

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

**6º Exercício Computacional de Algoritmos Numéricos II
Relatório**

**Matheus Gomes Arante de Souza
Vinícius Lucas dos Reis**

**Vitória
Junho de 2019**

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

**Matheus Gomes Arante de Souza
Vinícius Lucas dos Reis**

**6º Exercício Computacional de Algoritmos Numéricos II
Relatório**

Este exercício tem por objetivo analisar o desempenho de métodos iterativos estacionários e não estacionários considerando estratégias otimizadoras distintas para cada classe de métodos. Foi utilizado o ambiente de programação Octave para implementação dos métodos.

**Vitória
Junho de 2019**

Sumário

1	Introdução	1
2	Método das Diferenças Finitas	2
3	Implementação	3
3.1	GMRES	3
3.2	SOR	3
3.3	Plot 3D	4
4	Experimentos Numéricos	5
4.1	Validação 2 - Problema com solução conhecida	5
4.2	Aplicação Física 1 - Resfriador bidimensional	7
4.3	Aplicação Física 2 - Escoamento em Águas Subterrâneas	9
5	Conclusão	11
6	Referências	12

1 Introdução

Diversos fenômenos físicos podem ser modelados através de equações diferenciais e condições de contorno em um dado domínio, caracterizando um Problema de Valor no Contorno. Estes problemas podem ser resolvidos através da discretização do domínio e da aproximação das derivadas da equação diferencial, além da aplicação das condições de contorno.

Este exercício visa observar o comportamento e avaliar o desempenho de métodos iterativos estacionários e não estacionários (SOR e GMRES) frente à diferentes problemas de valor no contorno, considerando estratégias para melhorar o tempo computacional, além de armazenar os coeficientes sem uma estrutura matricial.

Nas seguintes seções apresentaremos um breve resumo de sobre o Método das Diferenças Finitas e sua utilização em problemas de valor de contorno, detalhes sobre a implementação dos métodos na linguagem Octave, a análise de diferentes aplicações frente aos métodos SOR e GMRES e por fim as conclusões obtidas a partir dos resultados.

2 Método das Diferenças Finitas

O método das diferenças finitas consiste em um método de resolução de equações diferenciais que se baseia na aproximação de derivadas por diferenças finitas. As aproximações vêm da série de Taylor da função derivada.

$$\frac{\partial u}{\partial x}(x_i, y_j) \approx \frac{u_{I+1} - u_{I-1}}{2h_x}, \theta(h_x^2)$$

$$\frac{\partial u}{\partial y}(x_i, y_j) \approx \frac{u_{I+n} - u_{I-n}}{2h_y}, \theta(h_y^2)$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{u_{I-1} - 2u_I + u_{I+1}}{h_x^2}, \theta(h_x^2)$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) \approx \frac{u_{I-n} - 2u_I + u_{I+n}}{h_y^2}, \theta(h_y^2)$$

$$I = 1, 2, \dots, m * n$$

Para solucionar Problemas de Valor no Contorno 2D através do método das Diferenças Finitas, devemos:

- Discretizar o domínio:

$$\Omega = \{(x, y), a < x < b, c < y < d\}$$

$$h_x = \frac{(b-a)}{(n-1)} \quad x_i = a + (i-1)h_x, \quad i = 1, \dots, n$$

$$h_y = \frac{(d-c)}{(m-1)} \quad y_j = c + (j-1)h_y, \quad j = 1, \dots, m$$

- Aplicar as aproximações de diferenças finitas nas derivadas da equação diferencial $-k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \beta_x \frac{\partial u}{\partial x} + \beta_y \frac{\partial u}{\partial y} + \gamma u = f$
para chegar à expressão

$$d_I u_{I-n} + b_I u_{I-1} + a_I u_I + c_I u_{I+1} + e_I u_{I+n} = f_I$$

e obter os coeficientes

$$a_I = \gamma_I + 2k \left(1/h_x^2 + 1/h_y^2 \right)$$

$$b_I = (-k/h_x^2) - (\beta_x)_I / (2h_x)$$

$$c_I = (-k/h_x^2) + (\beta_x)_I / (2h_x)$$

$$d_I = (-k/h_y^2) - (\beta_y)_I / (2h_y)$$

$$e_I = (-k/h_y^2) + (\beta_y)_I / (2h_y)$$

para chegar a um sistema resultante que é representado por uma matriz pentagonal.

