

Leandro Furlam Turi

**Aplicação de métodos iterativos  
estacionários em Problemas de Valor no  
Contorno em 2D**

Vitória

3 de dezembro de 2019

# Sumário

<b>I</b>	<b>TÉCNICAS E ORDENS DE APROXIMAÇÃO</b>	<b>4</b>
1	Método das diferenças finitas . . . . .	5
1.1	Discretização de domínio . . . . .	5
1.2	Operador diferença finita . . . . .	5
1.3	Método das diferenças finitas para problemas lineares . . . . .	6
1.3.1	Resolução de problemas unidimensionais de contorno (PVC1D) . . . . .	6
1.3.2	Tratamento de condição de contorno de valor prescrito . . . . .	7
1.3.3	Tratamento de condição de contorno de fluxo prescrito . . . . .	7
1.3.4	Tratamento de condição de contorno mista . . . . .	7
1.3.5	Resolução de problemas bidimensionais de contorno (PVC2D) . . . . .	8
1.3.6	Tratamento de condição de contorno de valor prescrito . . . . .	9
1.3.7	Tratamento de condição de contorno de fluxo prescrito . . . . .	10
1.3.8	Tratamento de condição de contorno mista . . . . .	10
<b>II</b>	<b>IMPLEMENTAÇÕES</b>	<b>11</b>
2	Implementações . . . . .	12
2.1	Aplicação . . . . .	12
2.1.1	Validação 1 . . . . .	12
2.1.2	Validação 2 . . . . .	12
2.1.3	Validação 3 . . . . .	13
2.1.4	Aplicação 1 . . . . .	14
2.1.5	Aplicação 2 . . . . .	15
2.2	Tratamento do contorno . . . . .	16
2.3	Sistema linear . . . . .	17
2.3.1	Tempo de execução . . . . .	18
2.4	Método SOR . . . . .	18
<b>III</b>	<b>RESULTADOS</b>	<b>21</b>
3	Experimentos numéricos . . . . .	22
3.1	Validação 1 . . . . .	22
3.2	Validação 2 . . . . .	22
3.3	Validação 3 . . . . .	27

3.4	Aplicação 1 . . . . .	30
3.5	Aplicação 2 . . . . .	35
	<b>Conclusões . . . . .</b>	<b>41</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>42</b>

# Introdução

O presente relatório trata-se da resolução numérica de equações diferenciais modelando problemas bidimensionais do mundo real, através do método das diferenças finitas. Para tanto, existem três problemas de validação dos algoritmos implementados cujas soluções são conhecidas e duas aplicações. A primeira representando um resfriador bidimensional e a segunda o escoamento em águas subterrâneas. Para que seja útil em aplicações, um problema de valor sobre o contorno deve ser bem definido. Por consequência, foi disponibilizado as equações que regem estes fenômenos bem como as condições para o tratamento de contorno. A técnica utilizada foi a da discretização de domínio bidimensional em um domínio discreto unidimensional. Todas as implementações ocorreram com o software livre *Octave*<sup>1</sup>.

---

<sup>1</sup> <https://www.gnu.org/software/octave/>

## Parte I

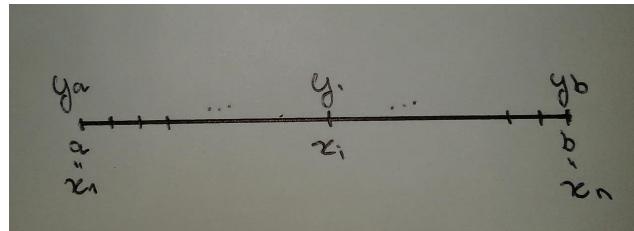
### Técnicas e ordens de aproximação

# 1 Método das diferenças finitas

## 1.1 Discretização de domínio

Normalmente, os problemas que modelam aplicações reais são problemas do tipo contínuo, uma vez que todos os pontos do domínio são incluídos tanto na descrição quanto na solução do problema. Como o computador trabalha com valores discretos, é necessário que seja feita uma transformação deste domínio contínuo num domínio discreto.

Figura 1 – Discretização de domínio 1D.



Fonte: elaborado pelo autor.

Dessa forma,  $x_{i+1} = a + (i - 1)h$ ,  $x_{i+1} = x_i + h$ ,  $h = \frac{b-a}{n-1}$ .

Com isso, temos a aproximação  $u_i \approx u(x_i)$ .

## 1.2 Operador diferença finita

Esse operador pode ser obtido a partir da série de Taulor para as seguintes funções (WIKIPÉDIA, 2018):

$$u(x_{i+1}) = u(x_i) + u'(x_i)h + \frac{u''(x_i)h^2}{2} + o(h^3)$$

$$u(x_{i-1}) = u(x_i) + u'(x_i)h + \frac{u''(x_i)h^2}{2} + o(h^3)$$

Portanto, a derivada primeira pode ser escrita de três formas distintas como uma diferença quociente mais um termo relacionado com  $h$ , obtido ao desprezar-se termos de ordem superior (WIKIPÉDIA, 2018):

$u'(x_i) = \frac{x_{i+1}-x_i}{h} + o(h)$ , que é conhecida como fórmula das diferenças adiantadas,

$u'(x_i) = \frac{x_i-x_{i-1}}{h} + o(h)$ , que é conhecida como fórmula das diferenças atrasadas,

$u'(x_i) = \frac{x_{i+1}-x_{i-1}}{2h} + o(h^2)$ , que é conhecida como fórmula das diferenças centradas.

Também é possível obter derivadas de ordem superior. Por exemplo, a derivada de segunda ordem é obtida a partir de

$$u(x_{i+1}) + u(x_{i-1}) = 2u(x_i) + u''(x_i)(x)h^2 + o(h^4)$$

$$\text{e é dada por } u''(x_i) = \frac{u(x_{i+1})-2f(x)+u(x_{i-1})}{h^2} + o(h^2)$$

### 1.3 Método das diferenças finitas para problemas lineares

A partir das aproximações por diferença-quotiente para derivadas de qualquer ordem, é possível transformar equações diferenciais em problemas lineares (WIKIPÉDIA, 2018). Para isso, é necessário ignorar o termo de erro e tornar  $h$  um número muito pequeno o bastante que gere boas aproximações das derivadas.

#### 1.3.1 Resolução de problemas unidimensionais de contorno (PVC1D)

O objetivo é resolver uma equação diferencial do tipo  $u''(x) + p(x)u'(x) + q(x)u(x) = f(x)$ , com  $x$  variando de  $a$  até  $b$ , e:

$u(a) = u_a$  e  $u(b) = u_b$  (condição de valor prescrito), ou

$u'(a) = \sigma_a$  e  $u'(b) = \sigma_b$  (condição de fluo prescrito), ou

$\alpha_a u'(a) + \beta_a u(a) = \gamma_a$  e  $\alpha_b u'(b) + \beta_b u(b) = \gamma_b$  (condição mista).

A equação é aproximada pelo método das diferenças finitas, com um erro de truncamento de ordem  $o(h^2)$ , substituindo-se as derivadas pelas suas representações numéricas, que são dadas por:

$$u'(x) = \frac{u(x_{i+1}) - u(x_{i-1})}{2h}$$

$$u''(x) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}$$

Teremos para  $i = 1 : n$

$$\left( \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \right) + p(x) \left( \frac{u(x_{i+1}) - u(x_{i-1})}{2h} \right) + q(x)u(x) = f(x)$$

Com alguns algebrismos e agrupando termos semelhantes, teremos algo do tipo:

$$a_i = q_i - \frac{2}{h^2}$$

$$b_i = \left( \frac{1}{h^2} \right) - \frac{p_i}{2h}$$

$$c_i = \left( \frac{1}{h^2} \right) + \frac{p_i}{2h}$$

$$b_i u_{i-1} + a_i u_i + c_i u_{i+1} = r_i,$$

$$\begin{bmatrix} a_1 & c_1 \\ b_1 & a_2 & c_2 \\ \dots & & \\ b_i & a_i & c_i \\ \dots & & \\ b_{n-1} & a_{n-1} & c_{n-1} \\ b_n & a_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_i \\ \dots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_i \\ \dots \\ r_{n-1} \\ r_n \end{bmatrix}$$

Nota-se que há valores faltantes na matriz na primeira ( $b_1$ ) e na última linha ( $c_n$ ). Desta forma é necessário fazer o tratamento destes contornos. Após o tratamento, a aproximação para  $u$  será dada pelos pontos que são solução do sistema  $Au = R$ .

### 1.3.2 Tratamento de condição de contorno de valor prescrito

$$u(a) = u_a, u(b) = u_b$$

O valor da função é conhecida em  $x_1 = a$  e/ou  $x_n = b$ , dessa forma o algebrismo é feito no sistema de forma a ter-se essa condição.

$$a_1 = 1, c_1 = 0, r_1 = u_a \text{ e/ou}$$

$$a_n = 1, b_n = 0, r_n = u_b.$$

### 1.3.3 Tratamento de condição de contorno de fluxo prescrito

$$u'(a) = \sigma_a, u'(b) = \sigma_b$$

O tratamento no contorno é feito substituindo os valores de  $u_{i-1}$  e  $u_{i+1}$  por algebrismos feitos na fórmula das diferenças atrasadas e na fórmula das diferenças adiantadas, respectivamente.

$$a_1 = a_1 + b_1, r_1 = r_1 + b_1 h \sigma_a \text{ e/ou}$$

$$a_n = a_n + c_n, r_n = r_n - c_n h \sigma_b.$$

### 1.3.4 Tratamento de condição de contorno mista

$$\alpha_a u'(a) + \beta_a u(a) = \gamma_a, \alpha_b u'(b) + \beta_b u(b) = \gamma_b$$

O travamento é feito análogo ao caso anterior.

