Universidade Federal de Juiz de Fora - UFJF
*Jonatas Dias Machado Costa - 202165077AC*
*Lucas Marins Ramalho de Lima - 202165555C*

**1.** Seja o problema de valor de contorno:

$$-\varepsilon\frac{d^2u}{dx^2} + u = 1, \quad x \in \Omega = [0,1] \tag{1}$$

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

A solução exata para o problema (1)-(2) é

$$u(x) = c_1 e^{-\frac{x}{\sqrt{\varepsilon}}} + c_2 e^{\frac{x}{\sqrt{\varepsilon}}} + 1 \tag{3}$$

onde $c_1 = -1 - c_2$ e $c_2 = \dfrac{e^{-\frac{1}{\sqrt{\varepsilon}}} - 1}{e^{\frac{1}{\sqrt{\varepsilon}}} - e^{-\frac{1}{\sqrt{\varepsilon}}}}$.

a) Visando apresentar uma discretização pelo método de diferenças finitas para o problema (1). Aplica-se a definição de derivada:

$$u'(x) = \frac{u(x+h) - u(x)}{h} \Rightarrow \tag{4}$$

$$u''(x) = \frac{u'(x+h) - u'(x)}{h} \Rightarrow \tag{5}$$

$$u''(x) = \frac{\frac{u(x+2h)-u(x+h)}{h} - \frac{u(x+h)-u(x)}{h}}{h} \Rightarrow \tag{6}$$

$$u''(x) = \frac{u(x+2h) - 2u(x+h) + u(x)}{h^2} \Rightarrow \tag{7}$$

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \tag{8}$$

Substituindo (8) em (1):

$$-\varepsilon\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + u(x) = 1 \Rightarrow \tag{9}$$

$$-\varepsilon\frac{u(x+h) - 2u(x) + u(x-h) - u(x)h^2}{h^2} = 1 \Rightarrow \tag{10}$$

$$-\varepsilon(u(x+h) - 2u(x) + u(x-h) - u(x)h^2) = h^2 \Rightarrow \tag{11}$$

$$-\varepsilon u(x+h) + 2\varepsilon u(x) - \varepsilon u(x-h) + \varepsilon u(x)h^2 = h^2 \Rightarrow \tag{12}$$

$$-\varepsilon u(x+h) + (2\varepsilon + h^2)u(x) - \varepsilon u(x-h) = h^2 \tag{13}$$

Passando a (13) para o contexto da implementação em um computador e considerando 'h' o valor da variação entre dois pontos a serem calculados:

$$-(\varepsilon)u_{i+1} + (2\varepsilon + h^2)u_i - (\varepsilon)u_{i-1} = h^2 \tag{14}$$

Utilizando agora o Algoritmo de Thomas para a resolução da matriz formada pelos coefientes que multiplicam **(u)**

$$\begin{bmatrix} 2\varepsilon+h^2 & -\varepsilon & & & 0 \\ -\varepsilon & 2\varepsilon+h^2 & -\varepsilon & & \\ & -\varepsilon & 2\varepsilon+h^2 & \ddots & \\ & & \ddots & \ddots & -\varepsilon \\ 0 & & & -\varepsilon & 2\varepsilon+h^2 \end{bmatrix} \begin{bmatrix} h^2 \\ h^2 \\ h^2 \\ \vdots \\ h^2 \end{bmatrix}$$

Na implementação, considera-se:

a = diagonal inferior
b = diagonal principal
c = diagonal superior
d = termo independente
x = aproximação
n = número de elementos na diagonal principal e no termo independente

Cria-se vetores auxiliares:

```
c_ = [ c[0] / b[0] ]
d_ = [ d[0] / b[0] ]
```

Eliminação Gaussiana:

```
for i in range(1, n):
    aux = b[i] - c_[i-1]*a[i-1]
    if i < n-1:
        c_.append( c[i] / aux )
    d_.append( (d[i] - d_[i-1]*a[i-1])/aux )
```

Substituição Regressiva:

```
x = [d_[-1]]
for i in range(n-2, -1, -1):
    x = [ d_[i] - c_[i]*x[0] ] + x
return x
```

b) Fixando o valor de $h$ e variando $\varepsilon = 0.1, 0.01, 0.001, 0.0001$ podemos obter seguintes aproximações dispostas na Figura 1.

