Resolução dos exercícios referentes a decomposição de domínio utilizando o método de Schwarz em laplacianas de 1D e 2D

Henrique Mariano Costa do Amaral[1]
Renan Victor Dias Costa[2]

[1]Graduado em Engenharia Civil (UEMA)
[1]Mestre em Engenharia de Sistemas e Computação (UFRJ)
[1]henriqueamaral@professor.uema.br

[2]Graduando em Eng. da Computação (UEMA)
[2]renancosta1@aluno.uema.br

São Luís
2024

Resumo

Para resolução de problemas físicos que tenham como base eventos da realidade e buscar um resultado com o menor custo computacional possível, existe uma técnica conhecida como decomposição de domínio. Para sua aplicação, o método de Schwarz é plenamente necessário, com uma aplicação em Laplacianas, que podem ser unidimensionais ou bidimensionais. Visando demonstrar essa probabilidade, será feita a resolução das atividades presentes no capítulo oito do livro [Danaila et al., 2007], que tratam justamente desse tema; porém, as resoluções serão voltadas para a computação científica e a resolução de problemas reais apartir de linhas de código. Além, será mostrada a teoria por trás da resolução, com a parte matemática, será dissertado sobre a resolução de cada um dos exercícios e será apresentado os erros relativos e absolutos.

Palavras-chave: Decomposição de Domínio, Laplacianas, Método de Schwarz

Abstract

To solve physical problems that are based on reality events and seek a result with the lowest possible computational cost, there is a technique known as domain decomposition. For its application, the Schwarz method is fully necessary, with an application in Laplacians, which can be one-dimensional or two-dimensional. In order to demonstrate this probability, the activities present in chapter eight of the book [Danaila et al., 2007] will be resolved, which deal precisely with this topic; however, the resolutions will be focused on scientific computing and solving real problems using lines of code. In addition, the theory behind the resolution will be shown, with the mathematical part, will be discussed on the resolution of each of the exercises and the relative and absolute errors will be presented.

Keywords: Domain Decomposition, Laplacians, Schwarz method

Sumário

1 Introdução

Com o desenvolvimento da computação, muitos problemas que antes só detinham uma resolução por meio de métodos matemáticos feitos de forma prática, sendo necessário um custo de tempo muito grande para aqueles que necessitavam realizar certas tarefas do cotidiano.

Com o desenvolvimento dos computadores e sua maior integração em meio a sociedade, além de uma praticidade em seu uso devido aos avanços tecnológicos. Muitos grandes problemas, ao invés de ser necessário uma resolução matemática totalmente nova toda vez que for feito um cálculo, só precisa do conhecimento sobre como fazer a resolução de um problema matemático e como ensinar ao computador como solucionar o problema, assim, gerando uma facilitação de cálculos simples e avançados.

Para demonstrar essa capacidade de resolução de problemas cotidianos usando métodos numéricos, foi criado o livro [Danaila et al., 2007], onde os autores escreveram doze capítulos, com doze casos cotidianos, onde eles demonstram a teoria para resolução do problema, apresentam propostas de exercícios que envolvem desenvolver códigos matemáticos e apresentam as soluções utilizando a linguagem Matlab.

Para fixação de conhecimento, os alunos da disciplina de Cálculo Numérico Avançado foram orientados a escolher um dos capítulos, estudar o escolhido, e fazer códigos computacionais para resolução dos exercícios do livro e elaborar um artigo científico para demonstrar suas respostas e o quão próximas elas estão dos valores que foram alcançados pelos autores do livro.

Para elaboração desse artigo, o capítulo selecionado foi o oito, chamado ”Decomposição de Domínio Usando o Método de Schwarz”, onde esse capítulo trás a teoria do uso do método de Schwarz aplicado a laplacianas unidimensionais e bidimensionais, para fins de realizar a decomposição de domínio, apresenta problemas que exigem elaboração de algoritmos para resolução e apresenta os resultados.

Então, para redigir esse artigo, a estrutura escolhida foi de três seções para demonstrar as três listas de exercícios, onde a primeira é unidimensional, a segunda é uma bidimensional e a terceira é uma bidimensional em um caso especial, onde em cada seção do artigo, será demonstrado o código escrito, onde a linguagem escolhida foi o *python*, para poder realizar essa resolução, além de explicar a teoria por trás da construção desse código, e o erro relativo e absoluto referente ao resultado esperado com base nas respostas contidas no livro e a resposta encontrada.