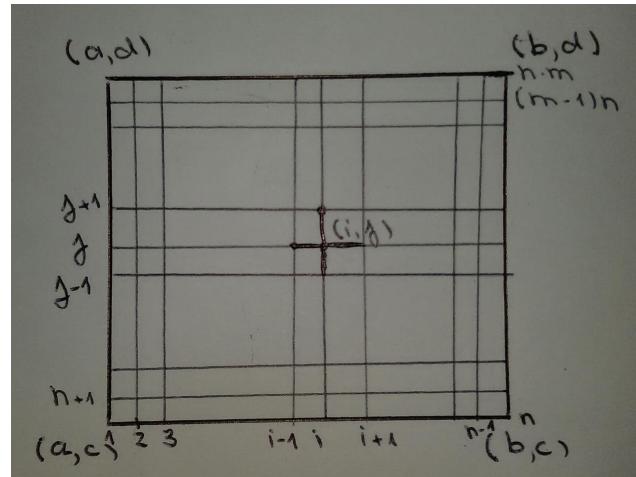
$$a_1 = a_1 + b_1 \left(1 + h \frac{\beta_a}{\alpha_a}\right), r_1 = r_1 + b_1 h \frac{\gamma_a}{\alpha_a} \text{ e/ou}$$

$$a_n = a_n + c_n \left(1 - h \frac{\beta_b}{\alpha_b}\right), r_n = r_n - c_n h \frac{\gamma_b}{\alpha_b}.$$

### 1.3.5 Resolução de problemas bidimensionais de contorno (PVC2D)

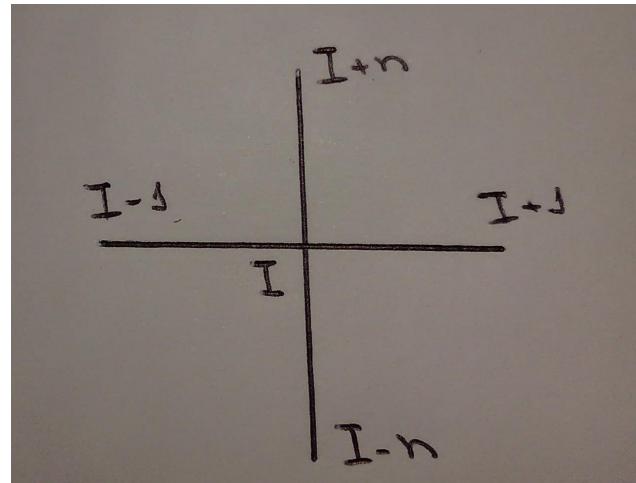
De forma a representar o problema de maneira linear e poder escrever como um sistema linear de matrizes, pode-se rearranjar os pontos de forma a estarem todos situados em uma única dimensão, em uma ordem pré definida. Uma ordem já conhecida de arrumar os pontos é a lexicográfica (CATABRIGA, 12 aug. 2019, 16 dec. 2019), onde obtém-se as seguintes discretizações:

Figura 2 – Discretização de domínio 2D.



Fonte: elaborado pelo autor.

Figura 3 – Ordem lexicográfica.



Fonte: elaborado pelo autor.

Supor  $\kappa, \beta_x(x, y), \beta_y(x, y), \gamma(x, y), g(x, y), h(x, y), f(x, y)$  conhecidas, encontrar  $u(x, y)$  tal que:

$$-\kappa\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \beta_x\left(\frac{\partial u}{\partial x}\right) + \beta_y\left(\frac{\partial u}{\partial y}\right) + \gamma u = f$$

Com condições de contorno conhecidas.

Aproximando as derivadas parciais com  $i = 1 : n$ ,  $j = 1 : m$  e  $I = n * n$ ,

$$\begin{aligned}\frac{\partial u}{\partial x}(x_i, y_j) &\approx \frac{u_{I+1} - u_{I-1}}{2h_x} \\ \frac{\partial u}{\partial y}(x_i, y_j) &\approx \frac{u_{I+1} - u_{I-1}}{2h_y} \\ \frac{\partial^2 u}{\partial x^2}(x_i, y_j) &\approx \frac{u_{I+1} - 2u_I + u_{I-1}}{h_x^2} \\ \frac{\partial^2 u}{\partial y^2}(x_i, y_j) &\approx \frac{u_{I+1} - 2u_I + u_{I-1}}{h_y^2}\end{aligned}$$

E substituindo

$$-\kappa \left( \frac{u_{I+1} - 2u_I + u_{I-1}}{h_x^2} + \frac{u_{I+1} - 2u_I + u_{I-1}}{h_y^2} \right) + \beta_x \left( \frac{u_{I+1} - u_{I-1}}{2h_x} \right) + \beta_y \left( \frac{u_{I+1} - u_{I-1}}{2h_y} \right) + \gamma u = f$$

Que com alguns algebrismos e agrupando termos semelhantes, teremos algo do tipo:

$$a_I = \gamma_I + 2\kappa \left( \frac{1}{h_x^2} + \frac{1}{h_y^2} \right)$$

$$b_I = \frac{-\kappa}{h_x^2} - \frac{\beta_x I}{2h_x}$$

$$c_I = \frac{-\kappa}{h_x^2} + \frac{\beta_x I}{2h_x}$$

$$d_I = \frac{-\kappa}{h_y^2} - \frac{\beta_y I}{2h_y}$$

$$e_I = \frac{-\kappa}{h_y^2} + \frac{\beta_y I}{2h_y}$$

$$\left[ \begin{array}{cccccc} a_1 & c_1 & & e_1 & & & \\ b_2 & a_2 & c_2 & & e_2 & & \\ & & \dots & & & & \\ d_{n+1} & & b_{n+1} & a_{n+1} & c_{n+1} & & e_{n+1} \\ & & & & \dots & & \\ & d_I & & b_I & a_I & c_I & e_I \\ & & & & & \dots & \\ & & d_{N-1} & & b_{N-1} & a_{N-1} & c_{N-1} \\ & & & d_N & & b_N & a_N \end{array} \right] \left[ \begin{array}{c} u_1 \\ u_2 \\ \dots \\ u_{n+1} \\ \dots \\ u_I \\ \dots \\ u_{N-1} \\ u_N \end{array} \right] = \left[ \begin{array}{c} r_1 \\ r_2 \\ \dots \\ r_{n+1} \\ \dots \\ r_I \\ \dots \\ r_{N-1} \\ r_N \end{array} \right]$$

Novamente, é necessário que haja tratamento nos contornos do retângulo discretizado. Isto será feito de maneira análoga ao caso de PVC unidimensional, diferindo apenas na quantidade de pontos a serem discretizados.

### 1.3.6 Tratamento de condição de contorno de valor prescrito

$$u(x_i, y_j) = g_I$$

O valor da função é conhecida em  $(x_i, y_j)$ , dessa forma o algebrismo é feito no sistema de forma a ter-se essa condição.

$$a_I = 1, b_I = 0, c_I = 0, d_I = 0, e_I = 0, r_I = g_I$$

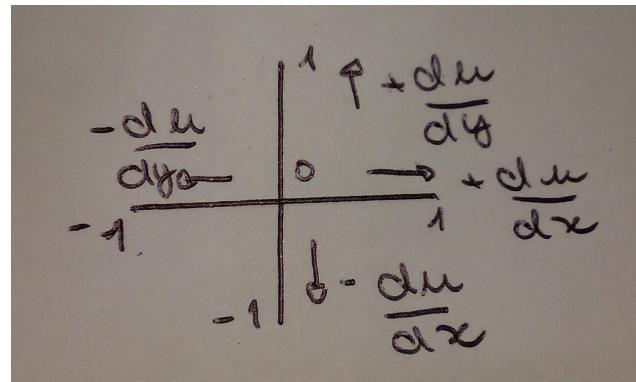
### 1.3.7 Tratamento de condição de contorno de fluxo prescrito

$$\frac{\partial u}{\partial \mathbf{n}} = \sigma_I$$

Com  $\mathbf{n}$  a normal exterior.

O tratamento no contorno é feito substituindo os valores de  $u_{i-n}, u_{i-1}, u_{i+1}, u_{i+n}$  por algebrismos feitos na fórmula das diferenças finitas, observando o seguinte comportamento:

Figura 4 – Comportamento de condição de contorno.



Fonte: elaborado pelo autor.

- **Left:**  $a_I = b_I, b_I = 0, f_i = f_i + b_i \frac{h_x}{\kappa} \sigma_I;$
- **Right:**  $a_I = c_I, c_I = 0, f_i = f_i + c_i \frac{h_x}{\kappa} \sigma_I;$
- **Bottom:**  $a_I = d_I, d_I = 0, f_i = f_i + d_i \frac{h_y}{\kappa} \sigma_I;$
- **Top:**  $a_I = e_I, e_I = 0, f_i = f_i + e_i \frac{h_y}{\kappa} \sigma_I;$

### 1.3.8 Tratamento de condição de contorno mista

- **Left:**  $a_I = b_I(1 - \frac{h_x \beta_q}{\alpha_q}), b_I = 0, f_i = f_i + b_i \frac{h_x q_I}{\alpha_q};$
- **Right:**  $a_I = c_I(1 - \frac{h_x \beta_q}{\alpha_q}), c_I = 0, f_i = f_i + c_i \frac{h_x q_I}{\alpha_q};$
- **Bottom:**  $a_I = d_I(1 - \frac{h_x \beta_q}{\alpha_q}), d_I = 0, f_i = f_i + d_i \frac{h_x q_I}{\alpha_q};$
- **Top:**  $a_I = e_I(1 - \frac{h_x \beta_q}{\alpha_q}), e_I = 0, f_i = f_i + e_i \frac{h_x q_I}{\alpha_q}.$

# Parte II

## Implementações

## 2 Implementações

Todos os códigos aqui mencionados serão enviados juntamente com este relatório. Para a execução dos testes, basta executar a função principal do arquivo equivalente. Por exemplo, para executar a Aplicação 1, basta fazer a chamada  $[u, tSOR, tDireto] = aplicacao1(tipo, w)$ , onde *tipo* refere-se ao tamanho da discretização definida e *w* ao valor de  $\omega$ .

### 2.1 Aplicação

Nestes, são definidos os termos que compõe a equação diferencial, bem como os valores e tipos de condição de contorno. Para tanto, em todos os termos construiu-se vetores linha ou coluna, de tamanho  $n * m$ , que contivesse os valores necessários. Ressalta-se que nos vetores de valores da condição de contorno, este é preenchido com zeros nas posições desnecessárias e com o valor do tratamento de contorno nas posições onde o mesmo será utilizado. Construiu-se dessa forma para que na função de tratamento das condições de contorno estes o índice desses vetores acompanhasse o índice dos elementos dentro de cada linha da matriz do modelo.

#### 2.1.1 Validação 1

Problema simples com solução trivial.

```

1 bx = repmat(0.0,1,n*m);
2 by = repmat(0.0,1,n*m);
3 gamma = repmat(0.0,1,n*m);
4 f = repmat(0.0,1,n*m);
5
6 kappa = 1;
7
8 left = 1;
9 right = 1;
10 bottom = 1;
11 top = 1;
12
13 gleft = repmat(10,n*m,1);
14 gright = repmat(10,n*m,1);
15 gbottom = repmat(10,n*m,1);
16 gtop = repmat(10,n*m,1);
```

#### 2.1.2 Validação 2

Problema com solução conhecida. Para o cálculo do erro, utilizou-se a saída do método SOR.

