UNIVERSIDADE DE SÃO PAULO
Instituto de Física de São Carlos

Potenciais e campos elétricos: a equação de Laplace

Aluno: Matheus Fernandes Sousa Lemes (N° USP: 9866506)
Curso: Bacharelado em Física
Disciplina: Eletromagnetismo computacional
Orientador: Prof. Dr. Guilherme Matos Sipahi

São Carlos - 2020

Sumário

1 Introdução

1.1 A equação de Laplace

Esse trabalho foi focado em solucionar os exercícios referentes a seção 5.1 do livro do Giordano [1], em que queremos determinar o potencial e campo elétrico para regiões do espaço que não contenham densidade de carga volumétrica, ou seja, $\rho = 0$. Substituindo na lei de Gauss em sua forma diferencial, temos

$$\boldsymbol{\nabla} \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} = 0. \tag{1}$$

Usualmente é mais fácil trabalharmos com o potencial elétrico, visto que ele é um escalar, vamos reescrever a equação acima em termos do potencial. Como $E = -\boldsymbol{\nabla} V$, temos

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0, \tag{2}$$

que é a conhecida *equação de Laplace*. Se assumirmos que o nosso problema é invariante na direção z, ou seja, $V(x, y, z) = V(x, y)$, a equação acima será simplesmente

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0. \tag{3}$$

Dada as condições de contorno do problema, podemos determinar o potencial elétrico. Ainda, podemos utilizar $V(x, y)$ para calcular o campo elétrico, visto que

$$E_\alpha = -\frac{\partial V_\alpha}{\partial x} \tag{4}$$

com $\alpha = x, y$.

1.2 Método das diferenças finitas

Para resolver a equação de Laplace numericamente é necessário que discretizemos o nosso espaço, em que os pontos serão especificados pelos números inteiros i, j, com $x = i\Delta x$ e $y = j\Delta y$. Logo, podemos escrever o potencial como $V(x, y) \equiv V(i, j)$.

Um dos métodos mais famosos para resolver equações diferenciais computacionalmente é o *método das diferenças finitas*, em que aproxima-se as derivadas parciais por diferenças finitas (para mais informações veja [2]). Substituindo as derivadas parciais de segunda ordem na equação (3) por suas aproximações de diferenças finitas centradas, temos

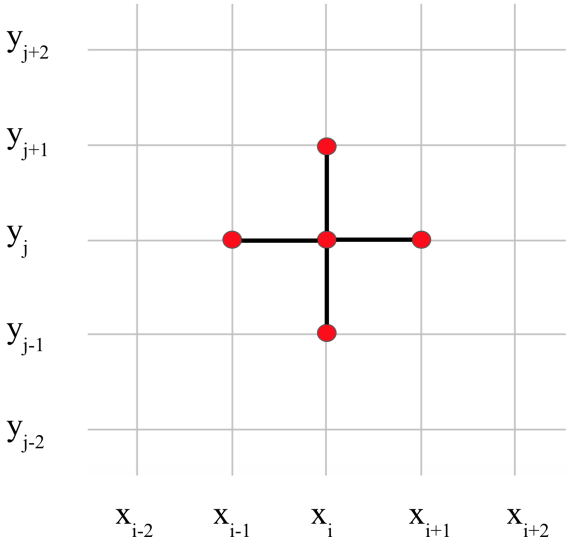
$$\frac{V(i+1, j) - 2V(i, j) + V(i-1, j)}{(\Delta x)^2} + \frac{V(i, j+1) - 2V(i, j) + V(i, j+1)}{(\Delta y)^2} = 0.$$

Simplificando a notação, vamos considerar que $\Delta x = \Delta y = h$. Logo, podemos reescrever a equação acima como

$$V(i,j) = \frac{1}{4}\left[V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j+1)\right]. \tag{5}$$

Esse é o chamado *estencil de 5 pontos* e nos mostra que o potencial $V(i,j)$ será determinado pela média dos seus vizinhos mais próximos. A figura 1 representa graficamente esse resultado. É importante ressaltar que os pontos na fronteira não podem ser calculados por essa fórmula, visto que sairia do domínio espacial em questão. Contudo, como os pontos na fronteira não irão mudar, esse fato não será um problema.

