IME - USP

MAP 2320 - Métodos Numéricos em Equações Diferenciais II

Trabalho Computacional 1

Erik Davino Vincent
10736584
December 6, 2020

Problema

"Considere a EDP parabólica,

$$u_t = u_{xx} + g(x), \qquad x \in [0, 1]$$

Vamos definir a solução manufaturada:

$$u(x, t) = e^{-t}.\cos(x) + x.\sin(x)$$

e deduzir g(x)."

Temos:

$$u_t = -e^t.\cos(x)$$

$$u_{xx} = -e^{-t}.\cos(x) + 2.\cos(x) - x.\sin(x)$$

Portanto:

$$g(x) = u_t - u_{xx} = -2.\cos(x) + x.\sin(x)$$

A.

"Aplique 1000 passos do método Forward Difference com $k = \Delta t$ ligeiramente abaixo do limite de estabilidade. Avalie o erro em relação a solução manufatura (norma 2 da diferença entre os valores para os pontos da discretização)."

Considere para a discretização espacial $m = 25$. Para a discretização no tempo, considere $t \in [0, 0.79]$ e $k = 0.00079$. Como $\alpha = 1$, encontramos $\lambda = k(\frac{\alpha}{h})^2 = 0.49375$, que está logo abaixo do limite de estabilidade de $\lambda = 0.5$. Como $T = 0.79$, temos exatamente 1000 passos usando $k = 0.00079$.

Inicialmente temos a configuração na figura 1. Após 1000 passos, obtemos a configuração da figura 2, com um erro em norma 2 de 0.0042. Na figura 3, podemos ver a evolução do erro em cada instante do tempo.

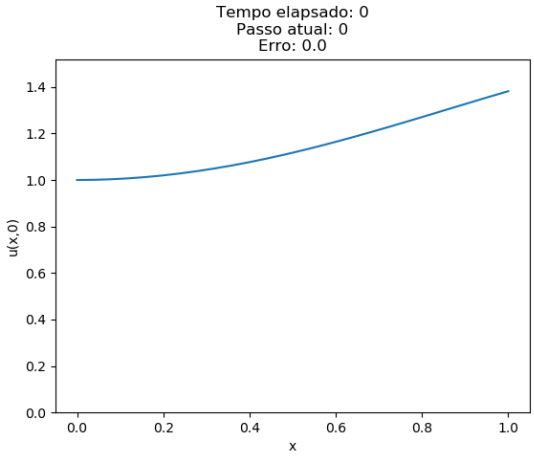

Figure 1: Instante inicial.

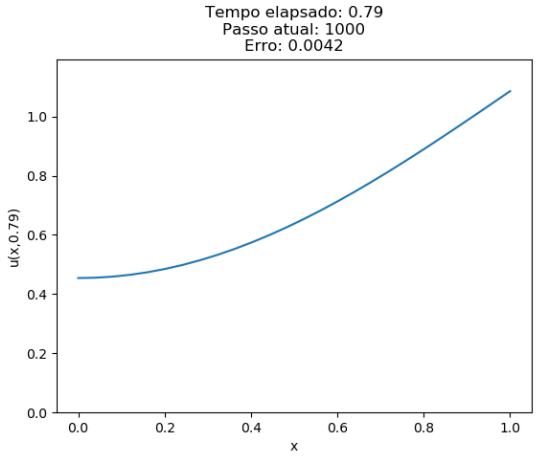

Figure 2: Instante final.

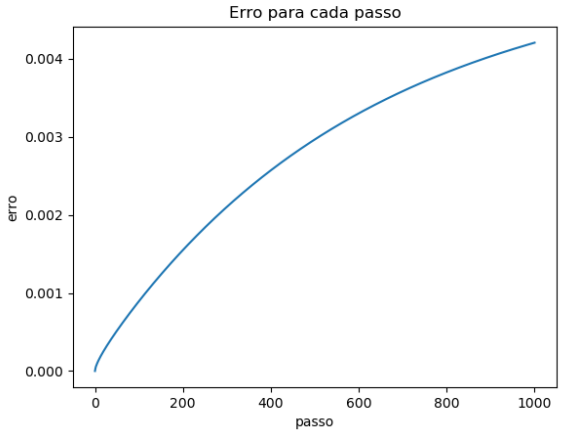

Figure 3: Evolução do erro a cada passo do método; no instante inicial o erro é sempre nulo.

B.

"Aplique o método de Crank-Nicolson até alcançar o mesmo instante T correspondente aos 1000 passos do item A, buscando o maior k de forma a conseguir o mesmo nível de erro."