Devido a complexidade da função, nesta utilizou-se softwares de cálculo simbólico na obtenção das derivadas, com uma precisão de seis casas decimais<sup>2</sup>.

$$u(x, y) = 10xy(1 - x)(1 - y)e^{x^{4.5}}$$

$$\frac{\partial u}{\partial x} = -45e^{x^{4.5}}(-x^{5.5} + x^{4.5} - 0.444444x + 0.222222)(y - 1)y$$

$$\frac{\partial u}{\partial y} = 10e^{x^{4.5}}(x - 1)x(2y - 1)$$

$$\frac{\partial^2 u}{\partial x^2} = 337.5(e^{x^{4.5}}(x^{4.5} - 0.733333x^{3.5} + 0.6(x^9) - 0.6(x^8) + 0.059259)(y - 1)y$$

$$\frac{\partial^2 u}{\partial y^2} = 20e^{x^{4.5}}(x - 1)x$$

```

1 I = 1;
2 for j = 1:m
3   for i = 1:n
4     bx(I) = 1.0;
5     by(I) = 20*y(j);
6     gamma(I) = 1.0;
7     f(I) = -((337.5*(exp(x(i)^4.5))*(x(i)^4.5 - 0.733333*x(i)^3.5 + 0.6*(x(i)^9) - 0.6*(x(i)^8) + 0.059259)*(y(j)-1)*y(j)) + (20*(exp(x(i)^4.5))*(x(i)-1)*x(i)) + bx(I)*(-45*(exp(x(i)^4.5))*(-x(i)^5.5 + x(i)^4.5 - 0.444444*x(i) + 0.222222)*(y(j)-1)*y(j)) + by(I)*(10*(exp(x(i)^4.5))*(x(i)-1)*x(i)*(2*y(j)-1)) + gamma(I)*(10*x(i)*y(j)*(1-x(i))*(1-y(j))*exp(x(i)^4.5)));
8     I++;
9   endfor
10 endfor
11
12 kappa = 1.0;
13
14 left = 1;
15 right = 1;
16 bottom = 1;
17 top = 1;
18
19 gleft = repmat(0,n*m,1);
20 gright = repmat(0,n*m,1);
21 gbottom = repmat(0,n*m,1);
22 gtop = repmat(0,n*m,1);

```

### 2.1.3 Validação 3

Problema com solução conhecida.

---

<sup>2</sup> <https://www.wolframalpha.com/widgets/view.jsp?id=f6279bc11dc9e954e5f1093156ab7f0>

```

1 bx = repmat(0.0,1,n*m);
2 by = repmat(0.0,1,n*m);
3 gamma = repmat(0.0,1,n*m);
4 f = repmat(0.0,1,n*m);
5
6 kappa = 1.0;
7
8 left = 1;
9 right = 2;
10 bottom = 2;
11 top = 1;
12
13 gleft = repmat(0,n*m,1);
14 j = 1;
15 for I = (0:m-1)*n+1
16     gleft(I,1) = 1 - y(j)^2;
17     j++;
18 endfor
19 gright = repmat(2,n*m,1);
20 gbottom = repmat(2,n*m,1);
21 gtop = repmat(0,n*m,1);
22 i = 1;
23 for I = ((m-1)*n+1):(m*n)
24     gtop(I,1) = x(i)^2 - 1;
25     i++;
26 endfor

```

#### 2.1.4 Aplicação 1

Resfriador bidimensional. Exemplos podem incluir o resfriamento de chips de computadores ou amplificadores elétricos.

```

1 bx = repmat(0,1,n*m);
2 by = repmat(0,1,n*m);
3 gamma = repmat(1,1,n*m);
4 f = repmat(70,1,n*m);
5
6 kappa = 1;
7
8 left = 1;
9 right = 3;
10 bottom = 1;
11 top = 1;
12
13 gleft = repmat(200,n*m,1);

```

```

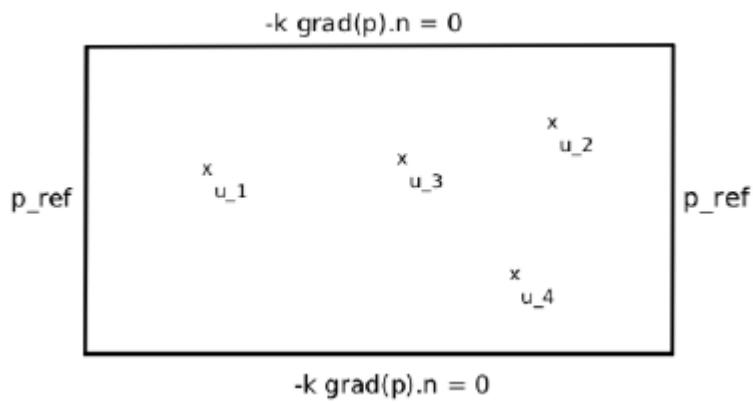
14 gright = [ repmat(1 ,n*m,1) , repmat(1 ,n*m,1) , repmat(70 ,n*m,1) ];
15 gbottom = repmat(70 ,n*m,1);
16 gtop = repmat(70 ,n*m,1);

```

### 2.1.5 Aplicação 2

Escoamento em águas subterrâneas, considerando um meio poroso superficial saturado retangular no plano  $xy$  com dois poços. Nas faces superior e inferior do retângulo assume-se que não existe fluxo na direção do contorno, porém, considerando um amplo abastecimento das fronteiras esquerda e direita de tal forma que a pressão seja conhecida.

Figura 5 – Esquema do Escoamento em águas subterrâneas - Exemplo com 4 poços.



Fonte: especificação do relatório.

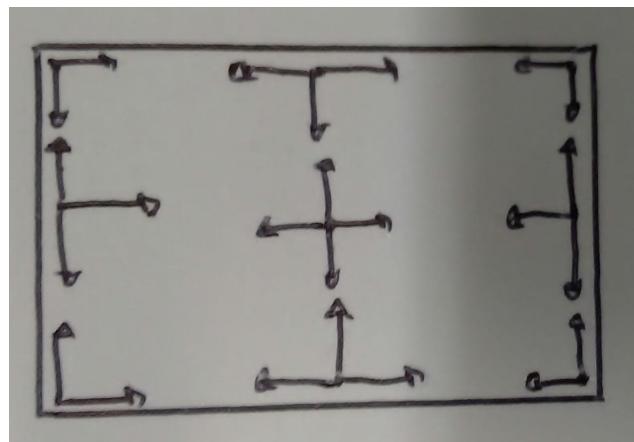
```

1 bx = repmat(0 ,1 ,n*m);
2 by = repmat(0 ,1 ,n*m);
3 gamma = repmat(0 ,1 ,n*m);
4 f = repmat(0 ,1 ,n*m);
5 %Localizacao dos poços
6 f(1 , 600*m/1000 + n*(1500*n/5000 - 1)) = -250;
7 f(1 , 250*m/1000 + n*(3200*n/5000 - 1)) = -250;
8
9 kappa = 1;
10
11 left = 1;
12 right = 1;
13 bottom = 2;
14 top = 2;
15
16 gleft = repmat(100 ,n*m,1);
17 gright = repmat(100 ,n*m,1);
18 gbottom = repmat(0 ,n*m,1);
19 gtop = repmat(0 ,n*m,1);

```

Já para o cálculo da velocidade, utilizou-se a definição dada na descrição, ao qual é necessário a obtenção das derivadas parciais da solução (valores de pressão). Dessa forma, utilizou-se das aproximações por diferença finitas que gerassem menor erro. Disto, procurou-se utilizar sempre diferenças centrais. Entretanto, nos contornos isto não é possível, uma vez que não consegue-se analisar passos a frente, já que os mesmos não existem. Assim, utilizou-se as diferenças finitas (adiantadas, centrais e atrasadas) seguindo o esquema representado na Figura 6, onde o tipo de diferença finita utilizado é o equivalente ao representado. Por exemplo, em toda a borda esquerda utiliza-se de diferenças finitas centrais para o cálculo de  $v_y$  e diferenças adiantadas para  $v_x$ .

Figura 6 – Contornos.

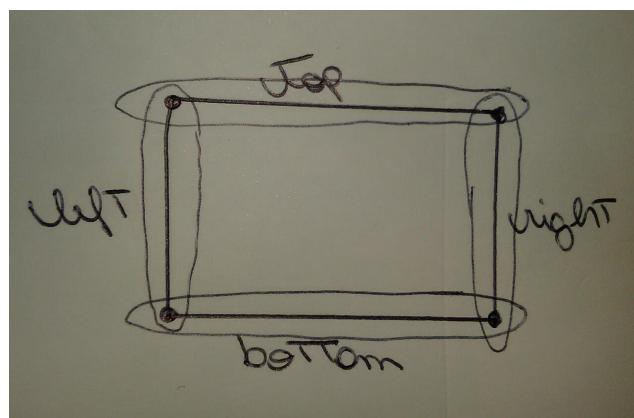


Fonte: elaborado pelo autor.

## 2.2 Tratamento do contorno

O tratamento do contorno foi construído seguindo o referencial teórico descrito anteriormente, sendo os contornos da seguinte forma:

Figura 7 – Contornos.



Fonte: elaborado pelo autor.

Nota-se que os cantos seriam tratados duas vezes cada um, o que é correto para condições

de fluxo prescrito e condições mistas. Se um determinado ponto, por exemplo  $(a, c)$ , possui um vetor de fluxo apontando para baixo e um vetor de fluxo apontando para a esquerda, estes devem ser tratados ambos neste ponto. Contudo, condições com valor prescrito possuem prioridade dentre os demais, já que representa a solução procurada naquele ponto. Se caso no exemplo acima conhecesse o valor exato da função no canto, este deveria ser utilizado. Se em ambos os contornos, esquerda e embaixo, conhecesse o valor exato da função, este deveria ser o mesmo no ponto acima citado. Dessa forma, a implementação para o caso *left*, por exemplo, deu-se conforme segue a seguir, com as demais seguindo o mesmo princípio.

```

1 % Condicoes de contorno para fluxo prescrito e condicao mista
2 % Condicoes de contorno left
3 if (left == 2)
4     for I = (0:m-1)*n+1
5         a(I) = a(I) + b(I);
6         f(I) = f(I) + b(I)*hx*gleft(I,1)/kappa;
7         b(I) = 0;
8     endfor
9 elseif (left == 3)
10    for I = (0:m-1)*n+1
11        a(I) = a(I) + b(I)*(1-(hx*gright(I,2)/gright(I,1)));
12        f(I) = f(I) + b(I)*hx*gleft(I,3)/gleft(I,1);
13        b(I) = 0;
14    endfor
15 endif

```

```

1 % Condicoes de contorno para valor prescrito
2 % Condicoes de contorno left
3 if (left == 1)
4     for I = (0:m-1)*n+1
5         b(I) = c(I) = d(I) = e(I) = 0;
6         a(I) = 1;
7         f(I) = gleft(I,1);
8     endfor
9 endif

```

Onde os valores das condições conhecidas encontram-se em vetores colunas, e no caso da condição mista, em um vetor tri-coluna.

