Aluna: Livia Barbosa Fonseca
DRE:118039721

Exercício 1.1-)
Nesse exercício queremos aproximar √10 com o Método da Bisseção no intervalo [0,20] e descobrir quantos passos você precisa executar no método da bisseção se o usuário pedir um erro máximo de 10^{-8}. Sabendo que o método da bisseção é tal que dado um intervalo [a,b], vamos dividindo este até encontrar um ponto na qual encontramos uma raiz da função dada.

```
##Função que recebe x e y e calcula a média entre eles
function média(x,y)
        return (x+y)/2 #Retorna a média de x e y
end

##Função que recebe f(a) e f(b) não nulos e retorna um booleano
function tem_sinais_opostos(f,a,b)
         return f(a)*f(b) < 0 #Retorna true se a função no ponto A tem
                                sinal oposto no ponto B
end

##Função que recebe uma função e um ponto A (f(a)) e retorna um
booleano
function é_raiz(f,a)
        return f(a)==0 #Retorna true se a função no ponto A dado é
                         uma raíz da função
end

#Função que recebe uma função e dois pontos com sinais opostos (f(a) e
f(b)) e o valor de um erro e retorna a aproximação pelo método da
bisseção e o número de iterações necessárias para o cálculo
function bissecao(f,a,b,erro)
#Se a função no ponto A for uma raiz, encontramos nossa aproximação e
retornamos A
    if é_raiz(f,a)
        return a
    end
#Se a função no ponto B for uma raiz, encontramos nossa aproximação e
retornamos B
    if é_raiz(f,b)
        return b
    end
```

```
#Se a função no ponto A e a função no ponto B não possuírem sinais
opostos, não vamos encontrar raízes, logo vamos retornar uma mensagem
de aviso
    if !(tem_sinais_opostos(f,a,b))
        return "Não tem sinais opostos"
    end

    #Variável que salva o número de iterações necessárias dado um erro
    iterações = floor(log2((b-a)/erro/2))+1

    #Lopping que vai se repetir até o número de iterações necessárias
    for i=1:iterações
        #Variável que salva a média entre a e b
        m=média(a,b)
        #Se m (metade do intervalo) é uma raíz, retorna m
        if é_raiz(f,m)
            return m
        end
        #Se a função no ponto A possui sinal oposto do ponto M temos
que b recebe m
        if tem_sinais_opostos(f,a,m)
            b=m
        #Caso contrário, a recebe m
        else
            a=m
        end
    end

    x_final=média(a,b)
#Retornamos o número aproximado que queríamos (raíz do intervalo) e o
número de iterações necessárias
    return x_final,iterações
end
```

Sabendo que podemos escrever √10 como f(x) = x^2 - 10. Vamos utilizar as funções anteriores para calcular o que foi pedido:

```
g(x)=x^2-10
erro = 0.00000001
bissecao(g,0,20,erro)
```

Encontramos 3.162277659866959 como a aproximação para √10 e podemos observar que encontramos 32 para o número de iterações necessárias.

```
(3.162277659866959, 32.0)

julia>
```

Exercício 1.2)
Queremos fazer uma função em Julia que recebe um polinômio de grau 5 e sua derivada e retorna uma raiz do polinômio no intervalo [−100, 100], caso o polinômio tenha sinais trocados e um retorna um aviso se não tiver. Iremos começar o programa com o método da bisseção e diminuir o intervalo para 10^{-2} e depois utilizar o método de Newton para encontrarmos nossa aproximação.

O método de Newton é tal que utiliza o conceito de tangência para descobrir o x quando a função é igual a zero. Dessa forma iremos calcular a derivada da função que é dada pela taxa de variação do eixo y dividida pela taxa de variação do eixo x. Dessa forma, calculamos:

$$x2 = x1 - \left(\frac{f(x1)}{f'(x1)}\right)$$

```
##Função média que calcula a média entre dois pontos x e y
function média(x,y)
    return (x+y)/2
end

##Função que recebe f(a) e f(b) não nulos e retorna um booleano
function tem_sinais_opostos(f,a,b)
    return f(a)*f(b) < 0 #Retorna true se a função no ponto A tem
                                sinal oposto no ponto B
end

##Função que recebe uma função e um ponto A (f(a)) e retorna um
booleano
function é_raiz(f,a)
    return f(a)==0 #Retorna true se a função no ponto A dado é
                         uma raíz da função
end

#Função que dada uma função f e sua derivada g retorna uma aproximação
desejada pelo método da bisseção e de Newton
function bissecao_e_Newton(f,g)
    #Intervalo dado pelo problema [-100,100]
    a = -100
```

```
    b = 100
    #Intervalo final desejado pelo problema
    intervalo = 0.01
      #Método da interpolação
#Se a função no ponto A for uma raiz, encontramos nossa aproximação e
retornamos A
    if é_raiz(f,a)
        return a
    end
#Se a função no ponto B for uma raiz, encontramos nossa aproximação e
retornamos B
    if é_raiz(f,b)
        return b
    end
#Se a função no ponto A e a função no ponto B não possuírem sinais
opostos, não vamos encontrar raízes, logo vamos retornar uma mensagem
de aviso
    if !(tem_sinais_opostos(f,a,b))
        return "Não tem sinais opostos"
    end
#Variável que salva o número de iterações necessárias dado um erro
    iterações = floor(log2((b-a)/intervalo))+1

#Lopping que vai se repetir até o número de iterações necessárias
    for i=1:iterações
#Variável que salva a média entre a e b
        m=média(a,b)
        #Se m (metade do intervalo) é uma raíz, retorna m
        if é_raiz(f,m)
            return m
        end
 #Se a função no ponto A possui sinal oposto do ponto M temos  que b
recebe m
        if tem_sinais_opostos(f,a,m)
            b=m
 #Caso contrário, a recebe m
        else
            a=m
        end
    end
#X recebe o número aproximado que queríamos (raíz do intervalo) e o
número de iterações necessárias pelo método da bisseção
    x=média(a,b)
```

```
#Método de Newton
#Tendo como base o valor dado pelo método da bisseção, vamos tentar
aproximar mais com o método de Newton.
    x2 = x - ((f(x))/g(x))
#Retorna o valor aproximado
    return x2
end
```

Exercício 1.3)

1. Nesse exercício queremos aproximar o valor de ln(3) utilizando o **método da bisseção**. Para isso vamos utilizar as funções do exercício 1.1. Porém, temos que realizar uma alteração tal que agora vamos realizar o input de um intervalo e não de um erro. Sabendo que o erro = intervalo/2 e intervalo = erro/2, vamos realizar as modificações necessárias no código:

```
##Função que recebe x e y e calcula a média entre eles
function média(x,y)
         return (x+y)/2 #Retorna a média de x e y
end

##Função que recebe f(a) e f(b) não nulos e retorna um booleano
function tem_sinais_opostos(f,a,b)
          return f(a)*f(b) < 0 #Retorna true se a função no ponto A tem
                                sinal oposto no ponto B
end

##Função que recebe uma função e um ponto A (f(a)) e retorna um
booleano
function é_raiz(f,a)
         return f(a)==0 #Retorna true se a função no ponto A dado é
                         uma raíz da função
end

#Função que recebe uma função e dois pontos com sinais opostos (f(a) e
f(b)) e o valor de um intervalo final desejado e retorna a aproximação
pelo método da bisseção
function bissecao(f,a,b,intervalo)
#Se a função no ponto A for uma raiz, encontramos nossa aproximação e
retornamos A
    if é_raiz(f,a)
        return a
    end
```

```
#Se a função no ponto B for uma raiz, encontramos nossa aproximação e retornamos B
    if é_raiz(f,b)
        return b
    end
#Se a função no ponto A e a função no ponto B não possuírem sinais opostos, não vamos encontrar raízes, logo vamos retornar uma mensagem de aviso
    if !(tem_sinais_opostos(f,a,b))
        return "Não tem sinais opostos"
    end
    #Variável que salva o número de iterações necessárias dado o intervalo
    iterações = floor(log2((b-a)/intervalo))+1

    #Lopping que vai se repetir até o número de iterações necessárias
    for i=1:iterações
        #Variável que salva a média entre a e b
        m=média(a,b)
        #Se m (metade do intervalo) é uma raíz, retorna m
        if é_raiz(f,m)
            return m
        end
        #Se a função no ponto A possui sinal oposto do ponto M temos que b recebe m
        if tem_sinais_opostos(f,a,m)
            b=m
        #Caso contrário, a recebe m
        else
            a=m
        end
    end

    x_final=média(a,b)
#Retornamos o número aproximado que queríamos (raíz do intervalo)
return x_final
end
```

Sabendo que sabemos calcular e^x para qualquer x dado e sabendo que podemos transformar ln(3) em $f(x) = e^x - 3$, vamos utilizar essa função e o código acima para nos ajudarmos a calcular o valor do ln(3) em um intervalo de 10^{-3}

```
g(x)= (exp(x))-3
intervalo = 0.001
```

```
bissecao(g,0,2,intervalo)
```

Rodando esse programa encontramos:

```
1.09814453125

julia> 
```

Logo temos que 1.09814453125 é o valor aproximado para ln(3) via Método da bisseção com intervalo de 10^{-3}.