2 Laplaciana unidimensional

2.1 Fundamentação teórica

Para aplicação da metodologia aplicada, seguindo o guia do [Danaila et al., 2007], onde no desenvolvimento do unidimensional, temos um método que pode vir, por exemplo, a modelar a deflexão de uma viga de comprimento (b - a); tomando noção de suas funções e das grandezas físicas ligadas a esse evento físico, pode-se então realizar a discretização por diferenças finitas da segunda derivada;do resultado conseguinte, com o desenvolvimento do método de Schwarz, é realizada a decomposição do domínio computacional em dois subdomínios com sobreposição e, em seguida, é feita a discretização por diferença finita, com o cálculo de dois vetores com critério de parada no loop de iteração sendo obtido medindo o intervalo entre as duas soluções dentro da região de sobreposição.

2.2 Resolução construída

O código solicita ao usuário o número de pontos na grade, o ponto de iteração e a tolerância, depois é chamada a função que implementa o método de Schwarz, onde ela inicializa os valores de V, W e U, que representam as soluções nos subdomínios. Então ele itera sobre o número especificado, atualizando os valores de acordo com o método de Schwarz. A cada iteração, os erros são calculados e armazenados em uma lista. O critério de parada é quando a norma da diferença entre U e V é menor que a tolerância. Conseguinte, a função plota as soluções V, W e U em cores diferentes, mostrando a convergência das soluções. Além, plota a evolução dos erros ao longo das iterações.

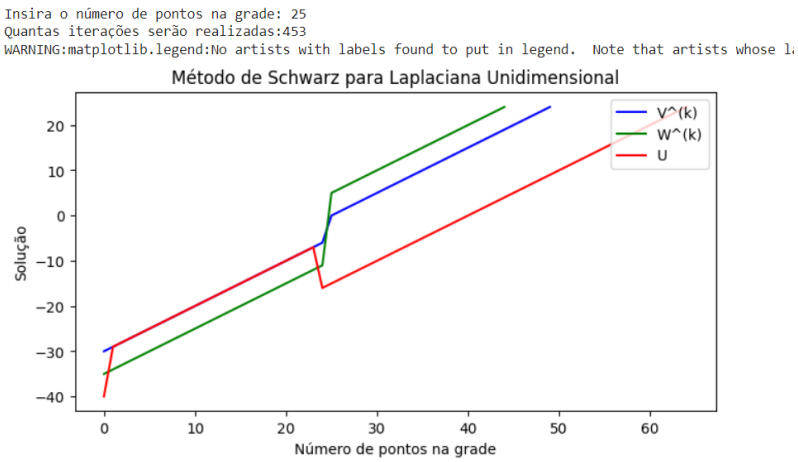

Figura 1: Resultado obtido da plotagem gráfica do exercício.

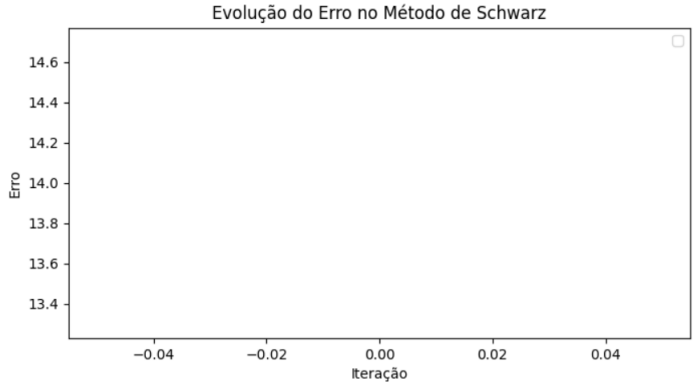

Figura 2: Plotagem da margem de erro detectada.

Na resolução do exercício conseguinte, a função executa o método de Schwarz para resolver a equação de Laplace. Ela recebe como entrada o tamanho inicial da grade (ninicial), o número de iterações (iteracao), a tolerância para o critério de parada (tolerancia), e o tamanho da sobreposição entre as regiões (tamanhooverlap). Uma vez que adentra à função, são inicializadas as variáveis V, W e U, que representam as soluções aproximadas em cada região, e é inicializada uma lista erros para armazenar os erros em cada iteração. O método continua iterando até que o erro entre duas iterações consecutivas seja menor que a tolerância especificada e, em paralelo, o tempo de cálculo e o número de iterações são registrados para fins de análise

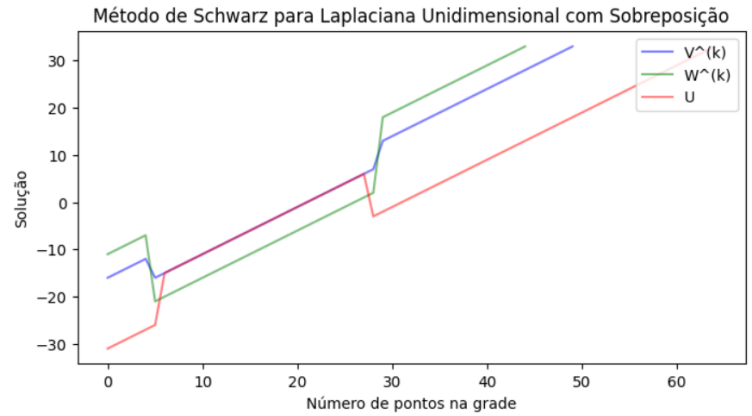

Figura 3: Resultado obtido da plotagem gráfica do exercício, com sobreposição.