Vamos considerar inicialmente a mesma configuração do esquema anterior: $m = 25$, $T = 0.79$ e $k = 0.00079$. Dessa forma executamos os mesmos 1000 passos do problema anterior, mas obtemos um erro no instante T de 0.0049. Esse erro está na mesma ordem de grandeza do outro método, porém é maior (ao contrario do que esperávamos).

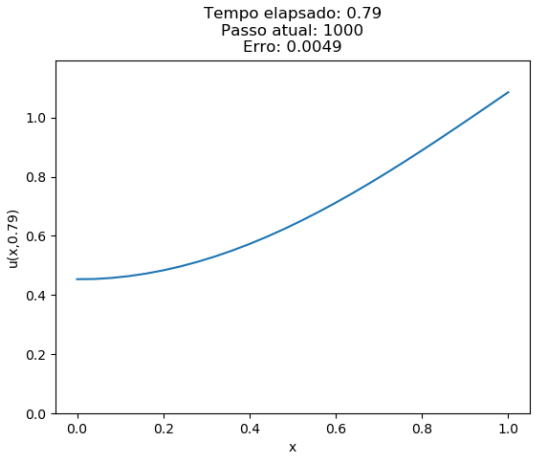

Figure 4: Instante final.

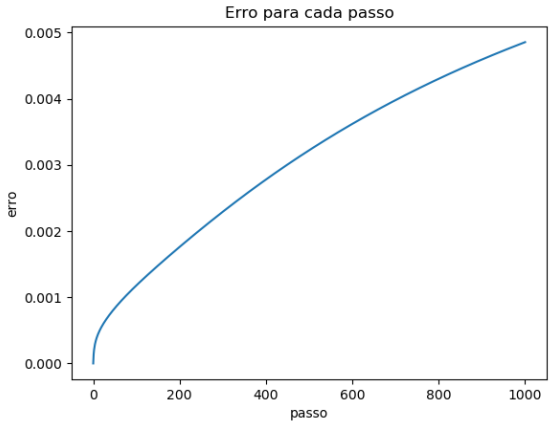

Figure 5: Evolução do erro a cada passo do método; no instante inicial o erro é sempre nulo.

C.

"Meça os tempos de cada método e compare os esforços computacionais."

Considerando T fixo, T = 0.79, vamos avaliar os tempos computacionais médios em 100 rodadas, para diferentes configurações. Os resultados obtidos para cada método se encontram na tabela 1.

	Forward Difference	**Crank Nicolson**
$m = 10;\ k = 0.00079$	0.05 s	0.04 s
$m = 10;\ k = 0.000079$	0.49 s	0.68 s
$m = 25;\ k = 0.00079$	0.11 s	0.05 s
$m = 25;\ k = 0.000079$	1.15 s	0.51 s

Table 1: Comparação dos tempos computacionais médios em 100 rodadas para os métodos Forward Difference e Crank Nicolson

D.

Ambos os métodos foram capazes de aproximar a solução com erro da ordem de $10^{-}3$. Porém, vemos que o erro do método de Crank-Nicolson é um pouco pior. A figura 6 mostra melhor essa comparação.
(Acreditamos que isso pode ter sido causado por algum erro de implementação do algorítimo que ainda não pudemos identificar).

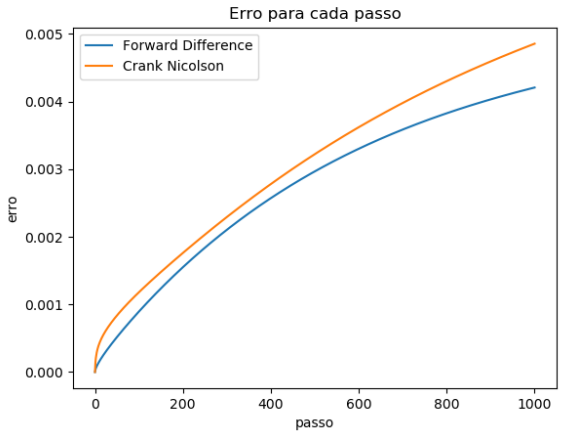

Figure 6: Evolução do erro a cada passo de ambos os métodos.

O tempo computacional do método de Crank Nicolson foi inferior em quase todos os casos. A implementação das operações matriciais foi econômica, o que possibilitou mais eficiência. Podemos concluir desses resultados que o tempo do método é menos afetado pelo afinamento da malha de discretização.

Extra

Implementação

Os algorítimos foram escritos em **Python 3.8**, com nomes *EP-1-Crank_Nicolson.py* e *EP-1-Forward_Difference.py*. Para obter os resultados obtidos acima basta executa-los sem altera-los. Para modificar algum parâmetro (e.g. T, m, L, etc.), basta editar o código dentro da função *main()*. Ambos os algorítimos possuem uma versão "fast", que não possui opção de plotagem e nem de gravar o erro; serve principalmente para analisar o tempo de execução.