Exercício 1.4)
Queremos calcular o cos(40) utilizando interpolação e para isso vamos construir uma função que realiza a interpolação dado 3 pontos. Nesse caso, vamos realizar a interpolação com base nos pontos da tabela abaixo:

x	**30**	**45**	**60**
cos(x)	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$

O método da interpolação é tal que queremos descobrir uma função que aproximadamente passe pelos pontos dados. Dessa forma, a interpolação serve para aproximar funções complexas por funções mais simples. Nesse exercício, vamos utilizar uma função de grau 2 (polinômio) para a criação da função interpoladora.

Para resolvermos esse exercício vamos utilizar:

$$p(x) = a + bx + cx^2$$

Onde substituindo os pontos dados temos:

$$\frac{\sqrt{3}}{2} = a + 30b + c30^2$$
$$\frac{\sqrt{2}}{2} = a + 45b + c45^2$$
$$\frac{1}{2} = a + 60b + c60^2$$

Com essas funções podemos montar uma matriz e utilizando o método de eliminação de gauss resolver e descobrir o valor de cada coeficiente. Porém, vamos realizar esse cálculo com ajuda do Júlia:

```
#Importação necessária para o funcionamento da função interpolação
using LinearAlgebra

#Função que dado um conjunto de três pontos, realiza uma interpolação e retorna um vetor contendo os coeficientes do polinômio interpolador calculado
function interpolação(x,y)
```

```
 #Cria a matriz V (primeira linha: todos os pontos 'x' elevados a 0, segunda linha: todos os pontos de 'x' elevados a 1 e terceira linha: todos os pontos de 'x' elevados a 2)
    V=[x.^0 x.^1 x.^2]
    #Inverte a matriz V (V^-1) e multiplica pela Y
    c=inv(V)*y
    return c #Retorna o vetor dos coeficientes do polinômio
end
```

Passando os valores do nosso caso e chamando a função temos:

```
x = [30,45,60]
y = [(((3)^(1/2))/2),(((2)^(1/2))/2),1/2]
interpolação(x,y)
```

```
3-element Vector{Float64}:
  1.0392981732142514
 -0.002563215075083285
 -0.00010708479686368129

julia> 
```

Encontramos que a = 1.0392981732142514, b = -0.002563215075083285 e c = -0.00010708479686368129

Assim, temos:

$$p(x) = a + bx + cx^2$$

$p(x)=1.0392981732142514-0.002563215075083285x-0.00010708479686368129x^2$

Como queremos a função no ponto x = 40, vamos substituir no polinômio encontrado:

```
y = 1.0392981732142514 + -0.002563215075083285*(40)
-0.00010708479686368129*(40)^2
```

```
0.7654338952290302

julia> 
```

Dessa forma encontramos que o valor de cos(40) = 0.7654338952290302 via polinômio interpolador.

Exercício 1.5-)

Nesse exercício, queremos descobrir o horário na qual ocorreu o assassinato dada a temperatura do corpo em relação ao tempo. Para isso, vamos realizar o método da interpolação para resolver o crime. Nesse caso, vamos realizar a interpolação com base nos pontos da tabela abaixo, interpolando com três pontos:

Tempo	15h	16:30h	17:30h
Temperatura	34oC	30oC	25oC

Utilizando a função interpolação já utilizada anteriormente para interpolar com três pontos, temos:

```
#Biblioteca necessária para a função inv(x)
using LinearAlgebra

function interpolação_três_pontos(x,y)
#Cria a matriz V (primeira linha: todos os pontos 'x' elevados a 0, segunda linha: todos os pontos de 'x' elevados a 1 e terceira linha: todos os pontos de 'x' elevados a 2)
   V=[x.^0 x.^1 x.^2]
   #Inverte a matriz V (V-1) e multiplica pela Y
   c=inv(V)*y
   return c #Retorna o vetor dos coeficientes do polinômio
end
```