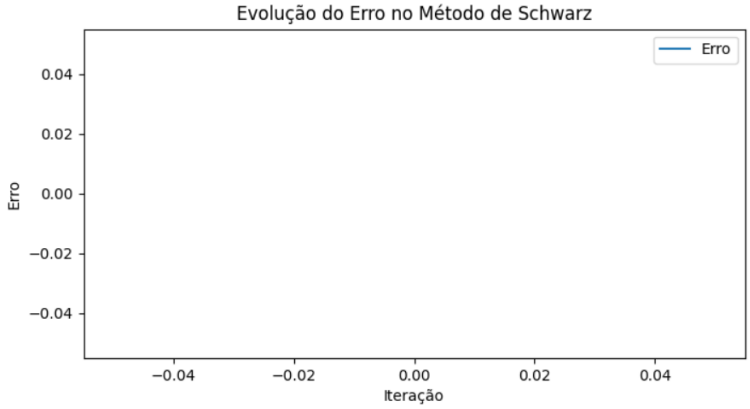

Figura 4: Plotagem da margem de erro detectada com a condição de sobreposição.

3 Laplaciana bidimensional

3.1 Fundamentação teórica

Com base nos estudos do [Danaila et al., 2007], foi estruturado de primeira forma impondo na fronteira as condições de Dirichlet não homogêneas. Além de apresentar uma aplicação real com um ensaio de choque térmico numa viga metálica.

Em continuidade, ocorre a restrição ao caso em que as dimensões do domínio são tais que é possível usar a mesma etapa de discretização em ambas as direções e em seguida, usando expansões de Taylor para aproximar as derivadas parciais em ambas as direções, se obtem a discretização Laplaciana com um esquema de cinco pontos, sendo cada linha do sistema linear possuidora de, no máximo, cinco termos diferentes de zero e podemos construir o simbolismo da matriz de blocos: os graus de liberdade são vizinhos na grade e também consecutivos na numeração global do graus de liberdade.

Para decomposição de domínio nesse caso, o domínio global foi decomposto em "n"subdomínios na direção x_2, então, para tornar mais fácil a implementação, é assumido que o domínio pode ser discretizado com o mesmo passo em ambas as direções, é imposto que todos os subdomínios tenham o mesmo tamanho e que todas as regiões de sobreposição tenham o mesmo número de células da grade. O livro demonstrou um exemplo de decomposição que satisfaz essas restrições.

A primeira iteração, tem a condição de contorno na extremidade direita do subdomínio como não definida, exceto na última, onde é a condição de contorno global, mas para todos os outros deve ser fixado arbitrariamente, por exemplo, por interpolação linear das condições de contorno globais.

3.2 Resolução construída

A função recebe os parâmetros a1, a2, n1, n2 e h. a1 e a2, que representam os limites do domínio no eixo x, n1 e n2 são o número de pontos de grade nos eixos x e y, respectivamente, e h é o tamanho do passo. A função começa criando uma matriz tridiagonal T de tamanho n1 x n1, que representa a discretização unidimensional da equação de Laplace no eixo x, onde esta matriz é criada usando scipy.sparse.diags, que permite criar matrizes esparsas tridiagonais. Após, uma matriz de identidade I de tamanho n1 x n1 é criada e a função usa scipy.sparse.kron para calcular o produto Kronecker entre T e uma matriz de identidade n2 x n2, representando a discretização no eixo y, e entre uma matriz de identidade n1 x n1 e T. Isso resulta em duas matrizes esparsas, que são somadas para obter a matriz A final de tamanho n1*n2 x n1*n2. O código então demonstra um exemplo de uso da função, especificando valores para a1, a2, n1, n2 e calculando h. Em seguida, ele chama a função DDMLaplaceDirichlet e imprime a matriz A.

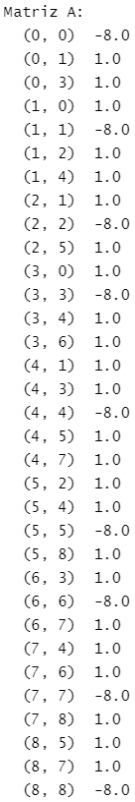

Figura 5: Construção da Matriz A.

