Universidade de São Paulo

Trabalho de Métodos Numéricos para Equações Diferenciais
Problemas de Valor de Contorno

Anna Lucia Moro Lanzuolo 13864436
Isabela Guarnier De Mitri 13862264

2 de maio de 2024

1 Introdução

Equações diferenciais desempenham um papel crucial em uma variedade de disciplinas, modelando fenômenos complexos que vão desde o movimento de corpos até o comportamento de sistemas biológicos. Entre essas equações, a equação de Van der Pol, proposta por Balthazar Van der Pol, destaca-se como um modelo essencial para sistemas auto-oscilatórios.

$$u'' + \mu \left(u^2 - 1\right) u' + u = 0$$

A equação de Van der Pol descreve oscilações não lineares em sistemas elétricos e mecânicos, sendo utilizada em uma variedade de aplicações, desde circuitos eletrônicos até modelagem de processos biológicos. Seu estudo permite compreender melhor fenômenos como batimentos cardíacos, oscilações neurais e ondas em meios fluidos.

Neste contexto, o presente trabalho propõe-se a analisar a equação de Van der Pol sob diferentes condições e parâmetros. Inicialmente, será realizada a discretização do problema, considerando uma malha uniforme e utilizando diferenças finitas de ordem 2. Em seguida, implementaremos e resolveremos este problema de valor de contorno utilizando o método de Newton para resolução do sistema não-linear. Serão produzidos gráficos com a solução do problema para diferentes valores do parâmetro μ, permitindo-nos observar as mudanças na solução e avaliar o impacto dessas variações na performance do código construído.

Esta análise contribuirá para uma melhor compreensão da equação de Van der Pol e seus efeitos sob diferentes condições, bem como para o desenvolvimento de habilidades na implementação de métodos numéricos para a resolução de problemas não lineares em equações diferenciais.

2 Desenvolvimento

2.1 Parte 1

Dada a equação de Van Der Pol:

$$u'' + \mu(u^2 - 1)u' + u = 0 \tag{1}$$

definida no intervalo $[0, 3\pi]$ com condições de contorno:

$$u(0) = 1, \tag{2}$$

$$u(3\pi) + u'(3\pi) = -1 \tag{3}$$

Primeiro, é necessário discretizar a equação (1) :

$$F_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mu\left((u_i)^2 - 1\right)\frac{u_{i+1} - u_{i-1}}{2h} + u_i \tag{4}$$

Para montar o sistema não linear, utilizamos a equação discretizada (4) analisando os possíveis valores de i.

- Para $i = 0$

Obtemos na equação u_{i-1} que não temos, portanto utilizamos a equação de contorno dada (2), sendo ela $u(0) - 1 = 0$.

- Para $0 < i < N$

Para esse caso, utilizamos a equação (4).

- Para $i = N$

Para $i = N$, obtemos pela equação discretizada a icógnita u_{N+1}, e então precisamos usar *ghost cell*. De acordo com a condição de contorno (3), vamos discretizá-la:

$$u(3\pi)+u'(3\pi) = -1 \Rightarrow u_N+\frac{u_{N+1} - u_{N-1}}{2h} = -1 \Rightarrow u_{N+1} = -2h(u_N+1)+u_{N-1}$$

Substituindo o valor de u_{N+1} na equação discretizada, temos:

$$\frac{-2h(u_n + 1) + 2u_{n-1} - 2u_n}{h^2} - \mu((u_n)^2 - 1)(u_n + 1) + u_n = 0$$

Segue que o sistema não linear fica como:

$$\begin{cases} u(0) - 1 = 0, \text{para } i = 0 \\ \dfrac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mu\left((u_i)^2 - 1\right)\dfrac{u_{i+1} - u_{i-1}}{2h} + u_i, \text{ para } 0 < i < N \\ \dfrac{-2h(u_n + 1) + 2u_{n-1} - 2u_n}{h^2} - \mu((u_n)^2 - 1)(u_n + 1) + u_n = 0, \text{ para } i = N \end{cases}$$

