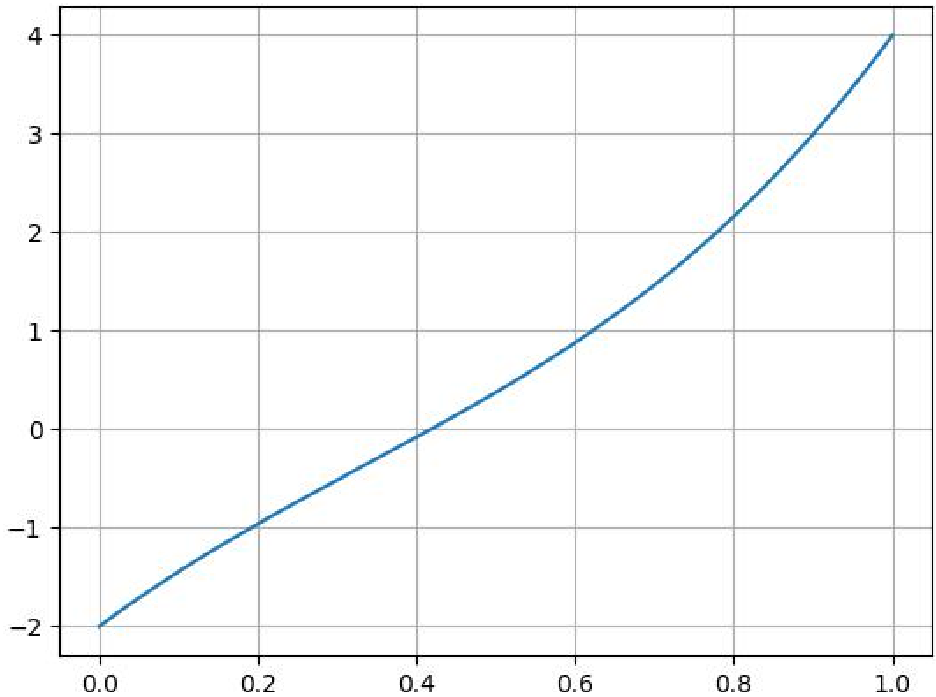
Questão 1

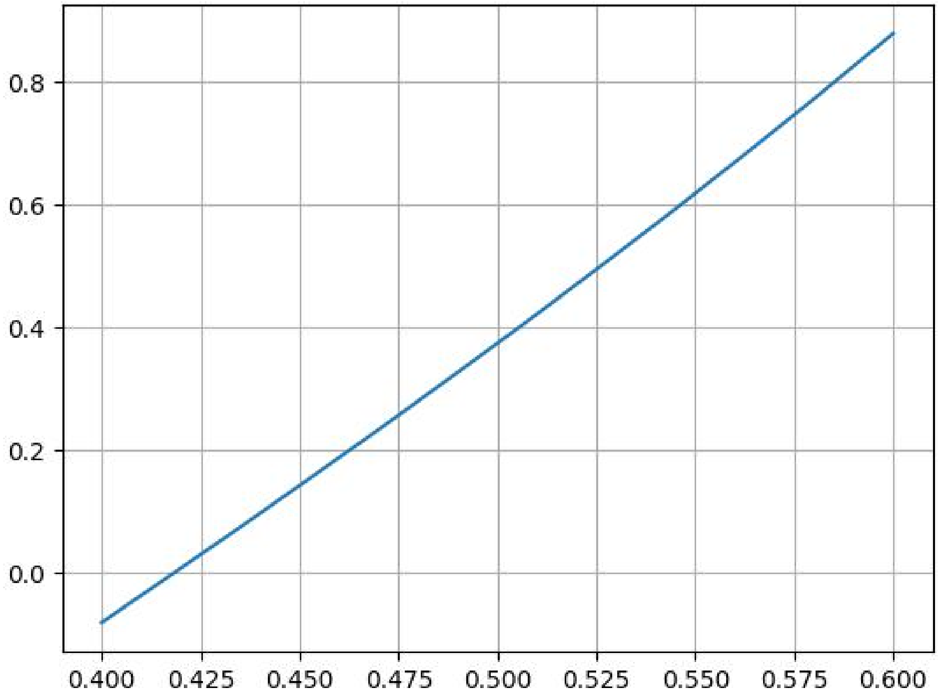
```python
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return 5*(x**3)-5*(x**2)+6*x-2
x=np.linspace(0,1,1000)
y=[]
for i in x:
  y.append(f(i))
plt.grid()
plt.plot(x,y)

def bisseccao(a,b,tol,maxiter):
  fa=f(a)
  fb=f(b)
  xr=(a+b)/2
  fxr=f(xr)
  erro = 1e6
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr<0:
      b=xr
      fb=fxr
    elif fb*fxr < 0:
      a = xr
      fa = fxr
    elif fxr == 0:
      return 0
    xra = xr
    xr = (a+b)/2
    fxr = f(xr)
    erro = math.fabs(xr-xra)/math.fabs(xr)
    it = it+1
  return (xr,it,erro)

print(bisseccao(0,1,0.1,100))
```