## 2.3 Sistema linear

Para a construção do sistema linear, trabalhou-se sobre as diagonais definidas das matrizes representando a modelagem. Para a solução do método direto, primeiro constrói-se a matriz a

partir destes vetores, para então solucioná-la.

```

1 % diagonal a_I
2 for I = 1:N
3     A(I,I) = a(I);
4     f(I) = fun(I);
5 endfor
6
7 % diagonal b_I e c_I
8 for I = 1:(N-1)
9     A(I+1,I) = b(I+1);
10    A(I,I+1) = c(I);
11 endfor
12
13 % diagonal d_I e e_I
14 for I = 1:(N-n)
15     A(I+n,I) = d(I+n);
16     A(I,I+n) = e(I);
17 endfor

```

### 2.3.1 Tempo de execução

Conforme foi requisitado, foi computado o tempo de execução pelo método SOR e pelo método direto para comparação. Para isso utilizou-se as funções `tic()` e `toc()`. Ressalta-se que o tempo de construção da matriz não é computado. A seguir é exemplificado.

```

1 [A,f] = sistema_linear(ai,bi,ci,di,ei,f,n,m);
2
3 tDireto = tic();
4 u = A\f;
5 tDireto = toc(tDireto);
6
7 tSOR = tic();
8 [u,iter,er] = sor(ai,bi,ci,di,ei,f,n,m,w,0.00001,1000);
9 tSOR = toc(tSOR);

```

## 2.4 Método SOR

Uma variante do método de Gauss-Seidel, o método SOR (*successive over relaxation*) possui um parâmetro de relaxação  $\omega$  definido em  $(0, 2)$ . Sua expressão geral livre de matriz e considerando os termos  $a_I, b_I, c_I, d_I, e_I$  construídos é

$$u_I^{k+1} = \frac{\omega}{a_I} (f_I - d_I u_{I-n}^{k+1} - b_I u_{I-1}^{k+1} - c_I u_{I+1}^k - e_I u_{I+n}^k) + (1 - \omega) u_I^k$$

Como a nossa matriz não possui todos os termos definidos em todas as posições, por exemplo,  $b_1$  não está definido na primeira linha, é necessário que isto seja respeitado durante a implementação do método. Disto, foram utilizados seis vetores adicionais, cada um representando um termo do método SOR, e seus cálculos foram feitos respeitando sua existência na matriz. Para isso, dividiu-se a matriz do modelo em cinco blocos, considerando o *lag* (por exemplo  $b_1$  relaciona-se com  $x_{I-1}$ ) de cada termo da equação do modelo.

$$\left[ \begin{array}{ccccc} a_1 & c_1 & & e_1 & \\ b_2 & a_2 & c_2 & & e_2 \\ & & \dots & & \\ d_{n+1} & & b_{n+1} & a_{n+1} & c_{n+1} & e_{n+1} \\ & & & & \dots & \\ & & d_I & b_I & a_I & c_I & e_I \\ & & & & & \dots & \\ & & & d_{N-1} & b_{N-1} & a_{N-1} & c_{N-1} \\ & & & & d_N & b_N & a_N \end{array} \right]$$

```

1  while ((iter <= maxiter) && (tol <= er(iter)))
2      x0 = x;
3
4      I = 1;
5      A(I) = a(I);
6      C(I) = c(I)*x0(I+1);
7      E(I) = e(I)*x0(I+n);
8      F(I) = f(I);
9      for I = 2:n
10         B(I) = b(I)*(x(I-1));
11         A(I) = a(I);
12         C(I) = c(I)*x0(I+1);
13         E(I) = e(I)*x0(I+n);
14         F(I) = f(I);
15     endfor
16     for I = (n+1):((m-1)*n)
17         D(I) = d(I)*x0(I-n);
18         B(I) = b(I)*(x(I-1));
19         A(I) = a(I);
20         C(I) = c(I)*x0(I+1);
21         E(I) = e(I)*x0(I+n);
22         F(I) = f(I);
23     endfor

```

```
24  for I = ((m-1)*n+1):(m*n-1)
25      D(I) = d(I)*x0(I-n);
26      B(I) = b(I)*(x(I-1));
27      A(I) = a(I);
28      C(I) = c(I)*x0(I+1);
29      F(I) = f(I);
30  endfor
31  I = n*m;
32  D(I) = d(I)*x0(I-n);
33  B(I) = b(I)*(x(I-1));
34  A(I) = a(I);
35  F(I) = f(I);
36
37  x = w./A.* (F - D - B - C - E + (1-w)).*x0;
38
39  iter++;
40  er(iter) = norm(x-x0, inf)/norm(x, inf);
41 endwhile
42 iter--;
```

# **Parte III**

## **Resultados**

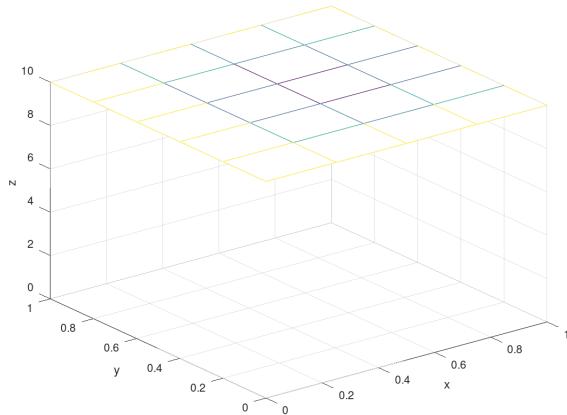
### 3 Experimentos numéricos

Em todos os testes realizados foram definidos diferentes tamanhos de discretização e em seguida, o sistema linear resultante foi calculado pelo método SOR com  $\omega = 0.25, 0.5, 0.75, 1, 1.25$ , tolerância  $tol = 10^{-5}$  e número máximo de iterações  $maxit = 1000$ . Esse número se mostrou pequeno nos testes de maior porte, entretanto, de forma a padronizar as comparações e conseguir observar bem a influência dos parâmetros, foi mantido esse valor de 1000. Apenas por curiosidade que serão apresentados resultados com mais iterações. Os gráficos apresentados referem-se apenas aos obtidos pelo método SOR, com o valor de  $\omega$  correspondente.

#### 3.1 Validação 1

Nesta validação espera-se que os valores no interior da placa sejam iguais ao valor escolhido ( $u_c = 10$ ) em todos os pontos da discretização. Fixando o eixo  $z$  e plotando a solução, nota-se através da Figura 8 que a mesma se comporta como o esperado.

Figura 8 – Validação 1 |  $n = 5, m = 6 | \omega = 0.25$ .

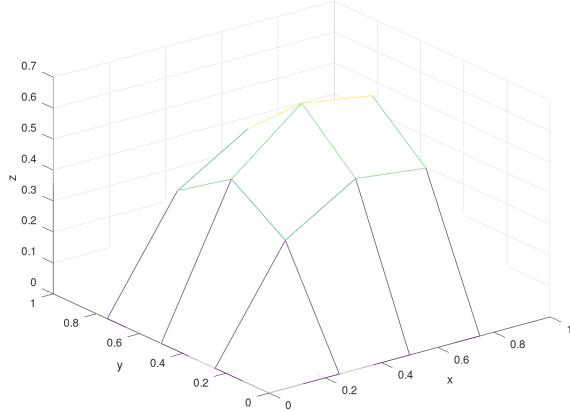


Fonte: elaborado pelo autor.

#### 3.2 Validação 2

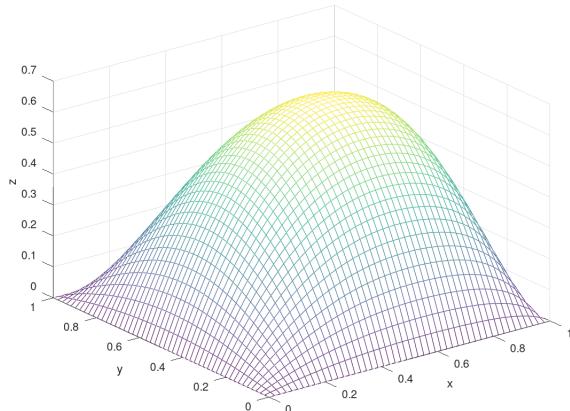
As Figuras 9 e 10 apresentam o gráfico da função solução real desta validação com malhas de diferentes espessuras.

Figura 9 –  $u(x, y) \mid n = 5, m = 5$ .



Fonte: elaborado pelo autor.

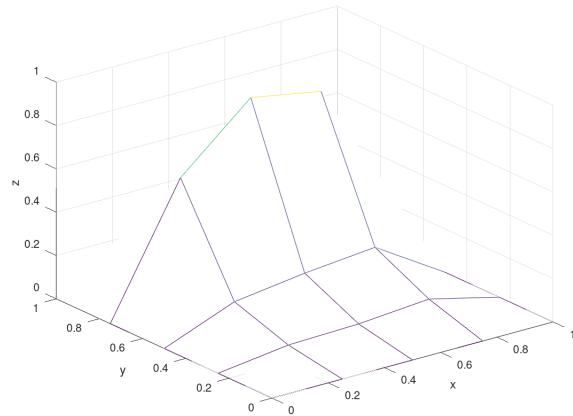
Figura 10 –  $u(x, y) \mid n = 50, m = 50$ .



Fonte: elaborado pelo autor.

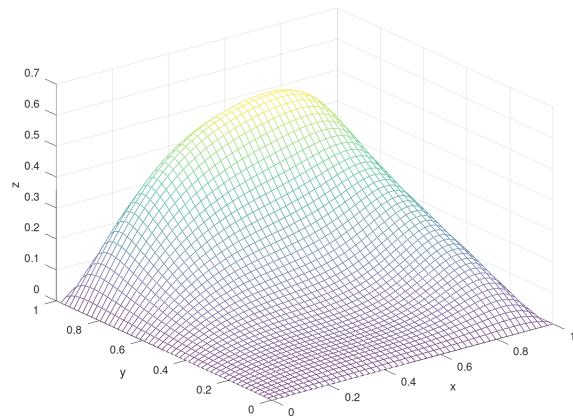
As Figuras 11 e 12, 13, 14, 15 apresentam o gráfico da solução obtida para discretizações correspondentes. Visualmente, percebe-se que quanto mais fina a malha mais o método aproxima sua solução à função, porém sempre com a “barriguinha” mais alongada e inclinada. Atrelado à esse fato está a variação de  $\omega$ , que quanto mais baixo, mais suave é o pico da curva. Quando compara-se as Figuras 12, 13, 14, 15 este fato fica evidente. Iniciando na Figura 12, com  $\omega = 0.25$  nota-se que o pico da curva vai se “afinando”, até a Figura 15, com  $\omega = 1$ .