Então, uma função calcula o vetor do lado direito (RHS - Right Hand Side) do sistema linear resultante da discretização da equação diferencial parcial bidimensional. Ela recebe como entrada os limites do domínio (a1, a2), o número de pontos discretizados em cada direção (n1, n2), o tamanho do passo (h) e a função que define o lado direito da equação diferencial (rhs2d), onde inicialmente, um vetor B de zeros é criado para armazenar os valores do lado direito do sistema linear. O código itera sobre os pontos internos da grade bidimensional, excluindo as bordas, pois as condições de contorno geralmente são aplicadas lá. Para cada ponto interno (i, j), o código calcula o índice global k correspondente e as coordenadas físicas (x1, x2) associadas a esse ponto e, utilizando as coordenadas (x1, x2) do ponto atual, a função rhs2d é chamada para calcular o valor da função do lado direito da equação diferencial. Conseguinte, o valor calculado é atribuído à posição correspondente no vetor B. Após preencher o vetor B com os valores do lado direito em cada ponto interno da grade, o vetor é retornado e a função de exemplo que calcula o valor da função do lado direito da equação diferencial.

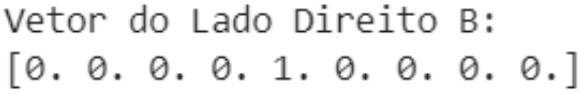

Figura 6: Valores do lado direito do vetor B.

Uma nova função utiliza o método de DDM para resolver a equação de Laplace bidimensional com condições de contorno definidas. Ela recebe como entrada os limites

do domínio (a1, a2, b1, b2), o número de pontos discretizados em cada direção (n1, n2), as funções que definem as condições de contorno (f1, f2, g1, g2), a função que define o lado direito da equação diferencial (rhs2d), e retorna a solução da equação, onde o código calcula o tamanho do passo de discretização h1 e h2 em cada direção, com base nos limites do domínio e no número de pontos discretizadose a função Laplace é chamada para construir a matriz do sistema linear A usando o método de discretização de diferenças finitas para a equação de Laplace. Esta matriz representa o operador Laplaciano discretizado no domínio bidimensional. Uma função RightHandSide2d é chamada para construir o vetor do lado direito B do sistema linear. Isso é feito calculando os valores da função do lado direito da equação diferencial em cada ponto interno da grade. O sistema linear A * X = B é resolvido utilizando a função solve do scipy.linalg. A solução obtida é convertida em uma matriz bidimensional Solm para facilitar a visualização e a manipulação e as condições de contorno são aplicadas aos bordos da grade bidimensional, utilizando as funções f1, f2, g1 e g2. Para fins de exemplificação, a função retorna zero em todos os pontos, mas pode ser modificada para representar qualquer função desejada.

```
Solução da equação de Laplace com condições de contorno:
[[ 0.  0.  0.  0.  0.]
 [ 0. -0. -0. -0.  0.]
 [ 0. -0. -0. -0.  0.]
 [ 0. -0. -0. -0.  0.]
 [ 0.  0.  0.  0.  0.]]
```

Figura 7: Plotagem de uma função 5 x 5 preenchida com zeros.

Para esse resultado, ocorre a criação dos pontos de grade, onde define intervalos ao longo dos eixos respectivamente. Há o uso de np.linspace para criar 100 pontos igualmente espaçados ao longo de cada intervalo. Após, usasse np.meshgrid para criar uma grade bidimensional com todos os pares ordenados de pontos. Em seguida é feita uma avaliação das funções nos pontos da grade e se usa os pontos de grade para avaliar as funções. Conseguinte, usa-se os comandos plot.plot para traçar as condições de contorno e plot.contourf para plotar o mapa de contorno da solução exata.

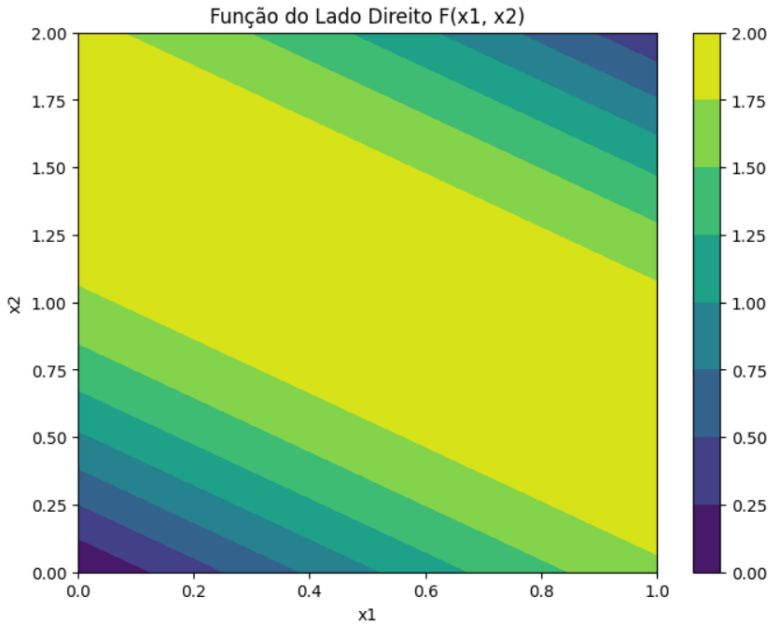

Figura 8: Representação da correlação de x1 e x2 na função do lado direito, com esquema de cores.