Para resolver o problema pelo método de Newton precisamos implementar a matriz jacobiana. Ela irá ser produzida da seguinte maneira:

- Para $0 < i < N$:

$$\frac{\partial F}{\partial u_{i-1}} = \frac{1}{h^2} - \mu \cdot \frac{u_i^2 - 1}{2h}$$

$$\frac{\partial F}{\partial u_i} = \frac{-2}{h^2} + \mu \cdot \frac{u_i(u_{i+1} - u_{i-1})}{2h}$$

$$\frac{\partial F}{\partial u_{i+1}} = \frac{1}{h^2} - \mu \cdot \frac{u_i^2 - 1}{2h}$$

- Para $i = 0$:

$$\frac{\partial F}{\partial u_{i-1}} = 0$$

$$\frac{\partial F}{\partial u_i} = 1$$

$$\frac{\partial F}{\partial u_{i+1}} = 0$$

- Para $i = N$:

$$\frac{\partial F}{\partial u_{i-1}} = \frac{2}{h^2}$$

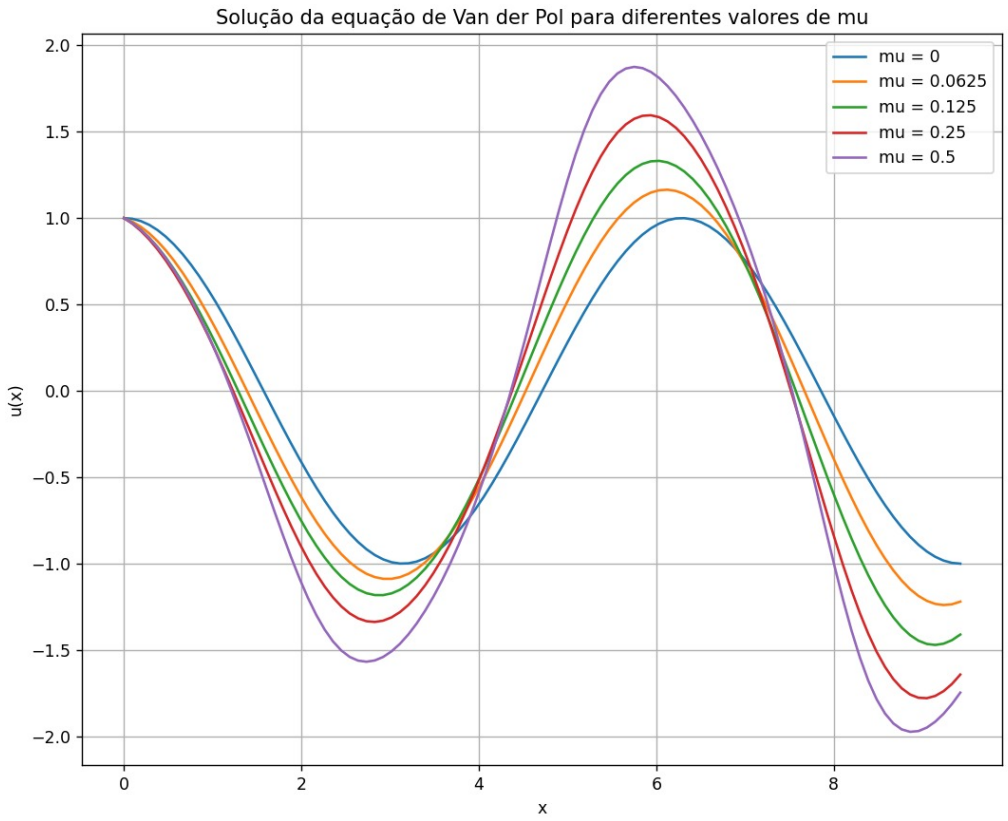
$$\frac{\partial F}{\partial u_i} = \frac{-2 - 2h}{h^2} - \mu(+3u_i^2 + 2u_i - 1) + 1$$

$$\frac{\partial F}{\partial u_{i+1}} = 0$$

A parte 1 do problema nos pede para resolver a equação para diferentes valores de μ utilizando o método de Newton.