Figura 11 – Validação 2 |  $n = 5, m = 5 | \omega = 0.75.$



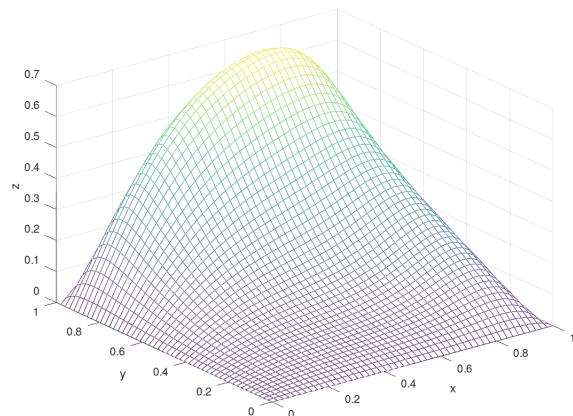
Fonte: elaborado pelo autor.

Figura 12 – Validação 2 |  $n = 50, m = 50 | \omega = 0.25.$



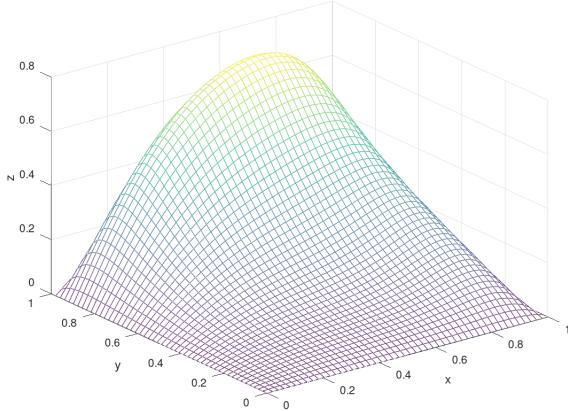
Fonte: elaborado pelo autor.

Figura 13 – Validação 2 |  $n = 50, m = 50 | \omega = 0.5.$



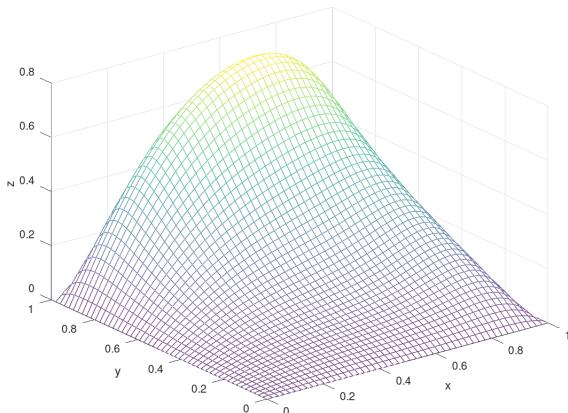
Fonte: elaborado pelo autor.

Figura 14 – Validação 2 |  $n = 50, m = 50 | \omega = 0.75.$



Fonte: elaborado pelo autor.

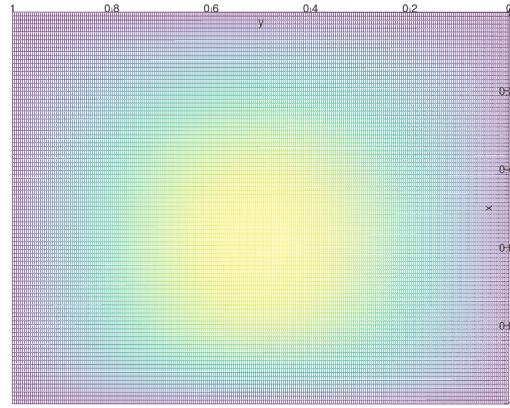
Figura 15 – Validação 2 |  $n = 50, m = 50 | \omega = 1.$



Fonte: elaborado pelo autor.

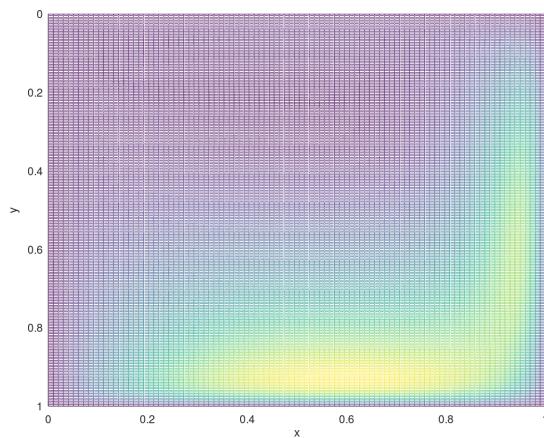
Para analisar melhor o comportamento da solução encontrada para a discretização analisada anteriormente, pode-se observar os gráficos de altura da solução em relação ao eixo  $z$ . As Figuras 16 (solução real) e 17 (solução aproximada) demonstram esta análise, onde cores mais frias representam níveis mais inferiores e cores mais quentes representam pontos de maior altura. Considerou-se importante esta análise gráfica pois através dela nota-se os locais de formação dos picos da função real e demonstra como a função aproximada está afastada.

Figura 16 – Validação 2 | altura alcançada pela solução real.



Fonte: elaborado pelo autor.

Figura 17 – Validação 2 | altura alcançada pela solução aproximada.



Fonte: elaborado pelo autor.

As Tabelas 1, 2 apresentam comparações entre os tempos de execução do método direto e do método SOR com uma precisão máxima de  $10^{-5}$ . Para  $\omega = 1.25$  não houve convergência do método. Para o cálculo do erro, utilizou-se a saída do método SOR, uma vez que a ordem de aproximação do erro cometido é bem conhecida.

n	5	50				
m	5	50				
w	método SOR	método direto	erro	método SOR	método direto	erro
0.25	262.207519	0.014207	0.545894	203.678872	0.011836	0.545894
0.5	206.701549	0.013349	0.488557	201.511168	0.015165	0.488557
0.75	207.549342	0.011830	0.509989	205.112767	0.011832	0.509989
1	207.992240	0.011929	0.529494	202.140698	0.011224	0.529494
1.25	207.289193	0.026444	Inf	202.495039	0.011757	Inf

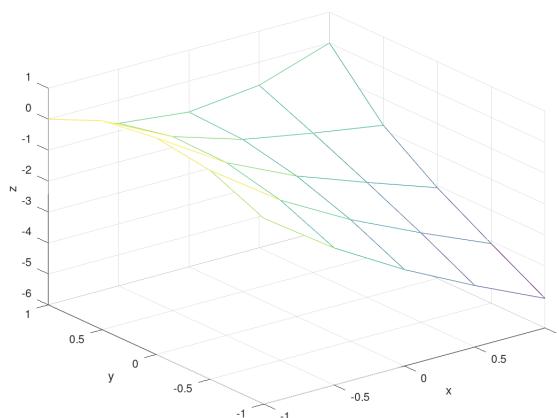
Tabela 1 – Comparação entre tempos de execução e erros gerados | Parte 1

n	100	500				
m	250	500				
w	método SOR	método direto	erro	método SOR	método direto	erro
0.25	2036.553745	0.187513	0.651450	2073.764822	0.197766	0.651450
0.5	2004.708764	0.168345	0.640813	2748.724422	0.260230	0.640813
0.75	1976.053408	0.236113	0.631269	2166.790269	0.197899	0.631269
1	1970.613341	0.184398	0.622407	2125.096263	0.175680	0.622407
1.25	1970.600868	0.184576	Inf	2139.715828	0.176485	Inf

Tabela 2 – Comparação entre tempos de execução e erros gerados | Parte 2

### 3.3 Validação 3

As Figuras 18 e 19, 20, 21, 22 apresentam o gráfico da solução encontrada desta validação com malhas de diferentes espessuras.

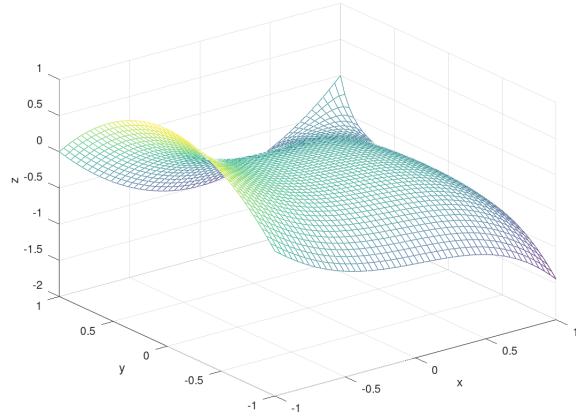
Figura 18 – Validação 3 |  $n = 5, m = 5 | \omega = 0.75$ .

Fonte: elaborado pelo autor.

Através das Figuras 19, 20, 21, 22 nota-se novamente a influência de  $\omega$  na qualidade da solução. Fixando o eixo  $z$  como foi feito nestas Figuras, nota-se que quanto mais aumenta-se  $\omega$

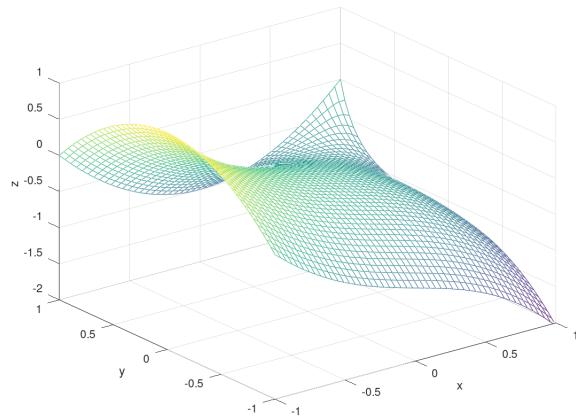
mas a superfície vai “escorrendo” em direção ao ponto  $(1, -1)$ , ponto este que na definição do problema é definido por duas condições de fluxo diferentes, e dessa forma, ambas são aplicadas a este ponto. É muito interessante notar que nos demais pontos, onde é conhecido o valor prescrito da função, não há quaisquer alterações.

Figura 19 – Validação 3 |  $n = 50, m = 50 | \omega = 0.25$ .



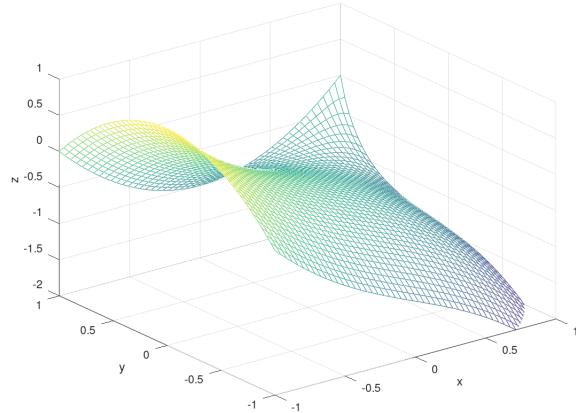
Fonte: elaborado pelo autor.

Figura 20 – Validação 3 |  $n = 50, m = 50 | \omega = 0.5$ .