- Código utilizado na questão 1

a)



Quando plotei o gráfico com os intervalos de 0 a 1 dados pela questão obtive esse gráfico. Podemos notar que a raiz da função está entre 0.4 e 0.6, então fiz outro plot dessa vez com os intervalos de 0.4 até 0.6.



Analisando o segundo gráfico podemos deduzir que a raiz da função é aproximadamente 0.42.

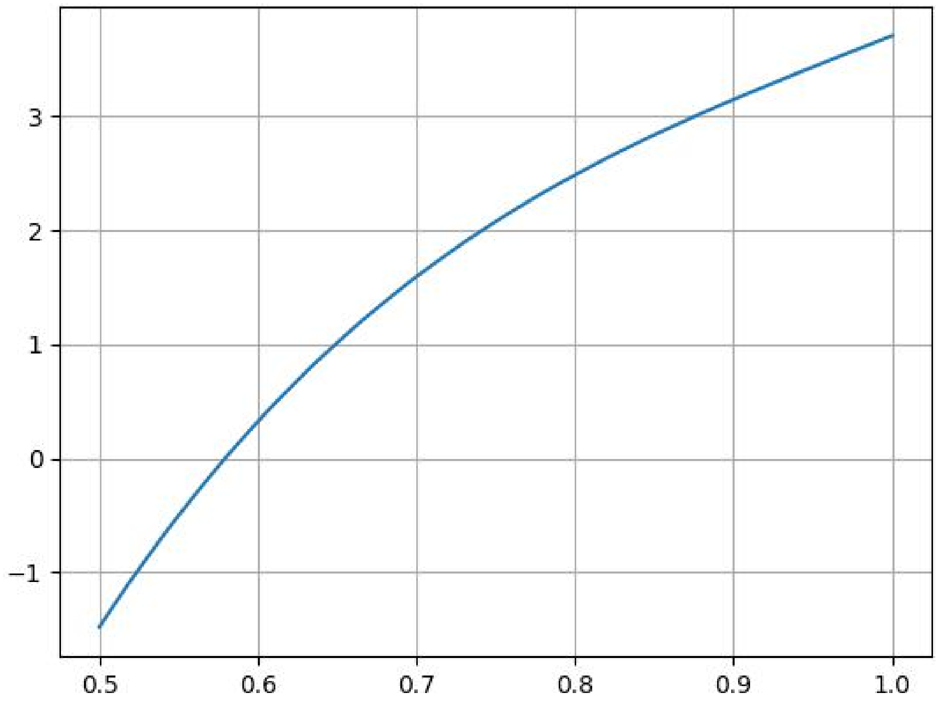
b) ao rodar o código o valor obtido para a raiz da função foi de 0.40625.

