UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE TECNOLOGIA - CT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEE
DISCIPLINA: TÉCNICAS DE OTIMIZAÇÃO (PARTE I)
PROFESSOR: RICARDO DE ANDRADE

Lista de Exercícios – Unidade I (Conceitos e Princípios Gerais em Cálculo Numérico)

GABARITO: Exercícios para Implementação Computacional (EI)

Maria do Rosário de Fátima Martins Ferreira

Questão 01 -

1 – Implemente os procedimentos computacionais necessários para obter o valor de e^x, por meio do seguinte somatório: $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. Os dados de entrada são:

a) Argumento da função (x);

b) Quantidade de termos a serem utilizados (n).

Os dados de saída são:

a) O valor de e^x;

i) Determine o valor de e^1, e^2, e^4, e^6 para as seguintes quantidades de termos: 3, 6, 9, 12 e 15.

Sugestão de implementação de função para calcular exponencial natural:

- ➔ **Função auxiliar Fatorial**
- ➔ **Sem uso de biblioteca na implementação; Usei math para conferir o valor “exato” fornecido pela biblioteca, apenas para comparação dos resultados;**
- ➔ **IDE: VS code;**

```
questao1.py 1 ●
C: > Users > Maria > Faculdade > ATUAIS > Otimização2 > questao1.py > ...

def fatorial(n): #função para calcular o fatorial de um número, recebe n como parâmetro
    if n == 0:
        return 1 #fatorial de 0 é 1
    else:
        return n * fatorial(n-1) #retorna fatorial de n


def e_ex(x, n): #função para calcular o valor de e^x, recebe x e n como parâmetros
    e = 0
    for i in range(n):
        e += x**i / fatorial(i)
    return e #retorna o valor de e^x
```

Sugestão de implementação para calcular e^1, e^2, e^4, e^6 para 3, 6, 9, 12 e 15 termos (utilizando as funções acima):

```
print("Esse programa serve para calcular aproximações de e^x para n termos da soma de Taylor. No final, será most
for x in [1, 2, 4, 6]: #para x = 1, 2, 4 e 6
    print("*** Valor de x = ", x, " ***") #imprime o valor de x
    for n in [3, 6, 9, 12, 15]: #para n = 3, 6, 9, 12 e 15
        print(f"→ Aproximação de e^{x} para {n} termos: {e_ex(x, n)}") #imprime o valor de e^x para n termos
    print("Valor exato de e^x: ", math.exp(x)) #imprime o valor exato de e^x
    print() #pula uma linha
```

Saída:
Terminal python, VScode

```
PS C:\Users\Maria> & C:/Users/Maria/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Maria/Faculdade/ATU
AIS/Otimizaçao2/questao1.py
Esse programa serve para calcular aproximações de e^x para n termos da soma de Taylor. No final, será mostrado o va
lor exato de e^x.
*** Valor de x =  1  ***
→ Aproximação de e^1 para 3 termos: 2.5
→ Aproximação de e^1 para 6 termos: 2.7166666666666663
→ Aproximação de e^1 para 9 termos: 2.71827876984127
→ Aproximação de e^1 para 12 termos: 2.718281826198493
→ Aproximação de e^1 para 15 termos: 2.71828182845823
Valor exato de e^x:  2.718281828459045

*** Valor de x =  2  ***
→ Aproximação de e^2 para 3 termos: 5.0
→ Aproximação de e^2 para 6 termos: 7.266666666666667
→ Aproximação de e^2 para 9 termos: 7.387301587301587
→ Aproximação de e^2 para 12 termos: 7.389046015712681
→ Aproximação de e^2 para 15 termos: 7.3890560703259105
Valor exato de e^x:  7.38905609893065

*** Valor de x =  4  ***
→ Aproximação de e^4 para 3 termos: 13.0
→ Aproximação de e^4 para 6 termos: 42.86666666666666
→ Aproximação de e^4 para 9 termos: 53.43174603174603
→ Aproximação de e^4 para 12 termos: 54.54818021484688
→ Aproximação de e^4 para 15 termos: 54.59706179769672
Valor exato de e^x:  54.598150033144236

*** Valor de x =  6  ***
→ Aproximação de e^6 para 3 termos: 25.0
→ Aproximação de e^6 para 6 termos: 179.8
→ Aproximação de e^6 para 9 termos: 341.8
→ Aproximação de e^6 para 12 termos: 395.3231168831169
→ Aproximação de e^6 para 15 termos: 402.86385043527895
Valor exato de e^x:  403.4287934927351
```

Questão 02 -

2 – Implemente os procedimentos computacionais necessários para obter o valor da raiz da equação $e^x - \text{sen}\, x - 2 = 0$, por meio da iteração ou aproximação sucessiva. Os dados de entrada são:

a) Tentativa inicial: $x^{(0)} = 0{,}5$;

b) Equação de recorrência: $x^{(i+1)} = x^{(i)} - \frac{e^{x^{(i)}} - \text{sen}\, x^{(i)} - 2}{e^{x^{(i)}} - \cos x^{(i)}}$;

c) Teste de parada: $\left|x^{(i+1)} - x^{(i)}\right| \leq 10^{-5}$.

Sugestão de implementação:

- ➔ **Usei a biblioteca math para obter os valores de sen e cos fornecidos pela biblioteca;**
- ➔ **IDE: VS code;**

UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE TECNOLOGIA - CT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEE
DISCIPLINA: TÉCNICAS DE OTIMIZAÇÃO (PARTE I)
PROFESSOR: RICARDO DE ANDRADE

Usei uma função auxiliar 'funcao' que representa a equação $e \quad - \operatorname{sen} x - 2 = 0$ e uma função 'recorrencia' que implementa equação de recorrência:

$$: x^{(i+1)} = x^{(i)} - \frac{e^{x^{(i)}} - \operatorname{sen} x^{(i)} - 2}{e^{x^{(i)}} - \cos x^{(i)}};$$

```python
questao2.py ×
questao2.py > aproximacao_sucessiva
import math

def funcao(x):
    return math.exp(x) - math.sin(x) - 2 # f(x) = e^x - sen(x) - 2

def recorrencia(x_atual):
    x_proximo = x_atual - funcao(x_atual) / (math.exp(x_atual) - math.cos(x_atual)) # x_n+1 = x_n - f(x_n) / f'(x_
    return x_proximo

```

A condição de parada foi representada como "erro" e foi adicionado um controle de quantidade máxima de iterações, para evitar loops infinitos. A função aproximação sucessiva também retorna o número de iterações necessárias para atingir a condição de parada. Isso e o controle de loop não foram requisitados na questão, mas são decisões interessantes e importantes de implementação que podem ser úteis em outras questões com diferentes entradas. Função aproximação sucessiva e função main:

```python
def aproximacao_sucessiva(x_inicial, erro, max_iter):
    x_atual = x_inicial
    i = 0

    while True:
        x_proximo = recorrencia(x_atual) # x_n+1 = x_n - f(x_n) / f'(x_n)
        i += 1
        print("Iteração ", i, ": ", x_proximo) # Printa o valor de x_n+1 a cada iteração, serve apenas para visualização


        if abs(x_proximo - x_atual) <= erro: # |x_n+1 - x_n| <= erro
            break

        if i >= max_iter: # Se o número de iterações for maior que o máximo, encerra o loop: evita loop infinito
            print("Número máximo de iterações alcançado.")
            break

        x_atual = x_proximo # Atualiza o valor de x_n para o próximo valor calculado

    return x_proximo, i

if __name__ == "__main__":
    res, iteracoes = aproximacao_sucessiva(0.5, 1e-5, 10000)
    print("A raiz da equação é aproximadamente: ", res, ". Com ", iteracoes, " iterações.")
```

Saída:

```
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2> & C:/Users/Maria/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Maria/Faculdade/ATUAIS/Otimizaçao2/questao2.py
Iteração  1 :  1.5772436377707693
Iteração  2 :  1.1973760625913226
Iteração  3 :  1.0683020029322148
Iteração  4 :  1.0542830971487671
Iteração  5 :  1.054127143233759
Iteração  6 :  1.0541271240912133
A raiz da equação é aproximadamente:  1.0541271240912133 . Com  6  iterações.
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2>
```

Questão 03 -

3 – Implemente os procedimentos computacionais necessários para obter o valor das seguintes séries: $sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \cdots$ e $cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \cdots$. Os dados de entrada são:

a) Argumento da função (x);
b) Precisão (número de casas decimais)

i) Determine o valor de $sen(x)$ e $cos(x)$ com uma precisão de 10 casas decimais para os seguintes valores de x: 0; 1; -0,2; 3,1415926535; 1,57 e 2,7;

ii) Determine o valor de $sen(x)$ e $cos(x)$ com uma precisão de 15 casas decimais para os seguintes valores de x: 0; 1; -0,2; 3,1415926535; 1,57 e 2,7.

Sugestão de implementação de função para calcular aproximação para sen x e cos x:

- ➔ **Uso de biblioteca Math para obter os valores “reais” de sen e cos, para efeito de comparação e obter valor de pi, para testes;**
- ➔ **IDE: VS code;**
- ➔ **Usei uma função auxiliar para calcular fatoriais**

UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE TECNOLOGIA - CT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEE
DISCIPLINA: TÉCNICAS DE OTIMIZAÇÃO (PARTE I)
PROFESSOR: RICARDO DE ANDRADE

questao3.py
questao3.py > calcula_aproximacoes

```python
import math

def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n - 1)

def sen(x, precisao):
    sen = 0 #inicializa a variável sen como 0
    n = 0 #inicializa a variável n como 0, n representa o número de termos da série e o número de iterações
    termo_atual = x #inicializa a variável termo como x, termo representa o termo atual da série
    sinal = 1 #inicializa a variável sinal como 1, sinal representa o sinal do termo atual

    while abs(termo_atual) >= 10**(-precisao): #enquanto o valor absoluto do termo atual for maior ou igual a 10 el
        sen += termo_atual
        n += 1
        sinal *= -1 #muda o sinal do termo atual, garante a oscilação entre positivo e negativo
        termo_atual = (x**(2*n + 1)) / fatorial(2*n + 1) * sinal
    return round(sen, precisao) #retorna o valor do seno de x com precisão de precisao casas decimais

def cos(x, precisao): #analogamente, função para calcular o cosseno de x, recebe x e precisão como parâmetros
    cos = 0
    n = 0
    termo_atual = 1
    sinal = 1

    while abs(termo_atual) >= 10**(-precisao):
        cos += termo_atual
        n += 1
        sinal *= -1
        termo_atual = (x**(2*n)) / fatorial(2*n) * sinal
    return round(cos, precisao)
```

Função calcula e imprime aproximações e valores “reais”. Alguns testes abaixo:

```python
def calcula_aproximacoes(x, p):
    print("*** Valor de x = ", x, " para ", p, " casas decimais***")
    print(f"-> Aproximação de sen({x}): {sen(x, p)}")
    print(f"-> Aproximação de cos({x}): {cos(x, p)}")
    print(f"Valor exato de sen({x}): ", math.sin(x))
    print(f"Valor exato de cos({x}): ", math.cos(x))

print("Esse programa serve para calcular aproximações de sen(x) e cos(x) com p casas decimais de precisão. No final, será mostrado o valo
calcula_aproximacoes(1, 5)
pi = math.pi
calcula_aproximacoes(pi/2, 5)
calcula_aproximacoes(pi, 5)
```

Saída:

```
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2> & C:/Users/Maria/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Maria/Faculdade/ATUAIS/Otimizaçao2/questao3.py
Esse programa serve para calcular aproximações de sen(x) e cos(x) com p casas decimais de precisão. No final, será mostrado o valor exato de sen(x) e cos(x).
*** Valor de x =  1  para  5  casas decimais***
→ Aproximação de sen(1): 0.84147
→ Aproximação de cos(1): 0.5403
Valor exato de sen(1):  0.8414709848078965
Valor exato de cos(1):  0.5403023058681398
*** Valor de x =  1.5707963267948966  para  5  casas decimais***
→ Aproximação de sen(1.5707963267948966): 1.0
→ Aproximação de cos(1.5707963267948966): -0.0
Valor exato de sen(1.5707963267948966):  1.0
Valor exato de cos(1.5707963267948966):  6.123233995736766e-17
*** Valor de x =  3.141592653589793  para  5  casas decimais***
→ Aproximação de sen(3.141592653589793): -0.0
→ Aproximação de cos(3.141592653589793): -1.0
Valor exato de sen(3.141592653589793):  1.2246467991473532e-16
Valor exato de cos(3.141592653589793):  -1.0
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2> 
```

Questão 04 -

4 – Implemente os procedimentos computacionais necessários para se calcular o valor de um polinômio $P(x)$ para $x = \bar{x}$, ou seja $P(\bar{x})$. Os dados de entrada são:

a) Grau do polinômio (n);

b) Coeficientes do polinômio $(a_n, a_{n-1}\, a_{n-2},.., a_2, a_1,a_0)$;

c) Argumento do polinômio $\bar{x}$.

i) Determine o valor de $P(x) = -x^5 + 2x^4 - 5x^3 + 2x^2 + 4x - 1$, para $\bar{x} = -2$;

ii) Determine o valor de $P(x) = 3x^9 + 2x^8 - 10x^7 + 2x^6 - 15x^5 - 3x^4 + 2x^3 - 16x^2 + 3x - 5$, para $\bar{x} = 2$.

Sugestão de implementação de função para calcular valor de polinômio:

- ➔ **Sem uso de biblioteca;**
- ➔ **IDE: VS code;**

```
questao4.py ×
questao4.py > ...
def calcular_polinomio(grau, coeficientes, x):
    res=0;
    for i in range(grau+1):
        res+=coeficientes[i]*(x**(grau-i))
        #print(f"{coeficientes[i]} *(({x}**{grau-i}))")
        #print(res)

    return res

```

Testes:

```python

resultado_teste_1 = calcular_polinomio(2, [1, 1, 1], 2)
print("Resultado do polinômio para x =", 2, "é:", resultado_teste_1)

#questão i - Determine P(x) = -x^5+2x^4-5x^3+2x^2+4x-1
resultado_teste_4 = calcular_polinomio(5, [-1, 2, -5, 2, 4,-1], -2)
print("Resultado do polinômio para x =", -2, "é:", resultado_teste_4)

#questão ii - Determine P(x) = 3x^9 + 2x^8 - 10x^7 + 2x^6 -15x^5 - 3x^4 + 2x^3 - 16x^2 + 3x - 5
resultado_teste_5 = calcular_polinomio(9, [3, 2, -10, 2, -15, -3, 2, -16, 3, -5], 2)
print("Resultado do polinômio para x =", 2, "é:", resultado_teste_5)
```

Saída:

```
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2> & C:/Users/Maria/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Maria/Faculdade/ATUAIS/Otimizaçao2/questao4.py
Resultado do polinômio para x = 2 é: 7
Resultado do polinômio para x = -2 é: 103
Resultado do polinômio para x = 2 é: 321
PS C:\Users\Maria\Faculdade\ATUAIS\Otimização2>
```

Questão 05 -

5 – Uma cidade A em 2020 possui 100 mil habitantes e tem uma taxa de crescimento de 1% ao ano. Uma cidade B em 2020 possui 30 mil habitantes e tem uma taxa de crescimento de 3% ao ano. Pode-se imaginar que, mantidas essas condições, ao longo de vários anos a cidade B terá mais habitantes que a cidade A. A questão é: em que ano a cidade B terá ultrapassado a cidade A em termos de quantitativo de habitantes? Qual será a população de cada cidade quando isso acontecer? Implemente os procedimentos computacionais necessários para determinar o ano que a população da cidade B terá ultrapassado a população da cidade A, bem como a população nessa data. O programa computacional deve mostrar a evolução populacional de cada cidade até o ano em que a cidade B terá uma população maior que a cidade A.

Sugestão de implementação de função para calcular valor de polinômio:

- ➔ **Biblioteca: Matplotlib (Gráficos)**
- ➔ **IDE: VS code;**

questao5.py

questao5.py > ...

```python
import matplotlib.pyplot as plt

def populacaoA(ano):
    return 100000 * ((1.01) ** (ano-2020))

def populacaoB(ano):
    return 30000 * ((1.03) ** (ano-2020))


'''print("População da cidade A em 2025:", populacaoA(2025))
print("População da cidade B em 2025:", populacaoB(2025))'''

# Dados das cidades
anos = list(range(2020, 2100))
populacao_A = [populacaoA(ano) for ano in anos]
populacao_B = [populacaoB(ano) for ano in anos]

# Cálculo do ano em que a população de B ultrapassa a população de A
ano = 2020
A = populacaoA(ano)
B = populacaoB(ano)
while B <= A:
    ano += 1
    A = populacaoA(ano)
    B = populacaoB(ano)
print(f"Em {ano} a população de B ultrapassará a população de A. A população de A será de {A} e a população de B se

```

Construção do gráfico:

```python

# Plotagem do gráfico
plt.figure(figsize=(10, 6))
plt.plot(anos, populacao_A, label='Cidade A')
plt.plot(anos, populacao_B, label='Cidade B')
plt.axvline(ano, color='r', linestyle='--', label=f'População de B ultrapassa a de A em {ano}')
plt.title('Crescimento da População ao Longo dos Anos')
plt.xlabel('Ano')
plt.ylabel('População')
plt.legend()
plt.grid(True)
plt.show()

```

Saída:

Em 2082 a população de B ultrapassará a população de A. A população de A será de 185321.23022052296 e a população de B será de 187512.05199246397.

UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE TECNOLOGIA - CT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEE
DISCIPLINA: TÉCNICAS DE OTIMIZAÇÃO (PARTE I)
PROFESSOR: RICARDO DE ANDRADE