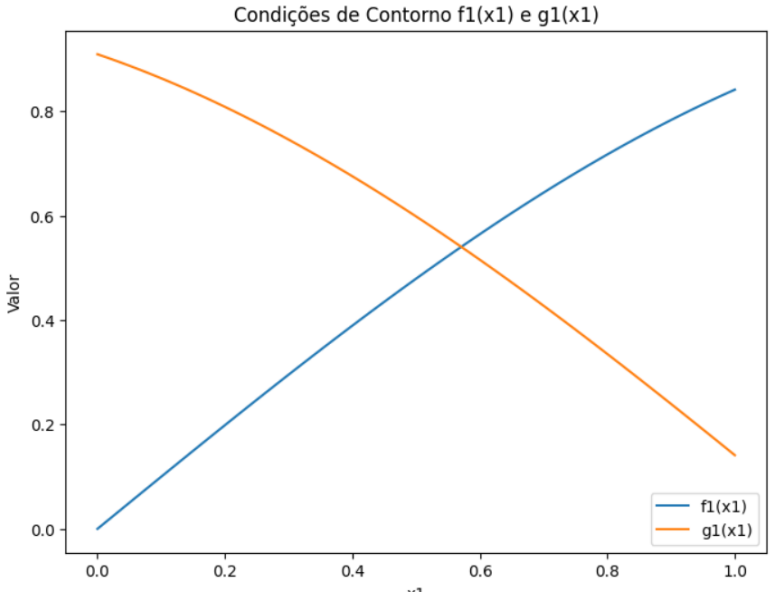

Figura 9: Condições de contorno, com a presente queda de valor de g1 e aumento de f1.

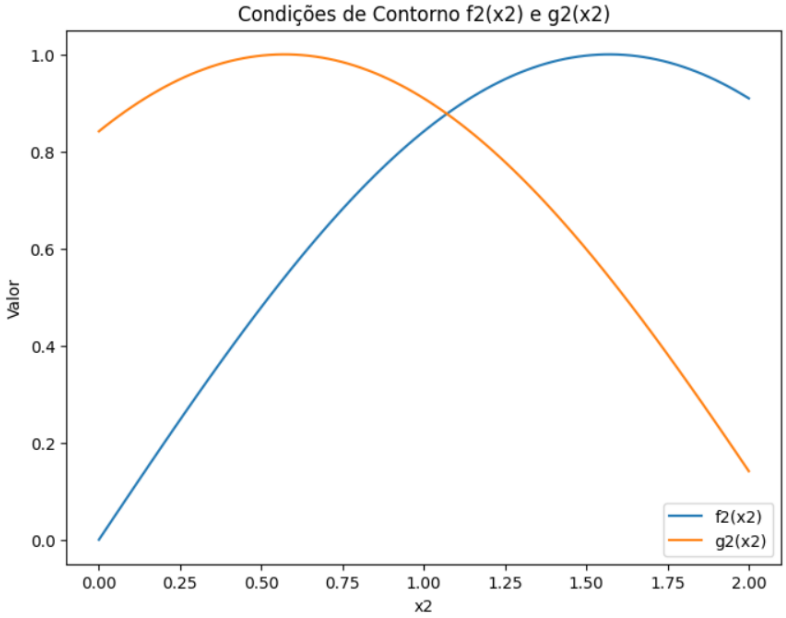

Figura 10: Condições de contorno, com a presente queda de valor de g2 e aumento eindicio de uma queda de f2

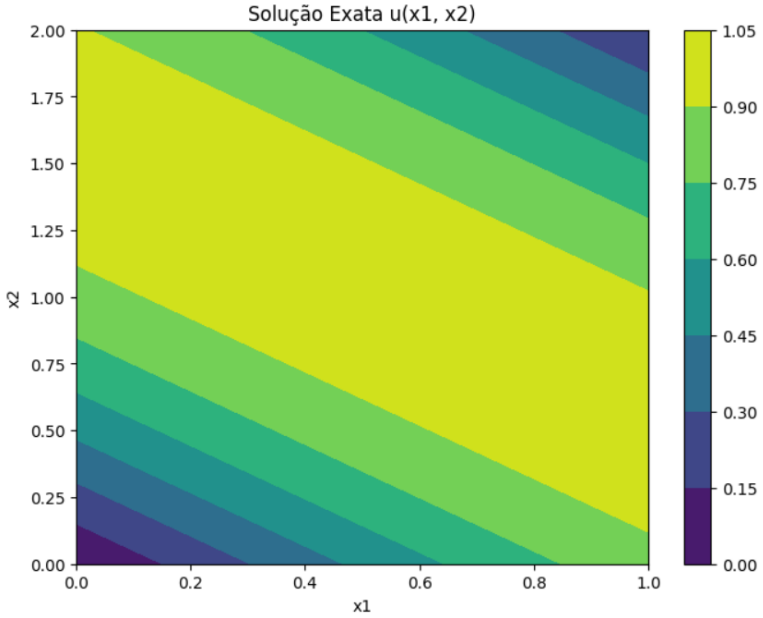

Figura 11: Resultado obtido da plotagem gráfica do exercício.