Figura 1: Estencil de 5 pontos em uma rede quadrada para calcular o potencial $V(i,j)$.



Além disso, o campo elétrico calculado por diferenças finitas será dado por

$$E_x(i,j) = -\frac{V(i+1,j) - V(i-1,j)}{2h}, \tag{6}$$

com expressão análoga para a direção y.

1.3 Algoritmo de relaxação de Jacobi

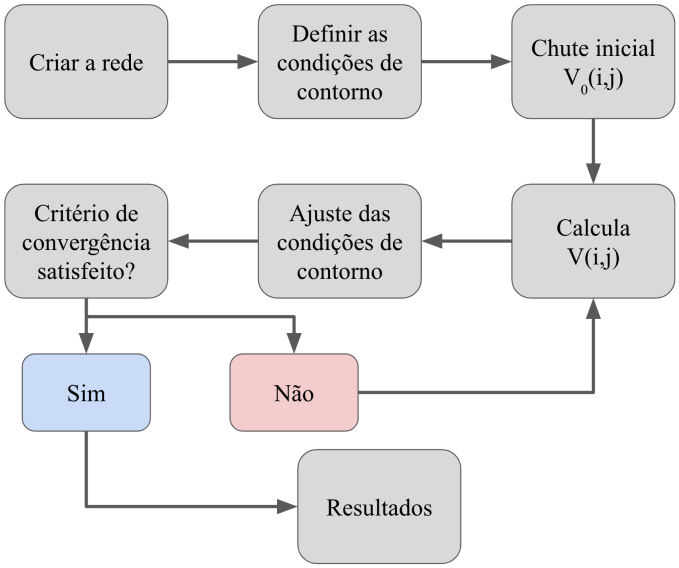
Para implementar o método de diferenças finitas descrito anteriormente é necessário a utilização de processos iterativos. Nós começamos com um chute inicial $V_0(i,j)$ e utilizamos a equação (5) para calcular um valor melhor do potencial $V_1(i,j)$. A partir de $V_1(i,j)$ calculamos um valor ainda melhor $V_2(i,j)$. Continuamos esse processo até que nossa solução atinga um critério de convergência. Esse é o conhecido *algortmo de relaxação de Jacobi* e definimos como critério de convergência a seguinte

expressão

$$\frac{\sqrt{\sum_{i=0,j=0}^{n} |V_{n+1}(i,j) - V_n(i,j)|^2}}{\sqrt{\sum_{i=0,j=0}^{n} |V_n(i,j)|^2}} < \epsilon, \tag{7}$$

em que ϵ representa o limite de convergência estabelecido. Um esquema geral desse algotitmo pode ser visto na figura 2.

Figura 2: Esquema geral do código utilizado.



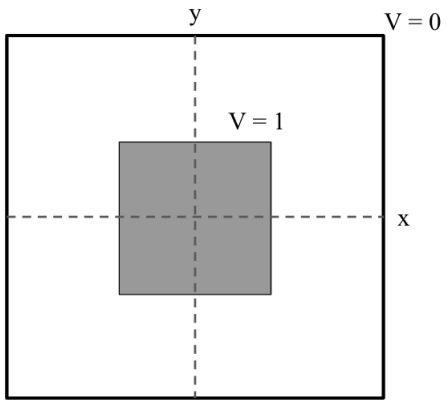
Escolhemos implementar os algoritmos em Python, visto que é uma linguagem simples de trabalhar e muito utilizada no meio científico, além de possuir uma extensa comunidade de usuários que podem ajudar na solução de problemas nos códigos.