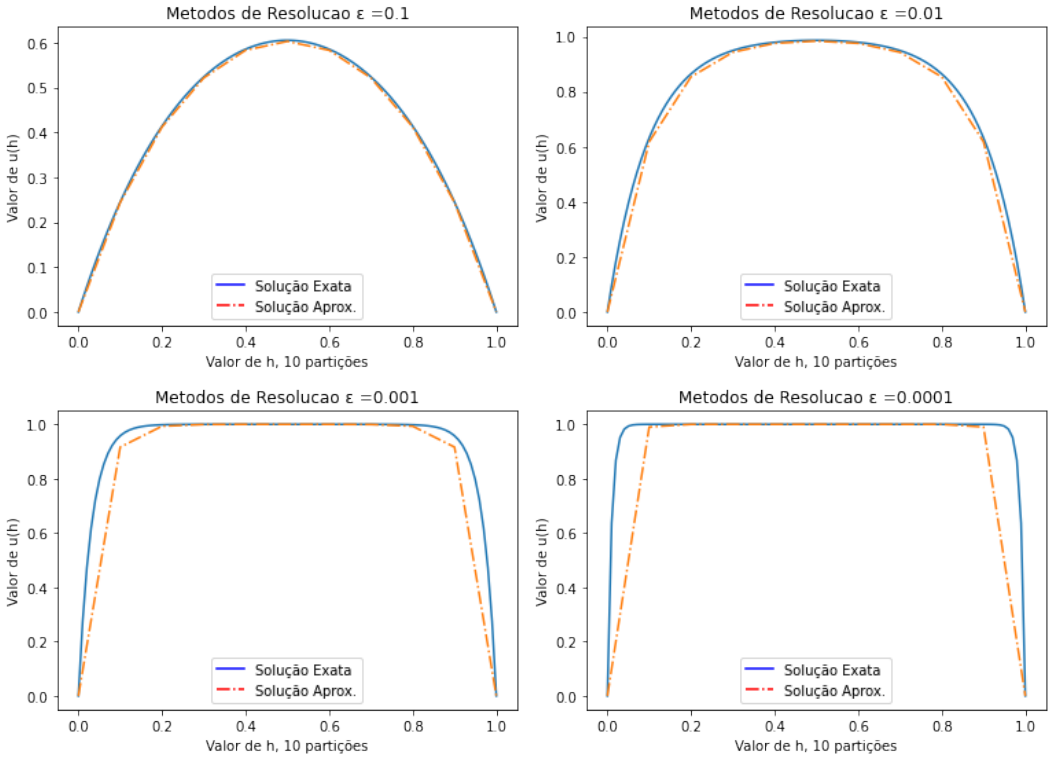


Figura 1: Gráficos comparando solução exata e solução aproximada pelo algoritmo de Thomas utilizando h = 0.1

– Com a redução gradual do valor de epslon, é perceptível a aproximação da solução encontrada em relação ao valor exato na parte central do eixo horizontal, e o distanciamento da mesma nos extremos do mesmo. Esse comportamento pode ser observado com mais clareza nos gráficos dispostos abaixo:

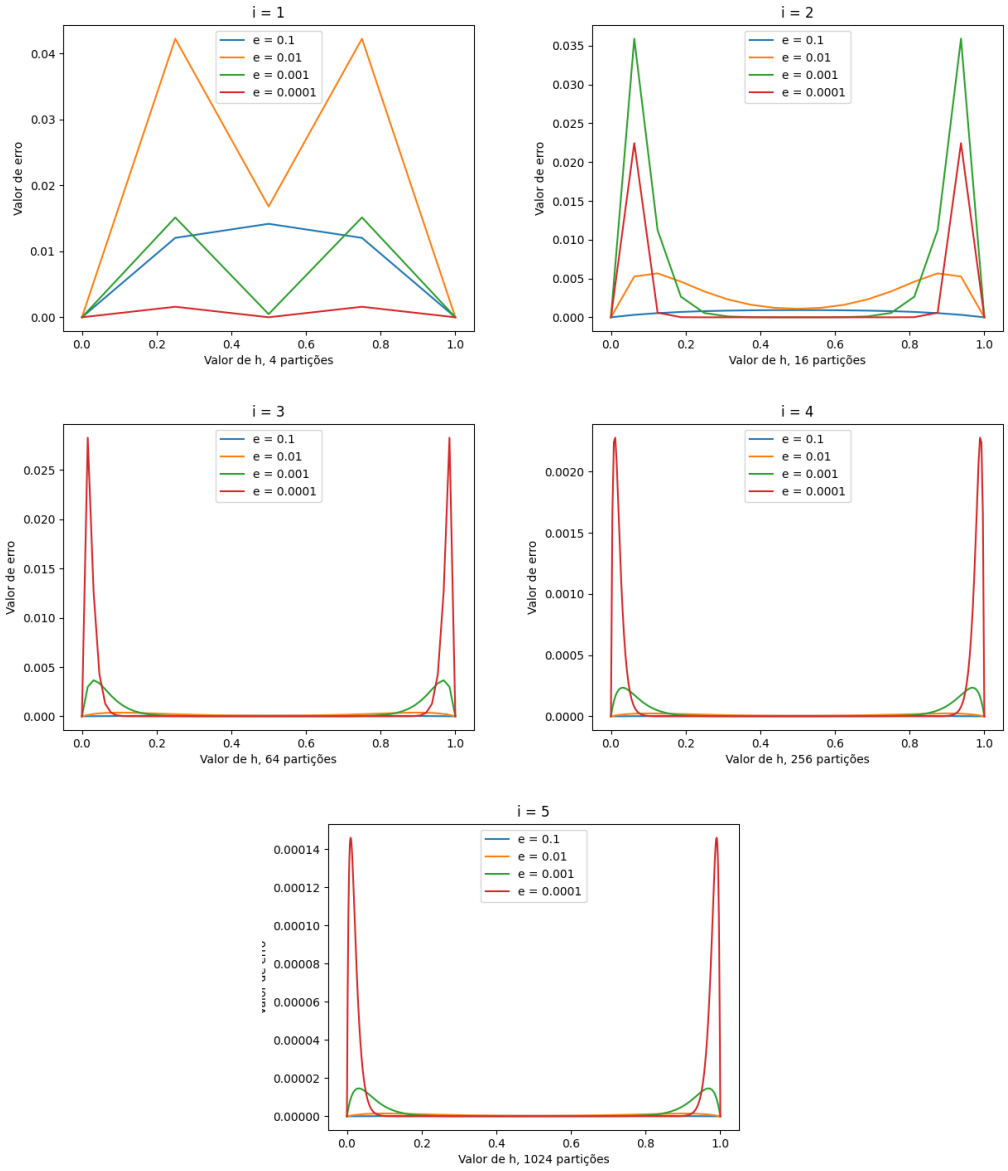


Figura 2: Gráficos com erros em grades com $4^i$ elementos para i = 1, 2, 3, 4, 5

c) Calculando o erro entre a solução exata e a solução aproximada utilizando a norma do máximo $\|u(x_n) - u^n\|_\infty$ para $\varepsilon = 0.1, 0.01, 0.001, 0.0001$ em uma malha formada por $4^i = 1, 2, 3, 4, 5$. Podemos calcular a taxa de convergência de acordo com a expressão abaixo:

$$tC = \frac{log(erro_j) - log(erro_1)}{log(\Delta t_j) - log(\Delta t_0)} \tag{15}$$

Em que j é o número de refinamentos feitos.

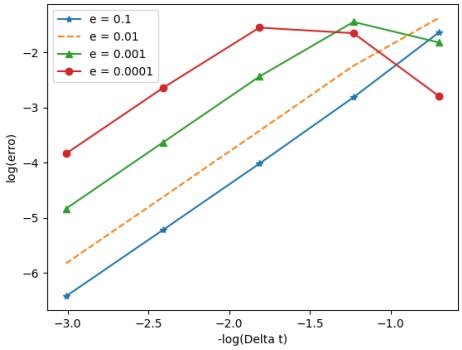


Figura 3: Curvas de crescimento do erro para cada valor de $\varepsilon$

- As taxas de convergência do método para cada valor de $\varepsilon$:
  Taxas de convergência:
  $\varepsilon = 0.1$: -2.0736633347818316446
  $\varepsilon = 0.01$: -1.9291915815484664353
  $\varepsilon = 0.001$: -1.3043677892420062743
  $\varepsilon = 0.0001$: -0.4491580435666738412
- Como observado na Figura 3, nota-se que o crescimento do erro não é linear. Isso se deve a característica do experimento analisado e a divisão do tempo total em partições. Como essa divisão é feita linearmente, os extremos possuem uma corbetura de pontos insufiente para que o método aproxime corretamente da solução exata.

(d) Os resultados dos itens (b) e (c) podem variar adotando precisão simples ou dupla na declaração das variáveis, isso se deve ao erro de variação de ponto flutuante instrínseco da expressão de números reais em computadores. Para representar um número decimal em binário é necessário pensá-lo como uma soma de bases 2 com expoentes negativos, e ao tentar fazer isso com um número definido de bits, pode haver overflow ou underflow o que causa uma pequena diferença nas últimas casas decimais que, em grande escala, pode acarretar imprecisões maiores. No cálculo em questão, as diferenças foram representadas na tabela a seguir:

Tabela 1: Tabela de comparação de representação de ponto flutuante do 75º valor calculado pelo método

| | $\varepsilon = 0.1$ | $\varepsilon = 0.01$ | $\varepsilon = 0.001$ | $\varepsilon = 0.0001$ |
|---|---|---|---|---|
| float32 | 0.185569897366358 | 0.514466095390101 | 0.898241087215579 | 0.99927089593787 |
| float64 | 0.185569899002455 | 0.514466091462338 | 0.898241092737526 | 0.999270895871395 |
| float128 | 0.185569899002452 | 0.514466091462337 | 0.898241092737533 | 0.999270895871396 |

A tabela acima foi construída com os valores calculados para o 75º ponto aproximado pelo método e indicam a diferença entre as precições de ponto flutuante. A discrepância entre os resultados representados em estrutura float32 e float64 pode ser observada a partir da 9ª casa decimal para $\varepsilon$ =0.1, sendo que em float 32 ocorreu um underflow. Para $\varepsilon$ =0.01, a partir da 9ª, com overflow, $\varepsilon$ =0.001, a partir da 8ª, com underflow, $\varepsilon$ =0.0001, a partir da 10ª, com overflow. Além disso, observe que, ao comparar o float64 com o float128, ainda é possível adquirir mais precisão ao representar as últimas casas decimais na estrutura de 128 bits.