Tarefa#01 — Tabelas de Convergência para a Verificação da Correta Implementação de um Método Numérico

Igor Pontes Tresolavy – NUSP 12553646 – tresolavy@usp.br
e
Thiago Antici de Souza Rodrigues – NUSP 12551411 – thiago_souza@usp.br
Escola Politécnica – Universidade de São Paulo

Neste relatório, verifica-se a ordem de convergência do Método de Euler através do Método de Soluções Manufaturadas para Problemas de Cauchy 1D e 2D, e através de artifícios matemáticos para um Problema de Valor Inicial bidimensional sem solução explícita conhecida.

I. Introdução

Ao se implementar (em código computacional) métodos numéricos para a obtenção de soluções aproximadas de equações diferenciais, é comum que haja a necessidade de verificar a corretude do método implementado para determinados intervalos de valores das variáveis independentes. Uma das maneiras de realizar tal tarefa consiste na comparação do comportamento do erro de aproximação entre soluções exatas conhecidas e soluções estimadas pelo código desenvolvido, tendo em vista o comportamento esperado previsto em teoria de métodos numéricos. Tal método de verificação é denominado Método de Solução Manufaturada [3].

Este relatório apresenta meios de verificação de implementações de métodos numéricos e exemplifica-as no âmbito da verificação de códigos computacionais para a aproximação de soluções de equações diferenciais unidimensionais e bidimensionais através do método de passo único de Euler. Na seção III, o conceito e significado das colunas de uma Tabela de Convergência são explicitados. Em IV, três aplicações de análise de convergência utilizando Tabelas de Convergência são apresentadas: duas pelo Método de Solução Manufaturada (uma para um Problema de Cauchy escalar e outra para um Problema de Cauchy bidimensional) e uma utilizando aproximações por meio de artifícios matemáticos (apresentados em IV B) para a estimativa do erro cometido quando não se conhece a solução exata das Equações Diferenciais Ordinárias (EDOs). A seção V conclui o trabalho.

II. Metodologia Numérica

Para a aproximação numérica dos Problemas de Cauchy aqui propostos, utilizou-se o Método de Euler Explícito. Esse método é derivado do Teorema de Taylor — mais especificamente, o Método de Euler constituí um método de aproximação por Taylor de ordem 1. Portanto, dado um Problema de Cauchy

$$\begin{cases} \dot{y}(t) &= f(t, y(t)) \\ y(t_0) &= y_0 \end{cases}$$

definido em um intervalo $[t_0, t_n]$, pode-se aproximar a solução do problema em cada um dos n pontos de malha do intervalo — separados por uma distância $h = \frac{t_n - t_0}{n}$ — através do polinômio

$$y_{k+1} = y_k + h \cdot f(t_k, y_k), k \in [0, n],$$

que corresponde à aproximação por polinômio de Taylor de ordem 1 do valor de y_{k+1} em torno de (t_k, y_k).

III. Tabelas de convergência

O Erro de Discretização Global e_k no instante $t_k \in [t_o, T]$ corresponde ao erro cometido (diferença entre o valor real da variável de estado e o valor calculado) em um passo de integração específico do método numérico usado na resolução de um Problema de Cauchy em um intervalo $[t_0, T]$ de valores da variável independente. Portanto, para métodos numéricos de passo único, temos que

$$e(t_k, h) \doteq e_k \doteq y(t_k) - y_k$$

com $y(t_k)$ correspondendo à solução exata no k-ésimo passo de integração, y_k, à aproximação numérica, n, à quantidade de passos no intervalo $[t_0, T]$ e $h = \frac{(T - t_0)}{n}$, à distância entre cada passo [3].

Ademais, se a função $f(t, y)$ associada ao Problema de Cauchy

$$\begin{cases} \dot{y}(t) &= f(t, y(t)) \\ y(t_0) &= y_0 \end{cases} \tag{1}$$

for de classe C^{N+2} em relação a y e limitada na faixa $\{(t, y) | a = t_0 \leq t \leq t_f = b, y \in \mathbb{R}^n, a, b < \infty\}$, e, dado que a solução numérica $\eta(t, y)$ — obtida pelo Método de Euler Explícito, de passo único — possui ordem $1 \leq N$, tem-se que a expansão da solução numérica admite a expansão assintótica em potências de h válida para todo $t \in [a, b]$ e para todo $h = \frac{T - t_0}{n}$

$$\eta(t, h) = y(t) + \sum_{j=1}^{N} h^j e_j(t) + h_{N+1} E_{N+1}(t, h),$$

com $e_j(t_0) = 0, \forall j \in \mathbb{N}[1, N]$. $E_{N+1}(t, h)$ é limitado para um t fixado e para todo h [3].

Segue que o Erro de Discretização Global $\eta(t,h) - y(t)$ associado à expansão assintótica é dado por

$$-e(t,h) = \sum_{j=1}^{N} h^j e_j(t) + h_{N+1} E_{N+1}(t,h). \tag{2}$$

De (2), conclui-se que o Erro de Discretização Global do Método de Euler Explícito é dado por um polinômio em h de ordem 1 ($O(h)$). Consequentemente, para valores de h suficientemente pequenos, o erro $e(t,h)$ deveria, na maioria dos casos, decair linearmente a medida que o passo de integração h é diminuído (*i.e.* se h for diminuído pela metade, o erro provavelmente também o será).

Partindo desse fato, define-se as colunas de uma Tabela de Convergência como

n	h	$\|e(t,h)\|$	$\log_r(\frac{\|e(t,r\cdot h)\|}{\|e(t,h)\|})$

Tabela I: Colunas de uma Tabela de Convergência

sendo r o decréscimo realizado para cada caso (cada linha da tabela) de estudo de comportamento do método.

Levando em consideração o que foi exposto em (2), espera-se que

$$\begin{cases} \lim_{n\to\infty} \|e(t,h)\| &= 0 \\ \lim_{n\to\infty} \frac{\|e(t,r\cdot h)\|}{\|e(t,h)\|} &= r \\ \lim_{n\to\infty} \log_r(\frac{\|e(t,r\cdot h)\|}{\|e(t,h)\|}) &= 1 \end{cases} \tag{3}$$

Utiliza-se os resultados de (3) para determinar a correta implementação do Método de Euler Explícito nas seções IV A 1 e IV A 2.

A. Erro de Discretização Global quando não se conhece a solução exata

É possível, ainda, depurar um método numérico implementado em código computacional quando não se conhece (ou não existe) a solução explícita de um Problema de Cauchy, necessária para o cálculo do erro apresentado em (2).

Para dois casos consecutivos de diminuição por um fator r do passo h de integração do Método de Euler Explícito, temos, de (2), que

$$\eta(t,h) - \eta\left(t, \frac{h}{r}\right) \approx e_1(t)\left[h - \left(\frac{h}{r}\right)\right]$$

$$= e_1(t)\left(\frac{h}{r}\right),$$

$$\therefore e_1(t)\left(\frac{h}{r}\right) \approx \eta(t,h) - \eta(t,\frac{h}{r}).$$

Pode-se, então, redefinir o erro $\|e(t,h)\|$ e a última coluna da Tabela de Convergência como, respectivamente:

$$||e(t,h)|| = \left|\left|\eta(t,h) - \eta\left(t, \frac{h}{r}\right)\right|\right|$$

$$\log_r(\left|\left|\frac{\eta(t, r\cdot h) - \eta(t,h)}{\eta(t,h) - \eta\left(t, \frac{h}{r}\right)}\right|\right|) \approx r^{\bar{p}}$$

Nota-se que é garantido que $\bar{p}$ aproxima-se de 1 para o Método de Euler Explícito caso as condições explicitadas para (1) sejam satisfeitas [3].

Desse modo, é possível analisar a Tabela de Convergência de maneira semelhante à da análise pelo Método de Solução Manufaturada. Utiliza-se o que foi aqui exposto na seção IV B.

IV. Exemplificando Tabelas de Convergência

Como demonstração do que foi exposto na seção III, escolheu-se duas EDOs e um sistema de EDOs; as duas primeiras as quais possuem solução exata — o que permite a verificação do código pelo Método de Solução Manufaturada — e o sistema de EDOs com soluções exatas desconhecidas.

Os programas referentes à esta seção foram desenvolvidos na linguagem de programação *Python* e utilizando-se as bibliotecas *numpy* [1] (para cálculos matemáticos) e *matplotlib* [2] (para o traçado de gráficos). Disponibilizou-se os programas **2_questao_computacional_itens_1_e_2.py** (referentes às seções IV A 1 e IV A 2) e **2_questao_computacional_item_3.py** (referente à seção IV B) junto à este relatório. Pode-se usá-los para obter-se as tabelas e figuras apresentadas nas seções a seguir (ver apêndice A).

A. Método de Solução Manufaturada

Como explicitado na seção III, a verificação de código por meio do Método de Solução Manufaturada consiste na comparação de uma solução exata conhecida de Problema de Valor Inicial com a aproximação numérica correspondente; doravante denominadas, respectivamente, $y_e(t)$ e $\eta(t,h)$.

As Tabelas de Convergência das seções IV A 1 e IV A 2 permitem a observação do comportamento do erro ao longo de vinte casos (*i.e.* vinte valores de n), com $n \in \mathbb{N}[4, 4\cdot 2^{19}]$ — ou seja, com fator multiplicativo $r = 2$. Utilizou-se o Método de Euler Explícito para a aproximação das soluções no intervalo $[0, \frac{3\cdot\pi}{4}]$, calculando-se o Erro de Discretização Global sempre no ponto final do intervalo: $t_n = \frac{3\cdot\pi}{4}$. Ou seja, aproximou-se a solução numérica de $t_0 = 0$ até $t_n = \frac{3\cdot\pi}{4}$ e calculou-se o Erro de Discretização Global neste ponto.

1. EDO 1D

Escolheu-se o seguinte Problema de Cauchy escalar para a verificação pelo Método de Solução Manufaturada:

$$\begin{cases} \dot{y}(t) &= f(t,y) &= -5y - e^{-5t}sin(t) \\ y(t_0) &= y_0 &= 1, \end{cases}$$

com solução exata

$$y_e(t) = e^{-5t}cos(t)$$

e aproximação numérica dada pelo Método de Euler Explícito

$$\eta(t_{k+1}, h) \ = y_{k+1} \ = y_k + h \cdot f(t_k, y_k), t_k \in [0, \frac{3 \cdot \pi}{4}].$$

A Tabela de Convergência apresenta-se na tabela II.

n	h	$\|e(t_n,h)\|$	$\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$
4	$5.890E-01$	$1.444E+01$	
8	$2.945E-01$	$2.542E-03$	$1.247E+01$
16	$1.473E-01$	$3.412E-06$	$9.541E+00$
32	$7.363E-02$	$6.744E-07$	$2.339E+00$
64	$3.682E-02$	$6.755E-07$	$-2.318E-03$
128	$1.841E-02$	$7.504E-07$	$-1.517E-01$
256	$9.204E-03$	$5.024E-07$	$5.788E-01$
512	$4.602E-03$	$2.863E-07$	$8.116E-01$
1024	$2.301E-03$	$1.523E-07$	$9.103E-01$
2048	$1.150E-03$	$7.851E-08$	$9.562E-01$
4096	$5.752E-04$	$3.985E-08$	$9.783E-01$
8192	$2.876E-04$	$2.007E-08$	$9.892E-01$
16384	$1.438E-04$	$1.007E-08$	$9.946E-01$
32768	$7.191E-05$	$5.046E-09$	$9.973E-01$
65536	$3.595E-05$	$2.525E-09$	$9.987E-01$
131072	$1.798E-05$	$1.263E-09$	$9.993E-01$
262144	$8.988E-06$	$6.318E-10$	$9.997E-01$
524288	$4.494E-06$	$3.159E-10$	$9.998E-01$
1048576	$2.247E-06$	$1.580E-10$	$9.999E-01$
2097152	$1.124E-06$	$7.899E-11$	$1.000E+00$

Tabela II: Estimativa da ordem de convergência do Método de Euler no instante $t_n = \frac{3\cdot\pi}{4}$, para o Problema de Cauchy unidimensional com solução conhecida.

Verifica-se pela tabela II que o erro $||e(t_n,h)||$ converge para 0 e o logaritmo $\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$, para 1, insinuando a corretude do método implementado. O primeiro gráfico da figura 1 apresenta a solução exata da EDO em comparação com soluções aproximadas para alguns valores de n. Observa-se que, conforme n aumenta em valor, o gráfico da aproximação numérica se sobrepõe à solução exata.

2. Sistema 2D de EDOs

Escolheu-se o seguinte Problema de Cauchy de ordem 2 para a verificação pelo Método de Solução Manufaturada:

$$\begin{cases} \ddot{y}(t) &= f(t,y,\dot{y}) &= -5\dot{y} - 4y \\ y(t_0) &= y_0 &= 0 \\ \dot{y}(t_0) &= \dot{y}_0 &= 3 \end{cases}$$

com solução exata

$$y_e(t) = e^{-t} - e^{-4t}$$

Para a utilização do Método de Euler Explícito, então, transformou-se a EDO de ordem 2 em dois sistemas de EDOs de ordem 1

$$\begin{cases} y_1(t) &= y(t) &\iff \dot{y_1}(t) &= y_2(t) \\ y_2(t) &= \dot{y}(t) &\iff \dot{y_2}(t) &= -4y_1(t) - 5y_2(t). \end{cases}$$

As funções de discretização do Método de Euler Explícito são dadas, então, por

$$\therefore \begin{cases} f_1(t, y_1(t), y_2(t)) &= y_2(t) \\ f_2(t, y_1(t), y_2(t)) &= -4y_1(t) - 5y_2(t). \end{cases}$$

Em notação matricial:

$$\begin{cases} \begin{pmatrix} \dot{y_1}(t) \\ \dot{y_2}(t) \end{pmatrix} = \begin{pmatrix} f_1(t,y_1,y_2) \\ f_2(t,y_1,y_2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -4 & -5 \end{pmatrix} \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}. \end{cases}$$

A aproximação numérica pelo Método de Euler Explícito é, portanto:

$$\begin{cases} \begin{pmatrix} y_{1,k+1} \\ y_{2,k+1} \end{pmatrix} = \begin{pmatrix} y_{1,k} \\ y_{2,k} \end{pmatrix} + \begin{pmatrix} h & 0 \\ 0 & h \end{pmatrix} \begin{pmatrix} f_1(t,y_{1,k},y_{2,k}) \\ f_2(t,y_{1,k},y_{2,k}) \end{pmatrix} \end{cases}$$

$$\begin{cases} \begin{pmatrix} \eta_1(t_{k+1},h) \\ \eta_2(t_{k+1},h) \end{pmatrix} = \begin{pmatrix} y_{1,k+1} \\ y_{2,k+1} \end{pmatrix}, t_k \in [0, \frac{3\cdot\pi}{4}] \end{cases}$$

Levando em conta a norma do máximo para o erro ($||e(t_n,h)|| = max\{|e_i(t_n,h)|\}, i = 1,2$), a tabela III corresponde à Tabela de Convergência para o sistema de EDOs.

n	h	$\|e(t_n,h)\|$	$\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$
4	$5.890E-01$	$1.360E+01$	
8	$2.945E-01$	$3.334E-02$	$8.672E+00$
16	$1.473E-01$	$1.653E-02$	$1.012E+00$
32	$7.363E-02$	$8.200E-03$	$1.011E+00$
64	$3.682E-02$	$4.079E-03$	$1.007E+00$
128	$1.841E-02$	$2.034E-03$	$1.004E+00$
256	$9.204E-03$	$1.015E-03$	$1.002E+00$
512	$4.602E-03$	$5.073E-04$	$1.001E+00$
1024	$2.301E-03$	$2.535E-04$	$1.001E+00$
2048	$1.150E-03$	$1.267E-04$	$1.000E+00$
4096	$5.752E-04$	$6.336E-05$	$1.000E+00$
8192	$2.876E-04$	$3.168E-05$	$1.000E+00$
16384	$1.438E-04$	$1.584E-05$	$1.000E+00$
32768	$7.191E-05$	$7.920E-06$	$1.000E+00$
65536	$3.595E-05$	$3.960E-06$	$1.000E+00$
131072	$1.798E-05$	$1.980E-06$	$1.000E+00$
262144	$8.988E-06$	$9.900E-07$	$1.000E+00$
524288	$4.494E-06$	$4.950E-07$	$1.000E+00$
1048576	$2.247E-06$	$2.475E-07$	$1.000E+00$
2097152	$1.124E-06$	$1.237E-07$	$1.000E+00$

Tabela III: Estimativa da ordem de convergência do Método de Euler no instante $t_n = \frac{3\cdot\pi}{4}$, para o Problema de Cauchy bidimensional com solução conhecida.

Assim como no caso escalar, nota-se que o erro do Problema de Cauchy bidimensional também se aproxima de 0 e o logaritmo $\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$, para 1, quanto menor o passo de integração.

Os dois últimos gráficos da figura 1 apresentam, respectivamente, comparações entre a solução exata e as aproximações numéricas para a variável de estado y_1, e comparações entre a derivada da solução exata e as aproximações numéricas para a variável de estado y_2 para diferentes quantidades de passos no intervalos. Em ambos os casos, nota-se que não é necessário muitas subdivisões do intervalo para obter-se soluções consideravelmente próximas da soluções exatas das EDOs.

B. Sistema 2D de EDOs sem solução exata conhecida

Caso não se saiba (ou não exista) solução exata para uma determinada EDO (ou sistema de EDOs), ainda é possível estimar a corretude da implementação de um método numérico de passo único como explicitado na seção III A.

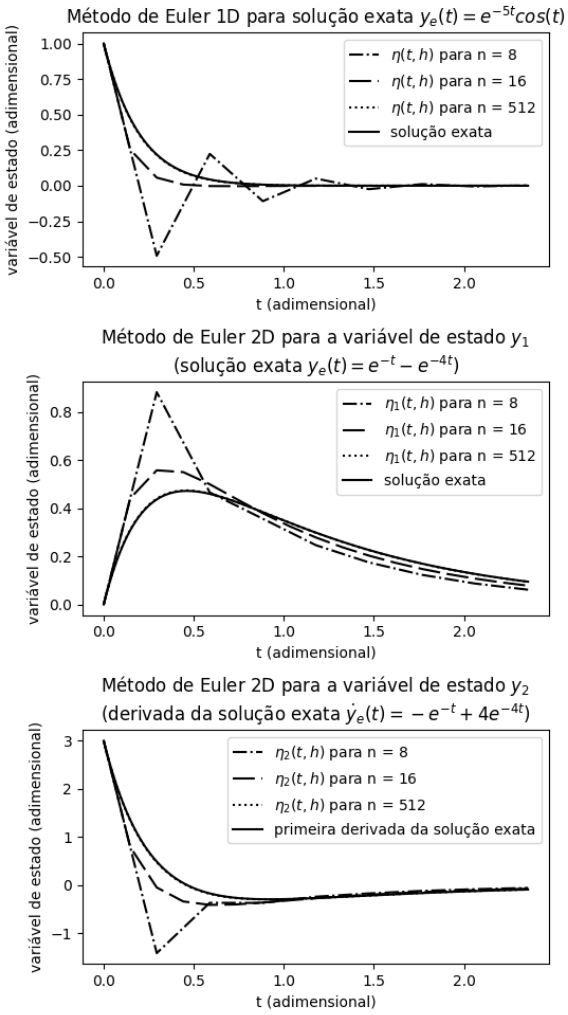

Figura 1: Comparações entre soluções exatas para Problemas de Cauchy unidimensionais e bidimensionais e aproximações para diversos valores de n

A interpretação da Tabela de Convergência segue análoga a da seção anterior. Também utilizou-se, aqui, 20 casos (mesmo intervalo de valores de n) para a análise do comportamento do erro do Método de Euler Explícito, com fator multiplicativo $r = 2$ entre eles, mas com a variável independente t variando no intervalo $[0, 50]$, de $t_0 = 0$ a $t_n = 50$. No entanto, continua-se observando o Erro de Discretização Global no ponto final do intervalo — $t_n = 50$, nesse caso.

O problema escolhido para a demonstração do método de verificação foi o de Lotka-Volterra com características de relações de mutualismo entre as populações (ver [4]) dado por

$$\begin{cases} \dot{p}(t) &= a\cdot p(t)\left(1-\frac{p(t)}{K}\right) - \frac{b\cdot p(t)\cdot q(t)}{1+b\cdot p(t)} &= f_1(t,p,q) \\ \dot{q}(t) &= m\cdot q\left(1-\frac{1}{k\cdot p(t)}\right) &= f_2(t,p,q), \end{cases}$$

que não possui solução exata.

Definiu-se, então, os valores dos parâmetros como $a = 0.2$, $m = 0.1$, $K = 500$, $k = 0.2$, $b = 0.1$ e as condições inciais como $p(t_0) = 10$ e $q(t_0) = 5$.

A aproximação pelo Método de Euler Explícito é dada, portanto, por

$$\begin{cases} \begin{pmatrix} p_{k+1} \\ q_{k+1} \end{pmatrix} = \begin{pmatrix} p_k \\ q_k \end{pmatrix} + \begin{pmatrix} h & 0 \\ 0 & h \end{pmatrix} \begin{pmatrix} f_1(t, p_k, q_k) \\ f_2(t, p_k, q_k) \end{pmatrix} \end{cases}$$

Utilizando-se a norma euclidiana para o erro ($||e(t_n, h)|| = \sqrt{e_1^2(t_n, h) + e_2^2(t_n, h)}$), produziu-se a tabela IV, que representa a Tabela de Convergência para esse Problema de Lotka-Volterra.

n	h	$\|e(t_n, h)\|$	$\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$
4	$1.250E+01$	$8.058E+07$	
8	$6.250E+00$	$8.059E+07$	$-1.713E-04$
16	$3.125E+00$	$2.814E+03$	$1.481E+01$
32	$1.562E+00$	$2.524E+00$	$1.012E+01$
64	$7.812E-01$	$2.562E+01$	$-3.343E+00$
128	$3.906E-01$	$1.537E+01$	$7.368E-01$
256	$1.953E-01$	$8.136E+00$	$9.181E-01$
512	$9.766E-02$	$4.163E+00$	$9.666E-01$
1024	$4.883E-02$	$2.104E+00$	$9.848E-01$
2048	$2.441E-02$	$1.057E+00$	$9.928E-01$
4096	$1.221E-02$	$5.298E-01$	$9.965E-01$
8192	$6.104E-03$	$2.652E-01$	$9.983E-01$
16384	$3.052E-03$	$1.327E-01$	$9.991E-01$
32768	$1.526E-03$	$6.637E-02$	$9.996E-01$
65536	$7.629E-04$	$3.319E-02$	$9.998E-01$
131072	$3.815E-04$	$1.660E-02$	$9.999E-01$
262144	$1.907E-04$	$8.298E-03$	$9.999E-01$
524288	$9.537E-05$	$4.149E-03$	$1.000E+00$
1048576	$4.768E-05$	$2.075E-03$	$1.000E+00$
2097152	$2.384E-05$	$1.037E-03$	$1.000E+00$

Tabela IV: Estimativa da ordem de convergência do Método de Euler no instante $t_n = 50$, para um Problema de Lotka-Volterra.

De maneira análoga à exposta nas seções IV A 1 e IV A 2, constata-se, pela tabela IV, que o erro $\|e(t_n, h)\|$ converge para 0 e o logaritmo $\log_2(\frac{\|e(t,2\cdot h)\|}{\|e(t,h)\|})$, para 1, corroborando a suposição de que o código computacional tenha sido implementado corretamente, mesmo que não haja solução exata do problema em análise.

A figura 2 apresenta aproximações para diversos valores de n para as variáveis de estado p e q, respectivamente. Nota-se que, para pequenos valores de n, o erro aumenta consideravelmente conforme o método aproxima-se do fim do intervalo. No entanto, para valores maiores de n, o método estabiliza-se, de maneira geral, e os gráficos das aproximações se sobrepõe.

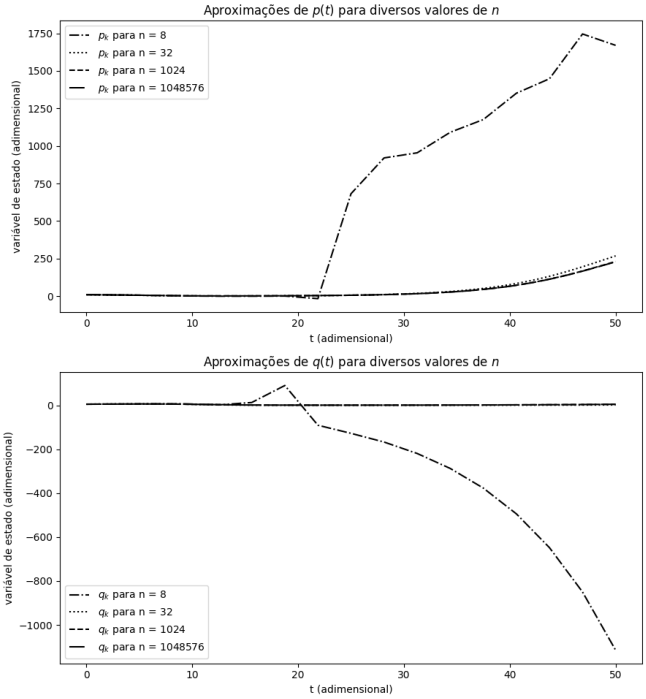

Figura 2: Comparações entre soluções aproximadas para um Problema de Lotka-Volterra para diversos valores de n

V. Conclusões

Neste trabalho, considerou-se a importância da verificação da corretude da implementação de um método numérico e apresentou-se duas maneiras de se alcançar esse fim. Uma das maneiras consiste no Método da Solução Manufaturada, quando se conhece a solução exata de uma EDO, sendo a outra uma estimativa realizada por meio de artifícios matemáticos, quando não se conhece a solução exata da EDO ou sistema de EDOs. Também explicitou-se modos de se analisar os resultados de uma Tabela de Convergência quando se sabe a ordem de convergência do método utilizado para gerá-la.

[1] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] Alexandre Megiorin Roma, Joyce Da Silva Bevilacqua, and Rudimar Luiz Nós. *Métodos para a solução numérica de equações diferenciais ordinárias a valores iniciais-Notas de Aula*. Br, IME- USP, 2022.

[4] Wikipedia. Mutualism (biology) — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Mutualism\%20(biology)&oldid=1134276088`, 2023. [Online; accessed 22-January-2023].

A. Código da seção IV

1. seções IV A 1 e IV A 2

```
# autor:  Igor Pontes Tresolavy
# autor:  Thiago Antice Rodrigues de Souza

"""
Confeccionando tabelas de convergência para os
    ↪ problemas manufaturados:
y_e_1(t) = e^(-5t)cos(t)
y_e_2(t) = e^(-t) - e^(-4t)

Problema de Cauchy 1D
- d[y(t)]/dt = f(t, y) = -5y - e^(-5t)sin(t)
- y(0) = 1

Problema de Cauchy 2D
d[(y1(t), y2(t))]/dt = ((0, 1), (-4, -5))*(y1(t), y2(t)
    ↪ )
(y1(0), y2(0)) = (0, 3)
"""

import matplotlib.pyplot as plt
import numpy as np

# definições
NO_DE_CASOS = 20
# condições iniciais
INICIO_INTERVALO = 0
CONDICAO_INICIAL_1 = 1
CONDICAO_INICIAL_2 = [0, 3]
FIM_INTERVALO = (3.0*np.pi)/4.0
QNTD_PASSOS_INICIAL = 4
FATOR_MULTIPLICATIVO = 2
PASSO_INICIAL = (FIM_INTERVALO - INICIO_INTERVALO)/
    ↪ QNTD_PASSOS_INICIAL

CASO = 0 # usado para distinguir condições inciais
CONDICOES_INICIAIS = [CONDICAO_INICIAL_1,
    ↪ CONDICAO_INICIAL_2]


# definindo a sequência de passos no tempo e
# a variável de estado
t = np.arange(INICIO_INTERVALO, FIM_INTERVALO +
    ↪ PASSO_INICIAL, PASSO_INICIAL)
y_k = [np.array(CONDICOES_INICIAIS[CASO])]

# Método de Euler Explícito
def euler(T, h_n, f):
    global t, y_k

    t_0 = INICIO_INTERVALO

    t = np.arange(t_0, T + h_n, h_n)
    y_k = [np.array(CONDICOES_INICIAIS[CASO])]

    for t_k in t[1:]:
        y_k.append(y_k[-1] + h_n*f(t_k, y_k[-1]))

    return y_k[-1]

# Solução exata do Problema de Cauchy 1D
def y_e_1(T):
    return np.cos(T)*np.exp(-5*T)
```

```
# Solução exata do Problema de Cauchy 2D
def y_e_2(T):
    return np.array([np.exp(-T) - np.exp(-4*T), -np.exp
        ↪ (-T) + 4*np.exp(-4*T)])

# f(t,y) do problema de Cauchy 1D
def f_1(t, y):
    return -5*y - np.exp(-5*t)*np.sin(t)

# f(t,y_1, y_2) do problema de Cauchy 2D
def f_2(t, y):
    return np.array([y[1], -4*y[0] -5*y[1]])

# Erro de Discretização Global
def global_error(T, h_n, y_e, numerical_approximation,
    ↪ f):
    return y_e(T) - numerical_approximation(T, h_n, f)

def main():
    global CASO, t, y_k

    y_e = [y_e_1, y_e_2]
    f = [f_1, f_2]
    fig, (ax1, ax2, ax3) = plt.subplots(3, figsize
        ↪ =(7,10))
    fig.tight_layout()
    linestyles = ["dotted", "dashed", "dashdot", (0,
        ↪ (10, 3))]

    print("TABELA DE VERIFICAÇÃO DE SOLUÇÃO
        ↪ MANUFATURADA (1D e 2D)");

    for i in range(0, 2):
        print("\n%dD" %(i+1));

        for caso in range(1, NO_DE_CASOS + 1):
            n = QNTD_PASSOS_INICIAL*(
                ↪ FATOR_MULTIPLICATIVO**(caso-1))
            h_n = (FIM_INTERVALO - INICIO_INTERVALO)/n
            e = np.max(abs(global_error(FIM_INTERVALO,
                ↪ h_n, y_e[i], euler, f[i]))) # norma
                do máximo
            ordem__p = np.log(abs(e__anterior/e))/np.log
                ↪ (FATOR_MULTIPLICATIVO) if caso != 1
                ↪ else 0

            print("%s %5d %s & %s %9.3e %s & %s %9.3e %s
                ↪  & %s %9.3e %s \\\\" % ("$", n, "$",
                ↪  "$", h_n, "$", "$",e, "$", "$",
                ↪ ordem__p, "$"));

            h_n__anterior = h_n
            e__anterior = e

            if caso == 2 or caso == 3 or caso == 8:
                if len(y_k[0].shape) == 0:
                    ax1.plot(t, y_k, color="black",
                        ↪ linestyle=linestyles[caso%4],
                        ↪  label="$\eta(t,h)$ para n =
                        ↪ {}".format(str(n)) )
                else:
                    ax2.plot(t, [y[0] for y in y_k],
                        ↪ color="black", linestyle=
                        ↪ linestyles[caso%4], label="$\
                        ↪ eta_1(t,h)$ para n = {}".
                        ↪ format(str(n)))
                    ax3.plot(t, [y[1] for y in y_k],
                        ↪ color="black", linestyle=
                        ↪ linestyles[caso%4], label="$\
                        ↪ eta_2(t,h)$ para n = {}".
                        ↪ format(str(n)))
```

```
        CASO = CASO + 1

    ax1.plot(t, y_e_1(t), color="black", linestyle="
        ↪ solid", label="solução exata")
    ax2.plot(t, y_e_2(t)[0], color="black", linestyle="
        ↪ solid", label="solução exata")
    ax3.plot(t, y_e_2(t)[1], color="black", linestyle="
        ↪ solid", label="primeira derivada da solução
        ↪ exata")

    ax1.set_xlabel("t (adimensional)")
    ax1.set_ylabel("variável de estado (adimensional)")
    ax1.set_title("Método de Euler 1D para solução
        ↪ exata $y_e(t) = e^{-5t}cos(t)$", size=12)
    ax1.legend(loc="upper right",)

    ax2.set_xlabel("t (adimensional)")
    ax2.set_ylabel("variável de estado (adimensional)")
    ax2.set_title("Método de Euler 2D para a variável
        ↪ de estado $y_1$ \n(solução exata $y_e(t) = e
        ↪ ^{-t} - e^{-4t}$)", size=12)
    ax2.legend(loc="upper right",)

    ax3.set_xlabel("t (adimensional)")
    ax3.set_ylabel("variável de estado (adimensional)")
    ax3.set_title("Método de Euler 2D para a variável
        ↪ de estado $y_2$ \n(derivada da solução exata
        ↪  $\dot{y}_e(t) = -e^{-t} + 4e^{-4t}$)", size
        ↪ =12)
    ax3.legend(loc="upper right")

    ax1.set_box_aspect(1/2)
    ax2.set_box_aspect(1/2)
    ax3.set_box_aspect(1/2)

    fig.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

2. seção IV B

```
# autor:  Igor Pontes Tresolavy
# autor:  Thiago Antici Rodrigues de Souza

"""
Sabe-se que o Método de Euler possui
erro de ordem de convergência 1.
Portanto, ao se estimar a ordem de convergência
pelos métodos descritos na seção 2.3.2 e 2.3.3
das notas de aula do Professor Roma, espera-se
obter um resultado semelhante ao da depuração
por solução manufaturada (seção 2.3.1).

Ademais, espera-se que a estimativa do erro global tamb
    ↪ ém
convirja para 0.

Usou-se o seguinte problema de Lotka-Volterra nesse
    ↪ programa
(Lotka-Volterra com características de relacionamento
    ↪ mutual;
ver https://en.wikipedia.org/wiki/Mutualism_(biology)\#
    ↪ Mathematical_modeling):

dp/dt = a*p(1-p/K) - b*p*q/(1+b*p)
```

```
dq/dt = m*q(1-1/(k*p))

com a = 0.2, m = 0.1, K= 500, k = 0.2, b = 0.1, p(0) =
    ↪ 10,
q(0) = 5
"""

import matplotlib.pyplot as plt
import numpy as np

# definições
NO_DE_CASOS = 20
INICIO_INTERVALO = 0
# condições iniciais
CONDICAO_INICIAL = [10, 5]
FIM_INTERVALO = (3.0*np.pi)/4.0
QNTD_PASSOS_INICIAL = 4
FATOR_MULTIPLICATIVO = 2
PASSO_INICIAL = (FIM_INTERVALO - INICIO_INTERVALO)/
    ↪ QNTD_PASSOS_INICIAL

def f(t, y):
    p = y[0]
    q = y[1]
    a = 0.2
    m = 0.1
    K = 500
    k = 0.2
    b = 0.1
    return np.array([a*p*(1-p/K) - b*p*q/(1+b*p), m*q
        ↪ *(1-1/(k*p))])

def euler(T, h_n, f):
    t_0 = INICIO_INTERVALO

    # definindo a sequência de passos no tempo
    t = np.arange(t_0, T + h_n, h_n)
    y_k = [np.array(CONDICAO_INICIAL)]

    for t_k in t:
        y_k.append(y_k[-1] + h_n*f(t_k, y_k[-1]))

    return y_k[-1]

def global_error_estimate(T, h_n,
    ↪ numerical_approximation, f):
    # ordem de convergência do método de euler = 1
    return numerical_approximation(T, h_n, f) -
        ↪ numerical_approximation(T, h_n/2, f)/(2**1 -
        ↪  1)

def main():

    print("TABELA DE VERIFICAÇÃO DE UM PROBLEMA DE
        ↪ LOTKA-VOLTERRA");

    for caso in range(1, NO_DE_CASOS + 1):
        n = QNTD_PASSOS_INICIAL*(FATOR_MULTIPLICATIVO**(
            ↪ caso-1))
        h_n = (FIM_INTERVALO - INICIO_INTERVALO)/n
        # norma euclidiana
        e = np.linalg.norm(global_error_estimate(
            ↪ FIM_INTERVALO, h_n, euler, f))
        ordem__p = np.log(abs(e__anterior/e))/np.log(
            ↪ FATOR_MULTIPLICATIVO) if caso != 1 else
            ↪ 0

        print("%5d & %9.3e & %9.3e & %9.3e \\\\" % (n,
            ↪ h_n,e,ordem__p));

        e__anterior = e

if __name__ == "__main__":
    main()
```