2 Potenciais e campos para algumas condições de contorno

2.1 Geometria 1: quadrado oco com um metal condutor no centro

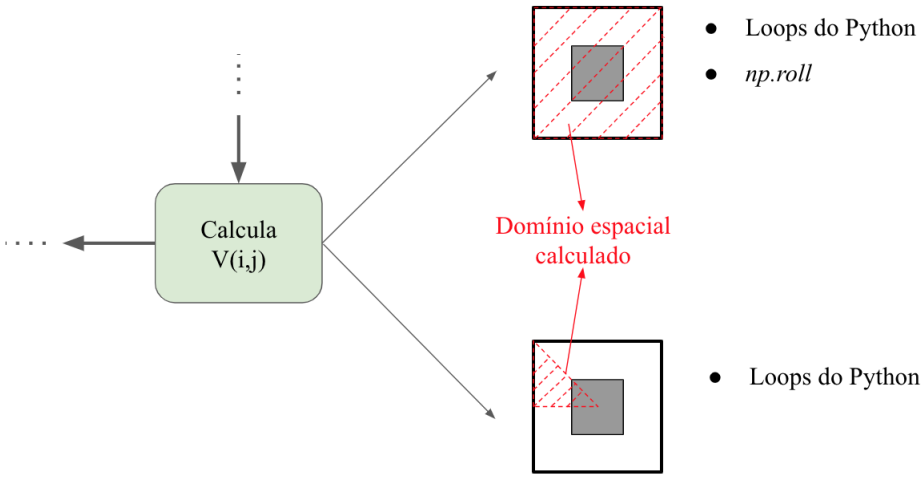
A primeira situação que resolvemos foi a de um quadrado oco aterrado com um metal condutor posto no centro, mantido a uma tensão de $V = 1$, como mostra a figura 4. É importante ressaltar que não estamos atribuindo unidade para as variáveis simplesmente por uma questão de simplicidade.

Figura 3: Esquemático da geometria 1.



Nós focamos nessa primeira geometria em explorar três métodos de resolução do problema. Para todo o domínio espacial utilizamos duas formas de calcular $V(i, j)$: loops no Python e a função $np.roll$. Além disso, devido a simetria do problema, foi possível calcular o potencial em todo o espaço apenas com um oitavo da geometria, utilizando também os loops do Python.

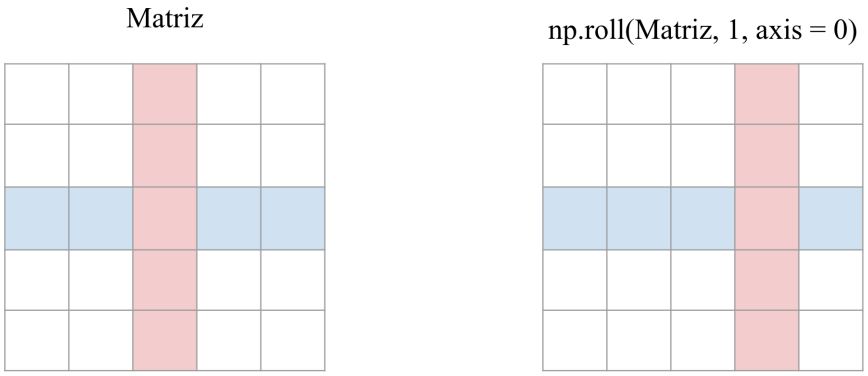
Figura 4: Métodos de cálculo de $V(i, j)$.



A grande diferença entre os loops e a função $np.roll$ é como cada elemento da matriz $V(i, j)$ é atualizado segundo a equação (5). As estruturas de repetição atualizam cada elemento de forma sequencial, ou seja, calculam o novo valor do potencial ponto a ponto. Já a função $np.roll$ atualiza todos os elementos da matriz de uma única vez. Ela consegue efetuar essa operação porque, em essência, o que ela faz é mover todos os elementos da matriz em uma determinada direção. Através da figura 5 é possível entender melhor como funciona essa função, em que aqui estamos destacando o que ocorre em apenas uma linha para fins didáticos, porém isso ocorre na matriz inteira. Perceba que

ao aplicar o $np.roll$, a linha em vermelho se deslocou para a direita de uma unidade. Se fizemos o mesmo para a esquerda e deslocassemos a linha em azul para cima e para baixo e, por fim, dividindo tudo por quatro, estaríamos atualizando o ponto central pela média dos seus vizinhos mais próximos. E é justamente isso que a equação (5) faz, porém o $np.roll$ permite realizar essa tarefa de forma muito mais rápida.

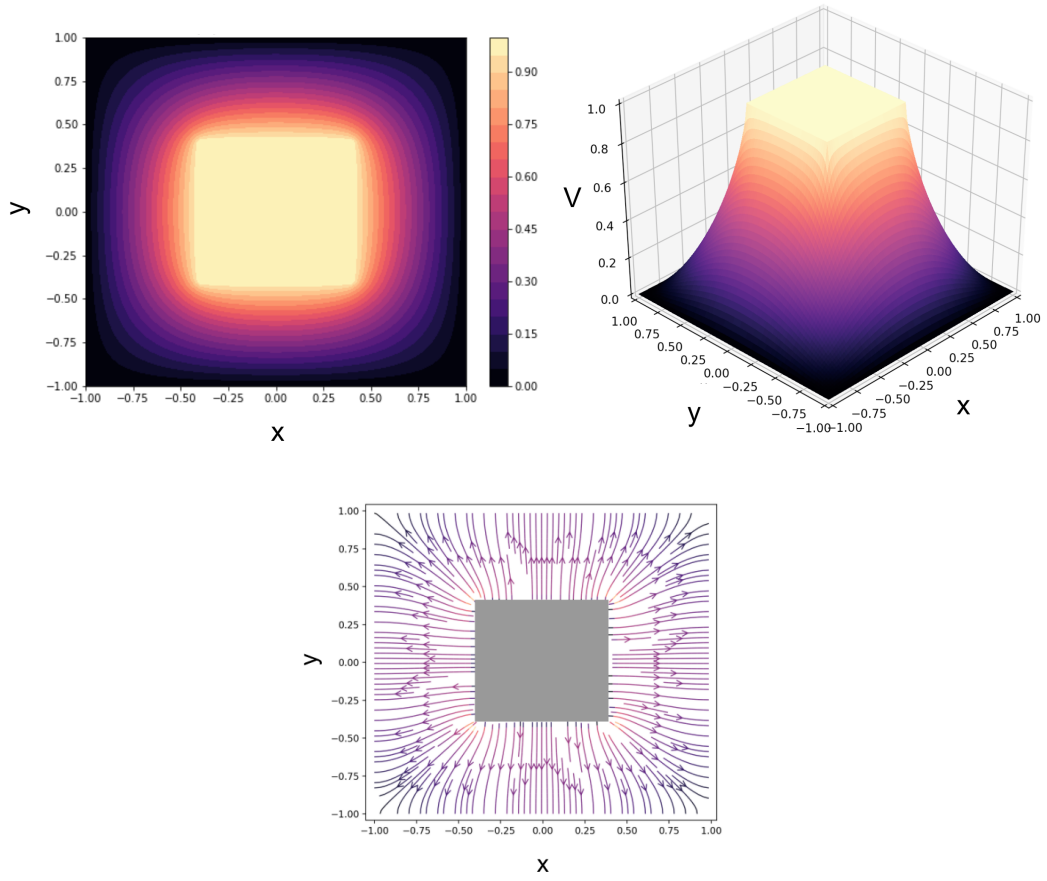
Figura 5: Funcionamento da função $np.roll$.



$$V(i,j) = \frac{1}{4}[V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j+1)]$$

```
V_novo = 0.25*(np.roll(V,1,axis=0)+np.roll(V,-1,axis=0)+np.roll(V,1,axis=1)+np.
↪roll(V,-1,axis=1))
```

Figura 6: Resultados para a geometria 1.



Na figura 6 é possível ver os resultados encontrados nos três métodos para o problema em questão. Embora o resultado final tenha sido o mesmo para os três métodos, alguns pontos importantes puderam ser observados na comparação entre eles. Em primeiro lugar, o cálculo de $V(i, j)$ para o espaço utilizando os loops no Python se mostrou o mais demorado, como pode ser visto na tabela 1, visto que ele não se aproveita da simetria rotacional do problema nem faz uso de funções específicas da linguagem para otimizar o código. Utilizando apenas um oitavo do domínio espacial foi possível observar um redução do tempo de processamento em aproximadamente 7.5 vezes. Por fim, a função $np.roll$ se mostrou a mais eficiente de todos, apresentando um tempo de processamento quase 50 vezes menor que o código que utiliza loops em Python. O único empecilho em relação ao $np.roll$ é que ela é específica para o Python, o que porventura poderia impossibilitar uma transposição do algoritmo para uma outra linguagem de programação. Dessa forma é preciso balancear entre desempenho e adaptabilidade na construção de algoritmos computacionais.

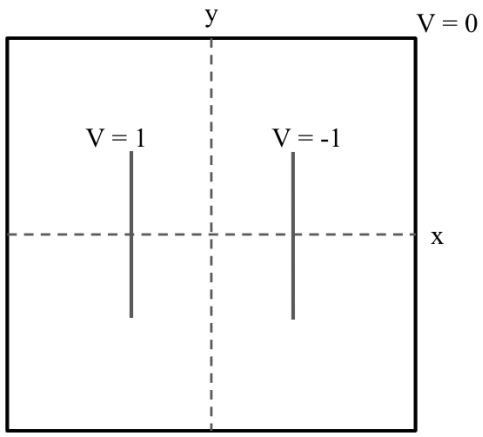
Tabela 1: Tempo de processamento dos 3 métodos de resolução da geometria 1 para uma rede de $L = 101$.

Método	Tempo (s)
Espaço completo	64.4
Um oitavo do espaço	8.6
$np.roll$	1.3

2.2 Geometria 2: capacitor de placas paralelas

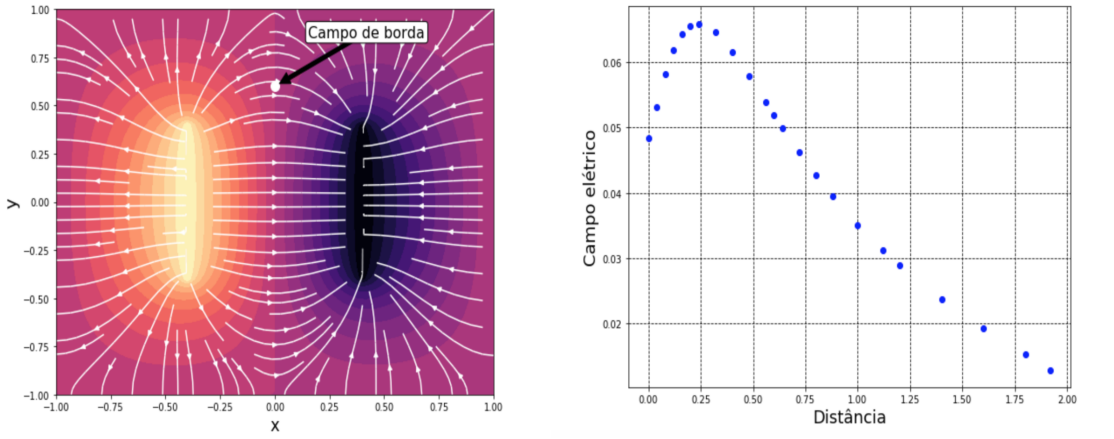
A segunda geometria que resolvemos foi a de um capacitor de placas paralelas dentro do mesmo quadrado oco aterrado anteriormente, como pode ser visto na figura 7. Nesse problema não focamos mais em analisar os métodos de resolução, mas sim em um aspecto mais físico do problema que é como a magnitude do campo elétrico na borda se comporta com a variação da distância entre as placas.

Figura 7: Esquemático da geometria 2.



Para a situação ideal de capacitores de placas paralelas infinitas, a magnitude do campo elétrico é inversamente proporcional a separação das placas, com as linhas de campo sempre paralelas. Contudo, no caso realista essa relação não é valída para o campo na borda das placas, como pode ser visto na figura 8 (a). Visando estudar a relação entre o campo elétrico e a distância entre placas, fixamos um ponto e calculamos a magnitude do campo nele para diferentes distâncias. Como pode ser visto na figura 8 (b), a relação E_{borda} *x distância* é bem mais complicada em uma situação física um pouco mais realista.

Figura 8: Resultados encontrados para os capacitores de placas paralelas. (a) Equipotenciais e campo elétrico para a geometria 2. (b) Campo na borda em relação à separação das placas.