Incorporação das condições de contorno

O método de Crank-Nicolson consiste em resolver o problema

$$\frac{w_{ij+1} - w_{ij}}{k} - \alpha^2[\frac{w_{i+1j} - 2w_{ij} + w_{i-1j}}{h^2} + \frac{w_{i+1j+1} - 2w_{ij+1} + w_{i-1j+1}}{h^2}]$$

para $m - 2$ pontos (de discretização espacial) de forma simultânea. Assim, obtém-se a forma matricial do problema, definida para cada j = 1,2,... por:

$$\begin{bmatrix} 2+2\lambda & -\lambda & \cdots & 0 \\ -\lambda & 2+2\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\lambda \\ 0 & \cdots & -\lambda & 2+2\lambda \end{bmatrix} w^{(j+1)} = \begin{bmatrix} 2-2\lambda & \lambda & \cdots & 0 \\ \lambda & 2-2\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda \\ 0 & \cdots & \lambda & 2-2\lambda \end{bmatrix} w^{(j)} \tag{1}$$

O lado direito possui informações conhecidas em cada instante de tempo, então podemos simplifica-lo para:

$$\begin{bmatrix} 2-2\lambda & \lambda & \cdots & 0 \\ \lambda & 2-2\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda \\ 0 & \cdots & \lambda & 2-2\lambda \end{bmatrix} w^{(j)} = \begin{bmatrix} (2-2\lambda)w_1^{(j)} + \lambda(w_0^{(j)} + w_2^{(j)}) \\ (2-2\lambda)w_2^{(j)} + \lambda(w_1^{(j)} + w_3^{(j)}) \\ \vdots \\ (2-2\lambda)w_{m-1}^{(j)} + \lambda(w_{m-2}^{(j)} + w_m^{(j)}) \end{bmatrix} \tag{2}$$

Para adicionar as condição de contorno de Dirichlet para $x = 0$, basta adicionar o termo:

$$\begin{bmatrix} 2-2\lambda & \lambda & \cdots & 0 \\ \lambda & 2-2\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda \\ 0 & \cdots & \lambda & 2-2\lambda \end{bmatrix} w^{(j)} = \begin{bmatrix} (2-2\lambda)w_1^{(j)} + \lambda(w_0^{(j)} + w_2^{(j)}) \\ (2-2\lambda)w_2^{(j)} + \lambda(w_1^{(j)} + w_3^{(j)}) \\ \vdots \\ (2-2\lambda)w_{m-1}^{(j)} + \lambda(w_{m-2}^{(j)} + w_m^{(j)}) \end{bmatrix} + \begin{bmatrix} \lambda.a \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3}$$

Para a condição de Neumann, considere a aproximação que leva em conta a 2_a derivada:
$w_{mj+1} = \dfrac{2hb + 4w_{m-1j+1} - w_{m-2j+1}}{3}$.
Substituindo em:

$$-\lambda w_{m-2j+1} + (2+2\lambda)w_{m-1j+1} - \lambda w_{mj+1} = RHS^{(j)}$$

obtemos:

$$-\frac{2}{3}w_{m-2j+1} + (2+\frac{2\lambda}{3})w_{m-1j+1} - \frac{\lambda 2hb}{3} = RHS^{(j)}$$
$$-\frac{2}{3}w_{m-2j+1} + (2+\frac{2\lambda}{3})w_{m-1j+1} = RHS^{(j)} + \frac{\lambda 2hb}{3}$$

Assim, a nova forma para o problema se torna:

$$\begin{bmatrix} 2+2\lambda & -\lambda & \cdots & 0 \\ -\lambda & 2+2\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\lambda \\ 0 & \cdots & -\lambda\frac{2}{3} & 2+\frac{2}{3}\lambda \end{bmatrix} w^{(j+1)} = \begin{bmatrix} (2-2\lambda)w_1^{(j)} + \lambda(w_0^{(j)} + w_2^{(j)}) \\ (2-2\lambda)w_2^{(j)} + \lambda(w_1^{(j)} + w_3^{(j)}) \\ \vdots \\ (2-2\lambda)w_{m-1}^{(j)} + \lambda(w_{m-2}^{(j)} + w_m^{(j)}) \end{bmatrix} + \begin{bmatrix} \lambda.a \\ 0 \\ \vdots \\ 0 \\ \lambda\frac{2hb}{3} \end{bmatrix} \tag{4}$$