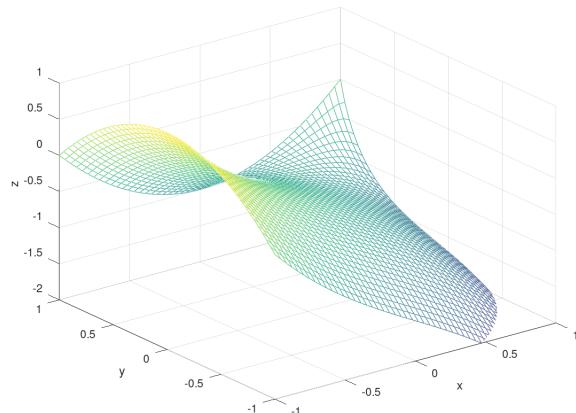
Fonte: elaborado pelo autor.

Figura 21 – Validação 3 |  $n = 50, m = 50 | \omega = 0.75.$



Fonte: elaborado pelo autor.

Figura 22 – Validação 3 |  $n = 50, m = 50 | \omega = 1.$



Fonte: elaborado pelo autor.

As Tabelas 3, 4 apresentam comparações entre os tempos de execução do método direto e do método SOR com uma precisão máxima de  $10^{-5}$ . Para  $\omega = 1.25$  não houve convergência do método.

$n$	5		50	
$m$	5		50	
$w$	método SOR	método direto	método SOR	método direto
0.25	194.848416	0.011403	199.323784	0.011278
0.5	196.612745	0.010046	196.147482	0.011164
0.75	196.924726	0.010035	196.374824	0.014365
1	196.603815	0.012511	196.690715	0.010099
1.25	195.470468	0.011243	197.768886	0.012515

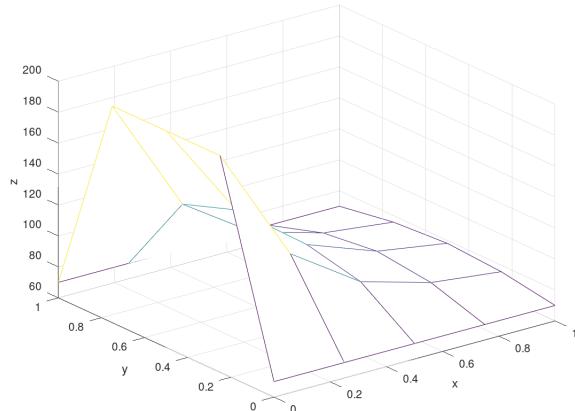
Tabela 3 – Comparação entre tempos de execução | Parte 1

n	100		500	
m	250		500	
w	método SOR	método direto	método SOR	método direto
0.25	1984.254397	0.181108	2122.074753	0.214291
0.5	1977.007107	0.181178	2168.319901	0.191120
0.75	1976.637267	0.224374	2161.171807	0.214462
1	1962.648861	0.181433	2152.066956	0.191425
1.25	1975.119747	0.237636	2450.109464	0.191064

Tabela 4 – Comparaçāo entre tempos de execuçāo | Parte 2

### 3.4 Aplicação 1

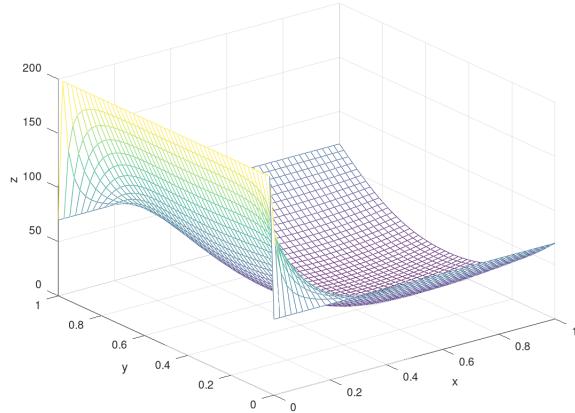
As Figuras 23, 24, 25, 26, 27 apresentam as soluções obtidas com diferentes valores de  $\omega$  e diferentes espessuras de malha.

Figura 23 – Aplicação 1 |  $n = 5, m = 5 | \omega = 0.25$ .

Fonte: elaborado pelo autor.

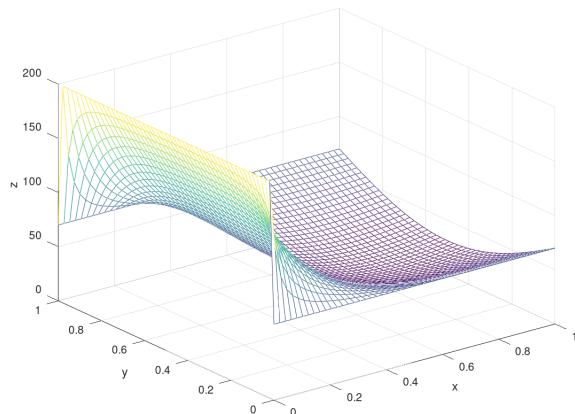
Pode-se notar a influência do valor de  $\omega$  para uma malha de tamanho médio, com  $n = m = 50$ , que é o caso das Figuras 24, 25, 26, 27. Quando observa-se a concavidade da curva de resfriamento (olhando pela “parte debaixo da curva”) nota-se que quanto maior o valor de  $\omega$ , mais rápido aparenta-se que a superfície resfria-se, isto é, mais suave vai ficando esta concavidade.

Figura 24 – Aplicação 1 |  $n = 50, m = 50$  |  $\omega = 0.25$ .



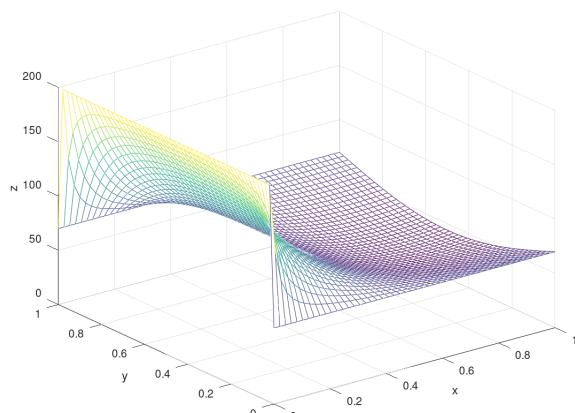
Fonte: elaborado pelo autor.

Figura 25 – Aplicação 1 |  $n = 50, m = 50$  |  $\omega = 0.5$ .



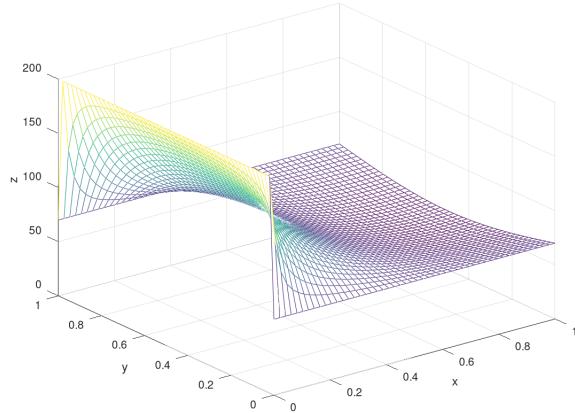
Fonte: elaborado pelo autor.

Figura 26 – Aplicação 1 |  $n = 50, m = 50$  |  $\omega = 0.75$ .



Fonte: elaborado pelo autor.

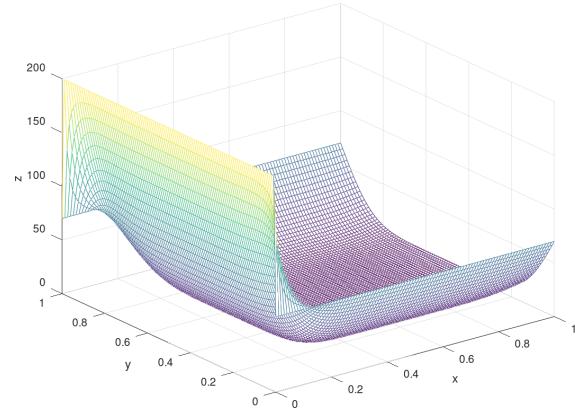
Figura 27 – Aplicação 1 |  $n = 50, m = 50$  |  $\omega = 1$ .



Fonte: elaborado pelo autor.

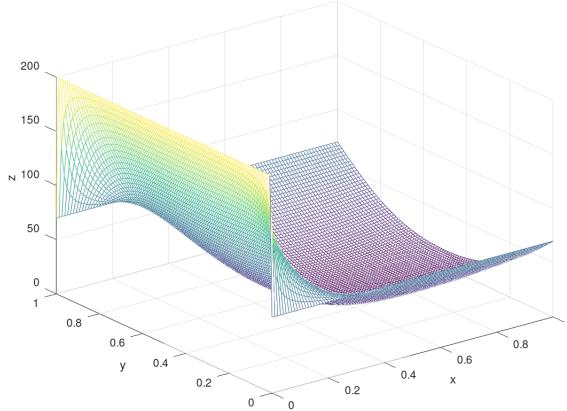
Aumentando a espessura da malha, obtemos as Figuras 28 e 29, a qual a mudança na concavidade da curva está bem aparente.

Figura 28 – Aplicação 1 |  $n = 500, m = 500$  |  $\omega = 0.25$ .



Fonte: elaborado pelo autor.

Figura 29 – Aplicação 1 |  $n = 500, m = 500$  |  $\omega = 1$ .



Fonte: elaborado pelo autor.

As Tabelas 5, 6 apresentam comparações entre os tempos de execução do método direto e do método SOR com uma precisão máxima de  $10^{-5}$ . Para  $\omega = 1.25$  não houve convergência do método.

$n$	5		50	
$m$	5		50	
$w$	método SOR	método direto	método SOR	método direto
0.25	161.614455	0.008977	162.068348	0.008960
0.5	160.699353	0.008999	161.294994	0.009910
0.75	160.856346	0.008964	160.832201	0.009010
1	161.447880	0.009002	161.256741	0.009005
1.25	160.932439	0.009846	161.348681	0.008962

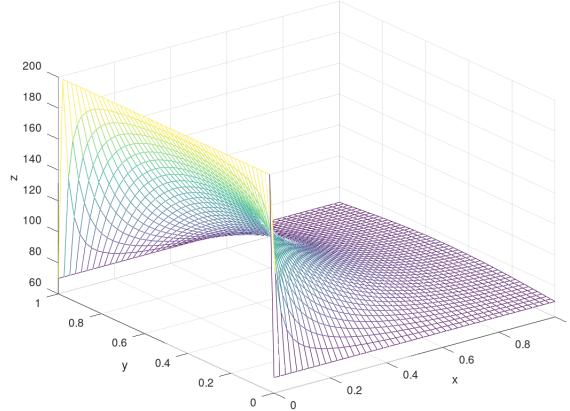
Tabela 5 – Comparaçāo entre tempos de execuāo | Parte 1

$n$	100		500	
$m$	100		500	
$w$	método SOR	método direto	método SOR	método direto
0.25	641.511945	0.047732	1130.134694	0.083242
0.5	639.326784	0.047711	892.002082	0.058133
0.75	640.024007	0.047989	872.894875	0.058152
1	638.880907	0.047659	856.252333	0.066900
1.25	640.215626	0.047647	856.230294	0.074547

Tabela 6 – Comparaçāo entre tempos de execuāo | Parte 1

Com os dados de tempos de execução apresentados nas Tabelas 5, 6 questionou-se que poderia ser que a solução não estava convergindo com a quantidade de iterações dada. Dessa forma, fez-se um teste com 100000 iterações limitando o método SOR, cuja saída gráfica está representada na Figura 30.

