Relatório da Tarefa 2

Erik Davino Vincent - Artur Ortiz
NUSP: 10736584 - 10734071
BMAC - Turma 54

August 27, 2019

IME - USP

1 Introdução

Usando o método de Euler, devemos encontrar a solução numérica para a EDO:

$$\dot{x}(t) = -100x(t),\ x(0) = 1 \tag{1}$$

utilizando diferentes valores para $\Delta t = \frac{0.1}{2^m},\ m = 0, 1, 2, 3, 4, 5$, com $t \in [0, 1]$.

Além disso, devemos utilizar o método de Euler Modificado para resolver a EDO para $t = 1$ no intervalo $[0, 1]$ da forma:

$$\dot{x}(t) = e^t(\sin(\omega t) + \omega\cos(\omega t)),\ x(0) = 1,\ \omega = 2 \tag{2}$$

que possui solução conhecida:

$$\begin{aligned} x(t) = e^t\sin(\omega t) + 1,\ \omega = 2 \\ x(1) = 3.471726672 \end{aligned} \tag{3}$$

Queremos verificar para o problema da equação (1) o comportamento da convergência através da observação dos gráficos gerados com Δts cada vez menores, uma análise menos técnica, que serve de exemplo para verificar se o método de está funcionando. Para o problema da equação (2) iremos fazer uma análise do erro absoluto das aproximações, construindo uma tabela de convergência para as aproximações numéricas, assim como da aproximação do erro para zero. Além disso, iremos verificar a ordem do método, p, através do log dos erros para cada Δt, pela forma:

$$p \approx \log_{\frac{\Delta t^k}{\Delta t^{k+1}}}\left(\left|\frac{erro^k}{erro^{k+1}}\right|\right) \tag{4}$$

Verificaremos que as estimativas para $p \longrightarrow 2$, a ordem do método de Euler Modificado.

Obs.: A ordem do método indica a velocidade de convergência de acordo com o valor de $(\Delta t)^p$, onde p é o valor da ordem.

1.1 Métodos e Implementação

O método de Euler pode ser escrito da forma:

$$\begin{aligned} y(t_0) &= y_0 \\ y_{k+1} &\approx y_k + \Delta t f(t_k, y_k) \\ t_{k+1} &= t_k + \Delta t \end{aligned} \tag{5}$$

onde $f(t, y)$ é uma função de passo do método (para nosso caso, a equação (1)). Definimos: $[t_0, t_f] \in Dom(y(t))$, $\Delta t = \frac{t_f - t_0}{n}$, n o número de passos do método. Em cada iteração calculamos a próxima aproximação de $y(t)$ utilizando a aproximação anterior.

O método de Euler Modificado, que é uma melhoria do método de Euler, converge de forma mais rápida (ordem 2). Pode ser escrito como:

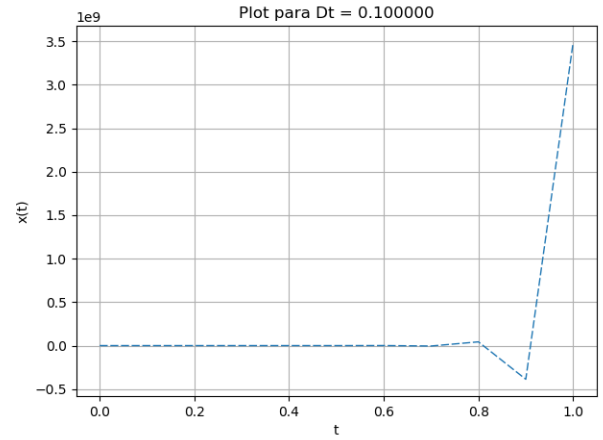
$$
\begin{aligned}
y(t_0) &= y_0 \\
y_{k+1} \approx y_k + \frac{\Delta t}{2}[f(t_k, y_k) + f(t_k + \Delta t, y_k + \Delta t f(t_k, y_k))] \quad &(6) \\
t_{k+1} &= t_k + \Delta t
\end{aligned}
$$

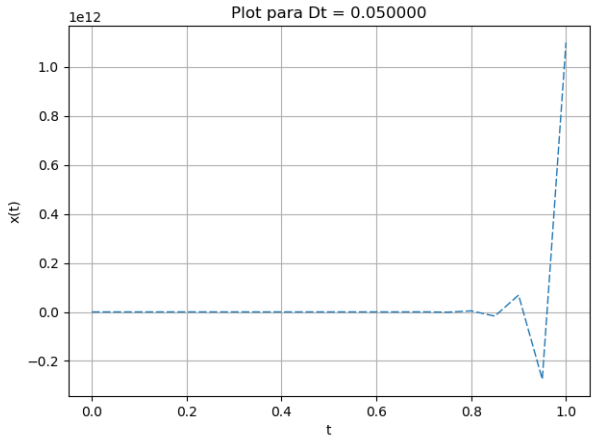
com as outras definições iguais as do método anterior.

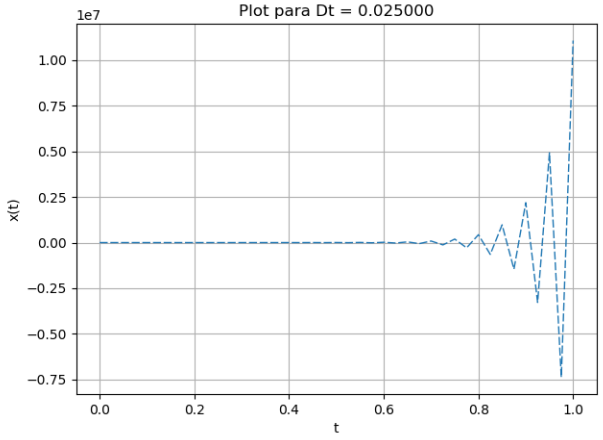
A implementação foi feita em Python e pode ser encontrada integralmente no arquivo *EPT2.py* ou no Apêndice 1 ao final do relatório. Nota-se que os parâmetros dos métodos devem ser alterados no próprio arquivo do programa, i.e. não recebe inputs de usuário. As mudanças devem ser "hardcoded" no código.

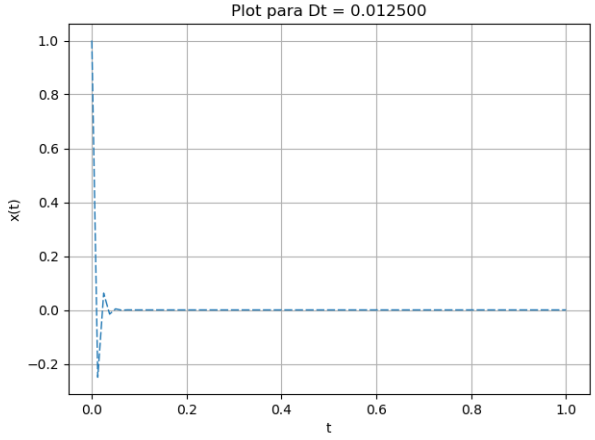
2 Resultados

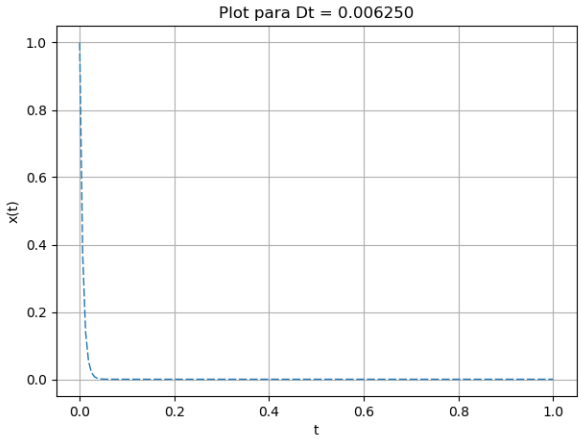
Vejamos abaixo os resultados obtidos para o problema da equação (1), utilizando Euler. Note que $Dt = \Delta t$. Note também que, por exemplo, para $Dt = 0.1$, temos que $m = 0$, $Dt = \frac{0.1}{2^0}$, para $Dt = 0.05$, temos que $m = 1$, $Dt = \frac{0.1}{2^1}$, e assim por diante, para cada valor de m.

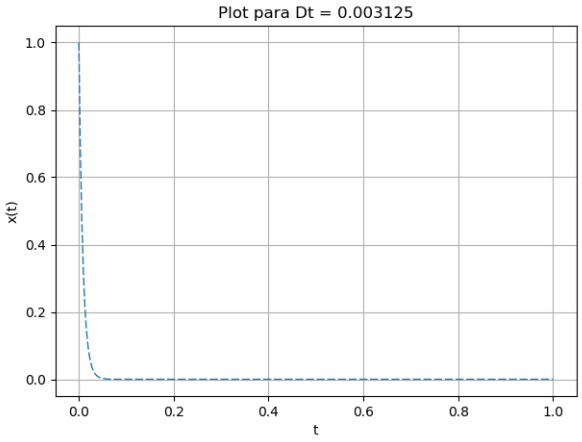












Verificamos a partir desses resultados que para $m = 0, 1, 2$ os resultados divergiram drasticamente do verdadeiro resultado, 0. Respectivamente obtivemos 3486784401, 1099511627776, 11057332. Para os outros valores de m, os resultados convergiram bem. Para explicar isso, podemos analisar o intervalo de estabilidade do método, i.e. o intervalo para o valor de Δt para qual o método converge:

2.1 Intervalo de estabilidade para Euler

Utilizamos a função $\dot{y}(t) = -\lambda y(t),\ y(0) = 1$ como o $f(t, y)$ e desenvolvemos:

$$\begin{aligned} y_{k+1} &= y_k + \Delta t(-\lambda y_k) \\ &= y_k(1 - \lambda\Delta t) \\ (1 - \lambda\Delta t) &= \text{"fator de amplificação do método"} \\ |1 - \lambda\Delta t| &< 1\ ^* \\ \Leftrightarrow -1 < 1 - \lambda\Delta t &< 1 \\ \Leftrightarrow 0 < -\lambda\Delta t &< 2 \\ \Rightarrow -\lambda\Delta t &\in [0, 2] \end{aligned} \tag{7}$$

Isso implica que $-\lambda\Delta t$ deve estar no intervalo $[0, 2]$. Para o caso específico da equação (1), como $\lambda = -100$, teríamos:

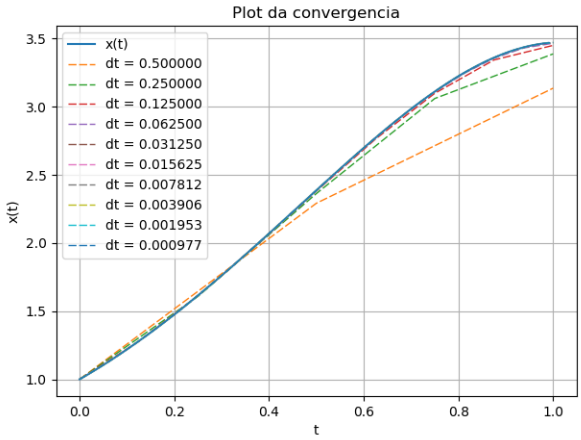
$$100\Delta t = 100\frac{0.1}{2^m} = \frac{10}{2^m} \in (0, 2] \Leftrightarrow m \in [2.32192..., \infty) \tag{8}$$

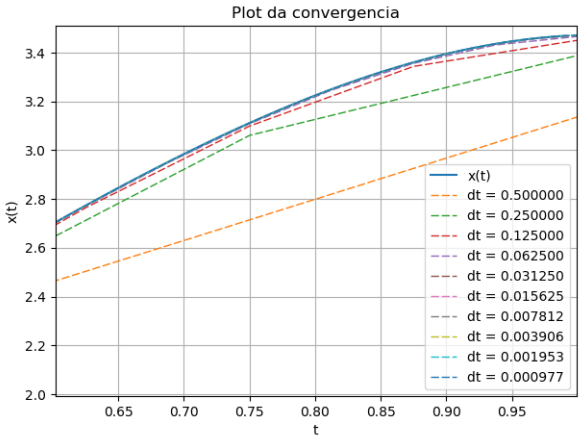
Assim, explicamos os resultados para $m = 0, 1, 2 \notin [2.32192..., \infty)$.

* Veja que o fator de amplificação do método descreve a convergência do método. Para o método converge ir, esse fator deve estar em um intervalo menor do que 1, caso contrário a aproximação "explode".

Veja também o Apêndice 2, para verificar os gráficos dos resultados obtidos pelo método de Euler Modificado.

Vejamos a seguir os resultados obtidos para o problema da equação (2) pelo método de Euler Modificado:





Δt	y_{k+1}	Erro $^{k+1}$	$\frac{\text{Erro}^k}{\text{Erro}^{k+1}}$	$p \approx \log_{\frac{\Delta t^k}{\Delta t^{k+1}}} \left(\left\| \frac{\text{Erro}^k}{\text{Erro}^{k+1}} \right\| \right)$
0.5000000000	3.1368129361			
0.2500000000	3.3885238240	0.083202848	4.0252676915	2.0090847297
0.1250000000	3.4509600147	0.0207666573	4.0065594941	2.0023638997
0.0625000000	3.4665371538	0.0051895182	4.0016541991	2.0005965029
0.0312500000	3.4704294269	0.0012972451	4.0004144426	2.0001494708
0.0156250000	3.4714023691	0.0003243029	4.0001037074	2.0000374041
0.0078125000	3.4716455968	8.10752e-05	4.0000260974	2.0000094126
0.0039062500	3.4717064032	2.02688e-05	4.0000071931	2.0000025944
0.0019531250	3.4717216048	5.0672e-06	4.0000044735	2.0000016135
0.0009765625	3.4717254052	1.2668e-06	4.0000118017	2.0000042565

Table 1: Tabela de convergência do erro e de p.

Analisando os gráficos, temos uma intuição visual de que de fato o método está convergindo para o verdadeiro valor de $x(1)$. Note que a partir de determinado dt não é possível distinguir a curva aproximada da verdadeira.

Analisando a tabela, podemos ver que o erro cai rapidamente e que a convergência é rápida. Podemos ver também que $p \longrightarrow 2$, pelo menos até a nona linha. Porém, da décima linha para frente (resultados omitidos) ocorre um aumento na estimativa de p. Isso pode ser explicado por erros computacionais para divisões com números muito pequenos.

Apêndice 1:

```python
import matplotlib.pyplot as plt
import numpy as np

# função de "passo", dx/dt
def f(t,x):
    return np.exp(t)*(np.sin(2*t) + 2 * np.cos(2*t))

def f2(t,x):
    return -100*x

# metodo de EULER
def euler(to, tf, n, xo, phi):
    H = (tf-to)/n

    vec_x = [xo] # vetor com cada ponto obtido de x
    vec_t = [to] # vetor com cada ponto de t

    for i in range(1, n+1, 1):
        vec_x.append(vec_x[i-1] + H*phi(vec_t[i-1],vec_x[i-1]))
        vec_t.append(vec_t[i-1] + H)

    return (vec_t, vec_x)

# metodo de Euler Modificado
def euler_mod(to, tf, n, xo, phi):
    H = (tf-to)/n

    vec_x = [xo] # vetor com cada ponto obtido de x
    vec_t = [to] # vetor com cada ponto de t

    for i in range(1, n+1, 1):
        tmp = phi(vec_t[i-1],vec_x[i-1]) # pequena otimização, para calcular a phi apenas uma vez
        vec_x.append(vec_x[i-1] + (H/2)*(tmp + phi(vec_t[i-1] + H, vec_x[i-1] + H*tmp)))
        vec_t.append(vec_t[i-1] + H)

    return (vec_t, vec_x)

def main():

    # variaveis de entrada:
    xo = 1 # x(to) inicial
    to = 0 # t inicial
    tf = 1 # t final
    n = 1 # n inicial/L
    L = 2 # fator de aumento de n
    fun = f # escolha da funcao (f ou f2)
    metodo = euler_mod # escolha do metodo (euler ou euler_mod)

    # plotagem da função verdadeira de f (nao de f2, pode ser ignorado se necessario)
    plt.figure(figsize = (7,5))
    u = np.arange(0.,1.,0.01)
```

```
52      plt.plot(u, np.exp(u)*np.sin(2*u)+1, label = "x(t)")
53
54
55      erro_ant = 1 #  erro anterior inicial (que não existe, mas é necessário para o cálculo não falhar)
56      for k in range(10):
57          n *= L
58          eu = metodo(to, tf, n, xo, fun) #chama o metodo escolhido
59          plt.plot(eu[0], eu[1], lw = 1, dashes = [6,2], label = ("dt = %f" %((tf-to)/n)))
60          # plot da função pelo método
61
62          erro = abs(eu[1][-1] - 3.471726672)
63          div_err = erro_ant/erro
64          print(round((tf - to)/n,10), " & " ,round(eu[1][-1],10), " & ",\
65          round(erro, 10), " & ", round(div_err,10), " & ",round(np.log(div_err)/np.log(L),10), "\\\\")
66          erro_ant = erro
67
68      # configuração do plot
69      plt.grid(True)
70      plt.xlabel("t")
71      plt.ylabel("x(t)")
72      plt.legend()
73      plt.title("Plot da convergencia")
74      plt.show()
75
76  main()
```

Note que na linha 64 do programa utilizamos $np.log(div\_err)/np.log(L)$. Isso vem da ideia de que:

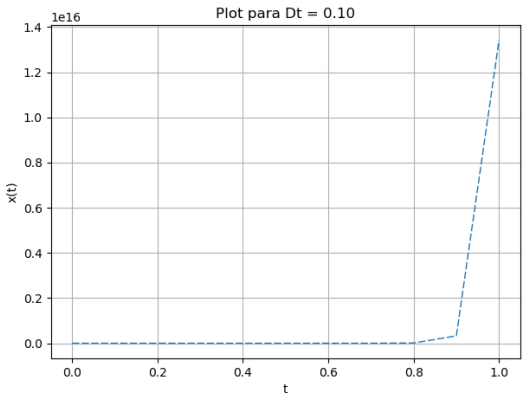
$$\log_c(x) = \frac{\log_a(x)}{\log_a(c)}$$

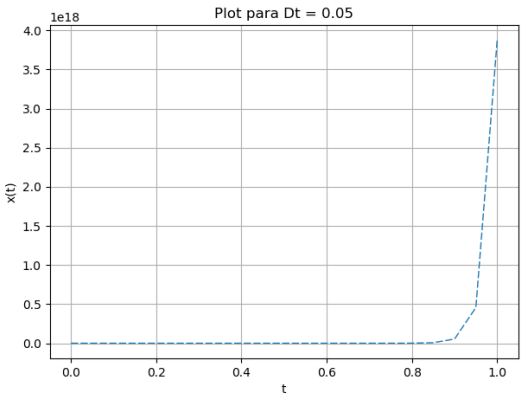
Utilizamos L, pois sabemos que a base do log será o próprio fator de aumento do n:

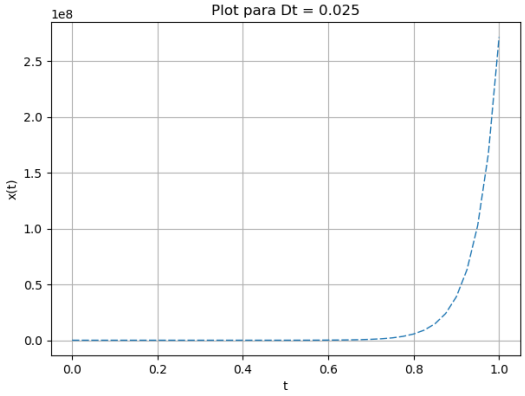
$$\frac{\frac{t_f - t_0}{n}}{\frac{t_f - t_0}{Ln}} = L$$

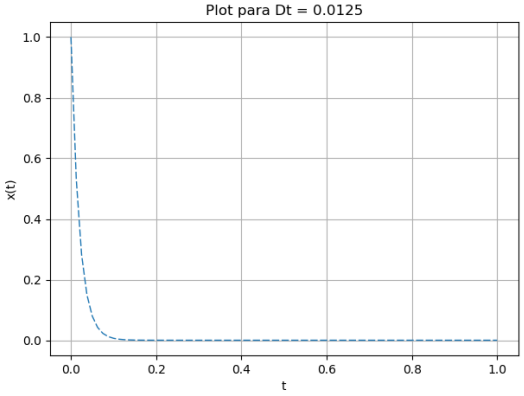
Note que $H = \Delta t$ nas linhas 13 e 26.

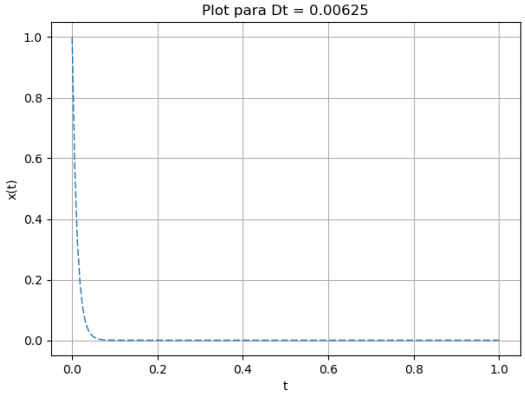
Apêndice 2:

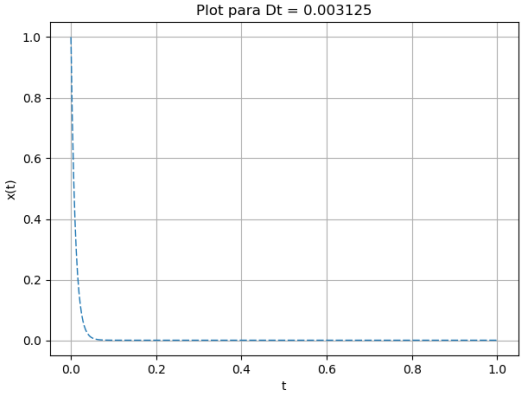












Note como a convergência é mais rápida e mesmo os gráficos que divergem são muito mais suaves do que pelo método de Euler.