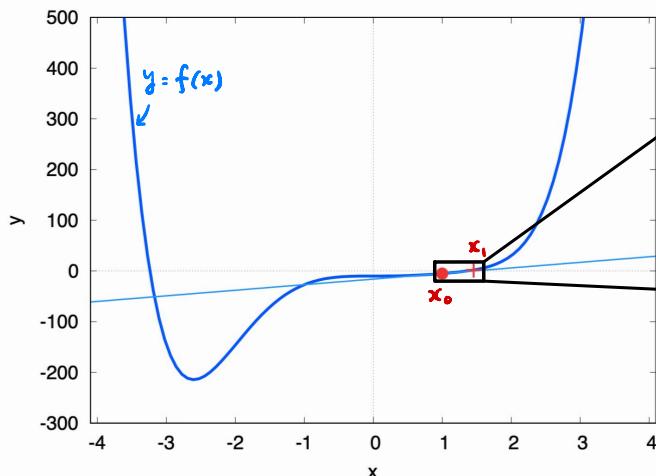
- ・このケースでは絶対誤差と残差が似た動きをしている

→ 異常な動き 小さければ絶対誤差も小さい

[〆切] 2020/10/22 19:00

[IV. 求根アルゴリズム (2)]

IV-A. ニュートン法により x の多項式 $f(x) = x^6 - 7x^4 + 11x^3 - 10$ の根を求めるプログラムを初期値を $x = 1$ として作成し、根の真値を 10 進 16 桁まで示した $\alpha = 1.357271472605337$ と比べて絶対誤差が 10^{-8} 以下となる最低の反復回数 n を求めよ。また n 回反復した時の根の近似値と絶対誤差も求めよ。近似値は 10 進 11 桁以降を切り捨てて求め、絶対誤差は 10 進 4 桁以降を切り捨ててよ。作成したプログラムも提出すること。プログラミング言語は問わない。



Python 3.6 のやり方

```
#def f(x):
#    x2 = x*x; x3 = x2*x
#    return ((x2-7)*x+11)*x3-10 } 見やすい

#define dfdx(x):
#    x2 = x*x
#    return ((6*x2-28)*x+33)*x2 } 違い

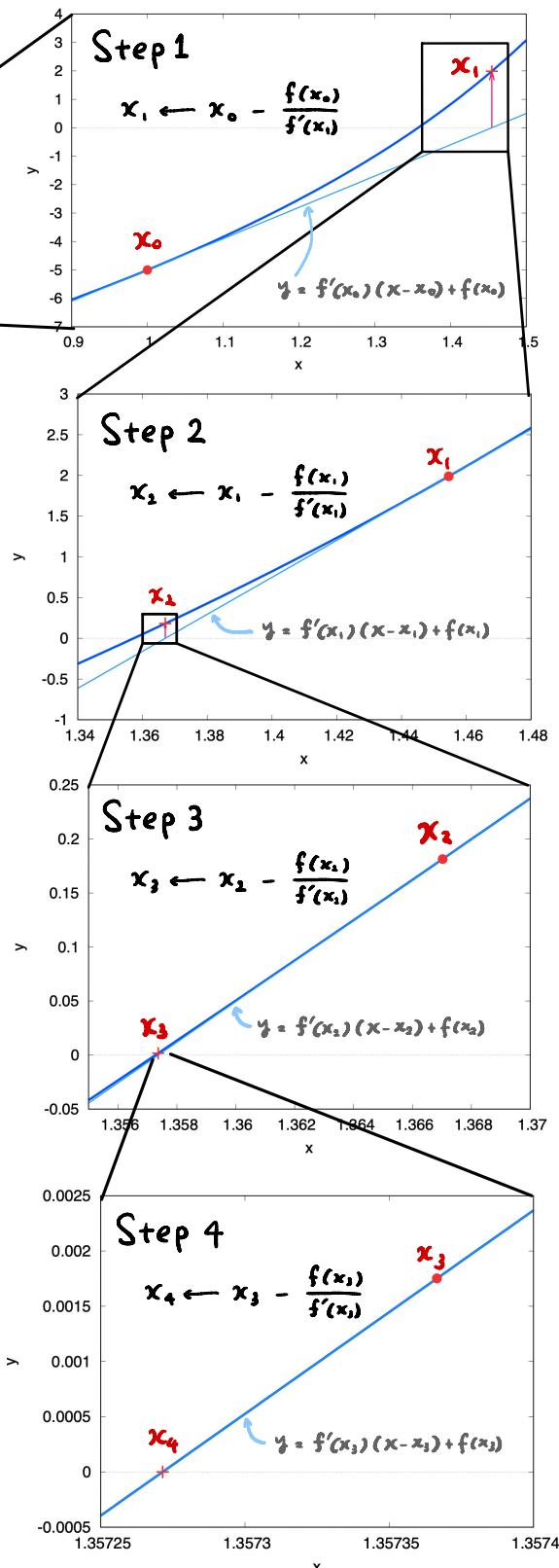
def f_and_dfdx(x):
    x2 = x*x; x3 = x2*x
    return ((x2-7)*x+11)*x3-10, ((6*x2-28)*x+33)*x2
```

```
x = 1
alpha = 1.357271472605337
i = 0
for i in range(100):
    f_0, f_1 = f_and_dfdx(x)
    if abs(alpha - x) <= 10**-8:
        break
    else:
        x = x - f_0 / f_1
    i += 1
```