Para isso, montamos um código em python que resolve o problema e monta os gráficos para cada um dos μ's.

Obtivemos como resposta os seguintes gráficos:



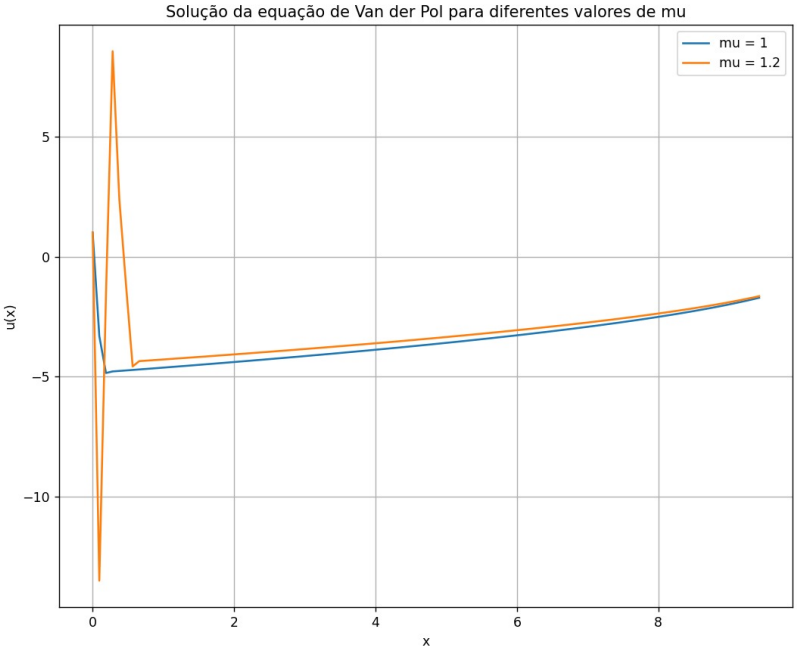


Tabela 1: Tabela de iterações para cada valor de μ

μ	Iterações
0	1
1/16	1
1/8	3
1/4	4
1/2	5
1	61
6/5	188

A alteração nos valores de μ podem afetar a estabilidade da solução e o comportamento oscilatório do sistema. Podemos observar que para valores mais baixos, como 0, $\frac{1}{16}$ e $\frac{1}{8}$, levam a um comportamento mais suave e estável. Por outro lado, valores mais altos levam à oscilações mais fortes e um sistema mais não linear.

As variações nos valores de μ também podem afetar a perfomance do código, especialmente se o problema se tornar mais não linear ou se as oscilações se tornarem mais acentuadas. Isso pode exigir um número maior de iterações para a convergência da solução. Além disso, valores extremos de μ podem introduzir instabilidades numéricas ou exigir uma malha mais fina

para capturar corretamente o comportamento da solução.

2.2 Parte 2

O objetivo é estudar numericamente a convergência da solução exata de uma equação diferencial modificada para um problema específico, comparando-a com soluções numéricas obtidas por discretização. Vamos considerar um valor fixo de $\mu = \frac{1}{16}$, e as condições de contorno conhecidas anteriormente, a equação diferencial é dada por

$$u'' + \mu(u^2 - 1)u' + u = \mu \sin^3(x) \tag{5}$$

Para este estudo, várias malhas serão utilizadas para discretizar o problema, e o erro será calculado em relação à solução exata. Será construído um gráfico do erro em função do espaçamento, com ambos os eixos em escala logarítmica, para visualizar o comportamento assintótico do erro. Além disso, uma tabela será criada com os valores das inclinações entre cada par de pontos no gráfico, para confirmar o comportamento visual a partir de dados numéricos.