Questão 2

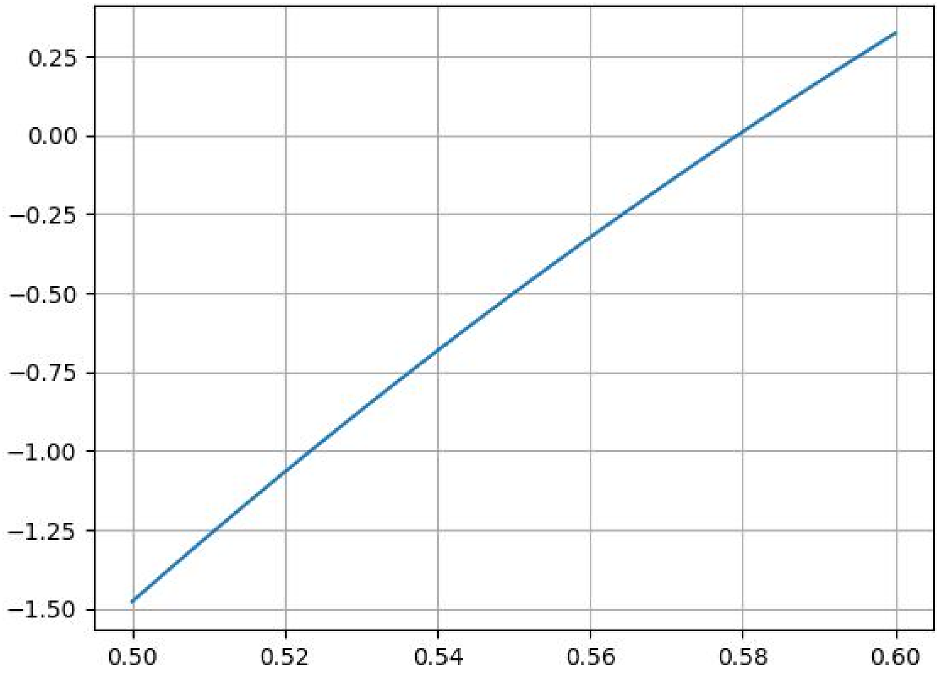
```
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return -25+(82*x)-(90*(x**2))+(44*(x**3))-(8*(x**4)

x=np.linspace(0.5,1,1000)
y=[]
for i in x:
  y.append(f(i))
plt.grid()
plt.plot(x,y)


def bisseccao(a,b,tol,maxiter):
  fa=f(a)
  fb=f(b)
  xr=(a+b)/2
  fxr=f(xr)
  erro = 1e6
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr<0:
      b=xr
      fb=fxr
    elif fb*fxr < 0:
      a = xr
      fa = fxr
    elif fxr == 0:
      return 0
    xra = xr
    xr = (a+b)/2
    fxr = f(xr)
    erro = math.fabs(xr-xra)/math.fabs(xr)
    it = it+1
  return (xr,it,f(xr))

print(bisseccao(0.5,1,0.1,100))
```

-Código usado na questão 2

a)



Quando plotei o gráfico com os intervalos de 0.5 a 1 dados pela questão obtive esse gráfico. Podemos notar que a raiz da função está entre 0.5 e 0.6, então fiz outro plot dessa vez com os intervalos de 0.5 até 0.6.



Analisando o segundo gráfico podemos deduzir que a raiz da função é aproximadamente 0.58

b) Ao usar o método da bissecção com os intervalos de 0.5 até 1 e com uma tolerância de 0.1 o valor obtido para a raiz da função foi 0.59375. Foi preciso 4 interações para alcançar esse valor e o erro final foi de 0.05263157894736842 aproximadamente 5% de erro.

```
def falsaPosicao(a,b,tol,maxiter):
  fa = f(a)
  fb = f(b)
  xr = (a*fb - b*fa)/(fb-fa)
  fxr=f(xr)
  xra = xr
  erro = abs(xr-a)/abs(xr)
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr < 0:
      b = xr
      fb = fxr
    else:
      a = xr
      fa = fxr
    xr = (a*fb - b*fa)/(fb-fa)
    fxr=f(xr)
    erro = abs(xr-xra)/abs(xr)
    xra = xr
    it = it+1
  return (xr,it,erro)

[raiz,iter,erro]=falsaPosicao(0.5,1,0.002,10)
print(raiz)
```