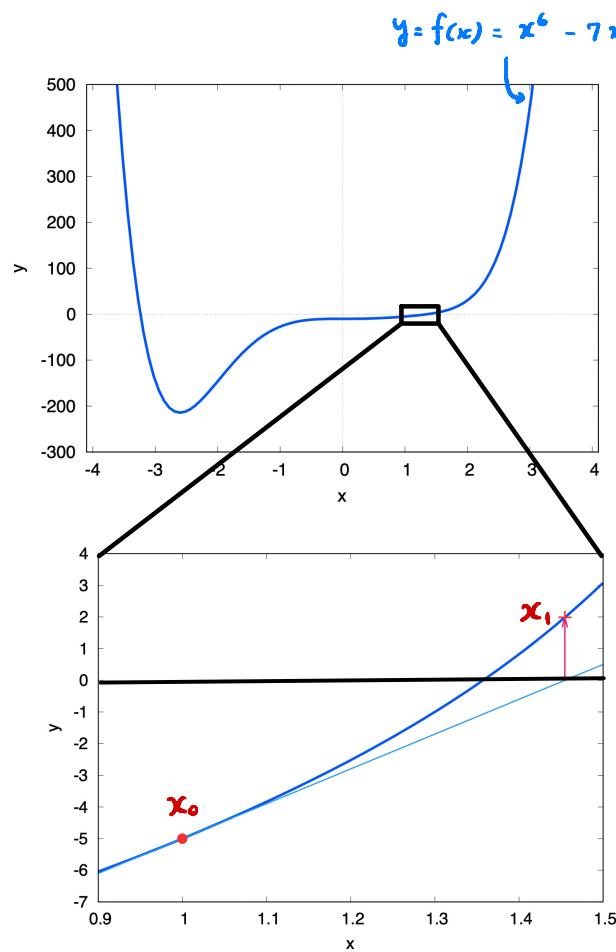
```
(i=0) 1.000000000000000 誤差 0.357271472605337 残差 5.000000000000000
(i=1) 1.454545454545455 誤差 0.097273981940118 残差 1.988161852738914
(i=2) 1.367024392736051 誤差 0.009752920130714 残差 0.181403933828856
(i=3) 1.357366563718892 誤差 0.000095091113555 残差 0.001751653873868
(i=4) 1.357271481600892 誤差 0.000000008995555 残差 0.000000165689585
```

反復回数 4

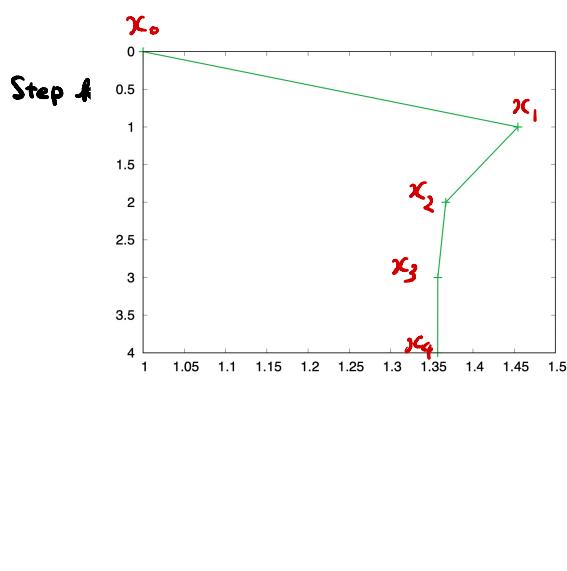
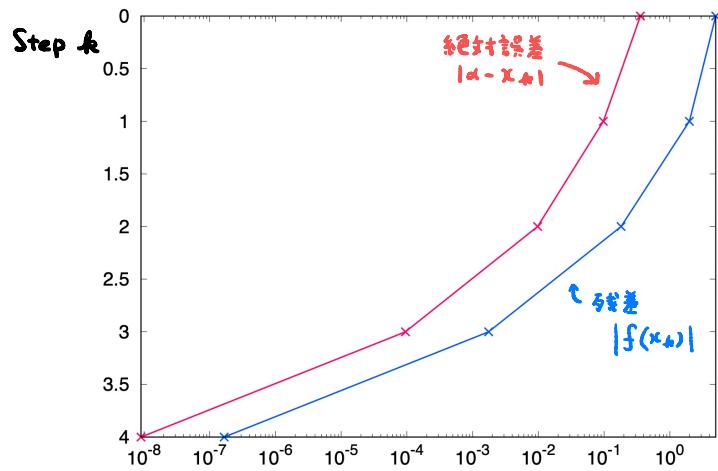
根の近似値 1.357271481

