Prova I

Exercício de Avaliação 01

Matheus Batista Honório – 20190097098

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Relatório apresentado à disciplina Cálculo Numérico

do curso de Engenharia de Computação.

Professor: Moisés Dantas dos Santos

João Pessoa, 2021

Para o desenvolvimento e resolução das questões, foi-se utilizado o **Google Colab** disponibilizado pelo monitor Joelder Victor Antonino Aguiar e o método escolhido para solucionar todas as questões foi o da Bissecção. O Colab pode ser acessado por este **link** com o e-mail acadêmico.

A seguir, uma breve explicação do método da bisseção ou método da bissecção.

Método da Bisseção

De acordo com o livro Cálculo Numérico – Introdução à Matemática Computacional:

Dados ε > 0 e f (x) contínua em [a, b] com f (a)f (b) < 0, o método da bisseção para a determinação de uma raiz da equação f (x) = 0 consiste em ir dividindo o intervalo ao meio até que ele fique suficientemente pequeno; daí, escolhemos o ponto médio do intervalo como sendo uma raiz aproximada. Consiste em se executar os seguintes passos:

1. Calculamos m = a+b 2 o ponto médio do intervalo; se f (m) = 0 então m é uma raiz da equação e encerramos;
2. Se Δ = |b − a| < ε, então dizemos que m é uma raiz aproximada da equação e encerramos;
3. Se os sinais de f (a) e f (m) coincidirem, então redefinimos a = m;
4. Se os sinais de f (b) e f (m) coincidirem, então redefinimos b = m;
5. Retornamos ao item (1).

Esse método faz uma pesquisa binária no intervalo [a, b] em busca da raiz da equação.

Questão 1

Resumidamente a questão propõe que o erro seja menor que 1% e sabendo que w = 10 N/m e os extremos do cabo têm coordenadas: x0 = 0, y0 = 5 m, xf = 50 m, yf = 12 m. Com a fórmula:

$$\mathbf{y = (TA / w)\ (cosh(w.x/TA) + y0 – TA / w}$$

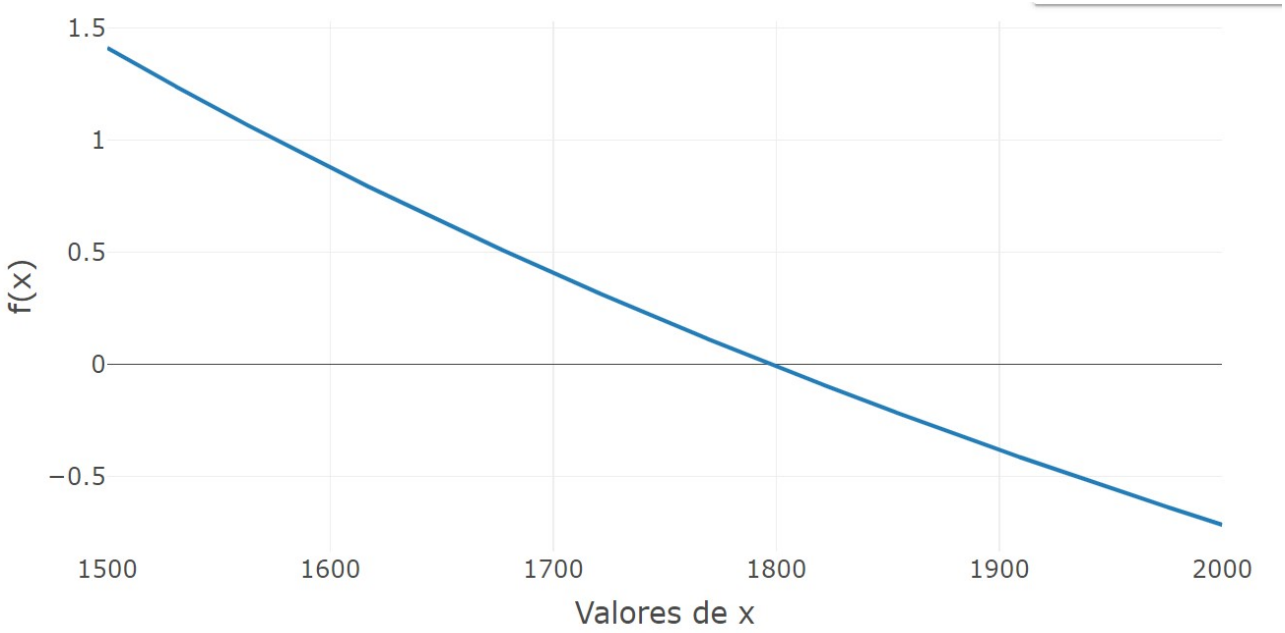
Após substituir algumas variáveis na função acima e utilizando o Symbolab, obtemos:

$$\mathbf{Fx = (x/10)*(cosh(500/x))-x/10-7}$$

Utilizando essa função no Colab e printando-a, exibe:

$$\frac{x\cosh\left(\frac{500}{x}\right)}{10} - \frac{x}{10} - 7$$

Plotando o gráfico com os pontos de 1500 até 2000:



Para deixar o método mais preciso e utilizando o método da Bissecção mencionado acima:
Os pontos **a** e **b** são respectivamente 1750 e 1850, com o Epsilon(ε) igual a 0.001, que significa que a taxa de erro será abaixo de 0,1% para respeitar a taxa de erro proposta na questão.

```
a = 1750
b = 1850
epsilon=0.001
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 1797.265625
Iterações: 8
```

Nos retornando como solução 1797,265625 e sendo feitas 8 iterações para ter obtido o resultado.

Uma visualização mais precisa das iterações, também pode ser feito em formato de tabela, como mostrado a seguir:

Então assim, é necessária uma força horizontal, ou seja, o T_A de 1797,265625.

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	1750.000	1800.000000	1800.000000	0.191580	-0.010787	-0.010787	50.000000
2	1775.000	1800.000000	1775.000000	0.088943	-0.010787	0.088943	25.000000
3	1787.500	1800.000000	1787.500000	0.038722	-0.010787	0.038722	12.500000
4	1793.750	1800.000000	1793.750000	0.013880	-0.010787	0.013880	6.250000
5	1796.875	1800.000000	1796.875000	0.001524	-0.010787	0.001524	3.125000
6	1796.875	1798.437500	1798.437500	0.001524	-0.004637	-0.004637	1.562500
7	1796.875	1797.656250	1797.656250	0.001524	-0.001558	-0.001558	0.781250
8	1796.875	1797.265625	1797.265625	0.001524	-0.000017	-0.000017	0.390625

Questão 2

A velocidade de ascensão de um foguete em vôo vertical próximo à superfície terrestre (**v**) pode ser aproximada pela seguinte expressão:

$$v = u.ln\left(\frac{M_0}{M_0 - c.t}\right) + g.t$$

A questão propõe os valores:
u = 200 m/s, m0 = 1600 Kg, g = 9.8 m/s2 e q = 27 Kg/s, **v=100 m/s.** Com o intervalo inicial ***[6, 8]*** e tolerância de **0,8%**.