Chamando a função e passando os valores:

```
y = [15,16.5,17.5]
x = [34,30,25]
interpolação_dois_pontos(x,y)
```

Encontramos os coeficientes do polinômio:

```
3-element Vector{Float64}:
  7.916666666666686
  0.8694444444444471
 -0.019444444444444486

julia>
```

Descobrindo o polinômio temos:

$$p(x) = a + bx + cx^2$$

$$p(x) = 7.916666666666686 + 0.8694444444444471x -0.019444444444444486x^2$$

Vamos encontrar x = 37, pois quando o corpo estava com essa temperatura ele ainda estava vivo, visto que essa é a temperatura normal do corpo humano. Logo, teremos a hora que a pessoa foi assassinada.

$$p(x) = 7.916666666666686 + 0.8694444444444471x + -0.019444444444444486x^2$$

Encontramos:

```
13.466666666666729

julia> 
```

Dessa forma, encontramos que o horário que a pessoa foi assassinada é 13,28h

Podemos observar que essa função está bem estranha e parece não representar a modelagem proposta. Até o momento, aprendemos a interpolar utilizando dois pontos (encontrando uma função linear) ou utilizando três pontos (encontrando um polinômio), porém, a função do resfriamento de um objeto não se dá por essas funções. Vamos analisar a função do resfriamento proposta por Newton:

$$T(t) = T_{amb} + (T_0 - T_{amb})e^{-t/k}$$

onde,
T_{amb} = temperatura do ambiente
T_0 = temperatura do corpo
k = tempo característico

Podemos observar que a função do resfriamento é dada por uma função f(x) = c + e^x. Por isso, nossa função não chega a contemplar o valor de 37 hora encontrada não está certa, pois tentamos aproximar uma função e por um polinômio de grau 2. Temos funções muito distintas, logo, a aproximação não é boa.

Exercício 1.6-)
Queremos realizar a interpolação para uma função de duas variáveis para determinar a melhor aproximação possível para a posição e a altura do maior pico ou vale de uma montanha. Primeiramente, vamos obter o formato da equação tendo 4 pontos e uma função de 2 variáveis e depois vamos substituir nos pontos dados:

A(x, y)	x0 = 1	x1 = 3
y0 = 2	800m	600m
y1 = 4	400m	500m

p(x,y) =
$$\frac{(x1-x)\,(y1-y)}{(x1-x0)\,(y1-y0)}q00 + \frac{(x-x0)\,(y1-y)}{(x1-x0)\,(y1-y0)}q10 + \frac{(x1-x)(y-y0)}{(x1-x0)(y1-y0)}q01 + \frac{(x-x0)(y-y0)}{(x1-x0)(y1-y0)}q11$$

$$p(x,y) = \frac{(3-x)\,(4-y)}{(3-1)\,(4-2)}800 + \frac{(x-1)\,(4-y)}{(3-1)\,(4-2)}600 + \frac{(3-x)(y-2)}{(3-1)(4-2)}400 + \frac{(x-1)(y-2)}{(3-1)(4-2)}500$$

Simplificando e resolvendo essa equação temos:

$$p(x,y) = -275y - 250x + 75xy + 1450$$

Essa equação p é o nosso polinômio interpolador.
Dessa forma, vamos calcular as derivadas parciais de x e y e igualar essas funções encontradas a zero para encontrarmos um ponto crítico da função.

Derivando em relação ao x:

$$\frac{\partial p(x,y)}{\partial x} = \frac{-275y - 250x + 75xy + 1450}{\partial x}$$

= -250 + 75y

Derivando em relação ao y:

$$\frac{\partial p(x,y)}{\partial y} = \frac{-275y - 250x + 75xy + 1450}{\partial y}$$

= -275 + 75x

Igualando os pontos a zero:

0 = -250 + 75y
0 = -275 + 75x

Dessa forma encontramos:

$y = \frac{10}{3}$

$x = \frac{11}{3}$

Temos que ($\frac{11}{3}$, $\frac{10}{3}$) é um ponto crítico da função.