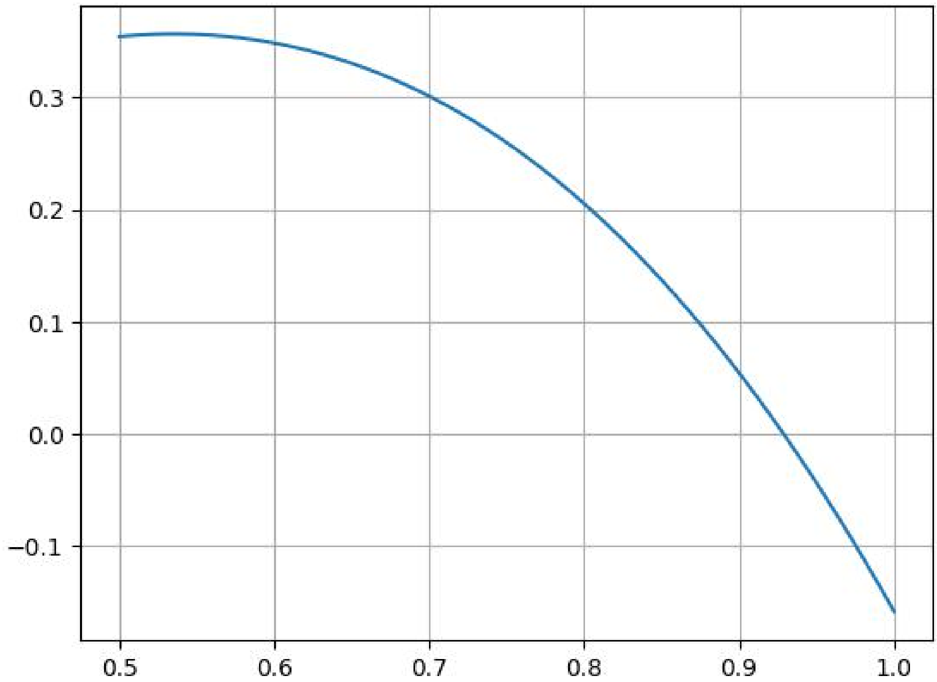
c) Ao usar o método da falsa posição com os intervalos de 0.5 até 1 e com um tolerância de 0.002 o valor obtido para a raiz da função foi 0.5795562476116108. Foi preciso 4 interações para alcançar esse valor e o erro final foi de 0.0016919782782566747 aproximadamente 0.17% de erro.

- Código da 2ª questão usando o método da falsa posição

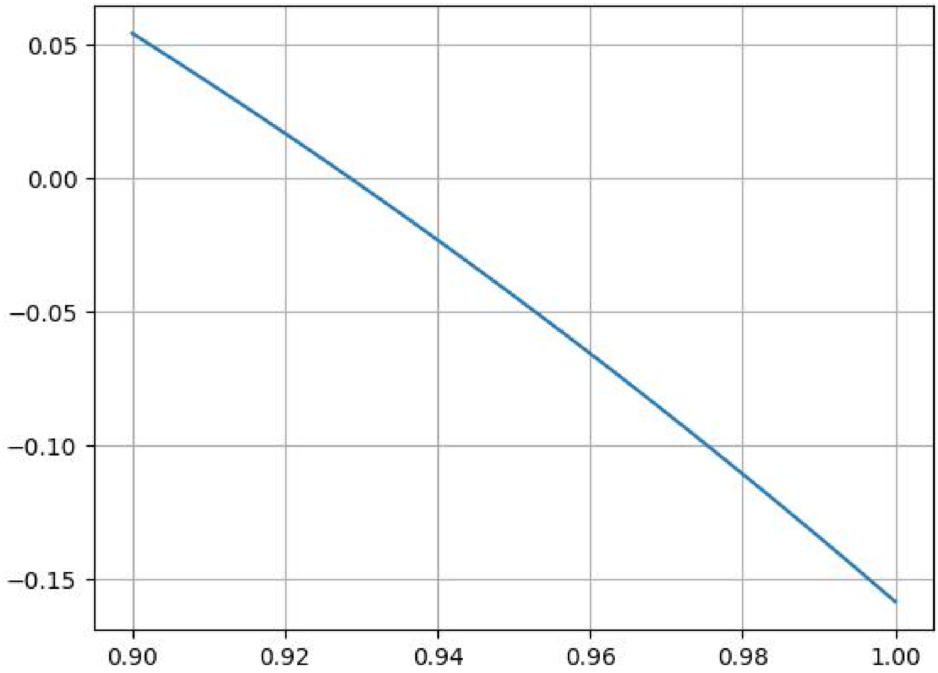
Questão 3)

```python
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return (math.sin(x))-(x**3)

x=np.linspace(0.5,1,1000)
y=[]
for i in x:
  y.append(f(i))
plt.grid()
plt.plot(x,y)

def bisseccao(a,b,tol,maxiter):
  fa=f(a)
  fb=f(b)
  xr=(a+b)/2
  fxr=f(xr)
  erro = 1e6
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr<0:
      b=xr
      fb=fxr
    elif fb*fxr < 0:
      a = xr
      fa = fxr
    elif fxr == 0:
      return 0
    xra = xr
    xr = (a+b)/2
    fxr = f(xr)
    erro = math.fabs(xr-xra)/math.fabs(xr)
    it = it+1
  return (xr,it,f(xr))

print(bisseccao(0.5,1,0.02,10))
```