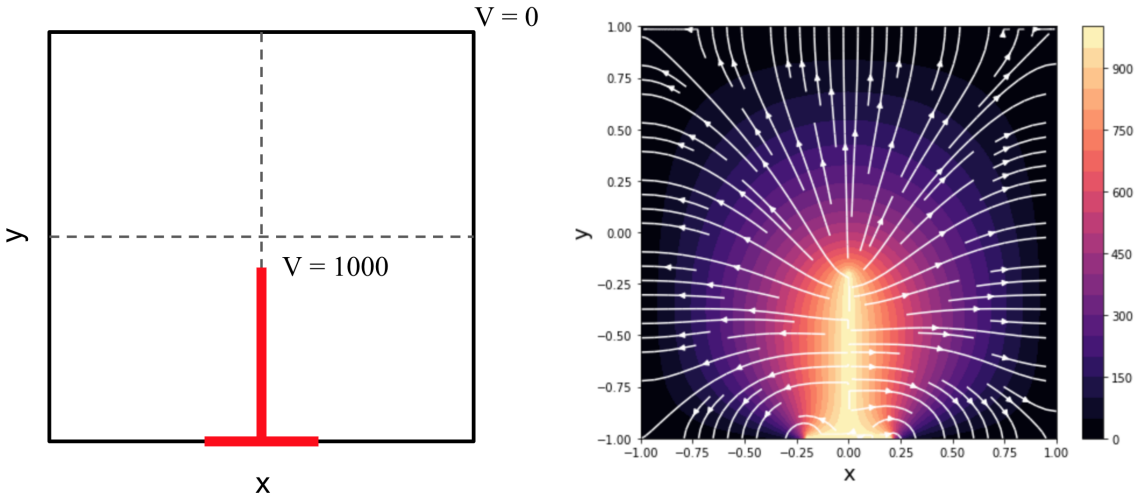
(a) (b)

O estudo do efeito do campo de borda é o primeiro passo para engenheiros e físicos que necessitem de valores mais precisos para os parâmetros de um capacitor. Um exemplo rápido de uma aplicação tecnológica que precisa considerar esse efeito é o carregamento wireless de veículos elétricos [3]. Logo, embora a situação descrita nesse problema ainda esteja longe da realidade, ela permite demonstrar que soluções computacionais são necessárias para solucionar problemas tecnológicos atuais.

2.3 Geometria 3: para-raios

A última geometria que foi analisada nesse trabalho foi a de um para-raios. Ele foi modelado como uma haste condutora fina, conectada a uma placa metálica na parte inferior, como poder ser visto na figura 9 (a). Definimos que a haste está em um potencial de $V = 1000$ e o restante do espaço em $V = 0$, visto que a informação relevante é a diferença de potencial entre a haste condutora e a fonte originadora dos raios (as nuvens na atmosfera). O resultado para as linhas equipotenciais e o campo elétrico podem ser vistos na figura 9 (b).

Figura 9: (a) Esquemático para a geometria 3. (b) Equipotenciais e campo elétrico para o para-raio da geometria 3.



(a) (b)

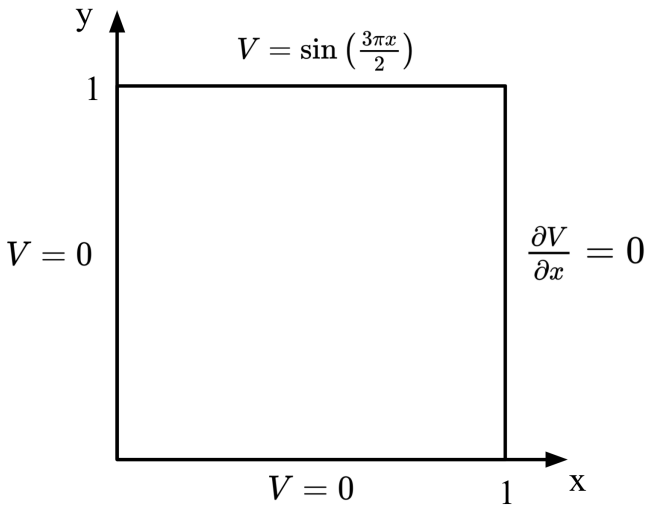
De acordo com [4], foi descoberto em 1934, pelos pesquisadores Schonland e Collens, que a maioria dos raios nuvem-terra eram oriundos da ligação entre o chamado líder negativo originado nas nuvens e o líder positivo originado na terra. A união desses dois objetos é o responsável pela transferência de carga principal, resultando nos terríveis estragos para as estruturas no entorno. O papel de um para-raio é proporcionar um caminho mais fácil para a conexão entre os líderes positivo e negativo, de forma que a transferência de carga ocorra em um circuito elétrico projetado para suportar tal descarga elétrica. Como poder ser visto na figura 9 (b), o campo elétrico na ponta da haste é praticamente vertical, o qual irá proporcionar que haja uma corrente elétrica entre a nuvem e o para-raio, assegurando assim a proteção do local.

3 Acurácia do algoritmo de relaxação

Nessa seção iremos analisar a acurácia do algoritmo de relaxação de Jacobi proposto nesse trabalho. Para fazer isso iremos comparar o resultado computacional para uma condição de contorno com o resultado analítico. A condição de contorno que escolhemos pode ser vista na figura 10. Escolhemos essa condição de contorno porque ela apresenta uma solução simples para comparar com o resultado computacional. A solução analítica é a seguinte expressão:

$$V_{an}(x,y) = \frac{\sinh\left(\frac{3\pi y}{2}\right)\sin\left(\frac{3\pi x}{2}\right)}{\sinh\left(\frac{3\pi}{2}\right)}. \tag{8}$$

Figura 10: Esquemático da situação de contorno utilizada para avaliar a acurácia do algoritmo de relaxação de Jacobi.

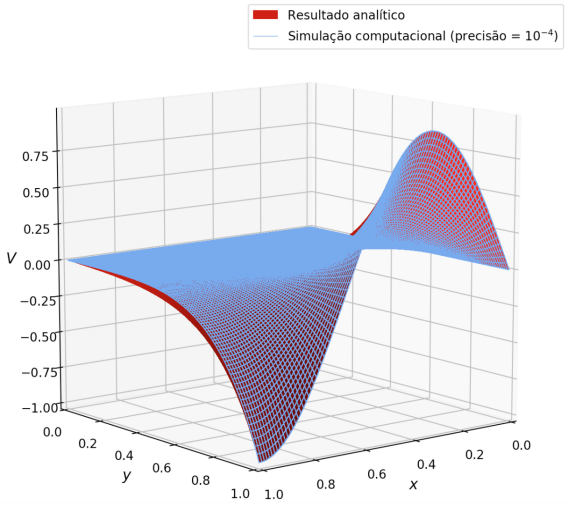


A figura 11 (a) apresenta um plot do potencial analítico e do calculado computacionalmente pelo algoritmo de relaxação com uma precisão de 10^{-4}. Para avaliar quantitativamente a diferença entre as duas soluções definimos que o erro será dado por

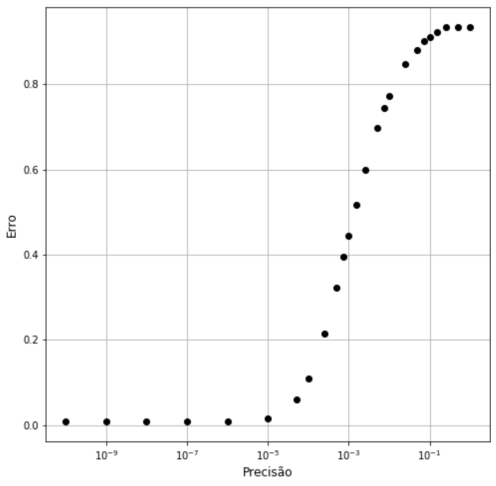
$$E = \max_{\{i,j\}} |V_{an}(i,j) - V_{comp}(i,j)|\,, \tag{9}$$

em que tomamos os valores de $V_{an}(x,y)$ nos pontos (i,j) da rede.

Figura 11: (a) Potencial analítico e computacional. (b) Erro da simulação computacional em função da precisão adotada.



(a)



(b)

O plot do erro em função da precisão adotada está representado na figura 11 (b). Ele nos mostra que o algoritmo de relaxação empregado é *consistente*, ou seja, a diferença entre o potencial analítico e o computacional tende à zero conforme a precisão do método tende para zero [5]. É importante destacar que nessa seção estamos analisando o comportamento do método numérico, i.e., a consistência da discretização. Dessa forma, ignoramos o erro de *round-off* do computador, que em alguns casos pode ser muito importante.