- Aplicar as condições de contorno, que variam de acordo com os dados que possuímos sobre o PVC. Estas podem ser de valor prescrito, fluxo prescrito ou condição de contorno mista.

3 Implementação

Para utilizar uma estrutura livre de matriz, a estratégia utilizada foi criar vetores a , b , c , d , e para armazenar somente as diagonais da matriz. Para inicializar estes vetores, é chamada a função `coefs(a,b,c,d,n,m,k,beta_x,beta_y,gama)`.

Após serem inicializados, é necessário aplicar as condições de contorno específicas do problema nestes vetores. Para isso, existem as funções específicas para cada exercício: `validacao2.m`, `aplicacao1.m` e `aplicacao2.m`.

Para o método GMRES e SOR, foram criadas as funções `solveByGMRES` e `solveBySOR`. Na chamada das funções, basta informar os intervalos do domínio (a, b, c, d), os valores para discretização (n, m), número de vetores para o restart (k) no caso do GMRES ou (w) no caso do SOR, além da tolerância, o número máximo de iterações e a função que aplica as condições de contorno específicas do problema.

3.1 GMRES

```
function [x,y,u,erro] = solveByGMRES(a,b,c,d,n,m,k,rtol,maxit,exerc)
```

Esta função utiliza a função `createMatrix(a,b,c,d,n,m,exerc)`, que utiliza `coefs` para gerar os vetores e a partir destes, construir a matriz dos coeficientes e o vetor f .

Além disso, as matrizes da fatoração LU são utilizadas como pré-condicionadores para acelerar a convergência do método, sendo passadas como parâmetros para a chamada da função `gmres(A,f,k,rtol,maxit,L,U)`.

3.2 SOR

```
function [x,y,u,erro] = solveBySOR(a,b,c,d,n,m,w,rtol,maxit,exerc,f_erro)
```

A função define as 5 diagonais da matriz através da função `coef` apresentada anteriormente e os utiliza para obter os coeficientes de cada linha da matriz A que seria criada, respeitando os intervalos de cada diagonal e as devidas condições de contorno.

Os retornos gerados são a discretização linear de cada eixo (x e y) e o vetor solução u , respectivamente. Esses valores serão passados posteriormente à função de plot para a geração de seu gráfico.

Para maior esclarecimento do parâmetro de entrada f_{erro} , esta serve para que o usuário possa escolher a forma como a qual ele deseja avaliar o erro da solução em cada iteração do método.

3.3 Plot 3D

Para ajuste da malha de discretização e dos resultados obtidos pelos métodos iterativos foi utilizado o seguinte código:

```
function plot3D(x,y,u,n,m)
[X,Y] = meshgrid(x,y);
U = reshape(u,m,n);
mesh(X,Y,U);
endfunction
```

4 Experimentos Numéricos

4.1 Validação 2 - Problema com solução conhecida

Neste exercício serão realizados alguns testes com diferentes tamanhos de discretização para ambos os métodos iterativos, SOR e GMRES.

n	m	Ordem (n,m) do Sistema Linear
n	= m	2500
n	\neq m	2500
n	= m	10000
n	\neq m	10000
n	\neq m	500000
n	= m	1000000

Tabela 1

Ordem	Iter.	Erro Rel.	Tempo (segundos)
2500(50,50)	4270	1e-10	1.49
2500(25,100)	7996	9.99e-11	2.82
10000(100,100)	16064	9.99e-11	8.32
10000(10,1000)	568724	1e-10	699
500000(500,1000)	†	†	†
1000000(1000,1000)	†	†	†

Tabela 2: Execução do SOR para a Tabela 1

Ordem	Iter.	Erro Rel.	Tempo (segundos)
2500(50,50)	27	1.71e-12	0.0413
2500(25,100)	27	2.15e-12	0.0428
10000(100,100)	30	8.13e-12	0.286
10000(10,1000)	44	9.61e-11	0.249
500000(500,1000)	156	7.55e-11	571
1000000(1000,1000)	†	†	†