- Código usado na 3ª questão



Quando plotei o gráfico com os intervalos de 0.5 a 1 dados pela questão obtive esse gráfico. Podemos notar que a raiz da função está entre 0.9 e 1.0, então fiz outro plot dessa vez com os intervalos de 0.9 até 1.0.



Analisando o segundo gráfico podemos deduzir que a raiz da função é aproximadamente 0.93

Ao usar o método da bissecção com os intervalos de 0.5 até 1 e com uma tolerância de 0.02 o valor obtido para a raiz da função foi 0.921875. Foi preciso 5 interações para alcançar esse valor e o erro final foi de 0.01694915254237288 aproximadamente 1.6% de erro. Ao aplicar o valor encontrado da raiz na função original obtive f(raiz) = 0.01327742392963882. Um valor muito próximo do erro relativo final.

Questão 4

```
#Questão 4
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return math.log(x**4)-0.7

def bisseccao(a,b,tol,maxiter):
  fa=f(a)
  fb=f(b)
  xr=(a+b)/2
  fxr=f(xr)
  erro = 1e6
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr<0:
      b=xr
      fb=fxr
    elif fb*fxr < 0:
      a = xr
      fa = fxr
    elif fxr == 0:
      return 0
    xra = xr
    xr = (a+b)/2
    fxr = f(xr)
    erro = math.fabs(xr-xra)/math.fabs(xr)
    it = it+1
  return (xr,it,erro)

[raiz,iter,erro]=bisseccao(0.5,2,0.01,3)
print(raiz)
print(erro)
```

a) Ao usar o método da bissecção com os intervalos de 0.5 até 2 e com uma tolerância de 0.01 e um limite de repetição de 3, o valor obtido para a raiz da função foi 1.0625 e o erro final foi de 0.17647058823529413, aproximadamente 17% de erro.

```python
#Questão 4
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return math.log(x**4)-0.7

def falsaPosicao(a,b,tol,maxiter):
  fa = f(a)
  fb = f(b)
  xr = (a*fb - b*fa)/(fb-fa)
  fxr=f(xr)
  xra = xr
  erro = abs(xr-a)/abs(xr)
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr < 0:
      b = xr
      fb = fxr
    else:
      a = xr
      fa = fxr
    xr = (a*fb - b*fa)/(fb-fa)
    fxr=f(xr)
    erro = abs(xr-xra)/abs(xr)
    xra = xr
    it = it+1
  return (xr,it,erro)

[raiz,iter,erro]=falsaPosicao(0.5,2,0.01,3)
print(raiz)
print(erro)
```

b) Ao usar o método da falsa posição com os intervalos de 0.5 até 2 e com um tolerância de 0.01 e um limite de repetição de 3, o valor obtido para a raiz da função foi 1.2175337173879655 e o erro final foi de 0.04413659202739157 aproximadamente 4.4% de erro.

raiz = 1.2175337173879655

- Ao analisar os erros podemos ver que o método da falsa posição tem um erro relativo menor, logo foi o método mais preciso nessa questão.