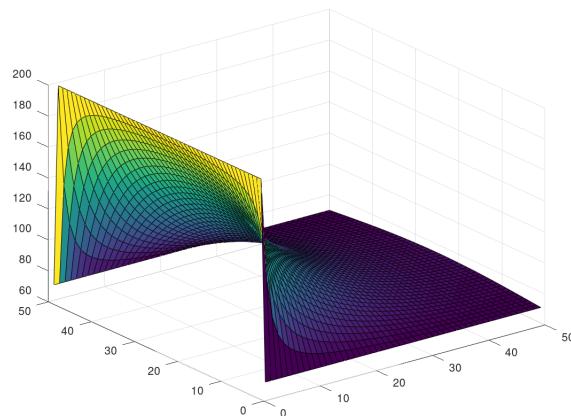
Figura 30 – Aplicação 1 | Superfície |  $n = 50, m = 50$  |  $\omega = 0.25$  | 100000 iterações.



Fonte: elaborado pelo autor.

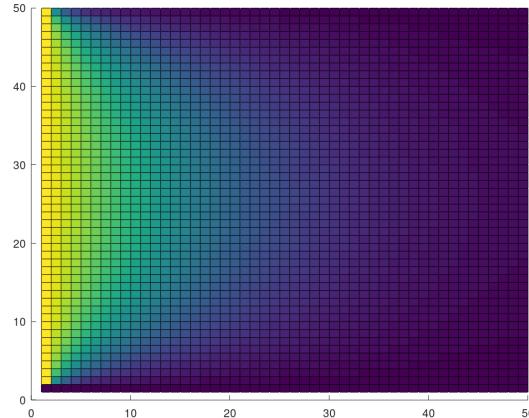
A partir do resultado melhorado e buscando observar o comportamento físico do problema, criou-se o gráfico de superfície do problema. As Figuras 31, 32 apresentam tal caso. Iniciando da Figura 31, temos a mesma representação dos gráficos anteriores, porém com uma forma gráfica melhorada, construída através do comando *surface* do *Octave*. A Figura 32 apresenta uma representação 2D da Figura 31, onde pode-se notar claramente como ocorre o resfriamento da superfície. A borda esquerda é onde está mais aquecido, pois há um valor de temperatura mais alto neste ponto. Conforme move-se em direção à borda direita, a temperatura vai caindo, uma vez que nesta há fluxo de calor bem como uma temperatura mais baixa ( $u_{ref}$ ). E essa temperatura de valor conhecido também é encontrada nas demais bordas, onde também pode-se notar diferenças de temperatura.

Figura 31 – Aplicação 1 | Superfície |  $n = 50, m = 50$  |  $\omega = 0.25$ .



Fonte: elaborado pelo autor.

Figura 32 – Aplicação 1 | Superfície |  $n = 50, m = 50$  |  $\omega = 0.25$ .

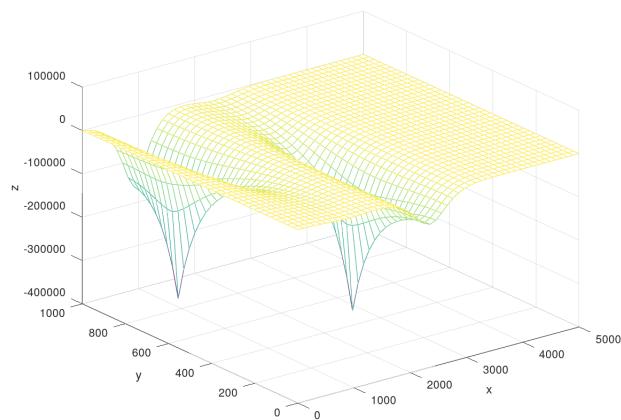


Fonte: elaborado pelo autor.

### 3.5 Aplicação 2

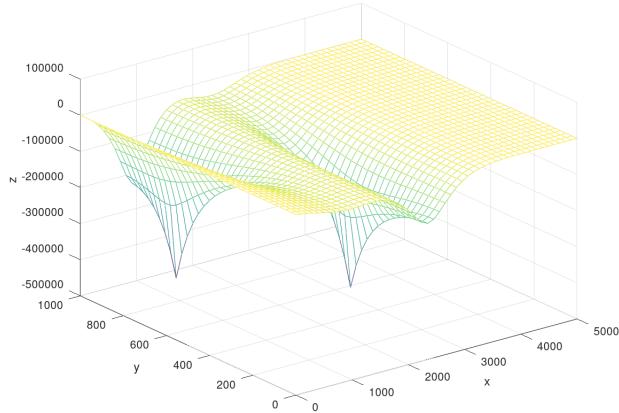
As Figuras 33, 34, 35, 36 apresentam as soluções obtidas com malhas de tamanho pequeno e diferentes valores de  $\omega$ . Novamente, nota-se a influência do valor de  $\omega$  na solução encontrada. Quanto maior o valor de  $\omega$ , mais “ondulada” a paisagem da solução se torna. Pensando no problema físico, quanto maior o valor de  $\omega$ , mais suave seria a variação da pressão, e maior seria os campos de velocidade em torno dos poços. Este fato será melhor apresentado através da análise do campo vetorial da velocidade, nas Figuras 37 e 38.

Figura 33 – Aplicação 2 |  $n = 50, m = 40$  |  $\omega = 0.25$ .



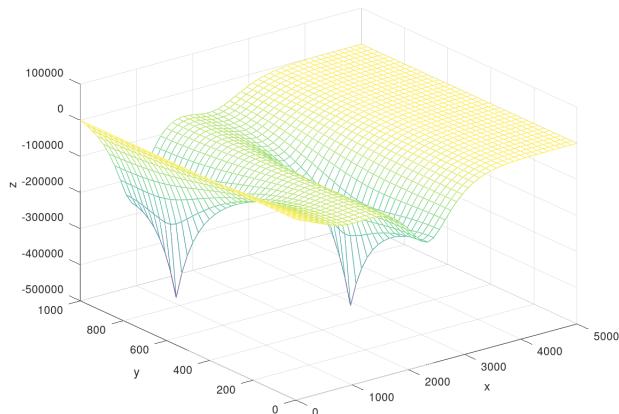
Fonte: elaborado pelo autor.

Figura 34 – Aplicação 2 |  $n = 50, m = 40 | \omega = 0.5.$



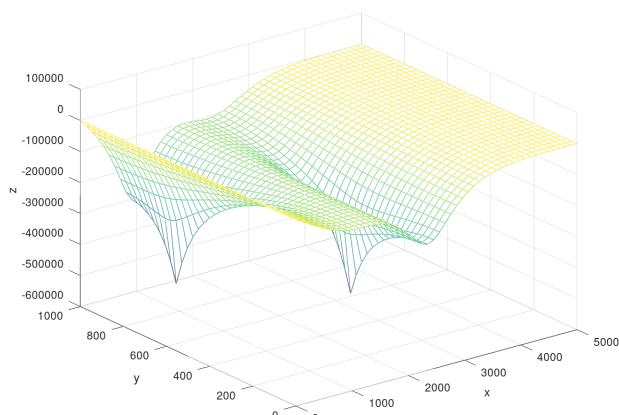
Fonte: elaborado pelo autor.

Figura 35 – Aplicação 2 |  $n = 50, m = 40 | \omega = 0.75.$



Fonte: elaborado pelo autor.

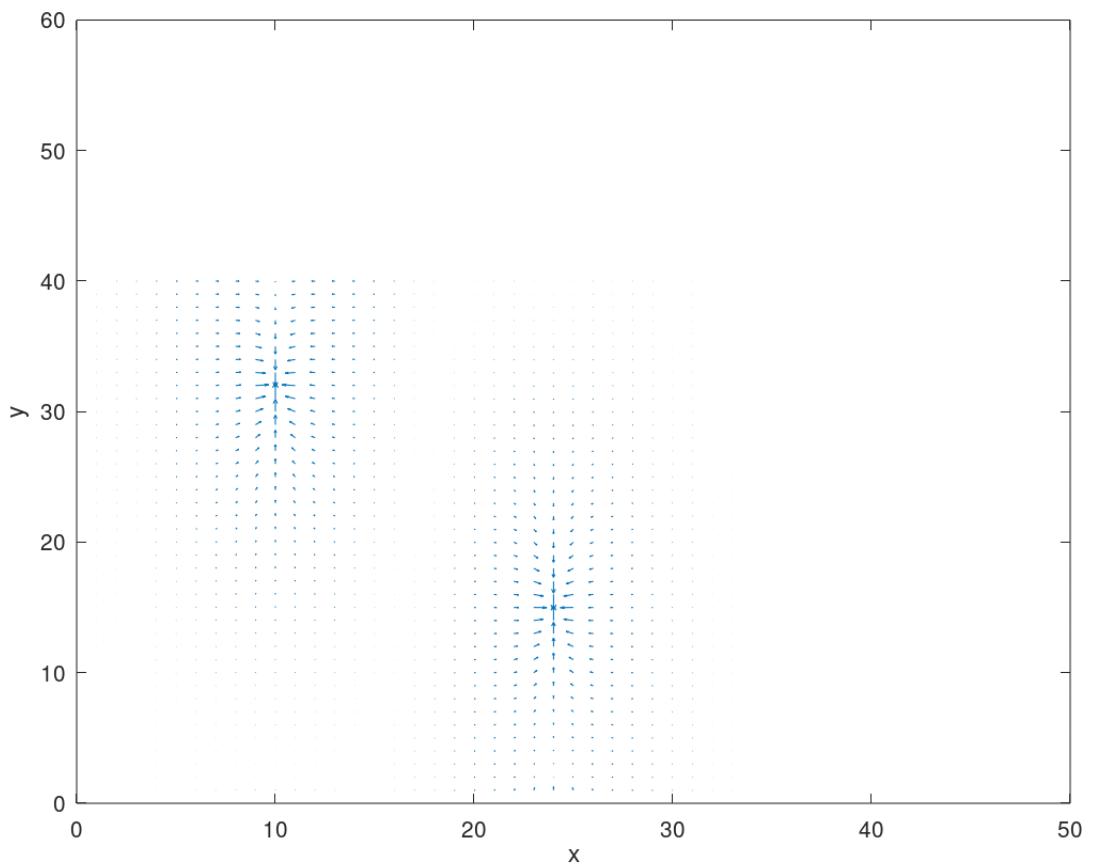
Figura 36 – Aplicação 2 |  $n = 50, m = 40 | \omega = 1.$



Fonte: elaborado pelo autor.