Tabela 3: Execução do GMRES para a Tabela 1

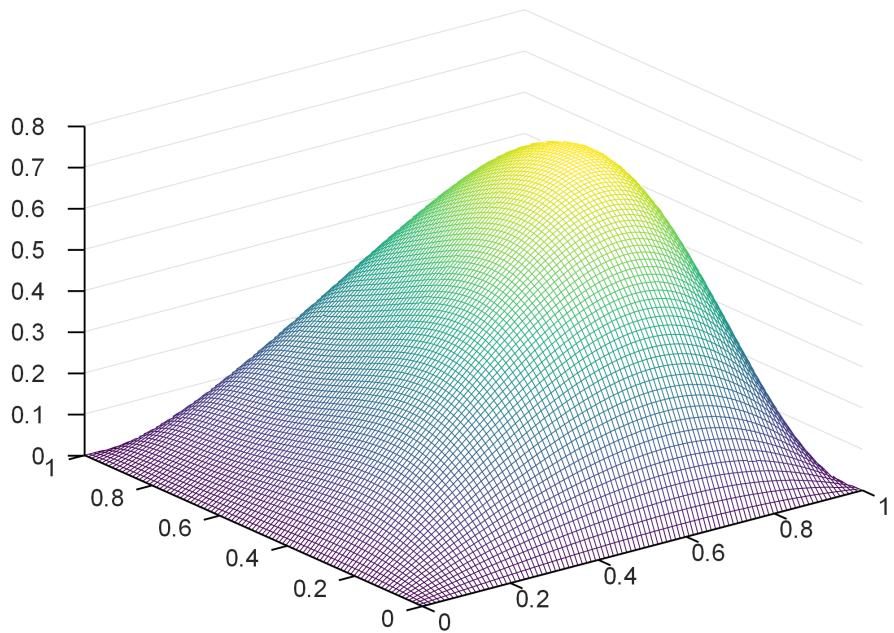


Figura 1: $n = 100, m = 100$

Para verificarmos a ordem de convergência do método das diferenças finitas basta analisarmos a ordem de decrescimento do erro em cada iteração dos métodos. Neste caso, foram usadas somente aproximações de segunda ordem na criação das matrizes, o que fez com que os procedimentos obtivessem uma convergência quadrática.

4.2 Aplicação Física 1 - Resfriador bidimensional

Método Iterativo	Iter.	Erro rel.	Tempo (segundos)
SOR - Condição Mista	38731	1e-10	34
SOR - Valor Prescrito	28882	1e-10	21.9
GMRES - Condição Mista	32	9.22e-12	0.395
GMRES - Valor Prescrito	31	1.22e-11	0.496

Tabela 4: Tempo de processamento, número de iterações e erro relativo para os dois métodos iterativos e respectivas estratégias de armazenamento

Com base nos gráficos a seguir, consegue-se notar de forma visual a diferença existente na qualidade da solução com relação as condições de contorno fornecidas pelo problema. Quando consideramos a condição mista na lateral direita ou (L, y), apesar de alcançar um erro menor do que o método com valor prescrito, a solução apresenta uma leve deformação no gráfico (em forma de curva). Entretanto, quando consideramos o valor prescrito, o gráfico apresenta uma linha reta, mais condizente e aproximada da solução esperada.