4 Algoritmo de super relaxação simultânea

Nessa última seção iremos explorar um algoritmo de relaxação mais eficiente que o de Jacobi. Ele é conhecido como algoritmo de super relaxação simultânea (do inglês *simultaneous overrelaxation - SOR*) e funciona da seguinte maneira. Se denotarmos $V^*(i,j)$ como sendo o valor calculado pelo algoritmo de Jacobi, podemos pensar que

$$\Delta V(i,j) = V^*(i,j) - V_{velho}(i,j) \tag{10}$$

será a atualização do potencial recomendada pelo algoritmo.

Contudo, essa escolha se mostra muito conservadora e podemos acelerar o processo da forma

$$V_{novo}(i,j) = \alpha \Delta V + V_{velho}(i,j), \tag{11}$$

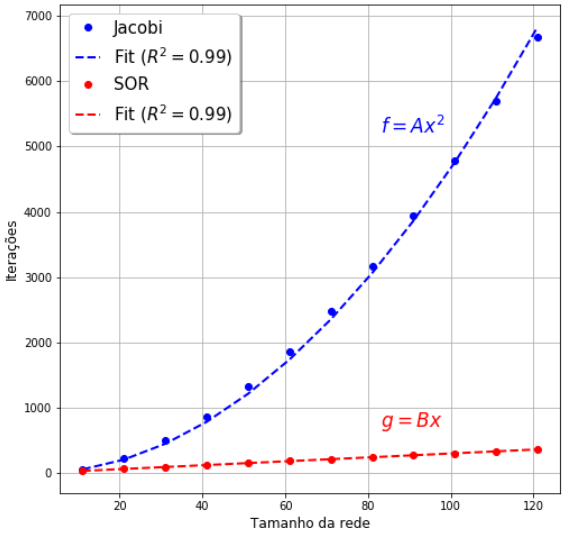
em que se tomarmos $\alpha = 1$ recuperamos o método de Jacobi. Para $\alpha \geq 2$ o método não converge, enquanto que $\alpha < 1$ corresponde ao regime de "underrelaxation". O SOR ocorre quando tomamos α entre 1 e 2. De acordo com [1], a melhor escolha de α para um rede bidimensional quadrada será:

$$\alpha \approx \frac{2}{1 + \pi/L}, \tag{12}$$

em que L representa o número de pontos da rede.

Para determinar se esse novo algoritmo realmente acelera o processo, resolvemos o problema do capacitor de placas paralelas (figura 7) pelo método de Jacobi e o SOR. Como pode ser visto na figura 12, o número de iterações do algoritmo de Jacobi é proporcional à L^2, enquanto que para o SOR o número de iterações é proporcional a L.

Figura 12: Comparação entre os algoritmos de Jacobi e SOR.



5 Conclusão

Nesse trabalho foi possível entender dois conceitos chaves no âmbito da física computacional: o método das diferenças finitas e o algoritmo de relaxação. Foi possível encontrar o potencial e campo elétrico para diferentes condições de contorno, bem como estudar a acurácia do algoritmo de Jacobi. Além disso, pudemos comparar a eficiência de dois tipos de algoritmo de relaxação: Jacobi e SOR. Dessa forma, embora tenham sido estudados problemas relativamente simples nesse projeto, as técnicas e formas de análise aqui trabalhadas e adquiridas serão de extrema importância na resolução de problemas atuais mais complexos.

Referências

[1] N.J. Giordano. *Computational Physics: 2nd edition*. Dorling Kindersley, 2012.

[2] Wikipedia. *Finite difference method*. `https://en.wikipedia.org/wiki/Finite_difference_method`.

[3] Xu Chen, Zhe Zhang, Shengbao Yu, and Tiberiu-Gabriel Zsurzsan. Fringing effect analysis of parallel plate capacitors for capacitive power transfer application. In *2019 IEEE 4th International Future Energy Electronics Conference (IFEEC)*, pages 1–5. IEEE, 2019.

[4] CB Moore, GD Aulich, and William Rison. The case for using blunt-tipped lightning rods as strike receptors. *Journal of Applied Meteorology*, 42(7):984–993, 2003.

[5] Douglas N Arnold. Stability, consistency, and convergence of numerical discretizations. *Encyclopedia of Applied and Computational Mathematics*, pages 1358–1364, 2015.