誤差 8.99×10^{-9} 

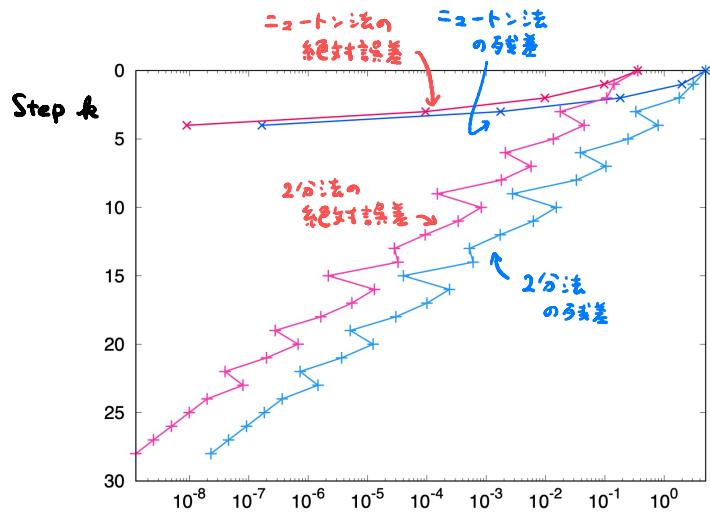
④ 絶対誤差と残差をプロット (IV-A)



```
(i=0) 1.000000000000000 誤差 0.357271472605337 残差 5.000000000000000
(i=1) 1.454545454545455 誤差 0.097273981940118 残差 1.988161852738914
(i=2) 1.367024392736051 誤差 0.009752920130714 残差 0.181403933828856
(i=3) 1.357366563718892 誤差 0.000095091113555 残差 0.001751653873868
(i=4) 1.357271481600892 誤差 0.000000089955555 残差 0.000000165689585
```



④ 2分法とニュートン法の比較



○ ニュートンの誤差公式の証明

テイラーの定理より $f(x)$ は $x = x_k$ において

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(\xi_k)}{2!} (x - x_k)^2$$

を満たす ξ_k を $x = x_k$ の附近に持つ。ここで $x = \alpha$ とおくと
 $f(\alpha) = 0$ より

$$0 = f(x_k) + f'(x_k)(\alpha - x_k) + \frac{f''(\xi_k)}{2!} (\alpha - x_k)^2$$

$$\Leftrightarrow 0 = \frac{f(x_k)}{f'(x_k)} + \alpha - x_k + \frac{1}{2} \frac{f''(\xi_k)}{f'(x_k)} (\alpha - x_k)^2 \quad \left(\begin{array}{l} \text{両辺を } f'(x_k) \\ \text{で割る。 } f'(x_k) \neq 0 \\ \text{を仮定} \end{array} \right)$$

$$\Leftrightarrow \alpha - \left(x_k - \frac{f(x_k)}{f'(x_k)} \right) = -\frac{1}{2} \frac{f''(\xi_k)}{f'(x_k)} (\alpha - x_k)^2$$

$$\therefore x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \text{ である。}$$

$$\alpha - x_{k+1} = -\frac{1}{2} \underbrace{(\alpha - x_k)^2}_{x_k \approx \alpha \approx} \frac{f''(\xi_k)}{f'(x_k)}$$

が成立する。

$x_k \approx \alpha$ は
 ある程度近くなると
 毎回誤差が小さくなる

さらに 両辺を $(\alpha - x_k)^2$ で割り、 $k \rightarrow \infty$ の極限を考える

$$\lim_{k \rightarrow \infty} \frac{\alpha - x_{k+1}}{(\alpha - x_k)^2} = \lim_{k \rightarrow \infty} \left\{ -\frac{1}{2} \frac{f''(\xi_k)}{f'(x_k)} \right\} = -\frac{f''(\alpha)}{2f'(\alpha)}$$