Porém, temos que analisar se esse ponto é um ponto de máximo e mínimo mesmo. Assim, vamos analisar se nesse ponto ele é um ponto de sela calculando:

$$f_{xx}\, f_y - (f_{xy})^2$$
$$0 - (75)^2$$
$$-5625 < 0$$

Logo, esse ponto é um ponto de sela e os pontos mínimos ou máximos estão nas bordas da função. Dessa forma, temos que as extremidades são:

p(x) = - 275*2 - 250x + 75x*2 +1450
p(x) = - 275*4 - 250x + 75x*4 +1450
p(y) = - 275y - 250 + 75y +1450
p(y) = - 275y - 250*3 + 75*3y +1450

Exercício 1.7)

Nesse exercício, queremos criar uma função que receba 5 pontos (x,y) e imprima o plot da função de grau 3 sem bicos. Dessa forma, temos que usar a interpolação por partes para resolver esse problema. Assim, vamos utilizar as seguintes equações para montar a matriz 8X8:

Formato: $F(X) = a + bx + cx^2 + dx^3$
P(X1) = Y1
P(X2) = Y2
P(X3) = Y3
Q(X3) = Y3
Q(X4) = Y4
Q(X5) = Y5
Q’(X3) = P’(X3)
P’(X) = 0

A equação Q’(X3) = P’(X3) é a que garante que as funções “se juntem” e não tenham bico neste ponto. Já a equação P’(X) = 0 é uma escolha para completar as equações da matriz.

Dessa forma temos:

$$P(X1) = a + bx1 + cx1^2 + dx1^3 = Y1$$
$$P(X2) = a + bx2 + cx2^2 + dx2^3 = Y2$$
$$P(X3) = a + bx3 + cx3^2 + dx3^3 = Y3$$
$$Q(X3) = e + fx3 + gx3^2 + hx3^3 = Y3$$
$$Q(X4) = e + fx4 + gx4^2 + hx4^3 = Y4$$
$$Q(X5) = e + fx5 + gx5^2 + hx5^3 = Y5$$
$$a + bx3 + cx3^2 + dx3^3 = e + fx3 + gx3^2 + hx3^3$$
$$b + 2cx + 2dx1^2 = 0$$

Criando a função temos:

```
#Biblioteca necessária para a função inv(k)
using LinearAlgebra
#Biblioteca necessária para desenhar os gráficos
using Plots

#Função que faz a interpolação de pontos dado os 5 pontos (x,y) dados
pelo usuário e retorna o plot dessa interpolação.
function interpolação(x,y)
    #Transformando os pontos y em uma matriz
    d = [y[1], y[2], y[3], y[3], y[4], y[5], 0, 0]
    #Criando cada linha da matriz do polinomio
    p1 = [x[1]^0 x[1]^1 x[1]^2 x[1]^3 0 0 0 0]
    p2 = [x[2]^0 x[2]^1 x[2]^2 x[2]^3 0 0 0 0]
    p3 = [x[3]^0 x[3]^1 x[3]^2 x[3]^3 0 0 0 0]
    p4 = [0 0 0 0 x[3]^0 x[3]^1 x[3]^2 x[3]^3]
    p5 = [0 0 0 0 x[4]^0 x[4]^1 x[4]^2 x[4]^3]
    p6 = [0 0 0 0 x[5]^0 x[5]^1 x[5]^2 x[5]^3]
    p7 = [0 -1 -2*x[3] -3x[3]^2 0 1 (2*x[3]) (3*x[3]^2)]
    p8 = [0 1 2*x[1] 3*x[1]^2 0 0 0 0]

    #Juntando as linhas e criando uma matriz
    k = [p1;p2;p3;p4;p5;p6;p7;p8]

    #Calculando a interpolação
    c = inv(k)*d

    #Obtendo as funções dado os coeficientes contidos em c
    h(x) = c[5] + (c[6]*x) + ((c[7])*x^2) + ((c[8])*x^3)
    g(x) = c[1] + (c[2]*x) + ((c[3])*x^2) + ((c[4])*x^3)
```

```
    #Criando os gráficos
    plot(h,-1000,40, lw=2, lab="Função H(x)") #Gráfico da função h(x)
    plot!(g,40,1000, lw=2, lab="Função G(x)") #Gráfico da função g(x)
end
```

```
#Exemplo de interpolação
x = [1, 17, 35, 25, 5]
y = [10 20 30 1 45]

interpolação(x,y)
```

```
: #Exemplo de interpolação
  x = [1, 17, 35, 25, 5]
  y = [10 20 30 1 45]

  interpolação(x,y)
```

: