UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ: CAMPUS DE FOZ DO IGUAÇU
CENTRO DE ENGENHARIAS E CIÊNCIAS EXATAS

Cálculo Numérico

Atividade #7

Instruções:

- Entrega individual, via “Tarefas” do Teams e arquivo único em .pdf;
- Use este arquivo .docx para fazer sua atividade, e ao finalizar, gere o .pdf.
- Além de incluir os algoritmos no .pdf, eles devem ser upados em anexo, cada um individualmente e um arquivo txt;

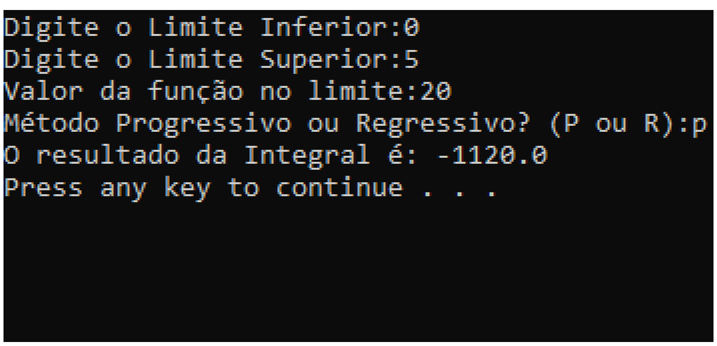
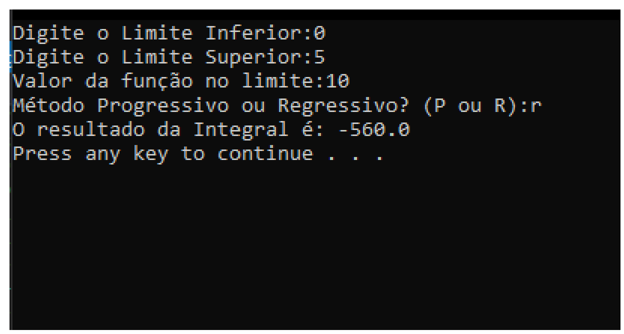
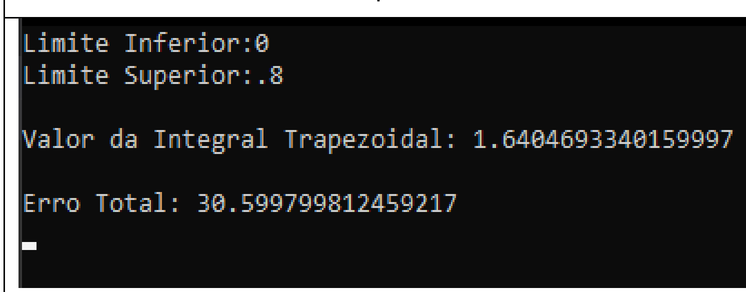
- **Discente**: Daniel Marques da Silva

1) Elabore funções genéricas para todos os casos.

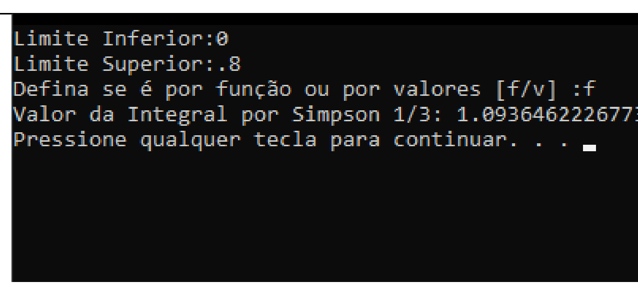
Resposta:

O primeiro exercício se referia a implementação de todos os casos de integração numéricas, excluídas os métodos de Romberg e Gauss. Esses deveriam efetuar a leitura usando dois tipos de entrada, ou uma função já conhecida, além de seus pontos de limite, ou a entrada deveria ser vetores de X e Y.

A tabela a seguir apresenta os resultados de saída de cada um dos métodos.

<table>
<tr><td>Euler Progressivo</td><td>Euler Regressivo</td></tr>
<tr><td>
Digite o Limite Inferior:0
Digite o Limite Superior:5
Valor da função no limite:20
Método Progressivo ou Regressivo? (P ou R):p
O resultado da Integral é: -1120.0
Press any key to continue . . .</td><td>
Digite o Limite Inferior:0
Digite o Limite Superior:5
Valor da função no limite:10
Método Progressivo ou Regressivo? (P ou R):r
O resultado da Integral é: -560.0
Press any key to continue . . .</td></tr>
<tr><td>Trapezoidal</td><td>Simpson 1/3</td></tr>
<tr><td>
Limite Inferior:0
Limite Superior:.8

Valor da Integral Trapezoidal: 1.6404693340159997

Erro Total: 30.599799812459217</td><td>
Limite Inferior:0
Limite Superior:.8
Defina se é por função ou por valores [f/v] :f
Valor da Integral por Simpson 1/3: 1.0936462226773
Pressione qualquer tecla para continuar. . .</td></tr>
<tr><td>Simpson 3/8</td><td></td></tr>
</table>

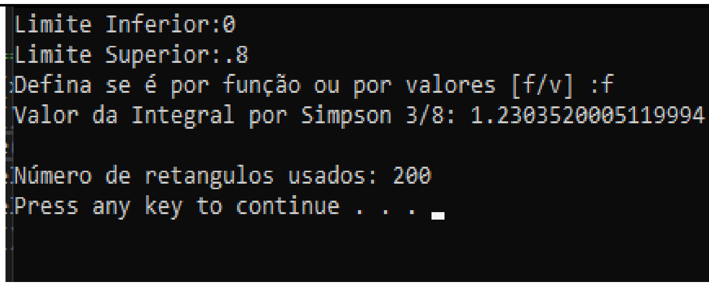
 Limite Inferior:0 Limite Superior:.8 Defina se é por função ou por valores [f/v] :f Valor da Integral por Simpson 3/8: 1.2303520005119994 Número de retangulos usados: 200 Press any key to continue . . .	

Tabela 1 – Resposta das funções

Como é observado, para a função dada por Euler $f(x) = x^2 - 18x + 36$, os limites selecionados não foram satisfatórios para o cálculo da integral. Além, a função em si é mal condicionada para uso desse método, esse que bastante simples e muitas vezes não se mostra suficiente para cálculos de 2° grau. Para os demais métodos, foi utilizado a função $f(x) = 400x^5 - 900x^4 + 675x^3 - 200x^2 + 25x + 0{,}2$, onde a resposta analítica é 1,6405, aproximadamente. E como é possível observar para a resposta dada pelo método trapezoidal é suficientemente próxima da real, quase convergindo ao real. Em respeito as respostas dadas pelos métodos de Simpson, a diferença entre eles foi de 0,136705778, algo relativamente baixo dependendo da necessidade de precisão.

Obs. Código-Projeto se encontra em anexo aos demais arquivos.

2) Comparações com valores analíticos

Resposta:

Prosseguindo com algumas comparações, foi realizado também uma comparação dos valores dados pelas funções de Simpson 3/8 e Trapezoidal para a função $f(x) = sen(x)$, onde os resultados podem ser apresentados segundo a Tabela 2.

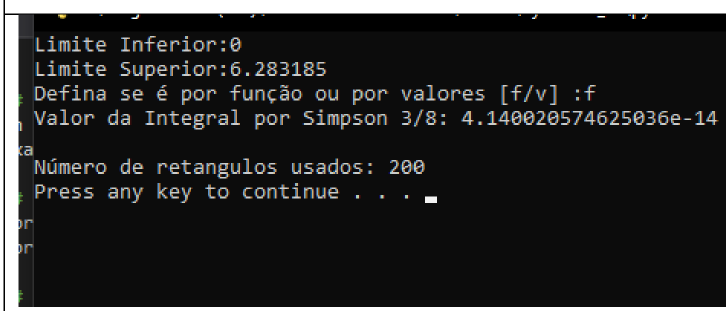
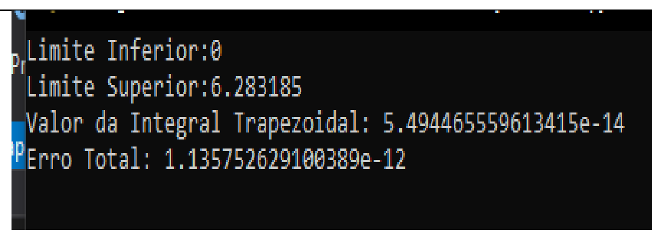
Simpson 3/8	Trapezoidal
 Limite Inferior:0 Limite Superior:6.283185 Defina se é por função ou por valores [f/v] :f Valor da Integral por Simpson 3/8: 4.140020574625036e-14 Número de retangulos usados: 200 Press any key to continue . . .	 Limite Inferior:0 Limite Superior:6.283185 Valor da Integral Trapezoidal: 5.494465559613415e-14 Erro Total: 1.135752629100389e-12

Tabela 2 – Solução para seno(x)

Como era de se esperar de um cálculo próximo do real, os valores retornados são extremamente pequenos, da ordem de 10^{-4} , onde o valor real deve ser zero. O que condiz com os valores analíticos estimados segundo anos a fio de pesquisa em Cálculo Diferencial Integral.

Obs. Código-Projeto se encontra em anexo aos demais arquivos.

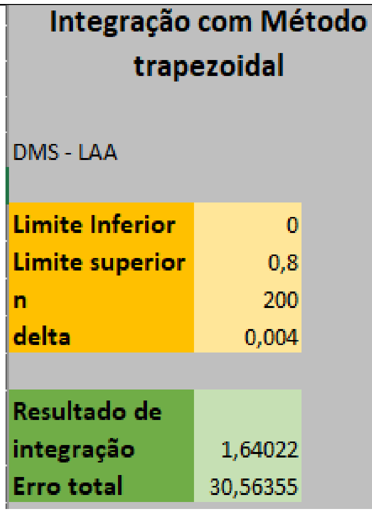
3) Executar o método Trapezoidal em Excel e SciLab

Resposta:

Para esse foi solucionado o método trapezoidal nas linguagens do Excel e SciLab e após uma comparação dada com os resultados em Python. É conveniente afirmar que os dados variam muito pouco entre essas, onde o

SciLab apresentou o resultado como sendo 1,6405, o que é muito mais próximo do real. A função usada foi a apresentada anteriormente no Exercício 1 para o suposto método.

Excel



SciLab

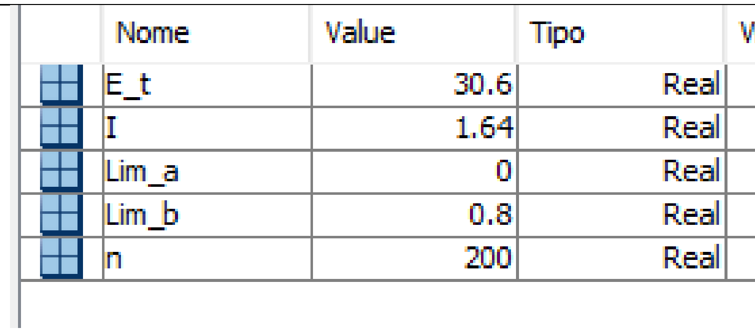


Tabela 3 – Resultados em SciLab e Excel

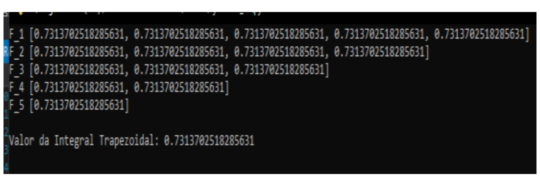
Obs. Código-Projeto se encontra em anexo aos demais arquivos.

4) Método de Romberg e Gauss

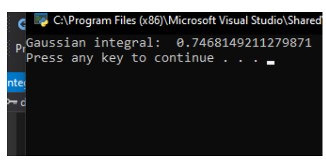
Resposta:

Para esse exercício foi feita a solução de 4,5,6. Onde foi utilizada a função $f(x) = e^{-x^2}$. Os resultados usados são apresentados na tabela a seguir, onde é possível verificar que os valores se aproximam bastante entre os métodos, Romberg foi o único que ficou fora do valor de 0,74 sendo sua solução 0,73 o que dependendo da necessidade de precisão, não chega a ser um erro muito grosseiro.

Romberg



Gauss



Trapézio

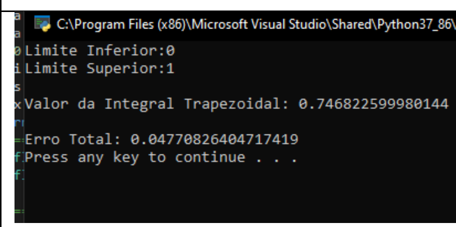


Tabela 4 – Resultados dos demais métodos

Obs. Código-Projeto se encontra em anexo aos demais arquivos.

5) Solução de Circuito eletrônico usando o Trapezoidal.

Resposta:

Dado um circuito RL, foi pedido para calcular a corrente que flui pelo circuito antes e depois do acionamento de uma chave que integra o Indutor ao circuito, conforme figura apresentada a seguir (de autoria do Autor e realizada no ATP). Também foi requisitado uma comparação dos dados retornados com um software de simulação de circuito, esse que foi selecionado o ATP.

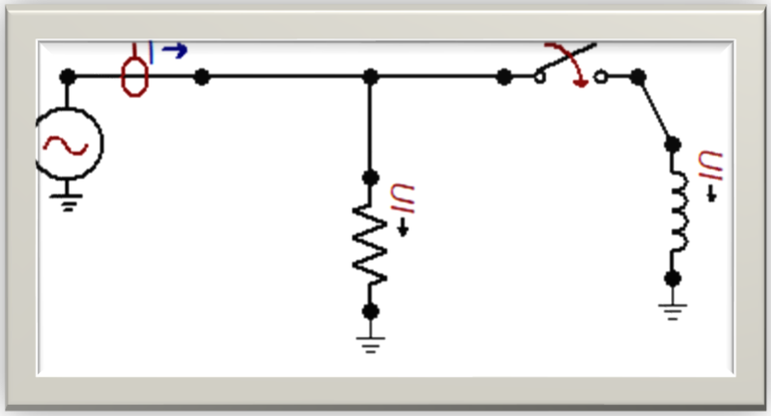


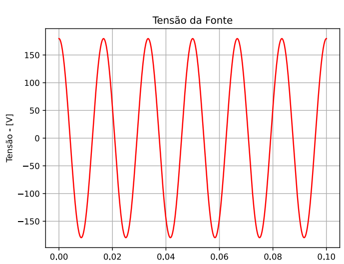
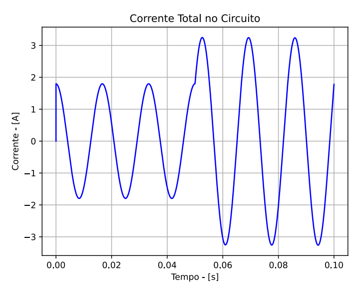
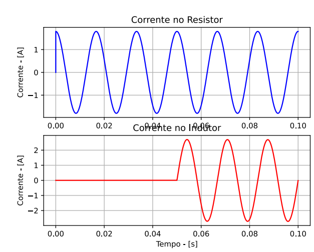
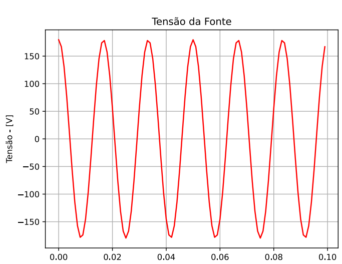
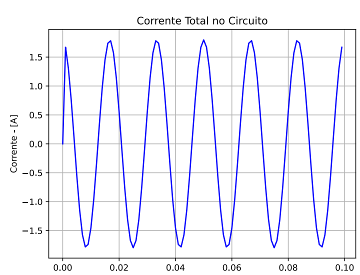
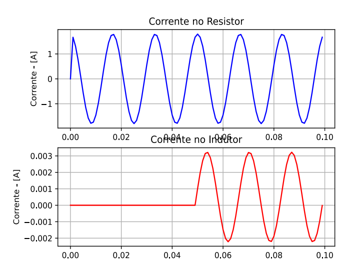
Figura 1 – Circuito de Análise em ATP

Para esse, os dados gerais do circuito são:

Tensão da Fonte	127 V
Frequência	60 Hz
Defasagem	0°
Resistencia	100Ω
Indutância	176mH
Tempo de Simulação	100ms
Fechamento da Chave	50ms

Tabela 4 – Definições Gerais do Circuito

Foram realizadas três simulações em ambos os programas, uma com tempo de amostras de 1µs, 1ms e 4ms, os gráficos resultantes são apresentados na tabela a seguir.

1µs			
1ms			

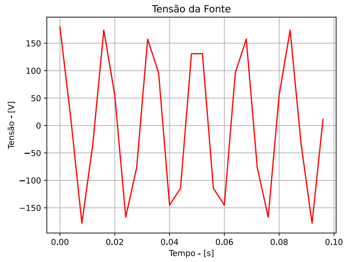
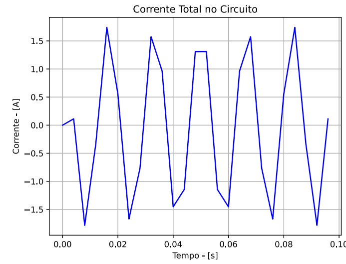
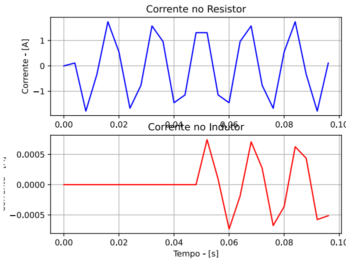
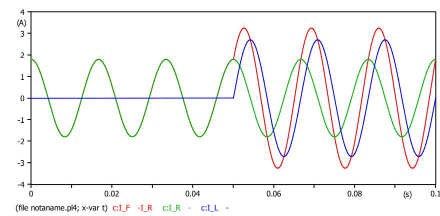
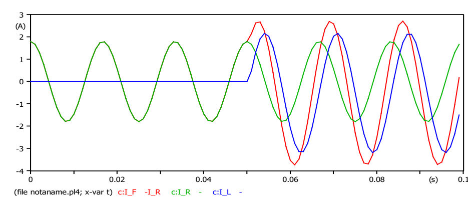
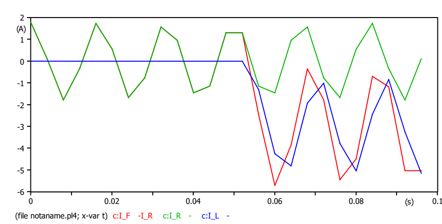
4ms			

Tabela 5 – Resultados dados em Python

Os resultados apresentados no software ATP são apresentados na tabela 6.

1μs	
1ms	
4ms	

Como é possível observar, alterando os subintervalos, ou tempo de amostra, a forma final da onda resultante sofre uma alteração bastante drástica, sendo que com 4 ms é quase um conjunto irregular de curvas. Todavia, para os casos anteriores, ambas as simulações apresentaram resultados bastante semelhantes o que demonstra a eficiência do método trapezoidal para solução de circuitos elétricos.

Obs. Código-Projeto se encontra em anexo aos demais arquivos, **Aqui as imagens podem sofrer ampliação, não há degradação da qualidade, formato .svg utilizado.**

DEMAIS CÓDIGOS PARA EXERCÍCIOS ESTÃO NO ARQUIVO .rar ANEXO NO MICROSOFT TEAMS

ARQUIVO PARA O ATP REALIZADO NA VERSÃO 7.0

Exercício 1/a-b ==

```
#==================================================================================
# ============================ Integração por Euler =============================
#################################
# This code made the progressive and regressive integrations
# Autor : Daniel Marques
# Electrical Engeneering - 2021
#################################
#==================================================================================

import numpy as np
import time

# ============================== Space for Functions ==========================
def f(x):
    return  x**2-15*x+36
def eulerprog(a,b,y,d,s):
    R = 0
    aux = a
    if (s == 'p'):
        if (aux < b):                       # Laço Progressivo da Integral
            R = R + y*d
            aux = aux + d
    else:
        aux = b
        if(aux > a):                        # Laço Regressivo da Integral
            R = R + y*d
            aux = aux - d
    return  R
# ============================== Space for Input ===============================
Lim_a = float(input('Digite o Limite Inferior:'))
Lim_b = float(input('Digite o Limite Superior:'))
y = float(input('Valor da função no limite:'))
select = input('Método Progressivo ou Regressivo? (P ou R):')
# ============================== Error Definition ==============================
d = 5*10^-6    # definição dos passos
# ============================== Main Loop/Output =============================
Result = 0
Result = eulerprog(Lim_a,Lim_b,y,d, select)

print('O resultado da Integral é:',Result)
# ============================== Space for Plots ===============================


# Exercício 1/c
```

```
#==================================================================================
#======================= Integral Por Simpson 1/3 ==============================
#==================================================================================
#################################
# Esse é feito usando da função para o cálculo;
# Caso seja conveniente, substituir a função f(x) por valores conhecidos para
# análise da integral.
#
# Autor : Daniel Marques
# Electrical Engeneering - 2021
#################################
#==================================================================================

import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt

# ============================== Space for Functions ==========================
def f(x):

    return 400*x**5-900*x**4+675*x**3-200*x**2+25*x+0.2

def simpson1_3(a, b, d, v):

    delta = (b - a) / d
    x = a
    s = 0
    if(v == 'f'):
        for i in range(d):
            s = s + (f(x)+f(x+delta))*delta/3
            x = x + delta
    else:
        x_a = np.linspace(a,b,len(v))
        y_a  = interpolate.CubicSpline(x_a,v)
        for i in range(d):
            s = s + (y_a(x)+y_a(x+delta))*delta/3
            x = x + delta

    return s
# ============================== Space for Input ===============================
curve = np.zeros(3)
Lim_a = float(input('Limite Inferior:'))
Lim_b = float(input('Limite Superior:'))
aux = input('Defina se é por função ou por valores [f/v] :')
```

```
if(aux == 'v'):
    for i in range(3):
        curve = float(input('Valores:'))
# ============================== Error Definition ==============================
n = int(1 / (5 * 10**-3))                              # Número de Subdivisões
xa = np.linspace(Lim_a,Lim_b,n)                        # Vetor de valores plot

# =============================== Main Loop/Output ==============================
print('Valor da Integral por Simpson 1/3:', simpson1_3(Lim_a, Lim_b, n,aux))

# ============================== Space for Plots ===============================
plt.plot(xa, f(xa), 'b')

plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()

// ---------------------------------------------------------------------------
# Exercício 1/d

#================================================================================
#====================== Integral Por Simpson 3/8 ==============================
#================================================================================
##################################
# Esse é feito usando da função para o cálculo;
# Caso seja conveniente, substituir a função f(x) por valores conhecidos para
# análise da integral.
#
# Autor : Daniel Marques
# Electrical Engeneering - 2021
##################################
#================================================================================

import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt

# ============================= Space for Functions ==========================
def f(x):                                    # Espaço para uso da função definida automaticamente

    return 400*x**5-900*x**4+675*x**3-200*x**2+25*x+0.2

def simpson3_8(a, b, d, v):
```

```
    delta = (b - a) / d                                # Cálculo dos retangulos contidos na função (Soma de Riemann)
    x = a
    s = 0
    if(v == 'f'):                                      # Efetua Integral usando a função f(x) definida anteriormente
        for i in range(d):
            s = s + 3*(f(x)+f(x+delta))*delta/8
            x = x + delta
    else:                                              # Efetua Integral interpolando os pontos dados pelo usuário
        x_a = np.linspace(a,b,len(v))
        y_a  = interpolate.CubicSpline(x_a,v)
        for i in range(d):
            s = s + (y_a(x)+y_a(x+delta))*3*delta/8
            x = x + delta

    return s
# ============================== Space for Input ==============================
curve = np.zeros(3)                                    # Vetor Y para uso da interpolação
Lim_a = float(input('Limite Inferior:'))
Lim_b = float(input('Limite Superior:'))
aux = input('Defina se é por função ou por valores [f/v] :')
if(aux == 'v'):
    for i in range(3):
        curve = float(input('Valores:'))
# ============================== Error Definition =============================
n = int(1 / (5 * 10**-3))                              # Número de Subdivisões
xa = np.linspace(Lim_a,Lim_b,n)                        # Vetor de valores plot

# ============================== Main Loop/Output =============================
print('Valor da Integral por Simpson 3/8:', simpson3_8(Lim_a, Lim_b, n,aux))
print('\nNúmero de retangulos usados:',n)

# ============================== Space for Plots ==============================
plt.plot(xa, f(xa), 'b')                               #Plotagem do gráfico real da função de análise
plt.grid()
plt.title('f(x) = 400x^5-900x^4+675x^3-200x^2+25x+0.2')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

//---

Exercício 1/e

```
#==============================================================================
# ============================ Integral Trapezoidal ===========================
#################################
# Considerations of project
# Autor : Daniel Marques
# Electrical Engeneering - 2021
#################################
#==============================================================================

import numpy as np
import time
import matplotlib.pyplot as plt
# ============================== Space for Functions ==========================
def f(x):

    return 400*x**5-900*x**4+675*x**3-200*x**2+25*x+0.2

def trapezio(a, b, d):

    delta = (b - a) / d
    x = a
    s = 0
    for i in range(d):
        s += (f(x) + f(x+delta)) * delta / 2
        x += delta
    return s
# ============================== Space for Input ==============================
Lim_a = float(input('Limite Inferior:'))
Lim_b = float(input('Limite Superior:'))

# ============================== Error Definition =============================
n = int(1 / (5 * 10**-3))                # Número de Subdivisões
xa = np.linspace(Lim_a,Lim_b,n)                # Vetor de valores plot

# ============================== Main Loop/Output =============================

I = trapezio(Lim_a, Lim_b, n)
E_t = -1/12*(-f(I)*(Lim_b-Lim_a)**3)
print('\nValor da Integral Trapezoidal:',I)
print('\nErro Total:',E_t)
# ============================== Space for Plots ==============================
plt.plot(xa, f(xa), 'b')
plt.title('F(x) = 400x^5-900x^4+675x^3-200x^2+25x+0.2')
plt.grid()
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.show()

//-----------------------------------------------------------------------------------

# Exercício 4

#==================================================================================
# ============================= Integral Romberg ============================
###################################
# Considerations of project
# Autor : Daniel Marques
# Electrical Engeneering - 2021
###################################
#==================================================================================

import numpy as np
import time
import matplotlib.pyplot as plt
# =============================== Space for Functions ===========================
def romberg(col1):

  col1 = [item for item in col1]
  n = len(col1)

  for j in range(n - 1):
    temp_col = [0] * (n - 1 - j)
    for i in range(n - 1 - j):
      power = j + 1
      temp_col[i] = (4 ** power * col1[i + 1] - col1[i]) / (4 ** power - 1)
    col1[:n - 1 - j] = temp_col
    print(f'F_{j+2}',temp_col)
  return col1[0]

def trapezio(f,a, b, d):

  n = int((b - a) / h)
  soma = 0
  for k in range(1, n):
    soma += f(a + k * h)
  return (h / 2) * (f(a) + 2 * soma + f(b))

def f(x):
```

```
    return np.exp(-x**2)



# ============================== Space for Input ==============================

Lim_a, Lim_b = [0, 1]

h = 0.5
k = 5
hs = [h / 2 ** i for i in range(k)]
col1 = [trapezio(f, Lim_a, Lim_b, hi) for hi in hs]
print('F_1', col1)
# ============================== Error Definition ==============================
n = 3                                       # Número de Subdivisões
xa = np.linspace(Lim_a,Lim_b,n)                        # Vetor de valores plot
e = 5*10**-6                                    # Erro requerido
E_a = float(0)
# ============================== Main Loop/Output ==============================
r = romberg(col1)


print('\nValor da Integral Trapezoidal:',r)
#print('\nErro Total:',E_a)
# ============================== Space for Plots ==============================
plt.plot(xa, f(xa), 'b')
plt.title('F(x) = Sen (x)')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()

// -----------------------------------------------------------------------------

#Exercício 5

#===============================================================================
# ============================ Integração Gauss ==============================
################################
# This code made the progressive and regressive integrations
# Autor : Daniel Marques
# Electrical Engeneering - 2021
################################
```

```
#=================================================================================

import numpy as np
import math

# =============================== Space for Functions ==========================
def gauss(f, a, b, E, A):
    x = np.zeros(3)
    for i in range(3):
        x[i] = (b+a)/2 + (b-a)/2 *E[i]

    return (b-a)/2 * (A[0]*f(x[0]) + A[1]*f(x[1]) + A[2]*f(x[2]))
# =============================== Space for Input ===============================

E = np.array([-0.774597, 0.000000, 0.774597]) # X
A = np.array([0.555556, 0.888889, 0.555556])  # Coeficientes
# =============================== Error Definition ==============================

# =============================== Main Loop/Output ==============================
f = lambda x: math.exp(-x**2)
a = 0.0; b = 1

areaGau = gauss(f, a, b, E, A)
print("Gaussian integral: ", areaGau)

# =============================== Space for Plots ===============================

# Exercício 7

#=================================================================================
#============================ Modelagem Circuito ============================
#=================================================================================
#################################
# Considerations of project
#
# Autor : Daniel Marques
# Electrical Engeneering - 2021
#################################
#=================================================================================

import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
```

```
# ============================== Space for Functions ==========================
def V_f(x):
    Vrms = 127
    f = 60
    Ang = 0
    return  Vrms*np.sqrt(2)*np.cos(2*np.pi*f*x + Ang)

# ============================== Space for Input ===============================

L = 176*10**-3                                    # Valor do indutor

dt = 0.1                                          # Tempo de Simulação
t = 4*10**-3                                      # Duração

f = 60                                            # Frequência
Vrms = 127                                        # Tensão Rms
Ang = 0                                           # Defasagem

vt = np.arange(0,dt,t)                            # vetor tensão
#=============================================================================
# ---------------------------- Componentes ----------------------------------

R_L = 2*L/dt                                      # Resistencia Indutor
I_L = np.zeros(len(vt))                           # Corrente Indutor

R_1 = 100                                         # Resistencia
I_R = np.zeros(len(vt))                           # Corrente Resistencia

I_T = np.zeros(len(vt))                           # Corrente da Fonte
# ============================== Error Definition =============================

# ============================== Main Loop/Output =============================
i = 1

while(i < len(vt)):

    if(vt[i] < 50*10**-3):

        I_R[i] = V_f(vt[i])/R_1
        I_T[i] = I_R[i]
    else:

        I_R[i] = V_f(vt[i])/R_1
        I_L[i] = (1/R_L)*V_f(vt[i]) + I_L[i-1] + (1/R_L) * (V_f(vt[i])-dt)
```

```
        I_T[i] = I_R[i] + 10**-5*I_L[i]

    i = i + 1

# ============================== Space for Plots ==============================
plt.figure(1)
plt.plot(vt, I_T, 'b')
plt.title('Corrente Total no Circuito')
plt.grid()
plt.xlabel('Tempo - [s]')
plt.ylabel('Corrente - [A]')
# \\------------------------------ \\ -------------------------------------\\
plt.figure(2)
plt.plot(vt, V_f(vt),'r')
plt.title('Tensão da Fonte')
plt.grid()
plt.xlabel('Tempo - [s]')
plt.ylabel('Tensão - [V]')
# \\------------------------------ \\ -------------------------------------\\
plt.figure(3)

plt.subplot(2,1,1)
plt.title('Corrente no Resistor')
plt.plot(vt,I_R,'b')
plt.grid()
plt.ylabel('Corrente - [A]')
# =============================
plt.subplot(2,1,2)
plt.plot(vt,I_L*10**-5,'r')
plt.title('Corrente no Indutor')
plt.grid()
plt.xlabel('Tempo - [s]')
plt.ylabel('Corrente - [A]')
# \\------------------------------ \\ -------------------------------------\\
plt.show()
```