Trabalho 2:

Leonardo Maximo Silva, 200022172

Sumário

1 Introdução

O trabalho é uma lista de 10 exercícios abordando a resolução de Equações Diferenciais Ordinárias (EDOs) a partir do uso do método das diferenças finitas implementado em Python. Os 6 primeiros exercícios consistem na comparação entre os Métodos de Euler Explícito, Euler Implícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. Os 5 últimos exercícios consistiram na resolução de problemas numéricos reais, como, por exemplo, a equação diferencial de segunda ordem correspondente a um pêndulo simples, de modo a se comparar a convergência dos métodos de Euler Explícito e Range-Kutta de quarta ordem. A seguir, será apresentado a resolução para cada exercício proposto.

2 Exercício 1

Para o exercício 1, foi calculado a solução y(t) da equação diferencial com valor inicial y(0) = 1 para $0 \leq t \leq 1$:

$$y^{'}(t) = 2 - 2 \cdot t + 4 \cdot t^2 - 4 \cdot t^3 - 4 \cdot t^4$$

A partir dos Métodos Numéricos para a resolução de Equações Diferenciais Ordinárias (EDOs) de Euler Explícito, Euler Implícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. Comparou-se, então, os resultados obtidos com a solução analítica da equação diferencial fornecida:

$$y(t) = 1 + 2 \cdot t - t^2 + \frac{4}{3} \cdot t^3 - t^4 - \frac{4}{5} \cdot t^5$$

De acordo com [Ros22], a expressão para cálculo de y_{k+1} para o método de Euler Implícito pode ser dada a partir da manipulação da equação $y_{k+1} = y_k + \Delta t \cdot f_{k+1}$, em que f corresponde a $y^{'}$. Logo, a função utilizada para o cálculo desse método é dada por:

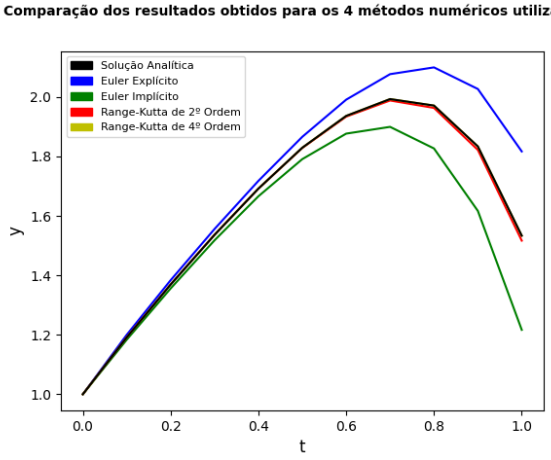
$$y_{k+1} = y_k + \Delta t \cdot f_{k+1}$$

$$y_{k+1} = y_k + \Delta t(\cdot 2 - 2 \cdot t + 4 \cdot t^2 - 4 \cdot t^3 - 4 \cdot t^4)$$

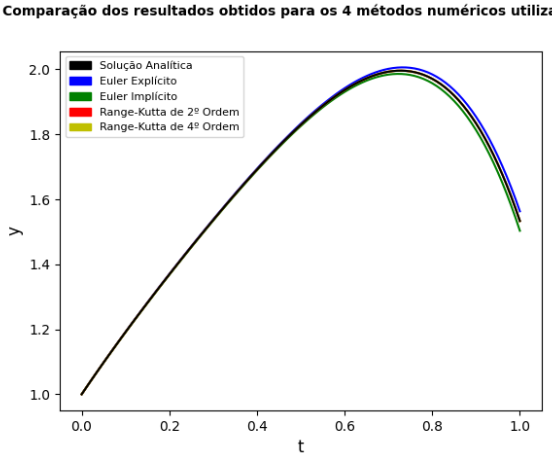
Em relação à implementação do problema, criou-se quatro funções correspondentes aos métodos de Euler Explícito, Euler Implícito, Range-Kutta de 2º Ordem, Range-Kutta de 4º Ordem e a solução analítica. Implementou-se também três funções correspondentes à derivada de y(t) ($y^{'}(t)$), à função utilizada para o cálculo do método de Euler Explícito e a Solução Analítica do problema de modo a tornar o código o mais escalável possível. Dessa forma, foi possível implementar os códigos dos exercícios 3,4,5,6. As funções utilizadas para a implementação dos métodos numéricos também foram posteriormente reaproveitadas para os demais exercícios.

Traçou-se, então, três gráficos com quatro valores diferentes para o passo de tempo, dt: 0.1, 0.01,0.001 e 0.0001.

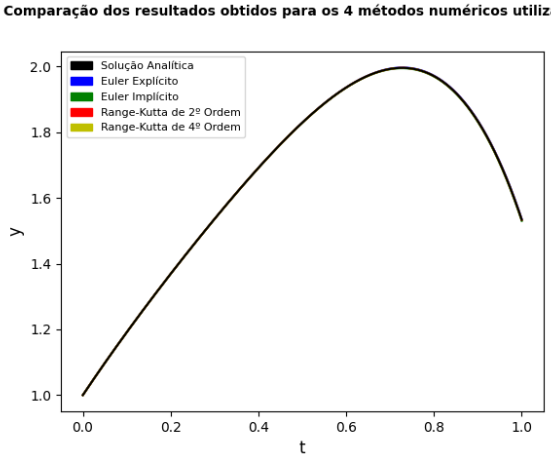
Em 1, é interessante notar que o método de Euler Explícito aproxima o valor da EDO para valores maiores que o valor exato, enquanto que o método de Euler Implícito aproxima o valor da EDO para valores menores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada decrescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor maior que o seu original. Por outro lado, o método de Euler Implícito aproxima a função para um valor menor que o original por levar em consideração um ponto futuro, ou seja, em que a derivada é menor que a do instante dt.



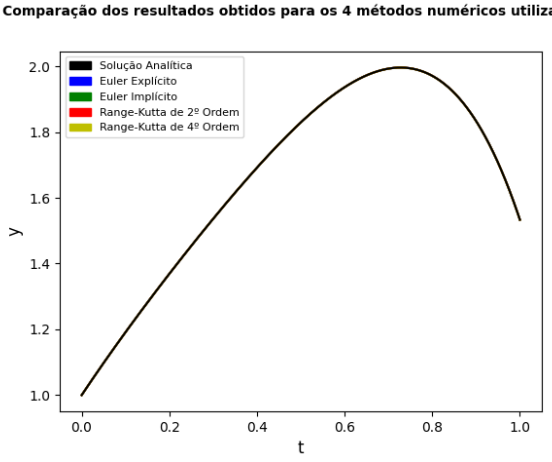
y(t) para dt = 0.1



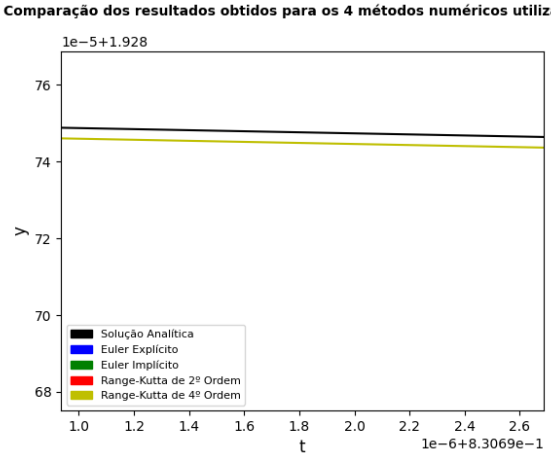
y(t) para dt = 0.01



y(t) para dt = 0.001



y(t) para dt = 0.0001



ampliação para dt = 0.1

Figura 1: Gráficos obtidos para diferentes valores de dt

A Equação Diferencial Finita (EDF), logo, apresenta um erro na quarta casa decimal para o passo de tempo de dt = 0.1, quando utilizado o método de Range-Kutta de 4º Ordem, apresentando, logo, boa convergência para esse método. Ainda que apresente um passo de convergência relativamente próximo ao método de Range-Kutta 4º Ordem, o Range-Kutta de 2º Ordem requer um valor de dt maior que para o Range-Kutta de 4º Ordem, como pode ser observado por, em 1, a linha vermelha do gráfico, correspondente a esse gráfico, ainda não ter convergido satisfatoriamente para o passo de dt = 0.1, convergindo, posteriormente, para os outros passos de dt escolhidos. Tanto o método de Euler Explícito quanto o de Euler Implícito convergem satisfatoriamente para o mesmo passo de tempo, dt = 0.001, o qual é menor que os passos utilizado para os outros dois métodos numéricos utilizados. O método de Range-Kutta de 4º Ordem, logo, converge para maiores valores de dt que os outros métodos utilizados, representando a melhor aproximação possível da EDO original a partir do método das Diferenças Finitas, sendo seguido pelo método de Range-Kutta de 2º Ordem e, posteriormente, pelos Métodos de Euler Explícito e Implícito.

3 Exercício 2

Para o exercício 2, solução y(t) foi calculada a partir da equação diferencial com valor inicial y(1) = 2 para $1 \leq t \leq 2$:

$$y^{'}(t) = 1 + \frac{y}{t}$$

A partir dos Métodos Numéricos para a resolução de Equações Diferenciais Ordinárias (EDOs) de Euler Explícito, Euler Implícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. Comparou-se, então, os resultados obtidos com a solução analítica da equação diferencial fornecida:

$$y(t) = tln(t) + 2t$$

De modo análogo ao realizado em 2, a função utilizada para o cálculo do método de Euler Implícito para a função y^t fornecida é:

$$y_{k+1} = y_k + \Delta t \cdot f_{k+1}$$

$$y_{k+1} = y_k + \Delta t \cdot (1 + \frac{y_{k+1}}{t})$$

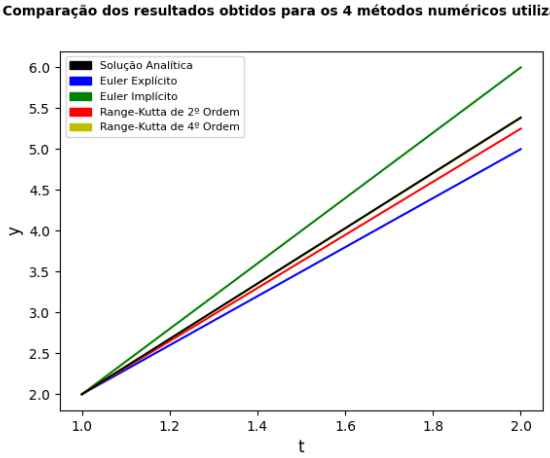
$$y_{k+1} \cdot (1 - \frac{\Delta t}{t}) = y_k + \Delta t$$

$$y_{k+1} = \frac{y_k + \Delta t}{1 - \frac{\Delta t}{t}}$$

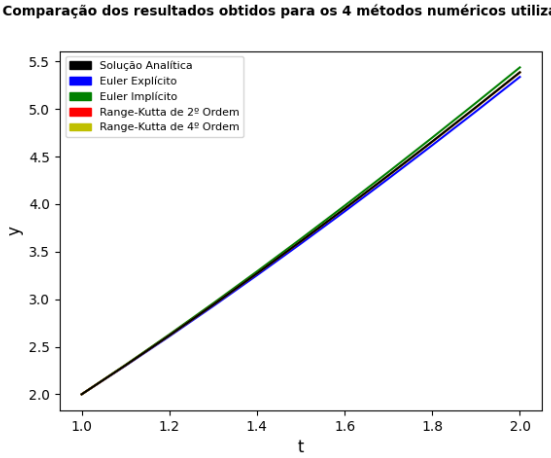
Traçou-se, então, três gráficos com quatro valores diferentes para o passo de tempo, Δt: 1.0, 0.1,0.01 e 0.001, obtendo-se 2.