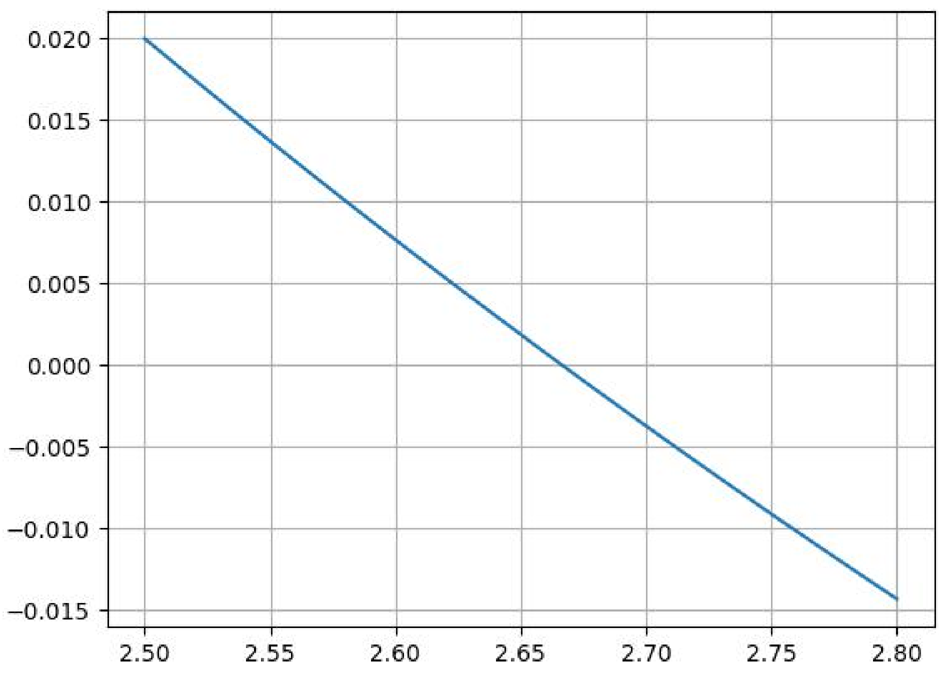
Questão 5

```python
def f(x):
  return (0.8-(0.3*x))/x

x = np.linspace(2.5,2.8,1000)
y=[]
for i in x:
  y.append(f(i))
plt.grid()
plt.plot(x,y)
plt.show()

def falsaPosicao(a,b,tol,maxiter):
  fa = f(a)
  fb = f(b)
  xr = (a*fb - b*fa)/(fb-fa)
  fxr=f(xr)
  xra = xr
  erro = abs(xr-a)/abs(xr)
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr < 0:
      b = xr
      fb = fxr
    else:
      a = xr
      fa = fxr
    xr = (a*fb - b*fa)/(fb-fa)
    fxr=f(xr)
    erroA = abs(xr-xra)
    erro = abs(xr-xra)/abs(xr)
    print('Erro Aproximado:',erroA,'Erro Verdadeiro:',erro)
    xra = xr
    it = it+1
  return (xr,it,erro)
```

a) Para achar analiticamente a função da equação eu fiz (0,8-0,3x) /x = 0. Como 1/x = 0 não existe no domínio dos reais, então eu só me preocupei em igualar a zero a parte de cima. Sege o passo-a-passo:

(0,8-0,3x) /x = 0 -> 0,8-0,3x = 0 -> x = 0,8/0,3 -> x = 2,666666667

b)



Como eu já sabia que a raiz estaria entre 2.60, eu fiz um plot da função entre os intervalos de 2.5 e 2.8. Ao analisar o gráfico podemos deduzir que a raiz está aproximadamente em 2.66.

C) Ao usar o método da falsa posição com os intervalos de 1 até 3 e com uma tolerância de 0.001 e um limite de repetição de 3, o valor obtido para a raiz da função foi (xr = 2.7480468749999996).

1ª Iteração

Erro Aproximado: 0.078125 Erro Verdadeiro: 0.027932960893854747

2ª Iteração

Erro Aproximado: 0.04882812500000444 Erro Verdadeiro: 0.017768301350391067

Mesmo alterando a tolerância para valores menores, o método só foi capaz de realizar 2 iterações, sendo que o valor obtido foi aproximadamente 2.75, o que está longe do valor calculado na questão a) de aproximadamente de 2.66. Podemos perceber que 2.75 está até mesmo depois do zero no gráfico.

