Tarefa2_JoaoBarbosa

April 18, 2021

1 Tarefa 2

1.0.1 Nome: João Vitor de Freitas Barbosa

1.0.2 DRE: 117055449

[3]:
```julia
function fatorial(n :: Integer)
    if n < 0
        error("Fatorial só pode ser n>=0")
    elseif n == 0
        return 1
    else
        return n * fatorial(n-1)
    end
end
```

[3]:
```
fatorial (generic function with 1 method)
```

1.0.3 Exercício 3.3:

Implemente em Julia o polinomial de Taylor para calcular ln(x) usando como exemplo esse link começando em 1:07:44. Na sua implementação, o usuário deve poder colocar um erro m´aximo como entrada (dica: exercício anterior).

Abaixo eu calculo o polinômio de Taylor de Ordem n para ln(x), onde as derivadas seguem f(x) = ln(x), f'(x) = 1/x, f''(x) = -1/(x^2), f'''(x) = 2/(x^3)

[262]:
```julia
function ln(x, n, atol)
    s = 0.0
    a = 1
    b = -1
    base = 1
    for k = 1:n
        s += ((a*(base^b))/fatorial(k))*((x-base)^k)
        a=a*b
        b+=-1
    end
    return s
end
```

[262]: ln (generic function with 2 methods)

[266]:
```
ln(1.5, 25, 1e-8)
```

[266]: 0.40546401996263515

[264]:
```
log(1.5)
```

[264]: 0.4054651081081644

1.0.4 Exercício 5.1:

Implemente em Julia o Método da Bisseção para calcular a solução da equação x^2 − 10 = 0 no intervalo [0,20]. Se precisar de ajuda use esse link começando em 11:00.

[118]:
```
f(x) = x^2 - 10
a, b = 0.0, 20.0
```

[118]: (0.0, 20.0)

[146]:
```
x = (a+b) / 2

if f(a) * f(x) < 0
    b = x
else
    a = x
end
x, f(x)
```

[146]: (3.1622767448425293, -5.789028136859997e-6)

[154]:
```
"""
    (x, fx, exitflag) = bisseccao(f, a, b)
"""
function bisseccao(f, a, b;
                   atol = 1e-8, rtol = 1e-8,
                   max_tempo = 10.0, max_iter = 1000)
    fa = f(a)
    fb = f(b)
      = atol + rtol * max(abs(fa), abs(fb))
     ba = atol + rtol * abs(b-a)
    if abs(fa) <=
        return a, fa, :sucesso
    elseif abs(fb) <=
        return b, fb, :sucesso
    elseif fa * fb >= 0
        return a, fa, :falha_sinais_opostos
```

```
    end

    x = (a + b)/2
    fx = f(x)
    iter = 0
    t0 = time()
     t = time() - t0

    resolvido = (abs(fx) <=   || abs(b - a) <=  ba)
    cansado = (iter >= max_iter ||  t >= max_tempo)

    while !(resolvido || cansado)
        if f(a) * f(x) < 0
            b = x
            fb = fx
        else
            a = x
            fa = fx
        end

        x = (a + b)/2
        fx = f(x)

        iter+=1
         t = time() - t0
        resolvido = (abs(fx) <=   || abs(b - a) <=  ba)
        cansado = (iter >= max_iter ||  t >= max_tempo)
    end
    exitflag = :desconhecido
    if resolvido
        exitflag = :sucesso
    elseif cansado
        if iter >= max_tier
            exitflag = :max_iter
        else
            exitflag = :max_tempo
        end
    end

    return x, fx, exitflag, iter, b-a
end
```

[154]: bisseccao

[155]:
```
f(x) = x^2 - 10
a, b = 0.0, 20.0
bisseccao(f, a, b)
```

[155]: (3.162277936935425, 1.7504285665381758e-6, :sucesso, 23, 2.384185791015625e-6)

1.0.5 Exercício 5.2:

Dado o intervalo [a, b] e o número n de passos da bissecção, qual´e o tamanho do intervalo [an, bn]?

Podemos ver abaixo que o tamanho do intervalo é dado pelo último valor de retorno: 2.384185791015625e-6

[156]:
```
f(x) = x^2 - 10
a, b = 0.0, 20.0
bisseccao(f, a, b)
```

[156]: (3.162277936935425, 1.7504285665381758e-6, :sucesso, 23, 2.384185791015625e-6)

1.0.6 Desafio 5.1:

Usando o exercício anterior, determine quantos passos você precisa executar no método da bissecção com intervalo [a, b] se o usuário pedir um erro máximo de $10\hat{\ }{-8}$. Deixe claro com você esta definindo o erro.

O erro no caso, está definido na própria função como **atol(erro absoluto)** e **rtol(erro relativo)** como 1e-8 = 10^-8

Como podemos ver abaixo, temos 23 passos para um erro máximo de 10^-8

[157]:
```
f(x) = x^2 - 10
a, b = 0.0, 20.0
bisseccao(f, a, b)
```

[157]: (3.162277936935425, 1.7504285665381758e-6, :sucesso, 23, 2.384185791015625e-6)