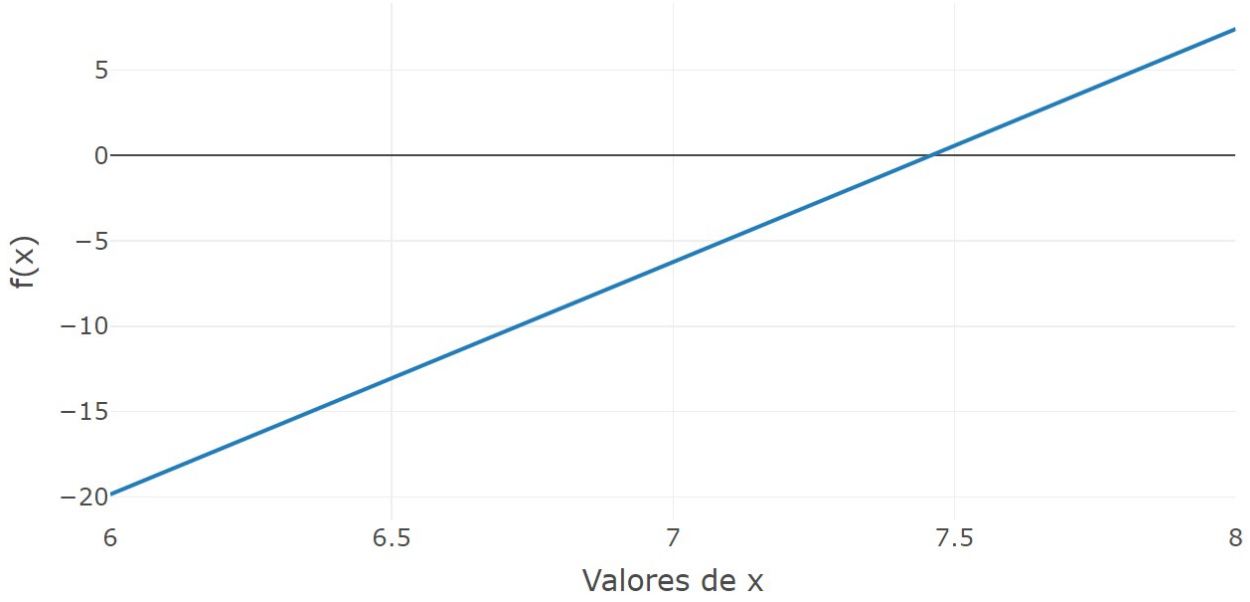
Após fazer a substituição dos valores na fórmula e formatarmos para que seja aceito para python, obtemos:

Fx= 200*sy.ln(1600/(1600-27*x)) + 9.8*x – 100

Utilizando essa função no Colab e printando-a, exibe:

$$9.8x + 200\log\left(\frac{1600}{1600 - 27x}\right) - 100$$

Plotando o gráfico com os pontos de 6 até 8:



O resultado pelo método é igual a 7,460938 e 8 iterações:

```
a = 6
b = 8
epsilon=0.008
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 7.460938
Iterações: 8
```

Para uma melhor visualização das iterações, utilizamos o formato de tabela:

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	7.000000	8.000000	7.000000	-6.259009	7.405154	-6.259009	1.000000
2	7.000000	7.500000	7.500000	-6.259009	0.563741	0.563741	0.500000
3	7.250000	7.500000	7.250000	-2.849945	0.563741	-2.849945	0.250000
4	7.375000	7.500000	7.375000	-1.143682	0.563741	-1.143682	0.125000
5	7.437500	7.500000	7.437500	-0.290116	0.563741	-0.290116	0.062500
6	7.437500	7.468750	7.468750	-0.290116	0.136776	0.136776	0.031250
7	7.453125	7.468750	7.453125	-0.076679	0.136776	-0.076679	0.015625
8	7.453125	7.460938	7.460938	-0.076679	0.030046	0.030046	0.007812

Explicitando a tolerância de cada iteração, respectivamente:

1. **100%** de taxa de erro: dos pontos 7,000 à 8,000 e x sendo igual a 7,000;
2. **50%** de taxa de erro: dos pontos 7,000 à 7,500 e x sendo igual a 7,500;
3. **25%** de taxa de erro: dos pontos 7,250 à 7,500 e x sendo igual a 7,250;
4. **12,5%** de taxa de erro: dos pontos 7,375 à 7,500 e x sendo igual a 7,375;
5. **6,25%** de taxa de erro: dos pontos 7,437 à 7,500 e x sendo igual a 7,4375;
6. **3,125%** de taxa de erro: dos pontos 7,4375 à 7,46875 e x sendo igual a 7,46875;
7. **1,5625%** de taxa de erro: dos pontos 7,453125 à 7,46875 e x sendo igual a 7,453125;
8. **0,7812%** de taxa de erro: dos pontos 7,453125 à 7,460938 e x sendo igual a 7,460938;