Para calcular o erro entre a solução exata e a solução aproximada de uma EDP, é calculado como a norma da diferença entre as duas soluções, usando a função np.linalg.norm. Depois se testa as funções previamente definidas para calcular a solução exata e as condições de contorno da EDP. Ela gera uma grade de pontos no domínio bidimensional especificado pelos limites a1, a2, b1 e b2, e calcula os valores da função do lado direito, das condições de contorno e da solução exata em cada ponto da grade. Ocorre a plotagem do erro entre a solução exata e a solução aproximada, imprimindo o valor do erro; os

parâmetros do problema são definidos (a1, a2, b1, b2, n1, n2), e as funções são testadas chamando TestFinDif2d. A solução exata e as condições de contorno são calculadas, e o erro entre a solução exata e ela mesma é impresso.

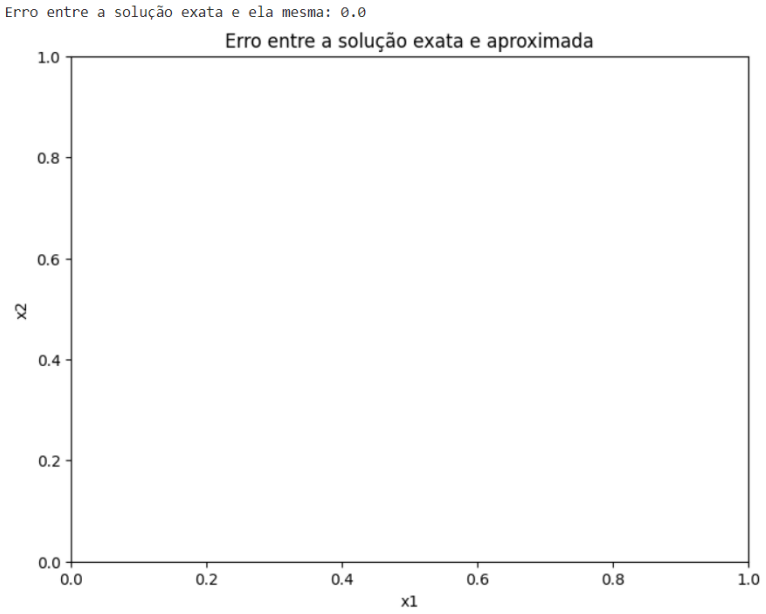

Figura 12: Margem de erro entre a solução exata e a aproximada encontrada

O lado direito da equação diferencial parcial (EDP) bidimensional para o problema do choque térmico. Neste caso, a função do lado direito é definida. Uma nova função calcula a solução exata da EDP bidimensional para o problema do choque térmico. Os limites do domínio são modificados, ampliando o domínio bidimensional para um intervalo maior. São criados vetores x1 e x2 espaçados uniformemente entre os limites do domínio, usando np.linspace. A função np.meshgrid é usada para criar uma grade bidimensional de pontos X1 e X2 a partir dos vetores x1 e x2, que representam as coordenadas dos pontos no domínio. O lado direito da EDP é calculado usando a função rhs2dThermalShock nos pontos da malha X1 e X2.

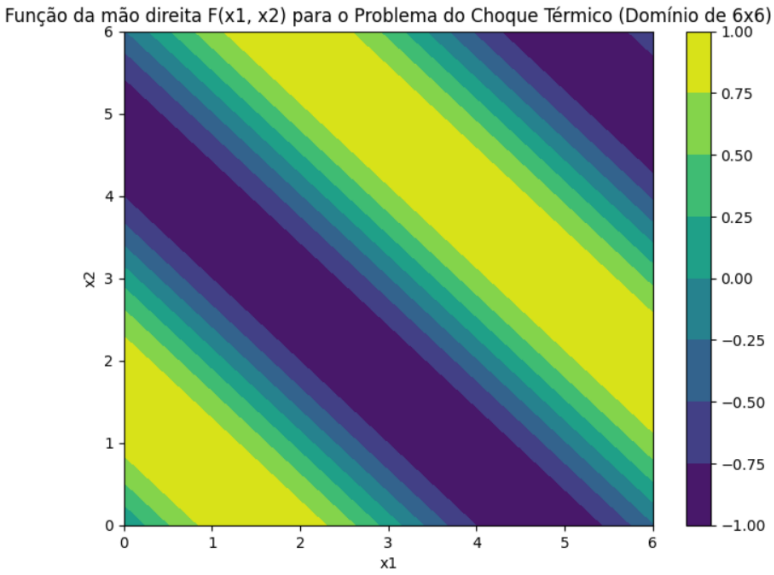

Figura 13: Plotagem da função da mão direita do choque térmico, em 6 x 6.