Questão 6

```
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return x**2 - 18

def falsaPosicao(a,b,tol,maxiter):
  fa = f(a)
  fb = f(b)
  xr = (a*fb - b*fa)/(fb-fa)
  fxr=f(xr)
  xra = xr
  erro = abs(xr-a)/abs(xr)
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr < 0:
      b = xr
      fb = fxr
    else:
      a = xr
      fa = fxr
    xr = (a*fb - b*fa)/(fb-fa)
    fxr=f(xr)
    erro = abs(xr-xra)/abs(xr)
    xra = xr
    it = it+1
  return (xr,it,erro)

[raiz,iter,erro]=falsaPosicao(4,5,0.05,20)
print(raiz)
```

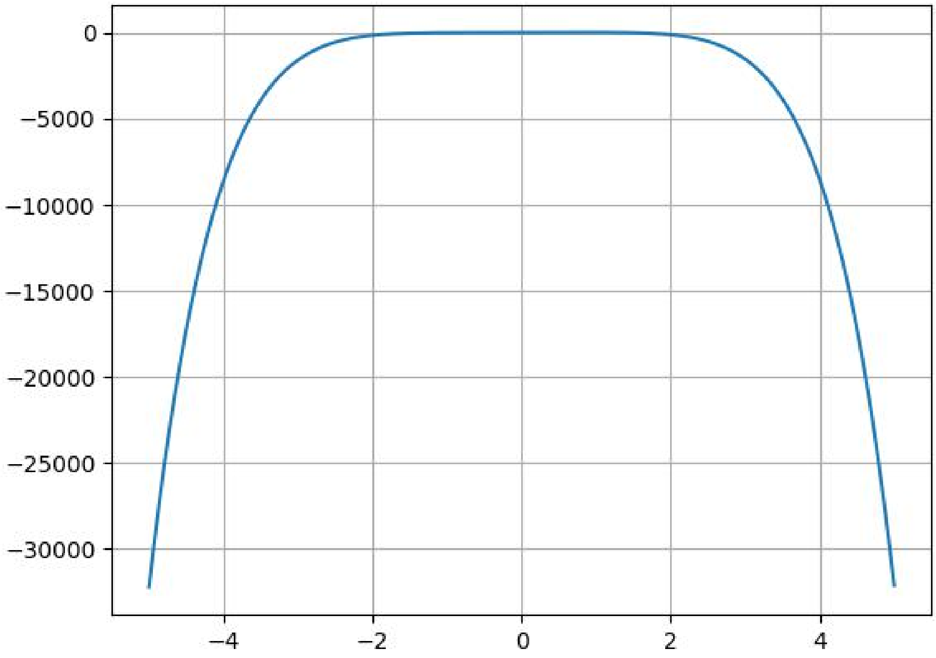
Valor obtido numa calculadora 4,2426406871

Valor obtido no programa: 4.240963855421687

Questão 7

```
#Questão 7
import matplotlib.pyplot as plt
import numpy as np
import math

def f(x):
  return (-12*x**5)-1.5*(4*x**3)+10

def falsaPosicao(a,b,tol,maxiter):
  fa = f(a)
  fb = f(b)
  xr = (a*fb - b*fa)/(fb-fa)
  fxr=f(xr)
  xra = xr
  erro = abs(xr-a)/abs(xr)
  it = 1
  while it < maxiter and erro > tol:
    if fa*fxr < 0:
      b = xr
      fb = fxr
    else:
      a = xr
      fa = fxr
    xr = (a*fb - b*fa)/(fb-fa)
    fxr=f(xr)
    erro = abs(xr-xra)/abs(xr)
    xra = xr
    it = it+1
  return (xr,it,erro)

[raiz,iter,erro]=falsaPosicao(0,1,0.05,20)
print(raiz)
```

- Primeiramente: para achar o máximo dessa função foi preciso deriva a equação original e igualar a zero. (Como a função tem concavidade para baixo, logo não foi preciso ter que fazer a derivada segunda para descobrir se era ponto de máximo ou mínimo).



-

Função derivada: $f'(x) = -12x^5 - 6x^3 + 10$

Ao usar o método da bissecção com os intervalos de 0 até 1 e com uma tolerância de 0.05, o valor obtido para a raiz da função foi 0.8665353042945969. Foi preciso 4 interações para alcançar esse valor e o erro final foi de 0.019055812750733602 aproximadamente 1.9% de erro.