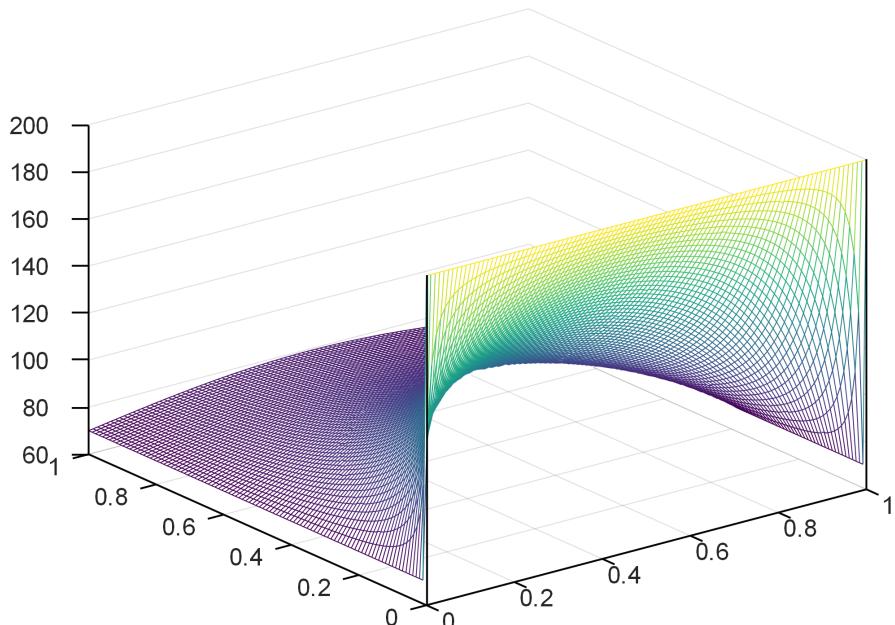


Figura 2: Condição Mista

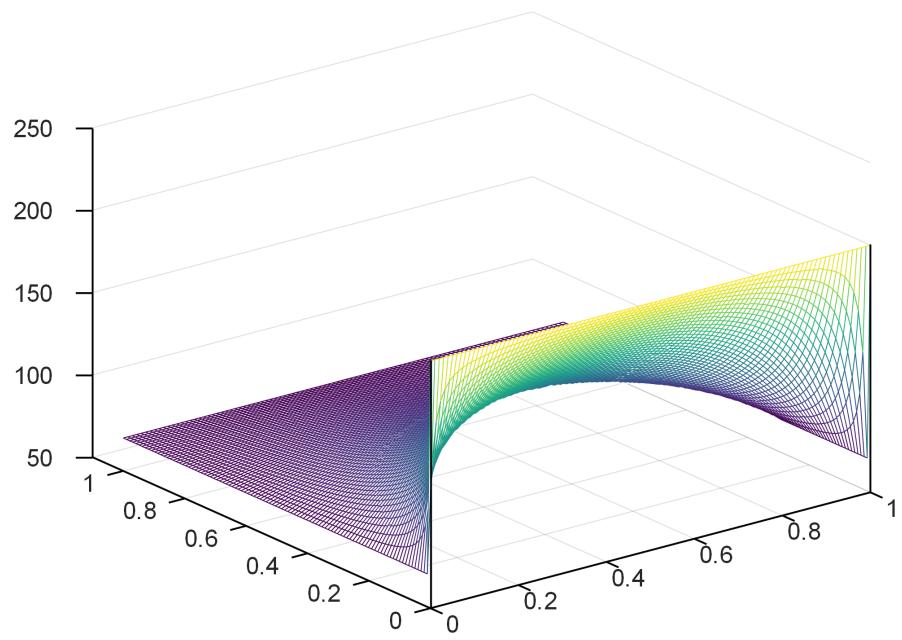


Figura 3: Valor Prescrito

4.3 Aplicação Física 2 - Escoamento em Águas Subterrâneas

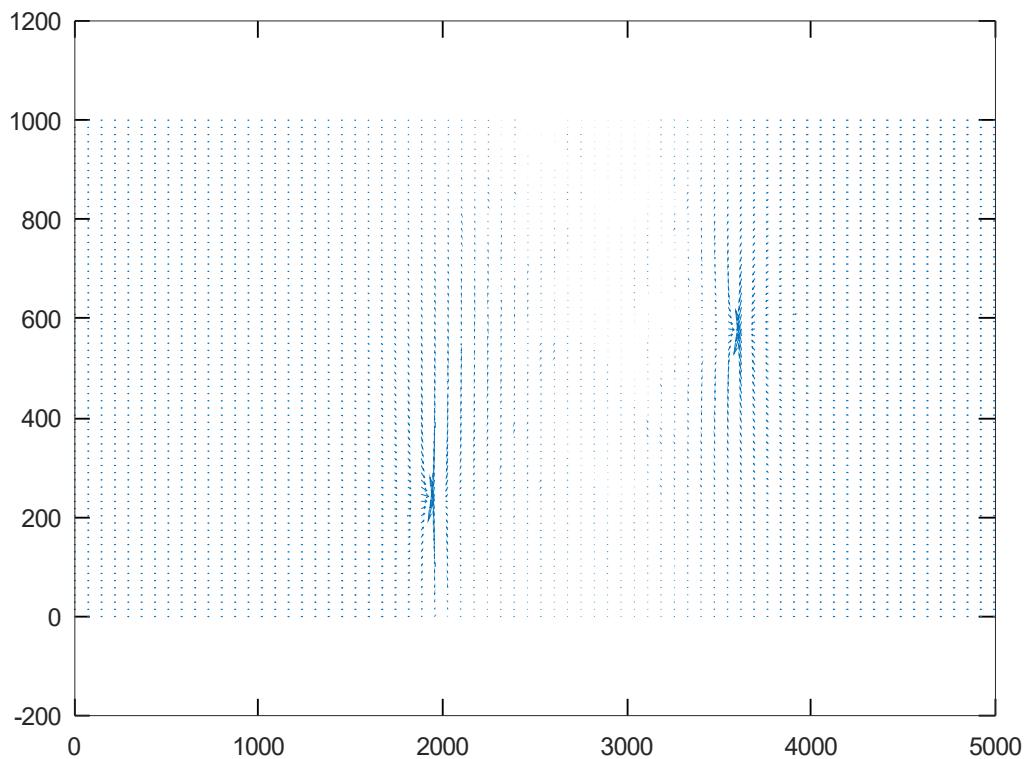


Figura 4: Gráfico da pressão e o campo de velocidade.

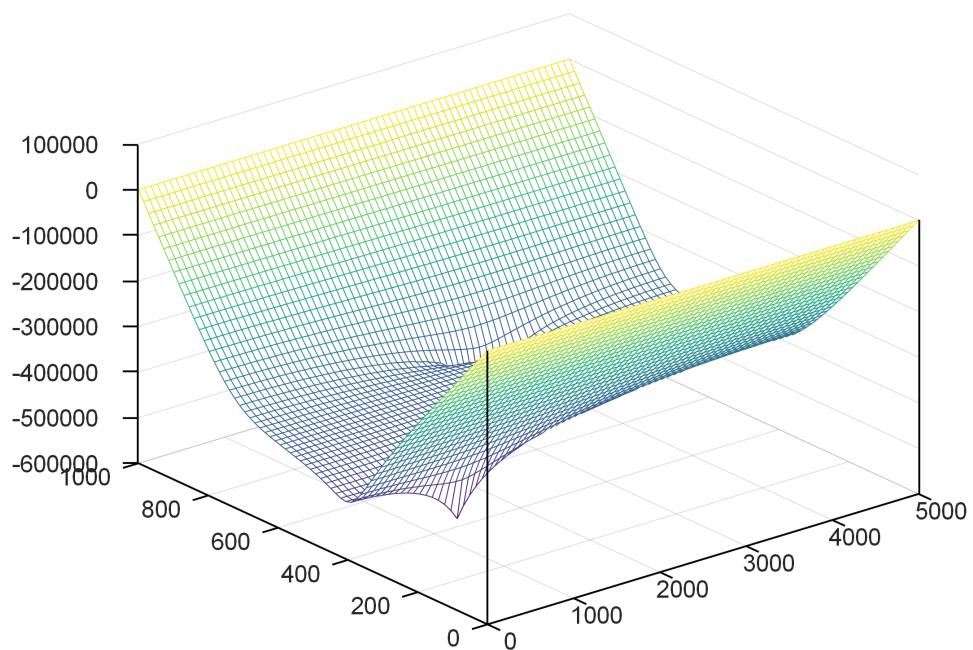


Figura 5: Problema de tamanho pequeno - $n = 51$, $m = 21$

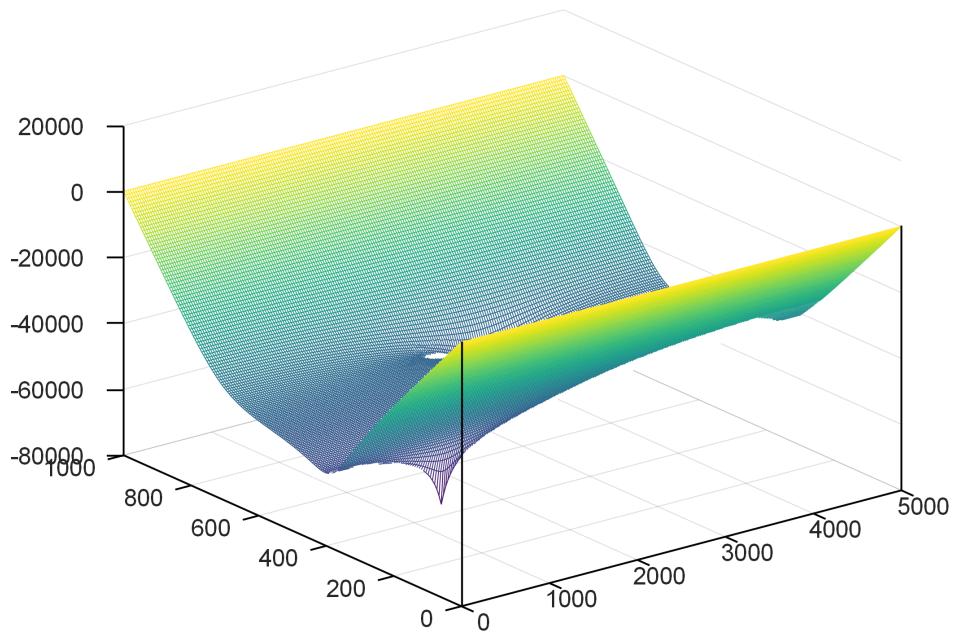


Figura 6: Problema de tamanho grande - $n = 101$, $m = 201$

Método Iterativo	Tempo
SOR - Prob. Pequeno	0.551
SOR - Prob. Grande	0.012
GMRES - Prob. Pequeno	1.113
GMRES - Prob. Grande	0.592

Tabela 5: Tempo de processamento para os dois métodos iterativos e respectivas estratégias de armazenamento

5 Conclusão

Com relação ao tempo computacional e número de iterações necessárias para obter a convergência dos métodos, o GMRES apresentou desempenho significativamente melhor, como é possível observar nas tabelas 2 e 3.

Ao utilizar uma estrutura livre de matriz, há um ganho em termos de memória utilizada, pois precisa-se armazenar apenas os vetores de cada diagonal não nula da matriz (neste caso 5, por ser um problema bidimensional), ou nem isso se os coeficientes forem calculados sempre em cada iteração. Entretanto, nesta abordagem livre de matriz, é necessário tratar cada região do domínio individualmente, o que acaba afetando o tempo computacional.

Em contrapartida, o método GMRES apesar de consumir mais memória ao gerar e armazenar a matriz A e as matrizes da fatoração LU, apresentou melhor tempo computacional. Isso acontece pois as matrizes LU são utilizadas como pré-condicionadores e aceleram a convergência do método. Além disso, como a função `gmres` é implementada pelo Octave, possui otimizações nos cálculos e nas operações sobre matrizes, acarretando em um menor tempo de execução.