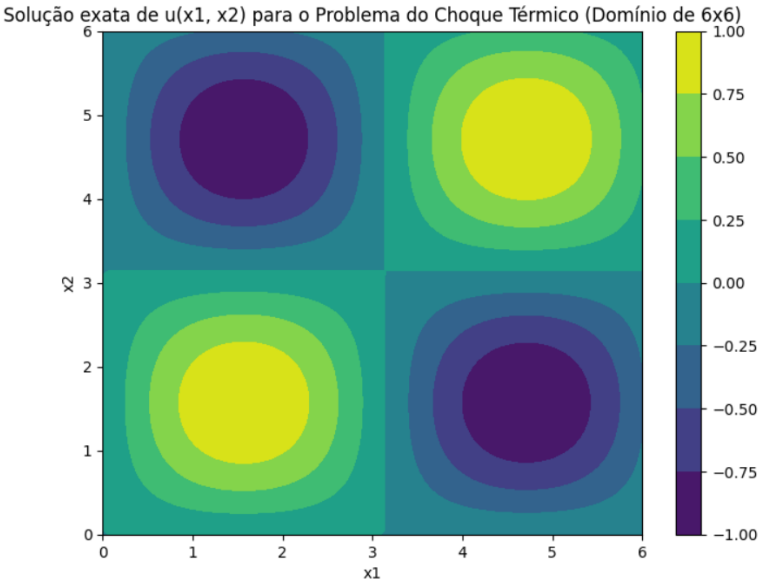

Figura 14: Solução exata dem 6 x 6.

Com a modificação do tamanho do domínio, os limites do domínio são modificados, ocorrendo o aumento a sua extensão. Os vetores x1 e x2 são espaçados uniformemente entre os limites do domínio estendido, utilizando np.linspace. Uma função np.meshgrid é usada para criar uma grade bidimensional de pontos X1 e X2 a partir dos vetores x1 e x2, representando as coordenadas dos pontos no domínio estendido. Então, o lado direito da equação diferencial parcial (EDP) é calculado usando a função rhs2dThermalShock nos pontos da malha X1 e X2 e a solução exata da EDP é calculada usando a função u2dThermalShock nos pontos da malha X1 e X2.

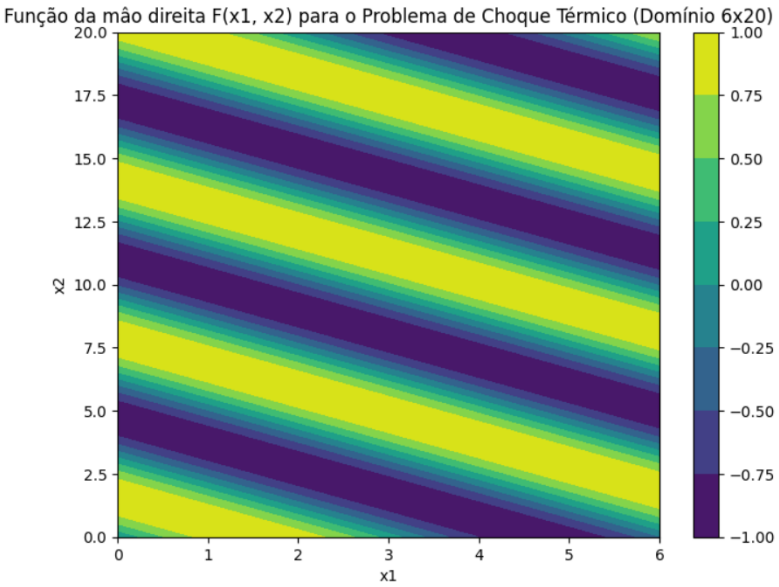

Figura 15: Plotagem da função da mão direita do choque térmico, em 6 x 20.

