**TRABALHO COMPUTACIONAL 1 -**

PARTE 1, 2 E 3

Nomes:

* Daiana Santos 120.357
* Isadora Muniz 120.431
* Luciana Bello 120.506
* Maria Victória 120.529

**PARTE 1**

1. Código em Python:

Método Bissecção

a = 1 #variável de intervalo (valor minimo)

b = 2 #variável de intervalo (valor máximo)

contador = 1 #variável de verificação de interações

x = a+b/2 #variável de aproximação da raiz

f = 2\*\*0.5 - x #função de verificação de valor da raiz

while(True): #laço de interação para aproximação do valor da raiz

if(abs(f)< 10\*\*(-6)): # condição de parada para as interações

break

elif(f < 0): #atualização do valor do intervalo (valor maximo)

b = x

else: #atualização do valor do intervalo (valor minimo)

a = x

x = (a+b)/2 #atualização da aproximação da raiz

f = 2\*\*0.5 - x #atualização da função de verificação

contador += 1 #incremento do numero de interações

print("Valor raiz")

print(2\*\*0.5)

print("Valor aprox")

print(x)

erro = 2\*\*0.5 - x

print(erro)

print(contador)

Foi adotado como condição de parada quando a função f= 2\*\* 0.5- x for menor que 10^6.

2. Código em Python:

Método de Heron

a = 2 # valor inicial

cont = 1 #variável contadora de interações

while(True):# laço para interações de aproximação da raiz

x = (a + (2/a))/2 #valor aproximado da raiz

if(abs(x\*\*2 - 2) < 10\*\*-6): #condição de parada da interação -> valor da função x mais proximo de zero

print(x)

break

a = x #atualização do valor de a

cont += 1 #incremento do numero de interações

erro = 2\*\*0.5 - x

print(erro)

print(cont)

3. O número de iterações obtidas em ao executar o programa com a estimativa inicial igual a 2 permanece o mesmo quando a estimativa inicial igual a 1, sendo feitas 4 iterações.

4. O erro absoluto obtido em ao executar o programa com os valores iniciais 1 e 2 foi de 1.594724 x 10^-12.

**PARTE 2**

1. Código em Python:

Método de Newton

x = 1 #valor inicial

cont = 1 #variável contadora de interações

while(True): # laço para interações de aproximação da raiz

f = x\*\*2 - 2 #função de verificação de valor da raiz

fd = 2\*x #derivada da função de verificação de valor da raiz

X = x - (f/fd) #valor aproximado da raiz

if(abs(X\*\*2 - 2) < 10\*\*-6): #condição de parada da interação -> valor da função x mais proximo de zero

print(X)

break

x = X

cont += 1 #incremento do numero de interações

print(cont)

Foi adotado como condição de parada quando a função f= X\*\*2-2 for menor que 10^6.

2. Tanto com x=1 e x=2, o número de iterações foi 4 e o erro foi o mesmo, tendo valor de 1.414.

3. Em comparação com o Método da Bisseção, o número de iterações foi menor, mas para o erro, o Método de Newton obteve um valor menor.

**PARTE 3**

1. Código:

x0 = 1 #valor inicial 1

x1 = 2 #valor inicial 2

cont = 1 #variável contadora de interações

while(True): # laço para iterações de aproximação da raiz

f1 = x1\*\*2 - 2 #função de verificação de valor da raiz 2

f0 = x0\*\*2 - 2 #função de verificação de valor da raiz 1

X = x1 - f1 \* ((x1 - x0)/ (f1 - f0)) #valor aproximado da raiz

if(abs(X\*\*2 - 2) < 10\*\*-6): #condição de parada da interação -> valor da função x mais proximo de zero

print(X)

break

x0 = x1 #atualização do valor inicial 1

x1 = X #atualização do valor inicial 2

cont += 1 #incremento do numero de interações

print(cont)

2. Tanto com a estimativa inicial sendo 2 quanto 1, o número de iterações permaneceu igual, entretanto com a estimativa inicial igual a 2, o valor da raiz de 2 obtido foi mais aproximado do valor real do que quando a estimativa inicial era 1.

3.Comparando o desempenho do método da secante com o da bissecção, o da secante é mais eficiente pois possui um erro mais preciso e um número de iterações menor. Já em comparação com o método de Newton e Heron , o método da secante torna-se menos eficiente, visto que o número de iterações foi maior e a precisão do erro menor do que ambos os métodos.