Vale ressaltar que como ambos os métodos possuem um parâmetro regulável para cada matriz (k para GMRES e w para SOR), um melhor refinamento de seus valores pode promover um ganho significativo no tempo computacional. Um exemplo disso seria a Validação 2, que ultrapassaram o limite de tempo estabelecido para os valores utilizados nas matrizes com ordem maiores de 100000, porém para valores mais adequados isso pode ser evitado.

Uma futura proposta para esta avaliação é implementar o SOR utilizado neste trabalho como uma função pré-compilada para poder comparar mais honestamente os dois métodos. Outra comparação que pode ser feita é a comparação tanto do SOR matricial quanto do SOR livre de matriz (ambos compilados) para verificar a real compensação do ganho de memória com relação ao tempo. Ou seja, verificar até que ponto o ganho de memória valida a perda de desempenho.

6 Referências

EATON, John W. Octave Documentation, 1996.
Disponível em: <<https://octave.org/doc/v4.2.2/>>.

COMMUNITY, Octave Forge. Octave Forge, 2002.
Disponível em: <<https://octave.sourceforge.io/docs.php>>.

CATABRIGA, Lucia. Problemas de Valor de Contorno (PVC), 2019.
Disponível em: <http://inf.ufes.br/~luciac/mn1/191-PVC_AlgoII.pdf>.