Questão 3

O objetivo da questão é calcular os três primeiros instantes em que o CG passa por sua posição de equilíbrio e erro tolerável menor que **0,1%**. Os dados da questão nos fornece que, em um automóvel, a massa **m** do carro é 1.2 10^6 g, a constante da mola **k** vale 1,25.10^9 g/s^2 e o amortecimento **c** corresponde a 1,4.10^7 g/s. Sabe-se que o deslocamento vertical do centro de gravidade do carro é dado por:

$$\mathbf{x(t) = x_0\,(t)\,e^{-nt}\,(\,cos(pt) + n/p(sen(pt)}$$

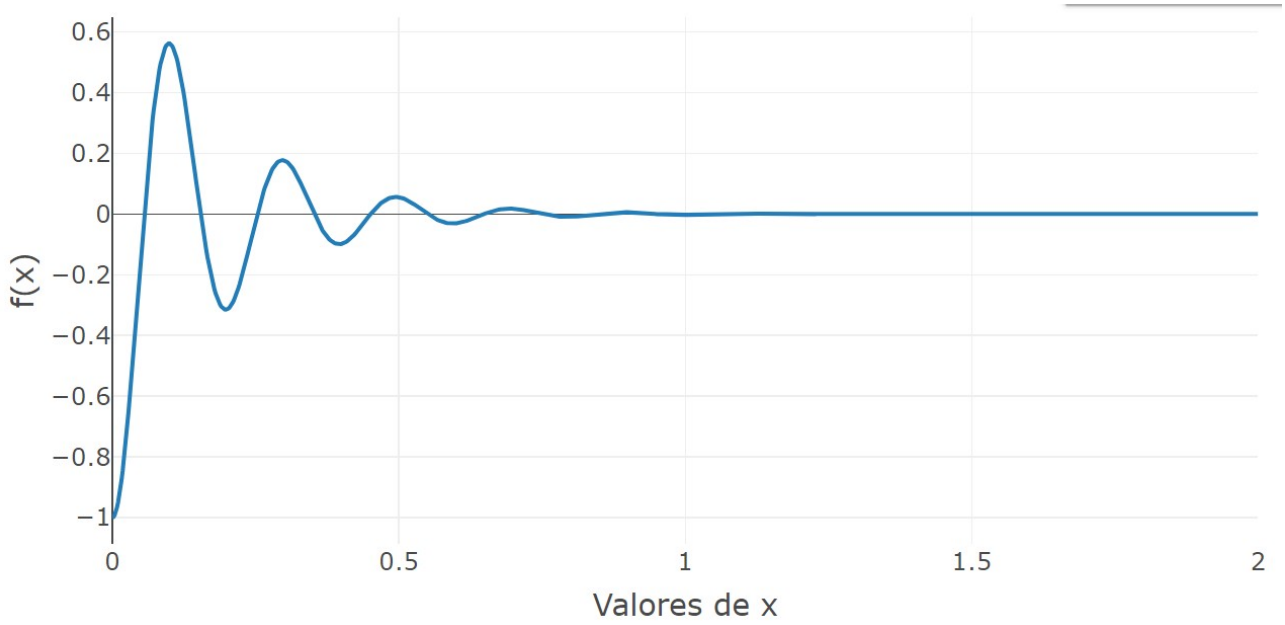
Após fazer a substituição dos valores na fórmula e formatarmos para que seja aceito para python, obtemos:

Fx=-1*exp(-5.833*x)*(cos(31.743*x) + (5.833 / 31.743)*sin(31.743*x))

Utilizando essa função no Colab e printando-a, exibe:

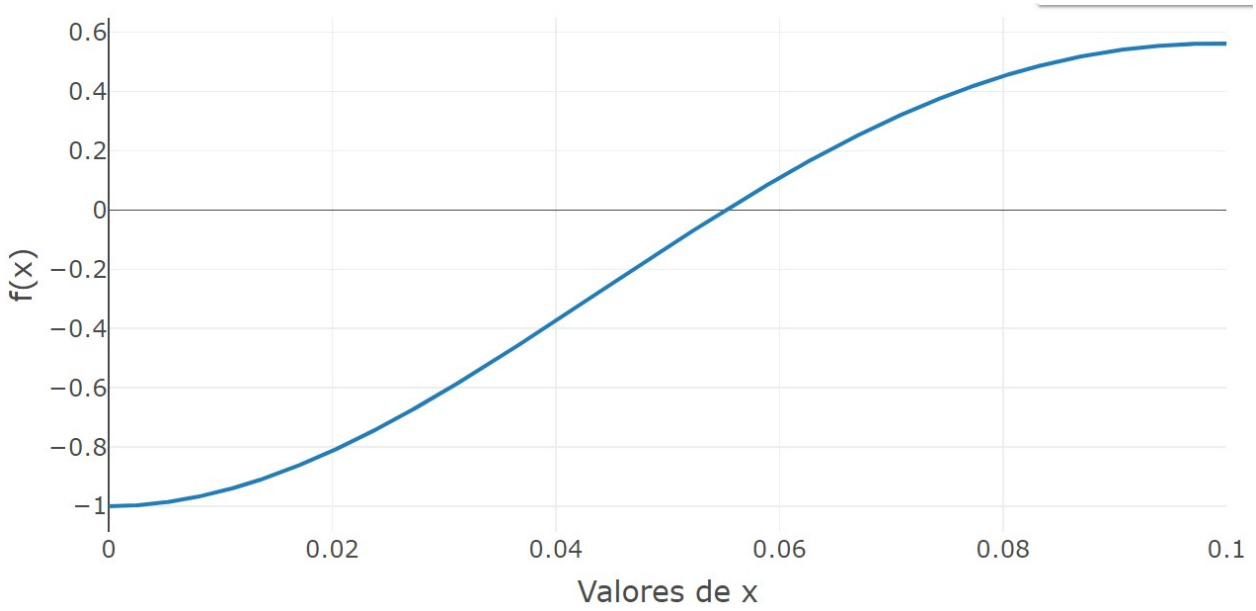
$$-\left(0.18375704879816\sin\left(31.743x\right)+\cos\left(31.743x\right)\right)e^{-5.833x}$$

Plotando o gráfico de 0 à 2, obtemos:



Para isolarmos os 3 primeiros pontos de equilibro, como vemos no gráfico que ocorrem antes do ponto 0,5, faremos uma primeira análise nos pontos de 0 à 0,1:

Ao plotarmos o gráfico de 0 à 0,1, obtemos:



O gráfico nos apresenta como resultado uma solução visualmente indicada entre os pontos 0,04 e 0,06, como mostrados a seguir:

```
a = 0
b = 0.1
epsilon=0.001
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 0.055469
Iterações: 7
```

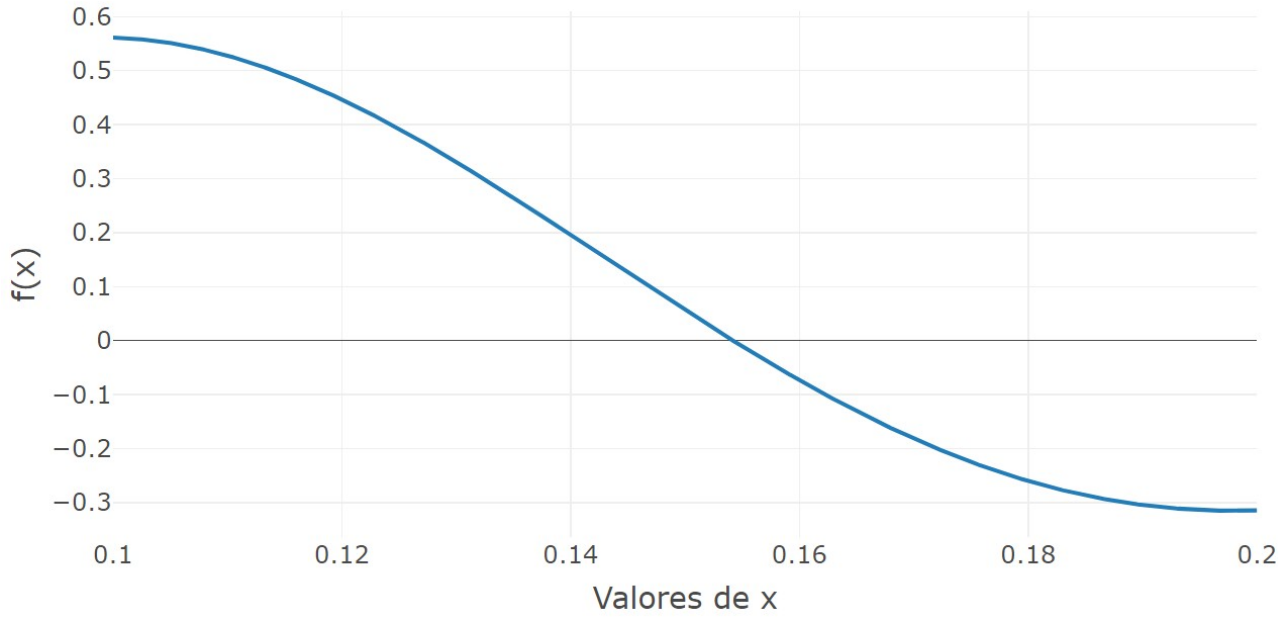
Para visualizarmos as iterações detalhadamente, utilizemos o formato tabelado dos dados:

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	0.050000	0.100000	0.050000	-0.125038	0.561109	-0.125038	0.050000
2	0.050000	0.075000	0.075000	-0.125038	0.385803	0.385803	0.025000
3	0.050000	0.062500	0.062500	-0.125038	0.161952	0.161952	0.012500
4	0.050000	0.056250	0.056250	-0.125038	0.024176	0.024176	0.006250
5	0.053125	0.056250	0.053125	-0.049322	0.024176	-0.049322	0.003125
6	0.054688	0.056250	0.054688	-0.012253	0.024176	-0.012253	0.001563
7	0.054688	0.055469	0.055469	-0.012253	0.006046	0.006046	0.000781

Assim, obtemos o primeiro ponto de equilíbrio, sendo: 0.055469, após 7 iterações.

Para o segundo ponto, analisaremos os pontos entre 0.1 e 0.2, assim:

Ao plotarmos o gráfico de 0,1 à 0,2, obtemos:



O gráfico nos apresenta como resultado uma solução visualmente indicada entre os pontos 0,14 e 0,16, como mostrados a seguir:

```
a = 0.1
b = 0.2
epsilon=0.001
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 0.153906
Iterações: 7
```

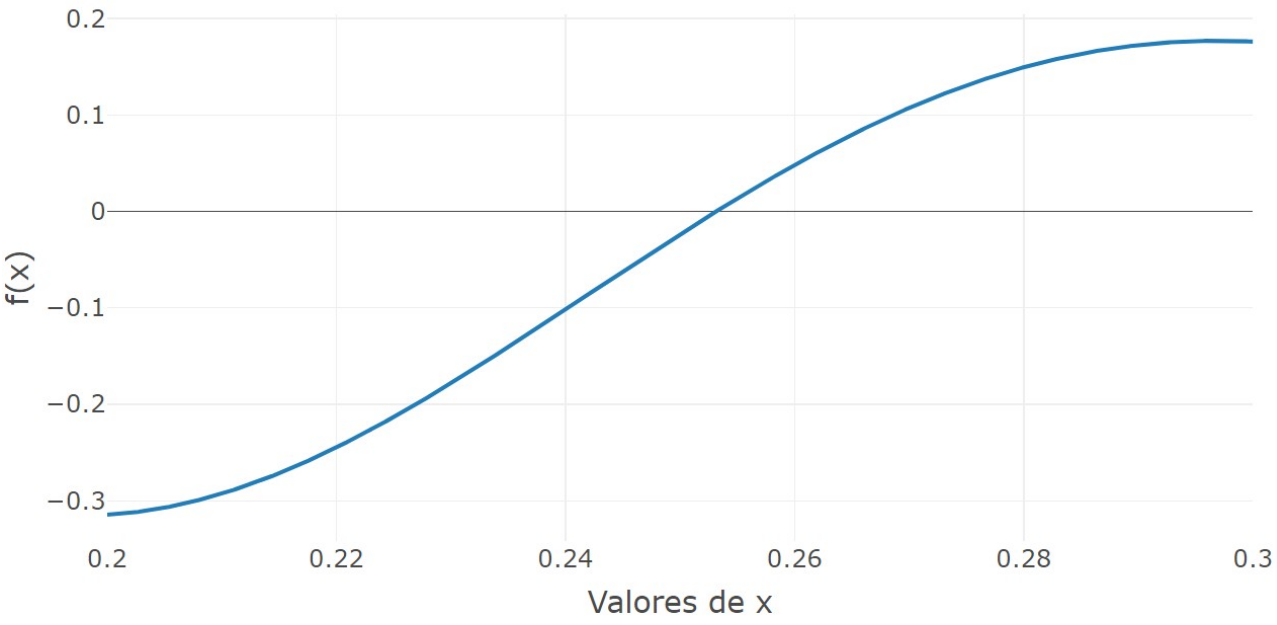
Para visualizarmos as iterações detalhadamente, utilizemos o formato tabelado dos dados:

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	0.150000	0.200000	0.150000	0.056068	-0.314499	0.056068	0.050000
2	0.150000	0.175000	0.175000	0.056068	-0.224877	-0.224877	0.025000
3	0.150000	0.162500	0.162500	0.056068	-0.102872	-0.102872	0.012500
4	0.150000	0.156250	0.156250	0.056068	-0.026842	-0.026842	0.006250
5	0.153125	0.156250	0.153125	0.013929	-0.026842	0.013929	0.003125
6	0.153125	0.154688	0.154688	0.013929	-0.006651	-0.006651	0.001563
7	0.153906	0.154688	0.153906	0.003593	-0.006651	0.003593	0.000781

Assim, obtemos o segundo ponto de equilíbrio, sendo: 0.153906, após 7 iterações.

Para o terceiro e último ponto, analisaremos os pontos entre 0.2 e 0.3, assim:

Ao plotarmos o gráfico de 0,2 à 0,3, obtemos:



O gráfico nos apresenta como resultado uma solução visualmente indicada entre os pontos 0,24 e 0,26, como mostrados a seguir:

```
a = 0.2
b = 0.3
epsilon=0.001
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 0.253125
Iterações: 5
```

Para visualizarmos as iterações detalhadamente, utilizemos o formato tabelado dos dados:

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	0.250000	0.30000	0.250000	-0.023605	0.176084	-0.023605	0.050000
2	0.250000	0.27500	0.275000	-0.023605	0.130704	0.130704	0.025000
3	0.250000	0.26250	0.262500	-0.023605	0.064319	0.064319	0.012500
4	0.250000	0.25625	0.256250	-0.023605	0.022413	0.022413	0.006250
5	0.253125	0.25625	0.253125	-0.000178	0.022413	-0.000178	0.003125

Assim, obtemos o terceiro e último ponto de equilíbrio, sendo: 0.253125, após 5 iterações.

Curiosamente, concluímos que o primeiro ponto de equilíbrio exclusivamente de nossa análise é 0.5 e os dois posteriores a cada 0.10, arrendondando.

Questão 4

A questão propõe que a equação abaixo pode ser usada para calcular o nível de oxigênio em um rio, após a chegada de uma descarga de esgoto.

$$\mathbf{C = 10 - 15\ (\ e^{-0.1x} - e^{-0.5x}\ )\ ,}$$

onde x é a distância rio abaixo. Determine em que ponto, após a descarga de esgoto, o nível de oxigênio terá caído para 4. Calcule com erro menor que 1% .

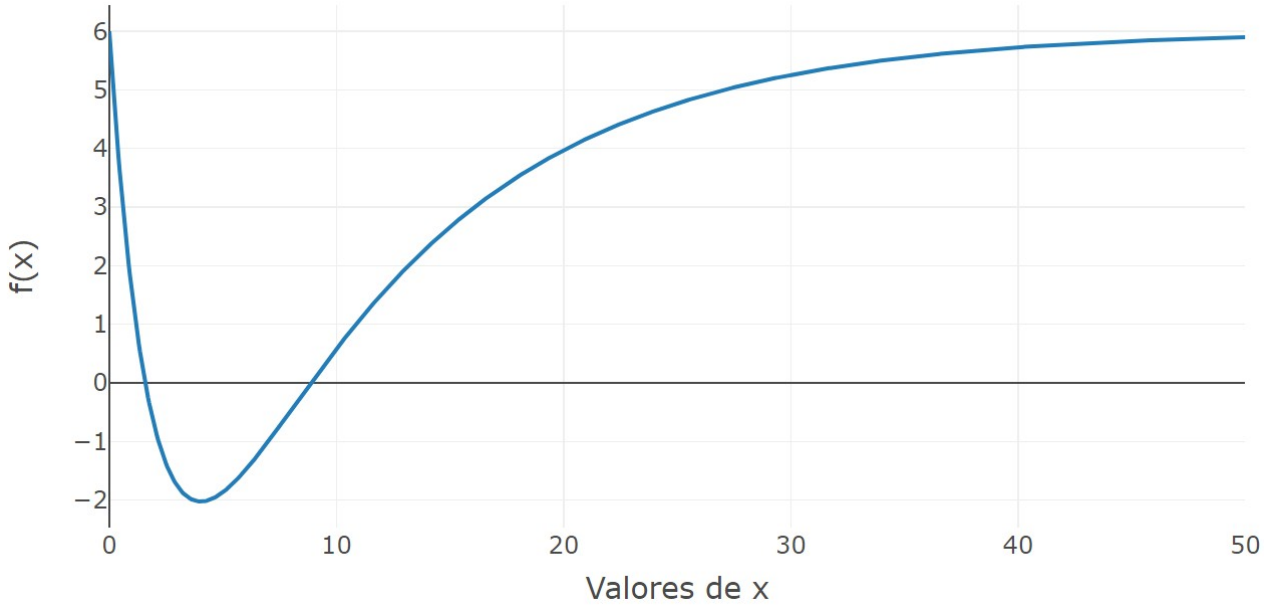
Após formatarmos a expressão apresentada acima para que seja aceito na linguagem de programação Python, obtemos:

Fx= 10 - 15*(exp(-0.1x) – (exp(-0.5*x)))-4

Utilizando essa função no Colab e printando-a, exibe:

$$6 + 15e^{-0.5x} - 15e^{-0.1x}$$

Plotando o gráfico com os pontos de 0 até 50, obtemos:



O gráfico nos apresenta dois pontos de equilíbrio, resultando em soluções visualmente indicada entre os pontos 0 e 10, como mostrados a seguir:

```
a = 0
b = 10
epsilon=0.001
x,i=bissec(a,b,f,epsilon)
print('Solução: {0:f} \nIterações: {1:d}'.format(x,i))
```

```
Solução: 1.579590
Iterações: 12
```

Assim, obtemos como resultado o ponto de equilíbrio, sendo: 1,579590, após 12 iterações.

Para melhor visualização das iterações, utilizemos uma visualização por tabela:

	a	b	x	f(a)	f(b)	f(x)	|b-a|
1	0.000000	5.000000	5.000000	6.000000	-1.866685	-1.866685	5.000000
2	0.000000	2.500000	2.500000	6.000000	-1.384440	-1.384440	2.500000
3	1.250000	2.500000	1.250000	0.791468	-1.384440	0.791468	1.250000
4	1.250000	1.875000	1.875000	0.791468	-0.561352	-0.561352	0.625000
5	1.562500	1.875000	1.562500	0.037321	-0.561352	0.037321	0.312500
6	1.562500	1.718750	1.718750	0.037321	-0.279867	-0.279867	0.156250
7	1.562500	1.640625	1.640625	0.037321	-0.125924	-0.125924	0.078125
8	1.562500	1.601562	1.601562	0.037321	-0.045489	-0.045489	0.039062
9	1.562500	1.582031	1.582031	0.037321	-0.004384	-0.004384	0.019531
10	1.572266	1.582031	1.572266	0.016393	-0.004384	0.016393	0.009766
11	1.577148	1.582031	1.577148	0.005986	-0.004384	0.005986	0.004883
12	1.579590	1.582031	1.579590	0.000796	-0.004384	0.000796	0.002441

Então, concluímos que o nível de oxigênio do esgoto terá caído para 4 no ponto 1,579590.