Em 2, o método de Euler Explícito aproxima o valor da EDO para valores menores que o valor exato, enquanto que o método de Euler Implícito aproxima o valor da EDO para valores maiores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada crescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor menor que o seu original. Por outro lado, o método de Euler Implícito aproxima a função para um valor maior que o original por levar em consideração um ponto futuro, ou seja, em que a derivada é maior que a do instante dt.

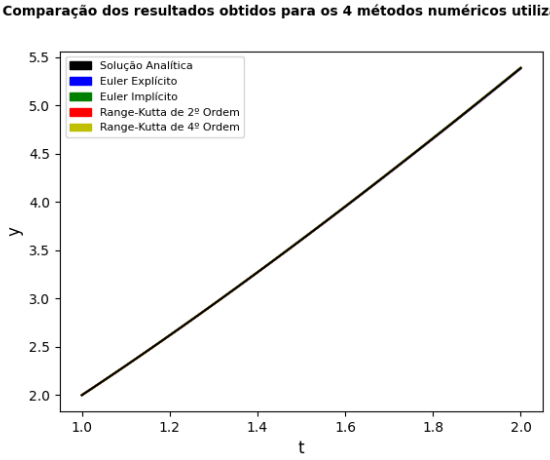
A Equação Diferencial Finita (EDF), logo, apresenta um erro na terceira e quarta casas decimais para os passos de tempo de, respectivamente, dt = 1.0 e dt = 0.1, quando utilizado o método de Range-Kutta de 4º Ordem, apresentando, logo, boa convergência para esse método. O método de Range-Kutta de 2º Ordem apresenta convergência relativamente pior que o método de Range-Kutta de 4º Ordem, como pode ser observado por, em 1, a linha vermelha do gráfico, correspondente a esse gráfico, ainda não ter convergido satisfatoriamente para o passo de dt = 1.0 e ter apresentado um erro cada vez maior conforme o tempo (t) aumenta, convergindo, posteriormente, para os outros passos de dt escolhidos. Tanto o método de Euler Explícito quanto o de Euler Implícito convergem satisfatoriamente para o mesmo passo de tempo, dt = 0.001, o qual é menor que os passos utilizado para os outros dois métodos numéricos utilizados. O método de Range-Kutta de 4º Ordem, logo, converge para maiores valores de dt que os outros métodos utilizados, representando a melhor aproximação possível da EDO original a partir do método das Diferenças Finitas, sendo seguido pelo método de Range-Kutta de 2º Ordem e, posteriormente, pelos Métodos de Euler Explícito e Implícito.



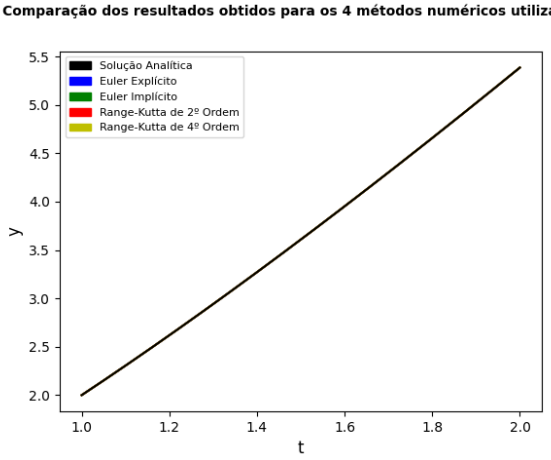
y(t) para dt = 1.0



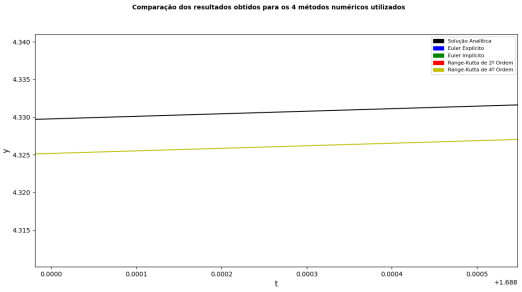
y(t) para dt = 0.1



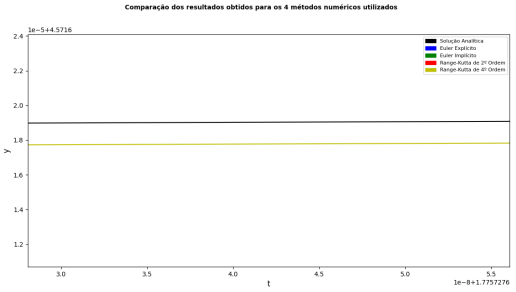
y(t) para dt = 0.01



y(t) para dt = 0.001



ampliação para dt = 1.0



ampliação para dt = 0.1

Figura 2: Gráficos obtidos para diferentes valores de dt

4 Exercício 3

A solução y(t) para o Exercício 3 foi calculada a partir da equação diferencial com valor inicial y(0) = 1 para $0 \leq t \leq 1$:

$$y^{'}(t) = t^2 y$$

A partir dos Métodos Numéricos para a resolução de Equações Diferenciais Ordinárias (EDOs) de Euler Explícito, Euler Implícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. Comparou-se, então, os resultados obtidos com a solução analítica da equação diferencial fornecida:

$$y(t) = e^{\frac{t^3}{3}}$$

De modo análogo ao realizado em 2, a função utilizada para o cálculo do método de Euler Implícito para a função y^t fornecida é:

$$y_{k+1} = y_k + \Delta t \cdot f_{k+1}$$

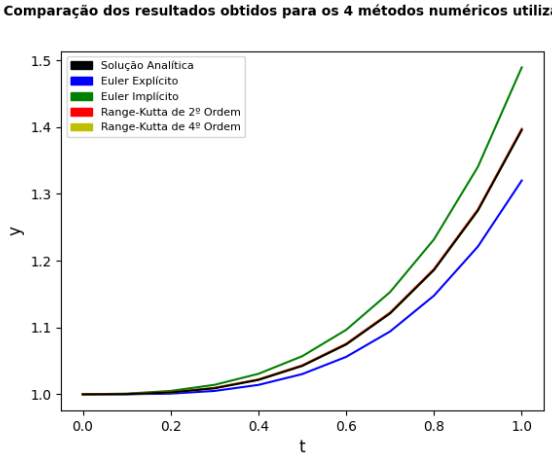
$$y_{k+1} = y_k + \Delta t^2 \cdot (t^2 \cdot y_{k+1} + 1)$$

$$y_{k+1} \cdot (1 - \Delta t \cdot t^2) = y_k$$

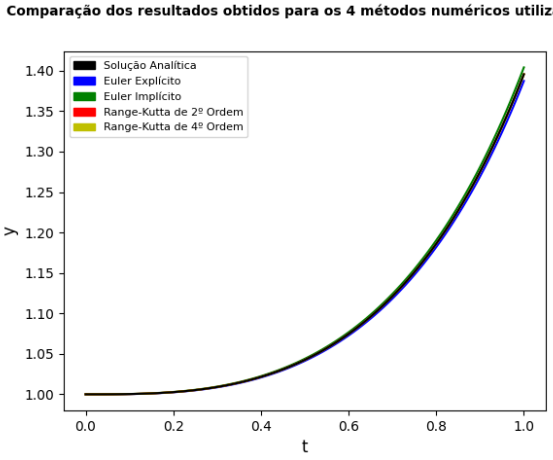
$$y_{k+1} = \frac{y_k}{1 - \Delta t \cdot t^2}$$

Traçou-se, então, três gráficos com quatro valores diferentes para o passo de tempo, Δt: 0.1, 0.01,0.001 e 0.0001.

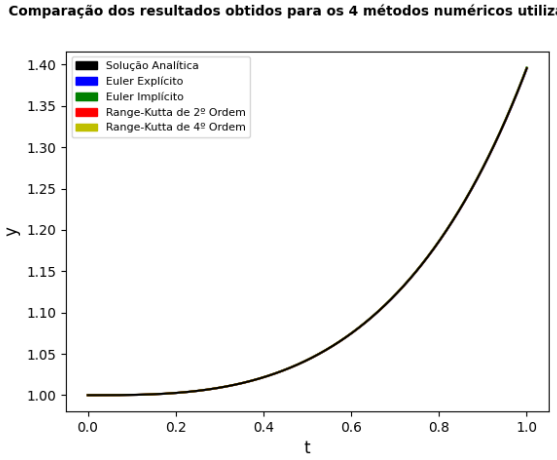
Em 3, o método de Euler Explícito aproxima o valor da EDO para valores menores que o valor exato, enquanto que o método de Euler Implícito aproxima o valor da EDO para valores maiores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada crescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor menor que o seu original. Por outro lado, o método de Euler Implícito aproxima a função para um valor maior que o original por levar em consideração um ponto futuro, ou seja, em que a derivada é maior que a do instante dt.



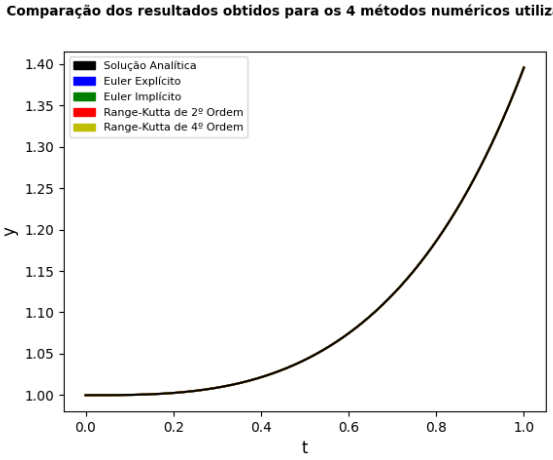
y(t) para dt = 0.1



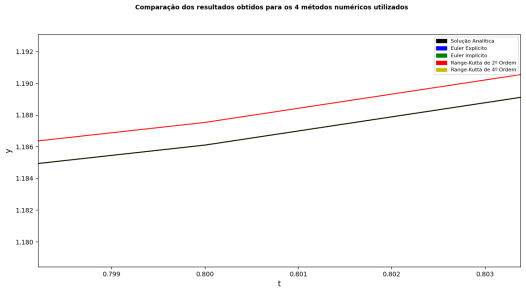
y(t) para dt = 0.01



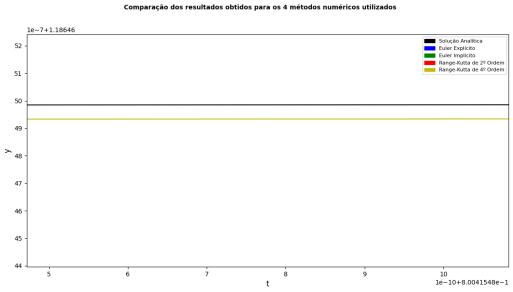
y(t) para dt = 0.001



y(t) para dt = 0.0001



ampliação para enxergar o erro da linha vermelha para dt = 0.1



ampliação para enxergar o erro da linha amarela para dt = 0.1

Figura 3: Gráficos obtidos para diferentes valores de d(t)

Em 3, o método de Euler Explícito aproxima o valor da EDO para valores menores que o valor exato, enquanto que o método de Euler Implícito aproxima o valor da EDO para valores maiores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada crescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor menor que o seu original. Por outro lado, o método de Euler Implícito aproxima a função para um valor maior que o original por levar em consideração um ponto futuro, ou seja, em que a derivada é maior que a do instante dt.