Na construção do código, é necessário definir os parâmetros iniciais, como μ, h (espaçamento) e N, o vetor com 5 malhas diferentes, assim como vetores para armazenar os valores do erro com diferentes normas. É feito um *loop* que calcula os espaçamentos da malha para cada número de pontos especificado em N. Depois é feito um segundo *loop* que itera sobre cada malha definida . É inicializado vetores e variáveis necessárias. Definimos a função discretizada que representa o sistema de equações não-lineares, depois a matriz Jacobiana do sistema. A equação discretizada é dada por:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mu\left((u_i)^2 - 1\right)\frac{u_{i+1} - u_{i-1}}{2h} + u_i = \mu \sin^3(x)$$

A matriz Jacobiana é a mesma da Parte 1, visto que o incremento não influencia nas derivadas. Implementamos o método de Newton para resolver o sistema não-linear. Os erros são calculados utilizando diferentes normas e é armazenado nos vetores. Plotamos o erro em função do espaçamento da malha em escala logarítmica. São plotadas três curvas representando os diferentes tipos de normas de erro.

∞-norm:

$$\|E\|_\infty := \max_{1 \leqslant j \leqslant m} |E_j| = \max_{1 \leqslant j \leqslant m} |U_j - u(x_j)|$$

discrete L_1-norm:

$$\|E\|_1 := h \sum_{j=1}^{m} |E_j|$$

discrete L_2-norm:

$$\|E\|_2 := \left(h \sum_{j=1}^{m} |E_j|^2 \right)^{\frac{1}{2}}$$

O código então exibe o gráfico do erro em função do espaçamento da malha para cada tipo de norma. Isso permite visualizar como o erro diminui à medida que a malha é refinada, e também como diferentes normas de erro se comportam em relação ao espaçamento da malha.

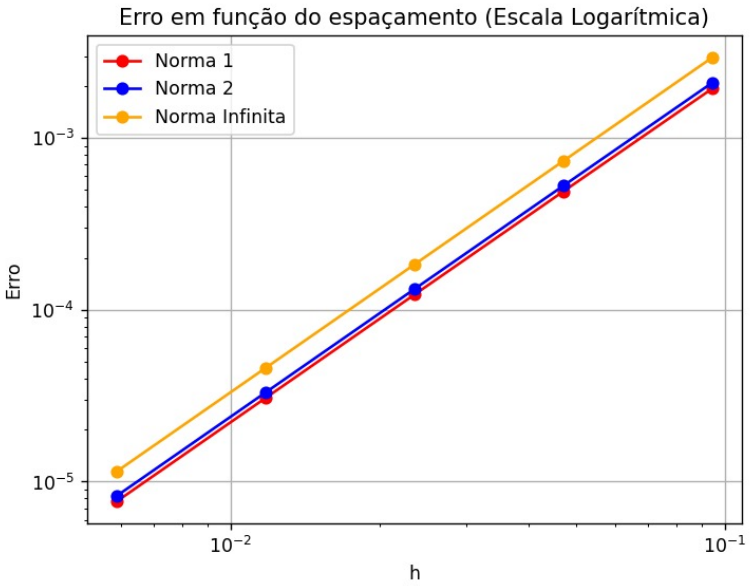


Figura 1: Erro

Para calcular a inclinação entre dois pares de pontos, que pode ser visto também como a ordem do erro, utilizamos a seguinte fórmula:

$$p \approx \frac{\log\left(|E\left(h_1\right)| \,/\, |E\left(h_2\right)|\right)}{\log\left(h_1/h_2\right)}$$

Usando ela, obtivemos como resultado da inclinação entre os pontos os seguintes valores:

Tabela 2: Tabela de valores de h_1, h_2 e a inclinação

h_1	h_2	Slope
0.0942	0.0471	1.9868
0.0471	0.0236	1.9942
0.0236	0.0118	1.9972
0.0118	0.0059	1.9987

Podemos concluir então que quanto menor é o h melhor é a aproximação do valor da função.