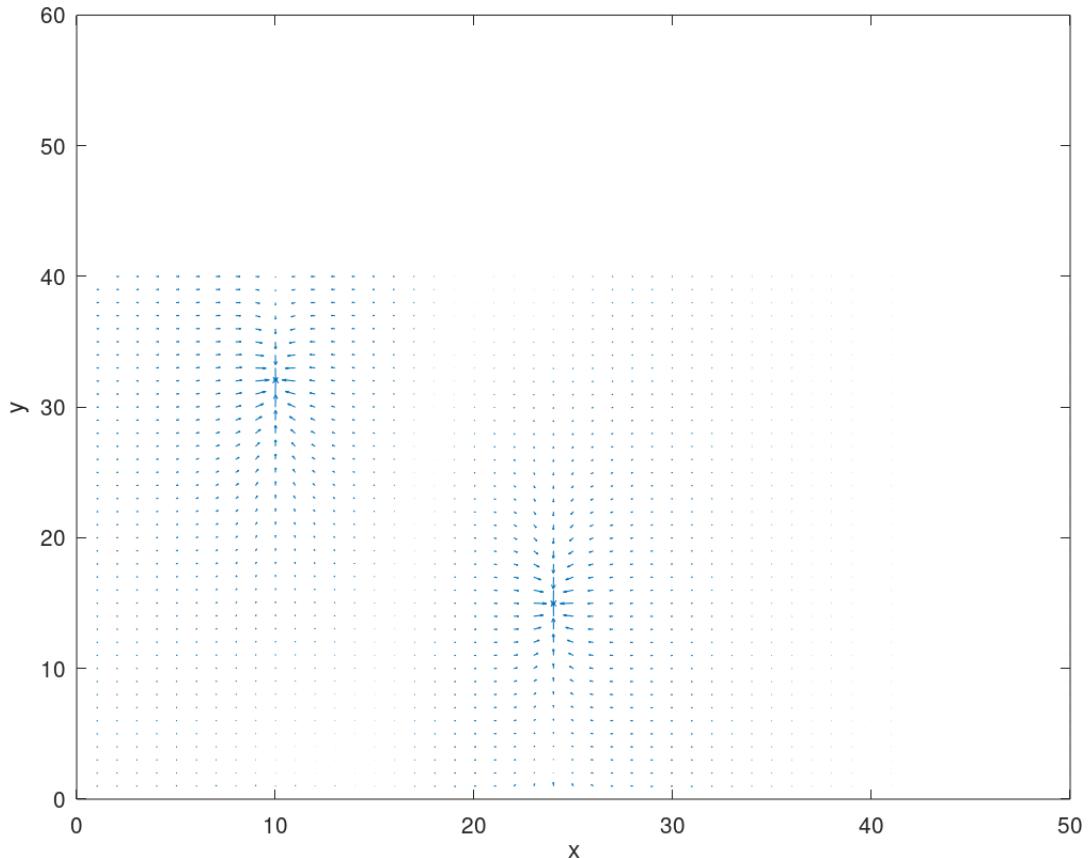
As Figuras 37 e 38 trazem o campo de velocidade do escoamento do fluído para a solução encontrada com  $n = 50$ ,  $m = 40$  e  $\omega = 0.25$  (Figura 33) e  $\omega = 1$  (Figura 36). Nota-se que quanto mais próximo aos poços, maior é a norma vetorial da velocidade, o que está de acordo com o comportamento físico do problema. Quanto mais próximo de um “buraco”, mais rápido um fluido escorrerá. Ainda, reafirma a hipótese sobre o valor de  $\omega$ , que quanto maior, mais cedo o fluido vai ganhando velocidade, isto é, haverá mais “vetorinhos” em torno do poço.

Figura 37 – Aplicação 2 |  $n = 50$ ,  $m = 40$  |  $\omega = 0.25$  | Campo de velocidade.



Fonte: elaborado pelo autor.

Figura 38 – Aplicação 2 |  $n = 50, m = 40$  |  $\omega = 1$  | Campo de velocidade.



Fonte: elaborado pelo autor.

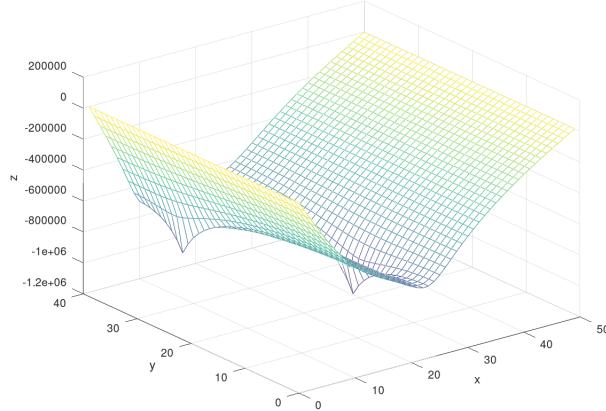
A Tabela 7 apresenta a comparação entre os tempos de execução para dois tamanhos de problema: um pequeno ( $n = 50, m = 40$ ), e um grande ( $n = 500, m = 400$ ).

n	50		500	
	m	40		400
w	método SOR	método direto	método SOR	método direto
0.25	127.767232	0.007413	12776.563720	4.070408
0.5	127.465428	0.007369	12769.725313	4.485020
0.75	126.089480	0.007431	12752.728190	3.616257
1	126.477611	0.007376	12728.240246	4.113870
1.25	126.437857	0.007399	12726.988104	3.365242

Tabela 7 – Comparaçāo entre tempos de execuāo

Novamente, fez-se mais um teste, com um limite de 100000 iterações para o método SOR, buscando observar qual seria uma melhor aproximação para o problema proposto. Os resultados gráficos estão apresentados na Figura 39. Ressalta-se a maior “movimentaāo” das bordas onde há fluxo, com uma queda de pressāo mais suave.

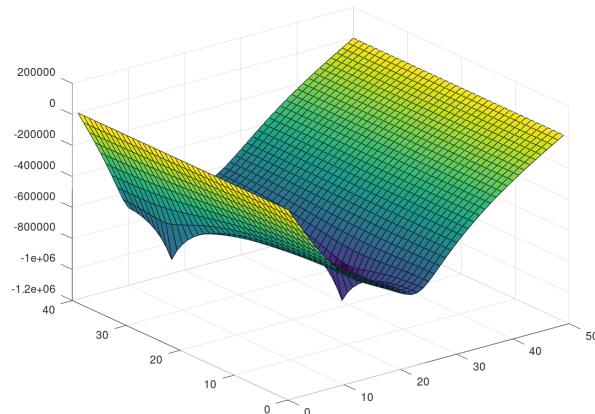
Figura 39 – Aplicação 2 | Superfície |  $n = 50, m = 40$  |  $\omega = 0.25$  | 100000 iterações.



Fonte: elaborado pelo autor.

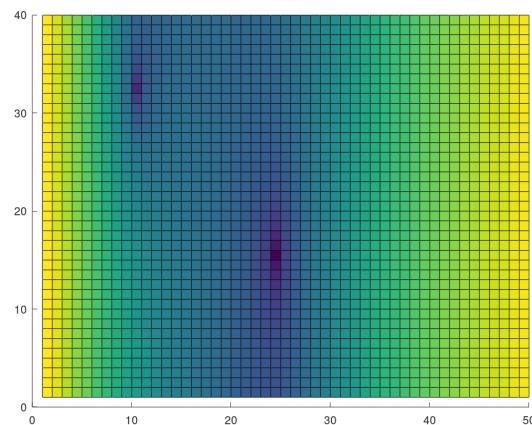
A partir do resultado melhorado, fez-se novamente a análise de superfície do problema, onde espera-se que próximos aos poços a pressão seja mais baixa (e com cores mais frias), e os pontos nas bordas, onde a pressão é constante, sejam também constantes. As Figuras 40, 41 confirmam este fato.

Figura 40 – Aplicação 2 | Superfície |  $n = 50, m = 40$  |  $\omega = 0.25$ .



Fonte: elaborado pelo autor.

Figura 41 – Aplicação 2 | Superfície |  $n = 50, m = 40$  |  $\omega = 0.25$ .



Fonte: elaborado pelo autor.

# Conclusões

Com os resultados gráficos obtidos pode-se concluir que o valor de  $\omega$  no método SOR não só influencia na convergência da solução do sistema linear, como também na característica da solução, tornando as concavidades mais ou menos côncavas. Dessa forma, como provavelmente em aplicações reais a espessura da malha será mais refinada que a utilizada neste relatório, será necessário utilizar métodos inteligentes para solucionar o sistema linear resultante da modelagem, como por exemplo o método SOR. Assim, é necessário antes de resolver o problema de fato, estudar e analisar a influência dos parâmetros requeridos pelo modelo, como por exemplo no caso do método SOR o valor de  $\omega$ . Isto, em conjunto com o conhecimento do problema físico, gerará soluções melhores e mais próximas do problema físico.

Entretanto, ressalta-se que o método direto sempre foi mais rápido que o método SOR. Isto pode ser explicado pois não sabe-se qual é a tolerância máxima do método direto, e talvez poderia ocorrer de estar pedindo demais do método SOR de forma que ele atingisse uma solução de qualidade “boa demais”. Ou também, uma vez que a matriz passada ao método direto é do tipo esparsa, e sua dimensão não era exorbitantemente grande, pode ser que o método direto esteja melhor preparado a esses casos. Ainda, há o fato de o método direto possuir implementação mais otimizada, quando comparado a implementação do método SOR. Analisando isto notou-se como as implementações poderiam ser melhoradas, aproveitando-se melhor dos laços utilizados ou utilizando linguagens mais robustas.

Também através dos testes nota-se a importância de se plotar a solução encontrada, não apenas para entender o contexto do problema, como também para avaliar quais e discretizações são mais adequadas para cada tipo de problema proposto, bem como definir da melhor forma possível qual será o critério de parada.

# Referências

CATABRIGA, L. Problemas de valor no contorno (pvc). *Notas de aula*, 12 aug. 2019, 16 dec. 2019. Citado na página 8.

WIKIPÉDIA. *Método das diferenças finitas — Wikipédia, a enclopédia livre*. 2018. [Online; acessado em 23 de novembro de 2019]. Disponível em: <[https://pt.wikipedia.org/w/index.php?title=M%C3%A9todo\\_das\\_diferen%C3%A7as\\_finitas&oldid=51019759](https://pt.wikipedia.org/w/index.php?title=M%C3%A9todo_das_diferen%C3%A7as_finitas&oldid=51019759)>. Citado 2 vezes nas páginas 5 e 6.