A Equação Diferencial Finita (EDF), logo, apresenta um erro na sexta casa decimal para o passo de tempo de dt = 0.1 quando utilizado o método de Range-Kutta de 4º Ordem, apresentando, logo, boa convergência para esse método. O método de Range-Kutta de 2º Ordem também apresenta boa convergência, ainda que levemente pior que para o método de Range-Kutta de 4º Ordem, como pode ser observado por, em 1, a linha vermelha do gráfico, correspondente a esse gráfico, ter convergido com um erro na primeira unidade para o valor de dt = 0.1, enquanto que o método de Range-Kutta de 4º Ordem apresenta um erro na sexta casa decimal, apresentando uma melhor convergência para os outros passos de dt escolhidos. Tanto o método de Euler Explícito quanto o de Euler Implícito convergem satisfatoriamente para o mesmo passo de tempo, dt = 0.001, o qual é menor que os passos utilizado para os outros dois métodos numéricos utilizados. É interessante notar, entretanto, que esses dois métodos apresentam um erro significativo quando aproximados para dt = 0.1, porém, um erro significativamente menor quando dt = 0.01, sendo notável, a partir da comparação com os gráficos de 2 e 3, que essa é a função que apresenta convergência para maiores valores de dt quando comparada às outras funções aproximadas. O método de Range-Kutta de 4º Ordem, logo, converge para maiores valores de dt que os outros métodos utilizados, representando a melhor aproximação possível da EDO original a partir do método das Diferenças Finitas, sendo seguido pelo método de Range-Kutta de 2º Ordem e, posteriormente, pelos Métodos de Euler Explícito e Implícito.

5 Exercício 4

A solução y(t) para o Exercício 4 foi calculada a partir da equação diferencial com valor inicial y(0) = 1 para $0 \leq t \leq 1$:

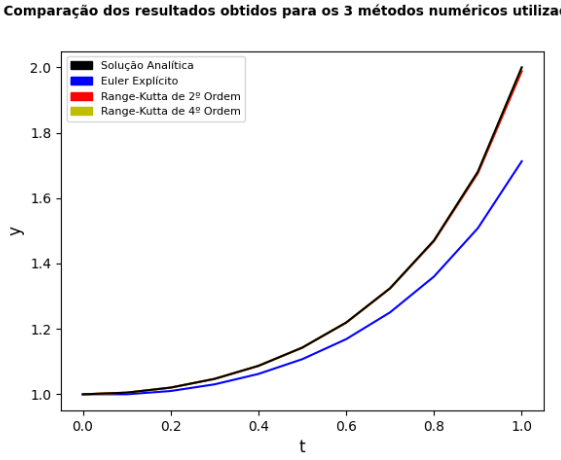
$$y^{'}(t) = ty^2$$

A partir dos Métodos Numéricos para a resolução de Equações Diferenciais Ordinárias (EDOs) de Euler Explícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. O método de Euler Implícito não foi requisitado para a realização da questão. Comparou-se, então, os resultados obtidos com a solução analítica da equação diferencial fornecida:

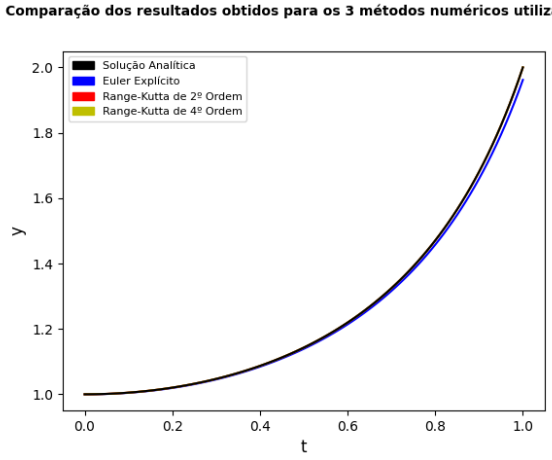
$$y(t) = -\frac{2}{t^2 - 2}$$

Traçou-se, então, três gráficos com quatro valores diferentes para o passo de tempo, Δt: 0.1, 0.01,0.001 e 0.0001.

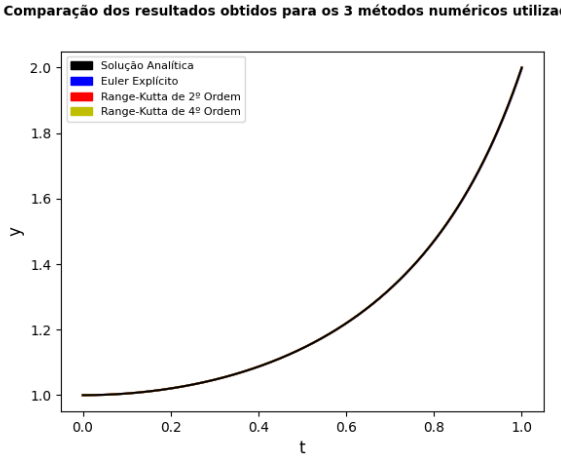
Em 4, o método de Euler Explícito aproxima o valor da EDO para valores menores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada crescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor menor que o seu original.



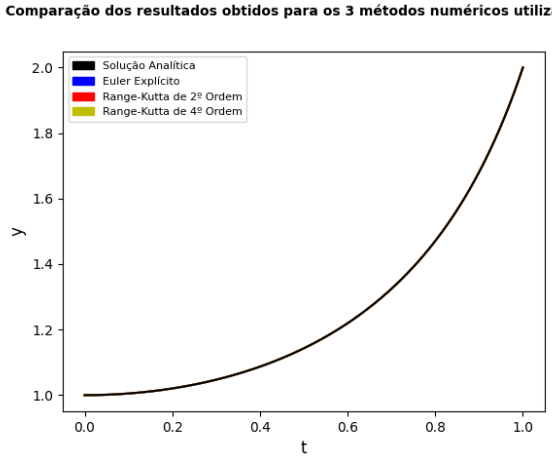
y(t) para dt = 0.1



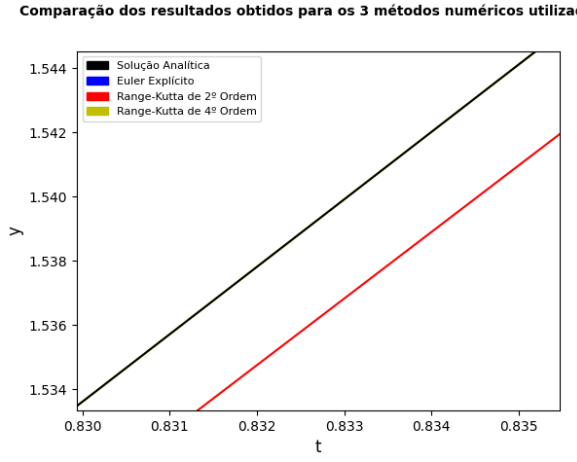
y(t) para dt = 0.01



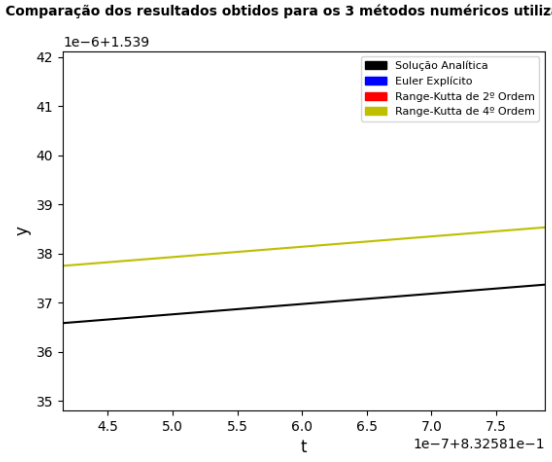
y(t) para dt = 0.001



y(t) para dt = 0.0001



ampliação para enxergar o erro da linha vermelha para dt = 0.1



ampliação para enxergar o erro da linha amarela para dt = 0.1

Figura 4: Gráficos obtidos para diferentes valores de d(t)

A Equação Diferencial Finita (EDF), logo, apresenta um erro na quarta casa decimal para o passo de tempo de dt = 0.1 ao se utilizar o método de Range-Kutta de 4º Ordem, apresentando, logo, boa convergência para esse método. O método de Range-Kutta de 2º Ordem apresenta convergência relativamente pior que o método de Range-Kutta de 4º Ordem, como pode ser observado por, em 1, a linha vermelha do gráfico, correspondente a esse gráfico, ter convergido para a primeira unidade para o passo de dt = 0.1 enquanto o método de Range-Kutta de 4º Ordem apresenta convergência na quarta casa decimal. O método de Euler Explícito converge satisfatoriamente para passo de tempo, dt = 0.001, o qual é menor que os passos utilizado para os outros dois métodos numéricos utilizados, entretanto, é interessante notar como o erro relacionado à sua convergência dimiui significativamente ao se comparar o passo de tempo dt = 0.1 com o passo de tempo dt = 0.01. O método de Range-Kutta de 4º Ordem, logo, converge para maiores valores de dt que os outros métodos utilizados, representando a melhor aproximação possível da EDO original a partir do método das Diferenças Finitas, sendo seguido pelo método de Range-Kutta de 2º Ordem e, posteriormente, pelo Método de Euler Explícito.

6 Exercício 5

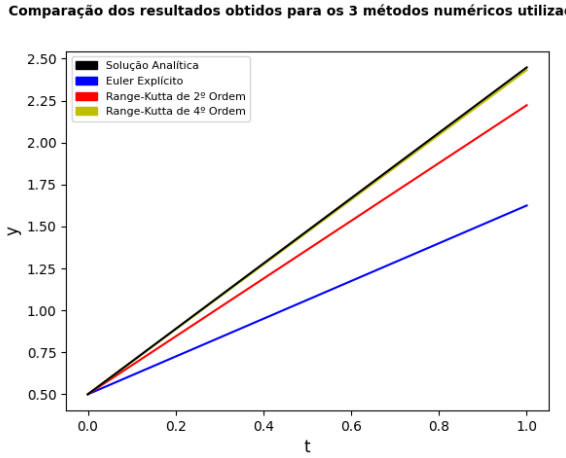
A solução y(t) para o Exercício 5 foi calculada a partir da equação diferencial com valor inicial y(0) = 0.5 para $0 \leq t \leq 1$:

$$y^{'}(t) = 1 + 0.5y^2$$

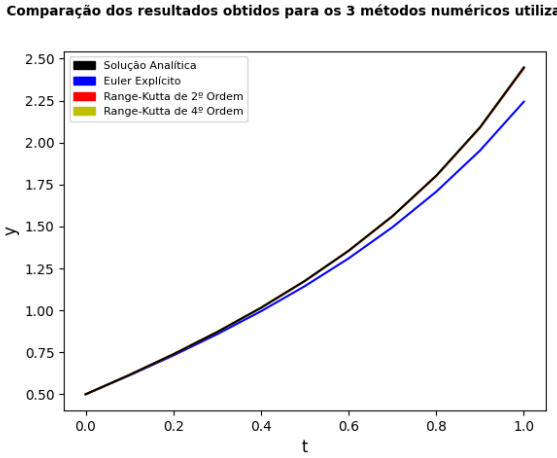
A partir dos Métodos Numéricos para a resolução de Equações Diferenciais Ordinárias (EDOs) de Euler Explícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem. O método de Euler Implícito não foi requisitado para a realização da questão. Comparou-se, então, os resultados obtidos com a solução analítica da equação diferencial fornecida:

$$y(t) = \frac{1}{\sqrt{0.5}} \cdot tan[\sqrt{0.5}t + tan^{-1}(0.5^{\frac{3}{2}})]$$

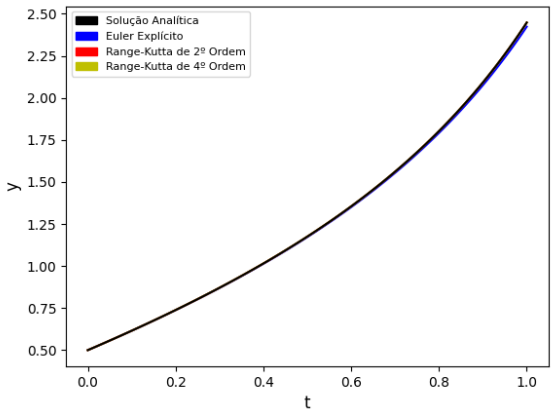
Traçou-se, então, três gráficos com quatro valores diferentes para o passo de tempo, Δt: 1.0, 0.1,0.01 e 0.001.



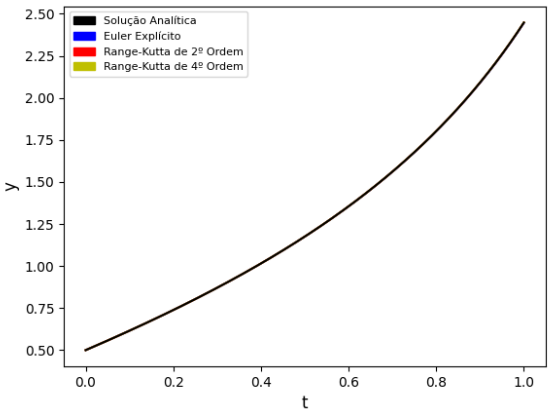
y(t) para dt = 1.0



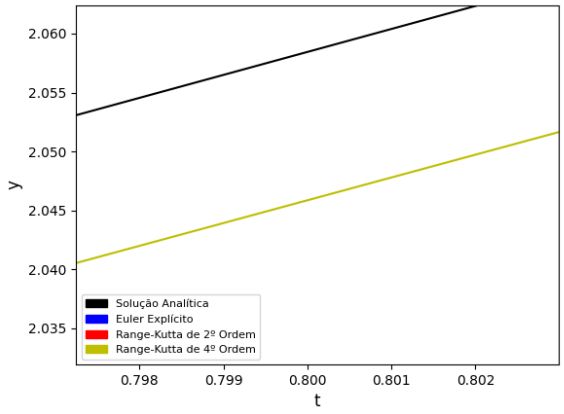
y(t) para dt = 0.1



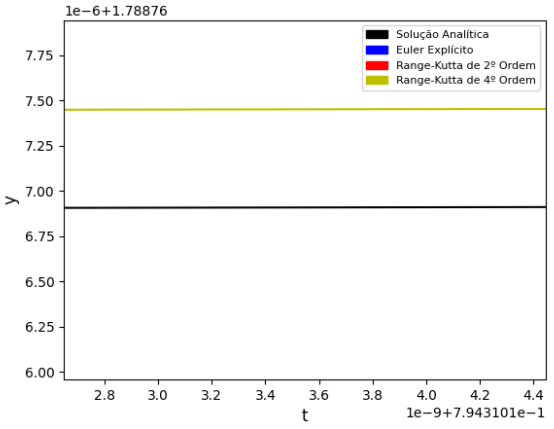
y(t) para dt = 0.01



y(t) para dt = 0.001



ampliação para enxergar o erro da linha vermelha para dt = 0.1



ampliação para enxergar o erro da linha amarela para dt = 0.1

Figura 5: Gráficos obtidos para diferentes valores de d(t)

Em 5, o método de Euler Explícito aproxima o valor da EDO para valores menores que o valor exato, muito provavelmente, pois, como o método de Euler Explícito leva em consideração a derivada em um instante dt, então, como a função apresenta uma derivada crescente ao longo do tempo, o método de Euler Explícito a aproxima para um valor menor que o seu original.

A Equação Diferencial Finita (EDF), logo, apresenta um erro na terceira e sexta casas decimais para o passo de tempo de, respectivamente de dt = 1.0 e dt = 0.1, quando utilizado o método de Range-Kutta de 4º Ordem, apresentando, logo, boa convergência para esse método. O método de Range-Kutta de 2º Ordem apresenta convergência relativamente pior que o método de Range-Kutta de 4º Ordem, como pode ser observado por, em 1, a linha vermelha do gráfico, correspondente a esse gráfico, ter convergido com um erro relativamente grande para o passo de dt = 1.0, enquanto que apresenta convergência satisfatória de 1 unidade para dt = 0.1. O método de Euler Explícito converge satisfatoriamente para passo de tempo, dt = 0.001, o qual é menor que os passos utilizado para os outros dois métodos numéricos utilizados, entretanto, é interessante notar como o erro relacionado à sua convergência dimiui significativamente ao se comparar o passo de tempo dt = 1.0 com o passo de tempo dt = 0.1. O método de Range-Kutta de 4º Ordem, logo, converge para maiores valores de dt que os outros métodos utilizados, representando a melhor aproximação possível da EDO original a partir do método das Diferenças Finitas, sendo seguido pelo método de Range-Kutta de 2º Ordem e, posteriormente, pelo Método de Euler Explícito.

7 Exercício 6

Para a função utilizada em 2, realizou-se um estudo do erro para os métodos de Euler Implícito, Euler Explícito, Range-Kutta de Segunda Ordem e Range-Kutta de Quarta Ordem para o valor correpondente a y(1), ou seja, para t = 1. A metodologia utilizada consistiu na criação de um array dt contendo valores entre 0.001 e 0.1 que sejam múltiplos de 1 de modo que, ao se traçar o gráfico correspondente ao estudo de erros, seja possível realizar um ajuste logarítmico de modo a se observar melhor o erro para o método de Range-Kutta de Quarta Ordem. Após a criação do array dt, realizaram-se os cálculos para os 4 métodos numéricos considerando cada valor de dt, escolhendo-se sempre o último valor, correspondente à aproximação de y(1). Realizou-se, então, o erro a partir do módulo da diferença do valor de y(1)

aproximado e o exato a partir da solução analítica. Traçou-se, então, o gráfico em escala logarítmica de dt em função dos erros, de modo a se obter o gráfico 2.

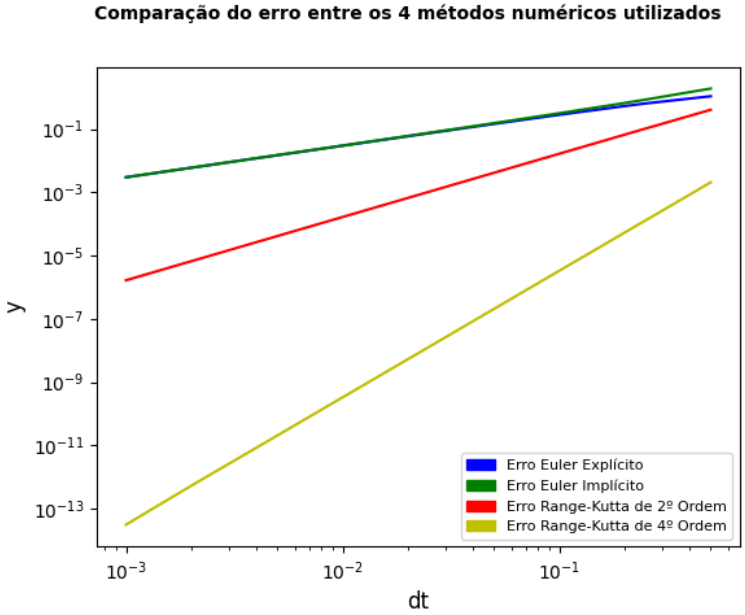


Figura 6: Comparação dos erros para os métodos numéricos utilizados

Percebe-se, a partir de 2, que o Erro relativo ao método Range-Kutta de 4º Ordem é significativamente menor que o erro dos outro métodos numéricos utilizados, variando da ordem de 10^{-13} a 10^{-4}. O método de Range-Kutta de 2º Ordem, por outro lado, apresenta um erro que varia da ordem de 10^{-6} a 10^{-3}, o qual é menor que o erro apresentado pelo método de Euler Explícito, que varia de pouco mais de 10^{-3} a cerca de 10^{-1}, e o método de Euler Implícito, que varia de pouco mais de 10^{-3} a cerca de 10^{0}. Essa diferença observada para os métodos de Euler Explícito e Implícito para valores de dt próximos a 10^{-1} deve-se, muito provavelmente, ao fato da própria função apresentar uma melhor convergência quando se leva em consideração sua derivada atual, como o método de Euler Explícito, que sua próxima derivada, como o método de Euler Implícito, muito provavelmente, pelo sinal da derivada se alterar, o que faz com que valores previstos, como os correspondentes ao Euler Implícito, apresent4m comportamento pior que valores calculados com base no tempo atual, como os correspondentes ao Euler Explícito.

O método de Range-Kutta de Quarta Ordem, logo, apresenta a melhor taxa de convergência entre os 4 métodos utilizados, sendo seguido pelo Range-Kutta de Segunda Ordem, Euler Explícito e Euler Implícito. É interessante notar que o método de Range-Kutta de Quarta Ordem é o mais complexo dos métodos implementados, envolvendo, uma ponderação entre 4 termos correspondentes a valores previstos para a variação de y(t), enquanto que o Range-Kutta de Segunda Ordem leva em consideração apenas dois termos e os Métodos de Euler Explícito e Implícito, apenas um. Percebe-se, então, que os métodos de Range-Kutta de Segunda e Quarta Ordem não são nada mais que melhores aproximações dos métodos de Euler Explícito e Implícito e, por isso, apresentam um erro menor.

8 Exercício 7

Para o Exercício 7, foi requisitado a resolução da equação diferencial de segunda ordem correspondente ao movimento de um pêndulo de modo a obter seu ângulo em radianos em relação a um referencial tanto para sua forma completa quanto aproximada. Ou seja, resolveram-se as seguintes equações:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \cdot sin(\theta)$$

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \cdot \theta$$

Para se resolver essa equação através do método das Diferenças Finitas, tomou-se uma nova variável ω tal que $\omega = \theta'$ de modo a se obter os seguintes sistemas de equações para θ real e θ aproximado:

$$\begin{cases} \omega = \frac{g}{L} \cdot sin(\theta) \\ \omega = \theta' \end{cases}$$

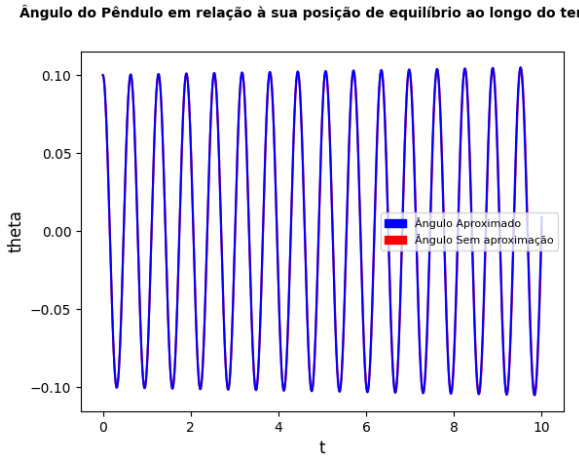
$$\begin{cases} \omega = -\frac{g}{L} \cdot \theta \\ \theta' = \omega \end{cases}$$

Tomando o método das diferenças finitas e os parâmetros $g = 9.81\frac{m}{s^2}$, $\theta(0) = 0.1$ ou $\theta(0) = 0.5$, $\theta' = 0$ e $0s \leq t \leq 10s$, obtém-se que:

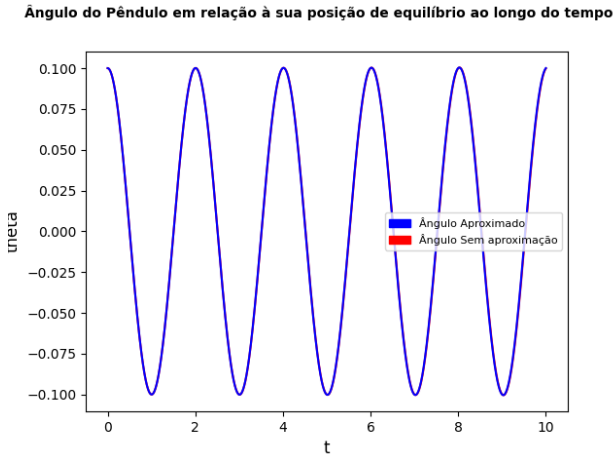
$$\begin{cases} \omega_{k+1} = \omega_k + \Delta t \cdot -\frac{g}{L} \cdot sin(\theta_k) \\ \theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \end{cases}$$

$$\begin{cases} \omega_{k+1} = \omega_k + \Delta t \cdot -\frac{g}{L} \cdot \theta_k \\ \theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \end{cases}$$

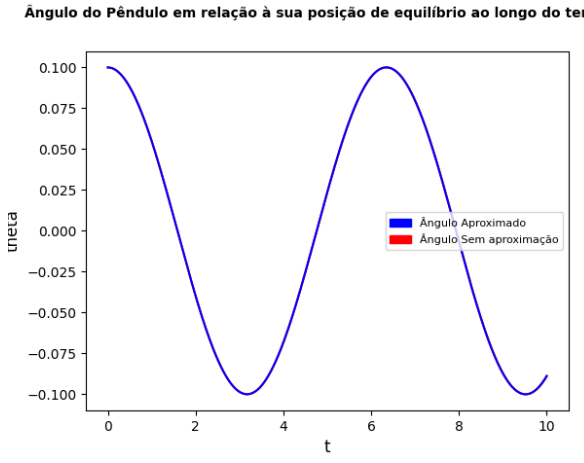
Aplicando essas duas equações no método de Euler Explícito, resolvendo, primeiro, a equação correspondente a ω para, posteriormente, resolver a correspondente a θ, obtiveram-se os gráficos 7 e 8.



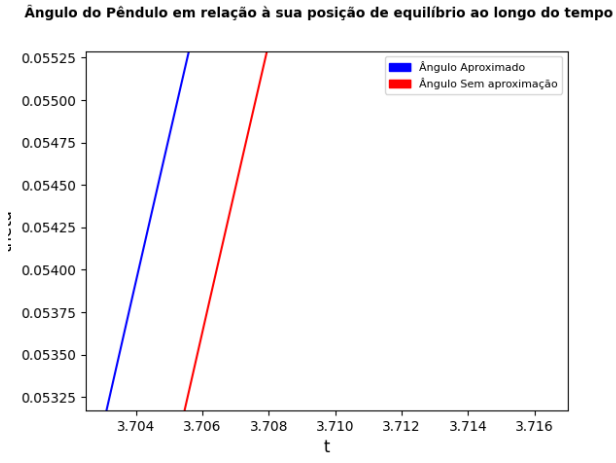
$\theta(t) para L = 0.1m$



$\theta(t) para L = 1.0m$

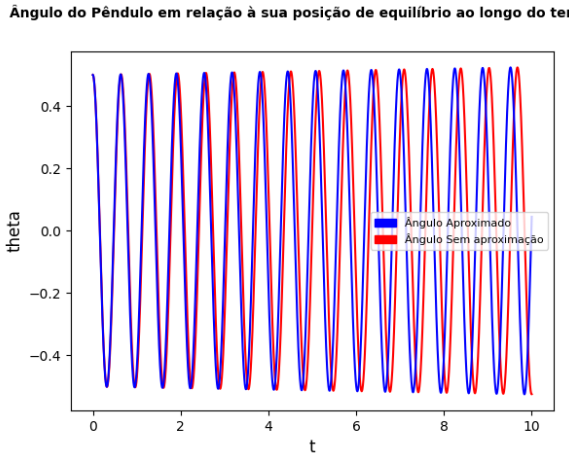


$\theta(t) para L = 10m$

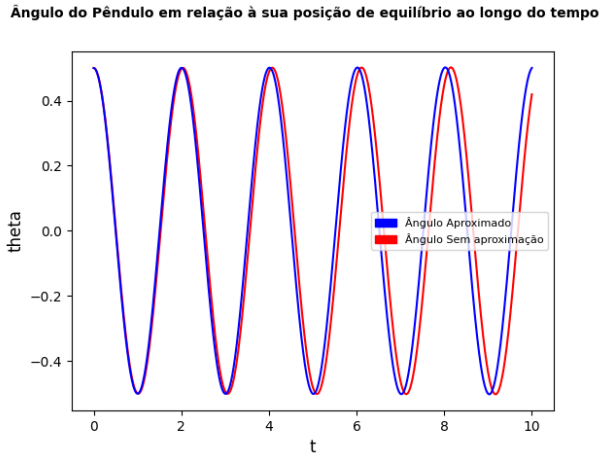


Ampliação no gráfico para dt = 0.1

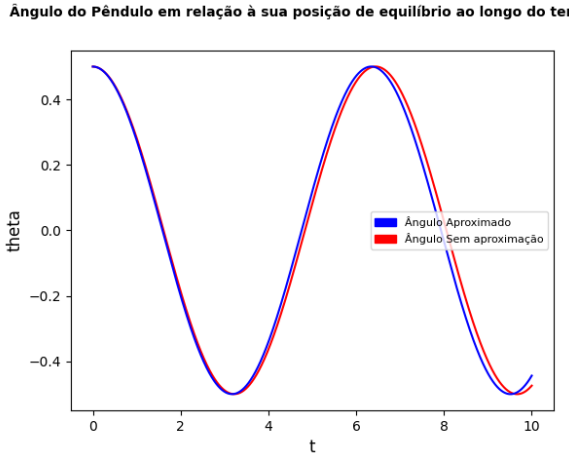
Figura 7: Gráficos obtidos para θ e $\theta_{aproximado}$ quando $\theta(0) = 0.1$



$\theta(t) para L = 0.1m$



$\theta(t) para L = 1.0m$



$\theta(t) para L = 10m$

Figura 8: Gráficos obtidos para θ e $\theta_{aproximado}$ quando $\theta(0) = 0.5$

Comparando-se os gráficos 7 e 8, percebe-se que o valor aproximado para o ângulo do pêndulo apresenta uma precisão de cerca de duas casas decimais em relação ao valor de ângulo exato, sendo, portanto, uma boa aproximação para valores de θ pequenos, tendo em vista que, para $\theta = 0.5rad$, os resultados apresentaram uma discrepância significativa de cerca de 1 cada decimal, a qual pode ser observada nos gráficos traçados. Em relação à variação do valor de L, percebe-se que o pêndulo oscila a uma frequência maior para valores de L pequenos e, a uma frequência menor, para valores de L grande, de modo

a se perceber que a frequência de oscilação do pêndulo está diretamente relacionada a seu valor de L. É interessante notar que o movimento pendular descrito é um Movimento Harmônico Simples de aspecto senoidal.

9 Exercício 8

Para o Exercício 8, foi requisitada a obtenção da corrente em função do tempo (i(t))de um Circuito RLC em paralelo. Por ser um Circuito de Segunda Ordem, o Circuito RLC pode ser representado por uma Equação Diferencial de Segunda Ordem que foi resolvida através dos métodos de Euler Explícito e Range-Kutta de Quarta Ordem.

Dessa forma, tomando a equação diferencial do circuito RLC para os parâmetro C = 0.3F; R = 1.4Ω e L = 1.7H e $0s \leq t \leq 10s$, obtém-se que:

$$\frac{di}{dt} = C\frac{d\epsilon^2}{dt^2} + (\frac{1}{R})\frac{d\epsilon}{dt} + \frac{1}{L}\epsilon$$

Considerando $\epsilon = e^{-0.06t}sin(2t-\pi)$, pode-se calcular, analiticamente, sua primeira e segunda derivadas, de modo a se obter:

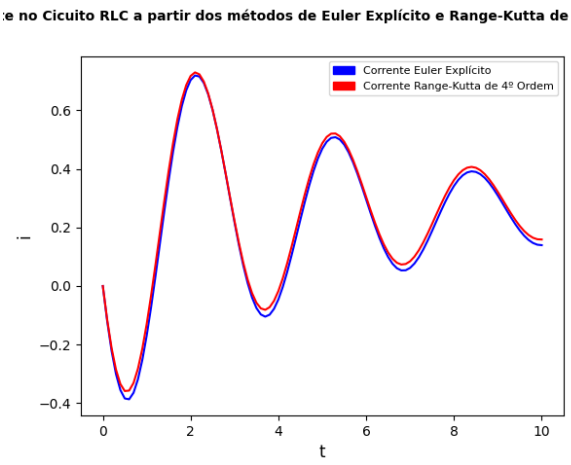
$$\frac{d\epsilon}{dt} = 0.06\pi e^{-0.06\pi t}sin(2t) - 2e^{-0.06}cos(2t)$$

$$\frac{d\epsilon^2}{dt^2} \simeq 3.96447e^{-0.06\pi t}sin(2t) + 0.753982e^{-0.06\pi t}cos(2t)$$

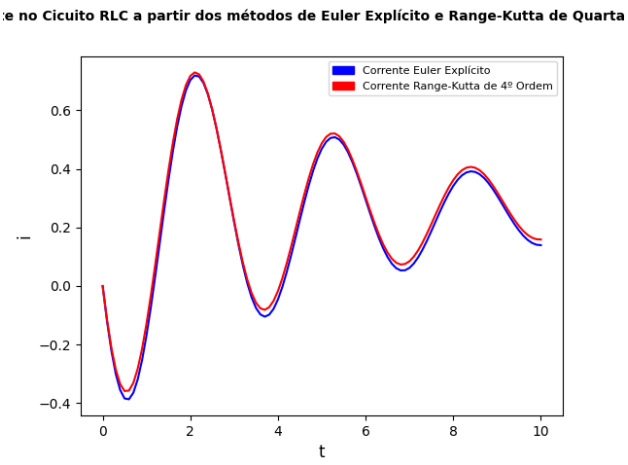
Como ϵ e suas derivadas primeira e segunda podem ser obtidas analiticamente, pode-se aplicar o método das diferenças finitas para se obter a corrente no circuito:

$$i_{k+1} = i_k + \Delta t \cdot (C\frac{d\epsilon^2}{dt^2} + (\frac{1}{R})\frac{d\epsilon}{dt} + \frac{1}{L}\epsilon)$$

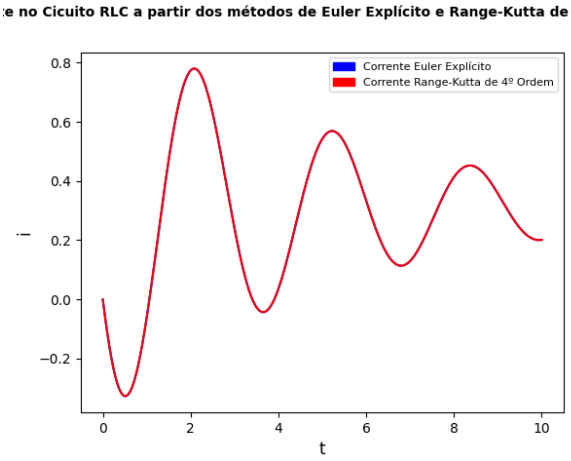
Substituindo as derivadas calculadas de ϵ na equação do método das diferenças finitas para a obtenção da corrente no circuito, obteve-se o gráfico 9.



i(0) para dt = 0.1



i(0) para dt = 0.01



i(t) para dt = 0.001

Figura 9: Gráficos obtidos para diferentes valores de d(t)

A partir do gráfico 9, percebe-se que o valor da corrente é amortecido pela exponencial $e^{-0.06t}$ enquanto que o sinal de saída senoidal é igual ao sinal senoidal de entrada. Percebe-se também que o método de Range-Kutta de Quarta Ordem e o método de Euler Explícito aproximam-se conforme dt torna-se menor, apresentando uma convergência satisfatória a dt = 0.001.

10 Exercício 9

Para o exercício 9, foi requisitado que se analisasse o crescimento de uma população N(t), ao qual foi modelado pela função:

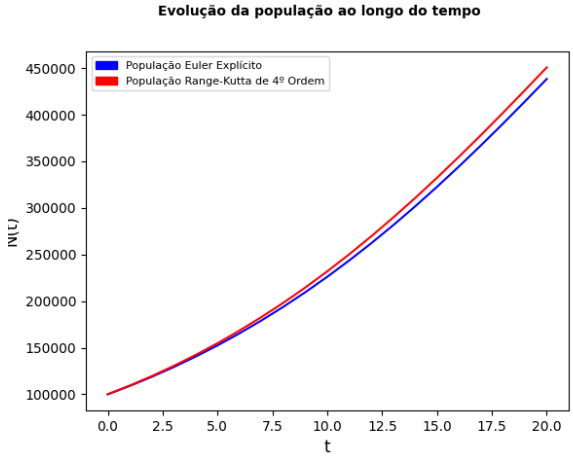
$$\frac{dN}{dt} = aN - bN^2$$

Aplicando-se o método das diferenças finitas, obteve-se que:

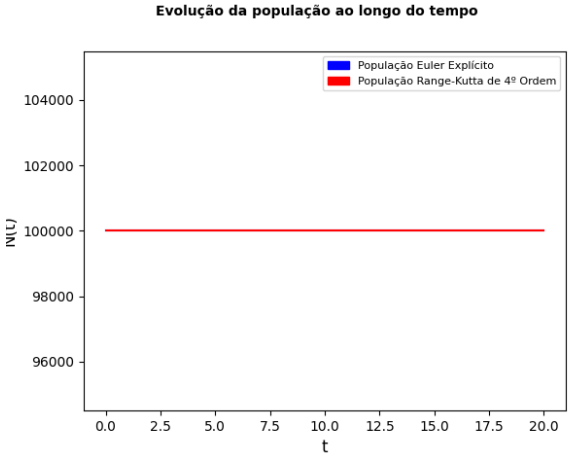
$$N_{k+1} = N_k + \Delta t(aN - bN^2)$$

Tomando os parâmetros N(0) = 100000, a = 0.1 e para t entre 0 e 20 anos, tomando os valores $b = 10^{-7}$, $b = 10^{-6}$, $b = 10^{-5}$ e $b = 10^{-8}$ obtiveram-se os gráficos 10.

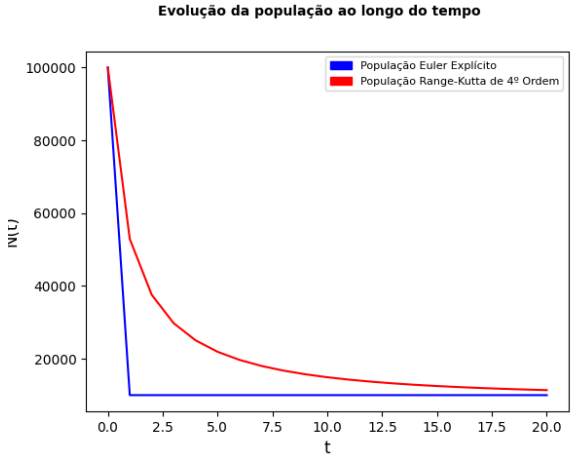
A partir do gráfico 10, percebe-se que, para $b = 10^{-7}$, a população cresce conforme o tempo com uma certa discordância entre os métodos de Range-Kutta de Quarta Ordem e método de Euler Explícito, em que o Euler Explícito converge para um valor levemente menor que o método de Range-Kutta de Quarta Ordem. Para $b = 10^{-6}$, a população não cresce nem decresce conforme o tempo, de modo que a sua derivada possui valor 0, ou seja, é constante. Para $b = 10^{-5}$, a população decresce. O método de Euler Explícito, entretanto, não apresenta um bom ajuste para os valores fornecidos, já que ocorreria um grande decrescimento da população em pouco tempo. Já para o método de Range-Kutta de Quarta Ordem, ocorre um decrescimento mais gradual da curva. Para $b = 10^{-8}$, a população cresce a uma taxa maior que quando $b = 10^{-7}$, de modo a ocorrer uma discordância maior entre o método de Range-Kutta de Quarta Ordem e o de Euler Explícito, principalmente, para valores maiores de t.



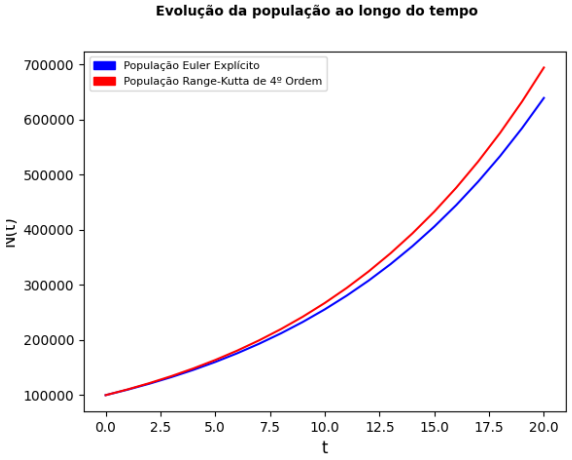
N(t) para b = 10^{-7}



(t) para b = 10^{-6}



N(t) para b = 10^{-5}



(t) para b = 10^{-8}

Figura 10: Gráficos obtidos de N(t) para diferentes valores de b

11 Exercício 10

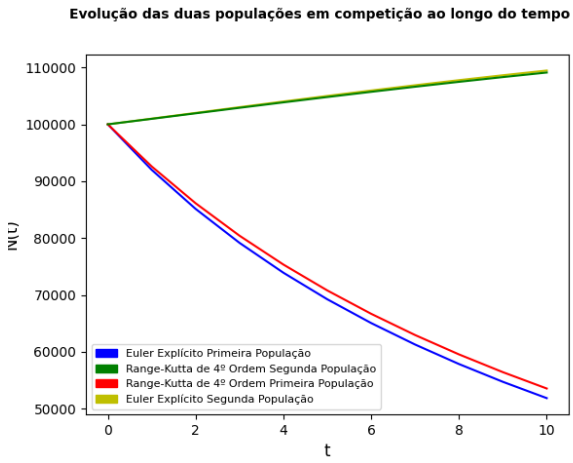
Para a questão 10, foi requisitado a curva de crescimento de duas populações que competem entre si, N1 e N2, as quais foram modeladas pelas funções:

$$\begin{cases} \frac{dN_1}{dt} = N_1(a_1 - b_1N_1 - c_1N_2) \\ \frac{dN_2}{dt} = N_2(a_2 - b_2N_2 - c_2N_1) \end{cases}$$

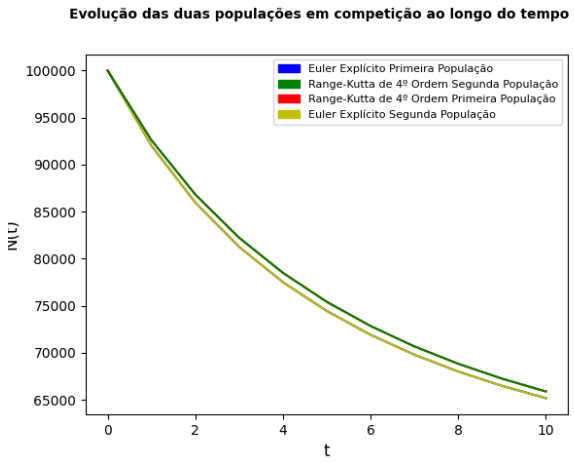
Aplicando o método das diferenças finitas, obtém-se que:

$$\begin{cases} N_1^{k+1} = N_1^{k+1} + N_1^k(a_1 - b_1 N_1^k - c_1 N_2^k) \\ N_2^{k+1} = N_2^{k+1} + N_2^k(a_2 - b_2 N_2^k - c_2 N_1^k) \end{cases}$$

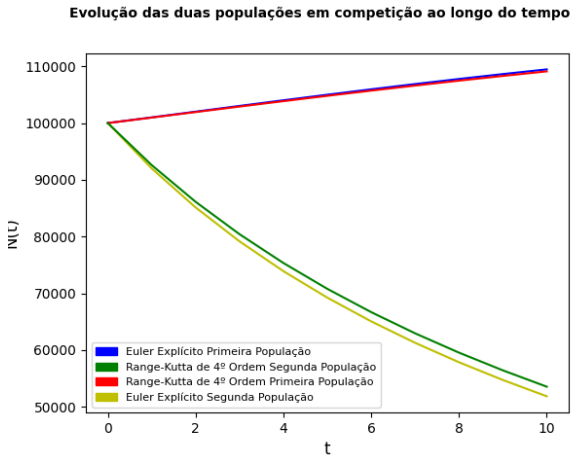
Tomando os parâmetros $N_1(0) = N_2 = 10^5$, $a_1 = 0.1$, $a_2 = 0.1$, $b_1 = 8 \cdot 10^{-7}$, $b_2 = 8 \cdot 10^{-7}$ e para t entre 0 e 10 anos, tomando os valores $c_1 = 10^{-6}$ e $c_2 = 10^{-7}$; $c_1 = 10^{-6}$ e $c_=10^{-6}$;$c_1 = 10^{-7}$ e $c_2 = 10^{-6}$; $c_1 = 10^{-5}$ e $c_2 = 10^{-7}$ obtiveram-se os gráficos 11.



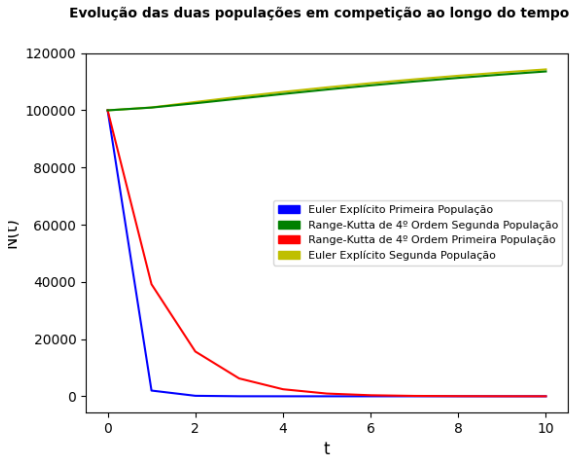
N(t) para $c_1 = 10^{-6}$ e $c_2 = 10^{-7}$



N(t) para $c_1 = 10^{-6}$ e $c_2 = 10^{-6}$



N(t) para $c_1 = 10^{-7}$ e $c_2 = 10^{-6}$



N(t) para $c_1 = 10^{-5}$ e $c_2 = 10^{-7}$

Figura 11: Gráficos obtidos para diferentes valores de c_1 e c_2

Percebe-se que, para $c_1 \geq c_2$, a primeira população decresce enquanto a segunda população cresce. O método de Range-Kutta de Quarta Ordem e de Euler Explícito apresentam boa correspondência para a população 1, que apresenta crescimento suave, enquanto divergem para a população 1, que apresenta decrescimento mais intenso.

Para $c_1 = c_2$, as duas populações decrescem ao mesmo ritmo em uma curva na qual pode-se ver uma pequena discordância entre os métodos de Runge-Kutta de Quarta Ordem e de Euler Explícito, principalmente, para t próximo a 20.0.

Para $c_1 \leq c_2$, a primeira população cresce enquanto a segunda população decresce. O método de Range-Kutta de Quarta Ordem e de Euler Explícito apresentam boa correspondência para a população 2, que apresenta crescimento suave, enquanto divergem para a população 2, que apresenta decrescimento mais intenso.

Para $c_1 < c_2$ tal que c_1 é 100 vezes maior que c_2, obtém-se uma curva de decrescimento ainda mais aguda para a curva correspondente à primeira população, de modo que o método de Euler Explícito apresenta uma maior divergência do método de Runge-Kutta de Quarta Ordem. Para a segunda população, os métodos de Euler Explícito e de Runge-Kutta de Quarta Ordem apresentam boa correspondência, assim como para o primeiro caso realizado.

A Código Exercício 1

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f(t,y):
    return 2.0-2.0*t+4*t**2-4*t**3-4*t**4

#funcao obtida a partir de manipulacoes algebricas para euler
    implicito
def f_implicito(t,y,dt):
    return y + dt*(2.0-2.0*t+4.0*t**2-4.0*t**3-4*t**4)
```

```
#funcao exata desejada
def f_exata(t):
    return 1.0+2.0*t-t**2.0+(4.0/3.0)*t**3-t**4-(4/5)*t**5

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de euler implicito
def euler_implicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = f_implicito(t[k+1],y[k],dt)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
```

```
        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

#funcao para o calculo da funcao exata

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
        y_exata[k] = f_exata(t[k])
    return y_exata

# Parmetros

dt = 0.1
t_inicial = 0.0
t_final = 1.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

#Calculo dos resultados obtidos

y_euler_explicito = euler_explicito(N,t,dt)
y_euler_implicito = euler_implicito(N,t,dt)
y_rk2 = rk2(N,t,dt)
y_rk4 = rk4(N,t,dt)
y_exata = exata(N,t)

#Inicializando o Grfico

graf = plt.figure()
```

```
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao dos resultados obtidos para os 4 mtodos
    numricos utilizados', fontsize = 10, fontweight = 'bold', usetex
    = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blk = mpatches.Patch(color = 'k',label = 'Soluo Analtica')

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito')

blg = mpatches.Patch(color = 'g',label = 'Euler Implcito')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 2 Ordem')

bly = mpatches.Patch(color = 'y',label = 'Range-Kutta de 4 Ordem')
plt.legend(handles = [blk,blp,blg,blr,bly], fontsize = 8)


# Traando os Grficos

plt.plot(t,y_euler_explicito,'b')
plt.plot(t,y_euler_implicito,'g')
plt.plot(t,y_rk2,'r')
plt.plot(t,y_rk4,'y')
plt.plot(t,y_exata,'k')
plt.show()
```

B Código Exercício 2

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f(t,y):
```

```
    return 1.0 + y/t

#funcao obtida a partir de manipulacoes algebricas para euler
    implicito
def f_implicito(t,y,dt):
    return (y + dt)/(1-dt/t)

#funcao exata desejada
def f_exata(t):
    return t*np.log(t) + 2*t

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 2.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de euler implicito
def euler_implicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 2.0
    for k in range(N):
        y[k+1] = f_implicito(t[k+1],y[k],dt)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 2.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
    return y
```

```
#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 2.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
        y_exata[k] = f_exata(t[k])
    return y_exata

dt = 1.0

t_inicial = 1.0
t_final = 2.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

y_euler_explicito = euler_explicito(N,t,dt)
y_euler_implicito = euler_implicito(N,t,dt)
y_rk2 = rk2(N,t,dt)
y_rk4 = rk4(N,t,dt)
y_exata = exata(N,t)

#Inicializando o Grfico
```

```
graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao dos resultados obtidos para os 4 mtodos
    numricos utilizados', fontsize = 10, fontweight = 'bold', usetex
    = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blk = mpatches.Patch(color = 'k',label = 'Soluo Analtica')

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito')

blg = mpatches.Patch(color = 'g',label = 'Euler Implcito')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 2 Ordem')

bly = mpatches.Patch(color = 'y',label = 'Range-Kutta de 4 Ordem')
plt.legend(handles = [blk,blp,blg,blr,bly], fontsize = 8)

# Traando os Grficos

plt.plot(t,y_euler_explicito,'b')
plt.plot(t,y_euler_implicito,'g')
plt.plot(t,y_rk2,'r')
plt.plot(t,y_rk4,'y')
plt.plot(t,y_exata,'k')
plt.show()
```

C Código Exercício 3

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
```

```
def f(t,y):
    return (t**2)*y

#funcao obtida a partir de manipulacoes algebricas para euler
    implicito
def f_implicito(t,y,dt):
    return y/(1-dt*t**2)

#funcao exata desejada
def f_exata(t):
    return np.exp((t**3)/3)

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de euler implicito
def euler_implicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = f_implicito(t[k+1],y[k],dt)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
```

```
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
        y_exata[k] = f_exata(t[k])
    return y_exata

dt = 0.1

t_inicial = 0.0
t_final = 1.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

y_euler_explicito = euler_explicito(N,t,dt)
y_euler_implicito = euler_implicito(N,t,dt)
y_rk2 = rk2(N,t,dt)
y_rk4 = rk4(N,t,dt)
y_exata = exata(N,t)
```

```
#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao dos resultados obtidos para os 4 mtodos
    numricos utilizados', fontsize = 10, fontweight = 'bold', usetex
    = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blk = mpatches.Patch(color = 'k',label = 'Soluo Analtica')

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito')

blg = mpatches.Patch(color = 'g',label = 'Euler Implcito')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 2 Ordem')

bly = mpatches.Patch(color = 'y',label = 'Range-Kutta de 4 Ordem')
plt.legend(handles = [blk,blp,blg,blr,bly], fontsize = 8)


# Traando os Grficos

plt.plot(t,y_euler_explicito,'b')
plt.plot(t,y_euler_implicito,'g')
plt.plot(t,y_rk2,'r')
plt.plot(t,y_rk4,'y')
plt.plot(t,y_exata,'k')
plt.show()
```

D Código Exercício 4

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

```
#funcao desejada
def f(t,y):
    return t*y**2

#funcao exata desejada
def f_exata(t):
    return -(2/(t**2-2))

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])
```

```
        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
        y_exata[k] = f_exata(t[k])
    return y_exata

dt = 0.1

t_inicial = 0.0
t_final = 1.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

y_euler_explicito = euler_explicito(N,t,dt)
y_rk2 = rk2(N,t,dt)
y_rk4 = rk4(N,t,dt)
y_exata = exata(N,t)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao dos resultados obtidos para os 3 mtodos
    numricos utilizados', fontsize = 10, fontweight = 'bold', usetex
    = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
```

```
blk = mpatches.Patch(color = 'k',label = 'Soluo Analtica')

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 2 Ordem')

bly = mpatches.Patch(color = 'y',label = 'Range-Kutta de 4 Ordem')
plt.legend(handles = [blk,blp,blr,bly], fontsize = 8)



# Traando  os  Grficos

plt.plot(t,y_euler_explicito,'b')
plt.plot(t,y_rk2,'r')
plt.plot(t,y_rk4,'y')
plt.plot(t,y_exata,'k')
plt.show()
```

E Código Exercício 5

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f(t,y):
    return 1+0.5*y**2

#funcao exata desejada
def f_exata(t):
    return
        (1.0/np.sqrt(0.5))*np.tan(np.sqrt(0.5)*t+np.arctan(0.5**(1.5)))

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 0.5
```

```
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 0.5
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 0.5
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
```

```
        y_exata[k] = f_exata(t[k])
    return y_exata

dt = 0.1

t_inicial = 0.0
t_final = 1.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

y_euler_explicito = euler_explicito(N,t,dt)
y_rk2 = rk2(N,t,dt)
y_rk4 = rk4(N,t,dt)
y_exata = exata(N,t)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao dos resultados obtidos para os 3 mtodos
    numricos utilizados', fontsize = 10, fontweight = 'bold', usetex
    = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blk = mpatches.Patch(color = 'k',label = 'Soluo Analtica')

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 2 Ordem')

bly = mpatches.Patch(color = 'y',label = 'Range-Kutta de 4 Ordem')
plt.legend(handles = [blk,blp,blr,bly], fontsize = 8)


# Traando os Grficos

plt.plot(t,y_euler_explicito,'b')
```

```
plt.plot(t,y_rk2,'r')
plt.plot(t,y_rk4,'y')
plt.plot(t,y_exata,'k')
plt.show()
```

F Código Exercício 6

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f(t,y):
    return 2.0-2.0*t+4*t**2-4*t**3-4*t**4

#funcao obtida a partir de manipulacoes algebricas para euler
    implicito
def f_implicito(t,y,dt):
    return y + dt*(2.0-2.0*t+4.0*t**2-4.0*t**3-4*t**4)

#funcao exata desejada
def f_exata(t):
    return 1.0+2.0*t-t**2.0+(4.0/3.0)*t**3-t**4-(4/5)*t**5

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k])
    return y

#funcao para o calculo de y pelo metodo de euler implicito
def euler_implicito(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):
        y[k+1] = f_implicito(t[k+1],y[k],dt)
```

```
    return y

#funcao para o calculo de y pelo metodo de range-kutta de segunda
    ordem
def rk2(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt,y[k]+a)

        y[k+1] = y[k] + 0.5*(a+b)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt):
    y = np.zeros(N+1)
    y[0] = 1.0
    for k in range(N):

        a = dt*f(t[k],y[k])

        b = dt*f(t[k]+dt/2,y[k]+a/2)

        c = dt*f(t[k]+dt/2,y[k]+b/2)

        d = dt*f(t[k]+dt,y[k]+c)

        y[k+1] = y[k] + (1/6)*(a+2*b+2*c+d)

    return y

def exata(N,t):
    y_exata = np.zeros(N+1)
    for k in range(N+1):
        y_exata[k] = f_exata(t[k])
    return y_exata
```

```
dt =
    np.array([0.001,0.002,0.0025,0.005,0.01,0.02,0.025,0.05,0.1,0.2,0.25,0.5])
y_erro_euler_explicito = np.zeros(len(dt))
y_erro_euler_implicito = np.zeros(len(dt))
y_erro_rk2 = np.zeros(len(dt))
y_erro_rk4 = np.zeros(len(dt))

t_inicial = 0.0
t_final = 1.0

for i in range(len(dt)):
    N = int((t_final-t_inicial)/dt[i])
    t = np.linspace(t_inicial,t_final,N+1)
    y_exata = exata(N,t)
    y_euler_explicito = euler_explicito(N,t,dt[i])
    y_euler_implicito = euler_implicito(N,t,dt[i])
    y_rk2 = rk2(N,t,dt[i])
    y_rk4 = rk4(N,t,dt[i])

    y_erro_euler_explicito[i] =
        np.abs(y_euler_explicito[len(y_euler_explicito)-1]
        -y_exata[len(t)-1])
    y_erro_euler_implicito[i] =
        np.abs(y_euler_implicito[len(y_euler_implicito)-1]
        -y_exata[len(t)-1])
    y_erro_rk2[i] = np.abs(y_rk2[len(y_rk2)-1] -y_exata[len(t)-1])
    y_erro_rk4[i] = np.abs(y_rk4[len(y_rk4)-1] -y_exata[len(t)-1])

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Comparao  do erro entre os 4 mtodos numricos
    utilizados', fontsize = 10, fontweight = 'bold', usetex = False)
ax.set_ylabel(r'y', fontsize = 12, usetex = False)
ax.set_xlabel(r'dt', fontsize = 12, usetex = False)
```

```
blp = mpatches.Patch(color = 'b',label = 'Erro Euler Explcito')

blg = mpatches.Patch(color = 'g',label = 'Erro Euler Implcito')

blr = mpatches.Patch(color = 'r',label = 'Erro Range-Kutta de 2
    Ordem')

bly = mpatches.Patch(color = 'y',label = 'Erro Range-Kutta de 4
    Ordem')

plt.legend(handles = [blp,blg,blr,bly], fontsize = 8)

plt.plot(dt,y_erro_euler_explicito,'b')
plt.plot(dt,y_erro_euler_implicito,'g')
plt.plot(dt,y_erro_rk2,'r')
plt.plot(dt,y_erro_rk4,'y')
plt.yscale('log')
plt.xscale('log')
plt.show()
```

G Código Exercício 7

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches


def eqsegundaordem(N,t,dt,c1):

    w = np.zeros(N+1)
    theta = np.zeros(N+1)

    theta[0] = 0.5
    w[0] = 0.0
    theta_aprox = np.copy(theta)
    w_aprox = np.copy(w)
```

```
    for k in range(N):

        theta[k+1] = theta[k] + dt*w[k]
        w[k+1] = w[k] + dt*(c1*np.sin(theta[k]))
        theta_aprox[k+1] = theta_aprox[k] + dt*w_aprox[k]
        w_aprox[k+1] = w_aprox[k] + dt*(c1*(theta_aprox[k]))

    return theta,theta_aprox

t_inicial = 0.0
t_final = 10.0
dt = 0.0001
L = 0.1
g = 9.81
c1 = -g/L
theta_inicial = 0.5
theta_linha_inicial = 0.0
N = int((t_final-t_inicial)/dt)
t = np.arange(t_inicial,t_final+dt,dt)

theta,theta_aprox = eqsegundaordem(N,t,dt,c1)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

#Parmetros do Grfico

graf.suptitle('ngulo do Pndulo em relao sua posio de
    equilbrio ao longo do tempo', fontsize = 10, fontweight =
    'bold', usetex = False)
ax.set_ylabel(r'theta', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)

blp = mpatches.Patch(color = 'b',label = 'ngulo Aproximado')

blr = mpatches.Patch(color = 'r',label = 'ngulo Sem aproximao')
```

```
plt.legend(handles = [blp,blr], fontsize = 8)

plt.plot(t,theta,'r')
plt.plot(t,theta_aprox,'b')
plt.show()
```

H Código Exercício 8

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

def f(t):
    return np.exp(-0.06*np.pi*t)*np.sin(2.0*t-np.pi)

def df(t):
    return
        0.06*np.pi*np.exp(-0.06*np.pi*t)*np.sin(2*t)-2*np.exp(-0.06*np.pi*t)*np.cos(2*t)

def df2(t):
    return 3.96447*np.sin(2*t) + 0.753982*np.cos(2*t)


def eqsegundaordem(N,t,dt,R,L,C):

    i = np.zeros(N+1)
    E = np.zeros(N+1)
    dE = np.zeros(N+1)
    dE2 = np.zeros(N+1)

    i[0] = 0.0

    for k in range(N):

        i[k+1] = i[k] + dt*((C*df2(t[k])) +
            (1/R)*(df(t[k]))+(1/L)*f(t[k]))

    return i
```

```
def rksegundaordem(N,t,dt,R,L,C):
    i = np.zeros(N+1)

    i[0] = 0.0
    for k in range(N):

        a = dt*((C*df2(t[k])) + (1/R)*(df(t[k]))+(1/L)*f(t[k]))

        b = dt*((C*df2(t[k]+dt/2)) +
            (1/R)*(df(t[k]+dt/2))+(1/L)*f(t[k]+dt/2))

        c = dt*((C*df2(t[k]+dt/2)) +
            (1/R)*(df(t[k]))+(1/L)*f(t[k]+dt/2))

        d = dt*((C*df2(t[k])) + (1/R)*(df(t[k]))+(1/L)*f(t[k]))

        i[k+1] = i[k] + (1/6)*(a+2*b+2*c+d)

    return i

t_inicial = 0.0
t_final = 10.0
dt = 0.001
C = 0.3
R = 1.4
L = 1.7
io = 0.0
N = int((t_final-t_inicial)/dt)
t = np.arange(t_inicial,t_final+dt,dt)

i = eqsegundaordem(N,t,dt,R,L,C)
i_rk4 = rksegundaordem(N,t,dt,R,L,C)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico
```

```
graf.suptitle('Corrente no Cicuito RLC a partir dos mtodos de Euler
    Explcito e Range-Kutta de Quarta Ordem', fontsize = 10,
    fontweight = 'bold', usetex = False)
ax.set_ylabel(r'i', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blp = mpatches.Patch(color = 'b',label = 'Corrente Euler Explcito')

blr = mpatches.Patch(color = 'r',label = 'Corrente Range-Kutta de 4
    Ordem')

plt.legend(handles = [blp,blr], fontsize = 8)


plt.plot(t,i,'b')
plt.plot(t,i_rk4,'r')
plt.show()
```

I Código Exercício 9

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f(t,y,a,b):
    return a*y-b*y**2

#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt,a,b):
    y = np.zeros(N+1)
    y[0] = 100000.0
    for k in range(N):
        y[k+1] = y[k] + dt*f(t[k], y[k],a,b)
    return y

#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
```

```
def rk4(N,t,dt,a,b):
    y = np.zeros(N+1)
    y[0] = 100000.0
    for k in range(N):

        k1 = dt*f(t[k],y[k],a,b)

        k2 = dt*f(t[k]+dt/2,y[k]+k1/2,a,b)

        k3 = dt*f(t[k]+dt/2,y[k]+k2/2,a,b)

        k4 = dt*f(t[k]+dt,y[k]+k3,a,b)

        y[k+1] = y[k] + (1/6)*(k1+2*k2+2*k3+k4)

    return y

dt = 1.0

t_inicial = 0.0
t_final = 20.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)
a = 0.1
b = 10.0**(-7)


y_euler_explicito = euler_explicito(N,t,dt,a,b)
y_rk4 = rk4(N,t,dt,a,b)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Evoluo da populao ao longo do tempo', fontsize =
    10, fontweight = 'bold', usetex = False)
ax.set_ylabel(r'N(t)', fontsize = 12, usetex = False)
```

```
ax.set_xlabel(r't', fontsize = 12, usetex = False)
blp = mpatches.Patch(color = 'b',label = 'Populao Euler Explcito')

blr = mpatches.Patch(color = 'r',label = 'Populao Range-Kutta de 4
    Ordem')

plt.legend(handles = [blp,blr], fontsize = 8)

plt.plot(t,y_euler_explicito,'b')
plt.plot(t,y_rk4,'r')
plt.show()
```

J Código Exercício 10

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

#funcao desejada
def f1(t,y,a1,b1,c1,a2,b2,c2,y2):
    return y*(a1-b1*y-c1*y2)

def f2(t,y,a1,b1,c1,a2,b2,c2,y2):
    return y*(a2-b2*y-c2*y2)


#funcao para o calculo de y pelo metodo de euler explicito
def euler_explicito(N,t,dt,a1,b1,c1,a2,b2,c2):
    y1 = np.zeros(N+1)
    y2 = np.zeros(N+1)
    y1[0] = 100000.0
    y2[0] = 100000.0
    for k in range(N):
        y1[k+1] = y1[k] + dt*f1(t[k],y1[k],a1,b1,c1,a2,b2,c2,y2[k])
        y2[k+1] = y2[k] + dt*f2(t[k],y2[k],a1,b1,c1,a2,b2,c2,y1[k])

    return y1,y2
```

```
#funcao para o calculo de y pelo metodo de range-kutta de quarta
    ordem
def rk4(N,t,dt,a1,b1,c1,a2,b2,c2):
    y1 = np.zeros(N+1)
    y2 = np.zeros(N+1)
    y1[0] = 100000.0
    y2[0] = 100000.0

    for k in range(N):

        k11 = dt*f1(t[k],y1[k],a1,b1,c1,a2,b2,c2,y2[k])

        k12 = dt*f1(t[k]+dt/2,y1[k]+k11/2,a1,b1,c1,a2,b2,c2,y2[k])

        k13 = dt*f1(t[k]+dt/2,y1[k]+k12/2,a1,b1,c1,a2,b2,c2,y2[k])

        k14 = dt*f1(t[k]+dt,y1[k]+k13,a1,b1,c1,a2,b2,c2,y2[k])


        k21 = dt*f2(t[k],y2[k],a1,b1,c1,a2,b2,c2,y1[k])

        k22 = dt*f2(t[k]+dt/2,y2[k]+k21/2,a1,b1,c1,a2,b2,c2,y1[k])

        k23 = dt*f2(t[k]+dt/2,y2[k]+k22/2,a1,b1,c1,a2,b2,c2,y1[k])

        k24 = dt*f2(t[k]+dt,y2[k]+k23,a1,b1,c1,a2,b2,c2,y1[k])

        y1[k+1] = y1[k] + (1/6)*(k11+2*k12+2*k13+k14)
        y2[k+1] = y2[k] + (1/6)*(k21+2*k22+2*k23+k24)

    return y1,y2

dt = 1.0

t_inicial = 0.0
t_final = 10.0
N = int((t_final-t_inicial)/dt)
t = np.linspace(t_inicial,t_final,N+1)

a1 = 0.1
```

```
b1 = 8*10.0**(-7)
c1 = 10.0**(-6)

a2 = 0.1
b2 = 8*10.0**(-7)
c2 = 10.0**(-7)



y_euler_explicito1,y_euler_explicito2 =
    euler_explicito(N,t,dt,a1,b1,c1,a2,b2,c2)
y_rk41,y_rk42 = rk4(N,t,dt,a1,b1,c1,a2,b2,c2)

#Inicializando o Grfico

graf = plt.figure()
ax = graf.add_subplot()

# Parmetros do Grfico

graf.suptitle(' Evoluo das duas populaes em competio ao longo
    do tempo', fontsize = 10, fontweight = 'bold', usetex = False)
ax.set_ylabel(r'N(t)', fontsize = 12, usetex = False)
ax.set_xlabel(r't', fontsize = 12, usetex = False)

blp = mpatches.Patch(color = 'b',label = 'Euler Explcito Primeira
    Populao')

bly = mpatches.Patch(color = 'y',label = 'Euler Explcito Segunda
    Populao')

blr = mpatches.Patch(color = 'r',label = 'Range-Kutta de 4 Ordem
    Primeira Populao')

blg = mpatches.Patch(color = 'g',label = 'Range-Kutta de 4 Ordem
    Segunda Populao')
plt.legend(handles = [blp,blg,blr,bly], fontsize = 8)
plt.plot(t,y_euler_explicito1,'b')
plt.plot(t,y_euler_explicito2,'y')
plt.plot(t,y_rk41,'r')
```

```
plt.plot(t,y_rk42,'g')
plt.show()
```

Referências

[Ros22] Adriano Possebon Rosa. *EDOs*. https://aprender3.unb.br/pluginfile.php/2182433/mod$_r$*esource*/*content*/6/*EDOs.pdf*. Fevereiro de 2022.