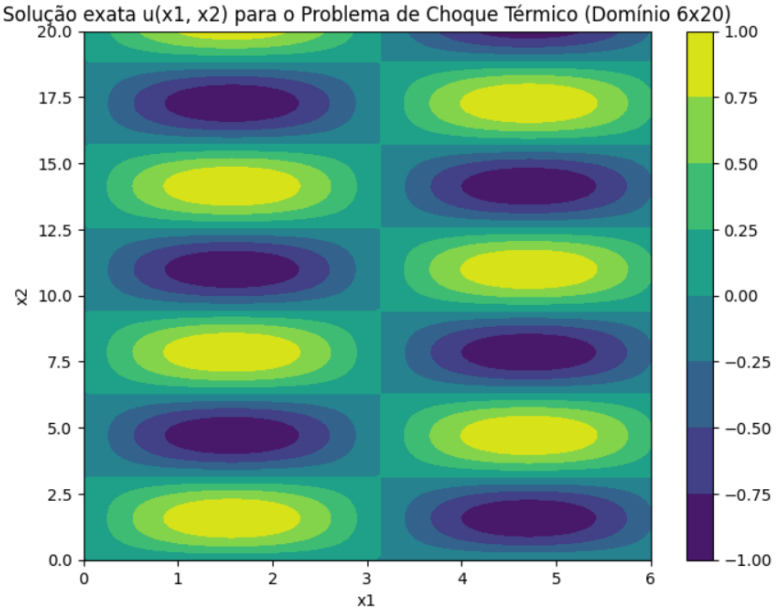

Figura 16: Solução exata em 6 x 20..

4 Laplaciana bidimensional, caso especial

4.1 Fundamentação teórica

Para proposta de um caso especial, o livro [Danaila et al., 2007] introdus as eguintes soluções para resolução do exercício seguinte: em casos de teste de condução de calor mais realistas há o requerimento da implementação de condições de contorno adicionais além da de Dirichlet que foi usada no anterior.

Para demonstração da resolução, ele relaciona o calor produzido pelo campo elétrico, sendo ele a uma taxa uniforme, onde são impostas temperaturas aos eletrodos na extremidade esquerda e na direita, sendo a temperatura da extremidade esquerda, maior.

As duas faces laterais da barra, bem como a face inferior, são isoladas, o que significa que uma condição de contorno de Neumann deve ser imposta. Na face superior, é imposto uma condição de contorno de Fourier, ou Robin, para modelar o fenômeno natural de resfriamento por convecção. Após, temos o coeficiente de transferência térmica e a temperatura externa, além de termos atribuido o coeficiente de difusidade.

No final, com a união de tudo, temos um caso particular. Para discretizar estas condições de contorno de Neumann e Fourier, os graus de liberdade correspondentes a estes nós são introduzidos no sistema, gerando 4 fórmulas, sendo a condição de Neumann, a condição de Fourrier e duas discretizações correlacionadas a cada condição.

4.2 Resolução construída

Para criação da condição especial, o domínio retangular é discretizado em uma grade usando os parâmetros especificados (comprimento, largura e número de pontos na grade). A temperatura inicial da placa é definida como sendo igual à temperatura ambiente em toda parte, exceto na primeira coluna (representando um eletrodo), onde é definida uma temperatura mais alta. As condições de contorno são aplicadas na borda da placa, onde uma condição de convecção é modelada na borda esquerda. Logo, o tempo é discretizado em intervalos regulares, e os parâmetros de tempo e espaço são calculados com base nas dimensões da placa e nas taxas de transferência de calor e função update temperature calcula a temperatura em cada ponto da grade para o próximo passo de tempo com base na equação do calor e nas condições de contorno e, a cada intervalo de tempo, a distribuição de temperatura é plotada em um gráfico tridimensional usando matplotlib. O loop só é interrompido quando o tempo de simulação atinge o tempo final especificado.

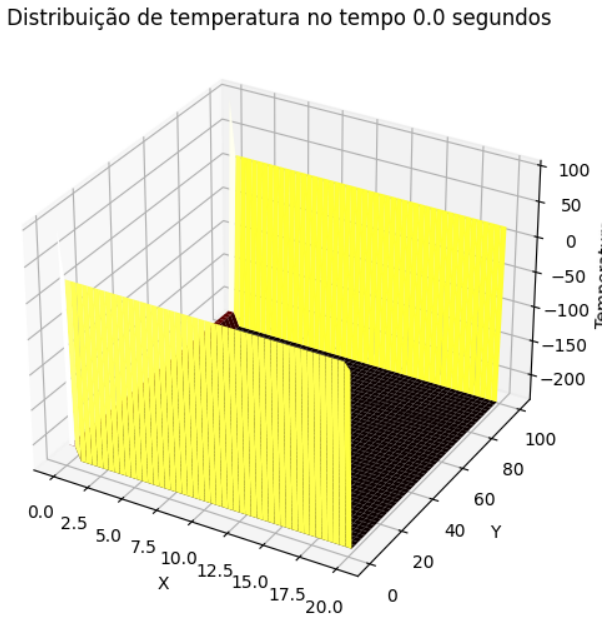

Figura 17: Plotagem do gráfico 3D no tempo de 0 segundos.

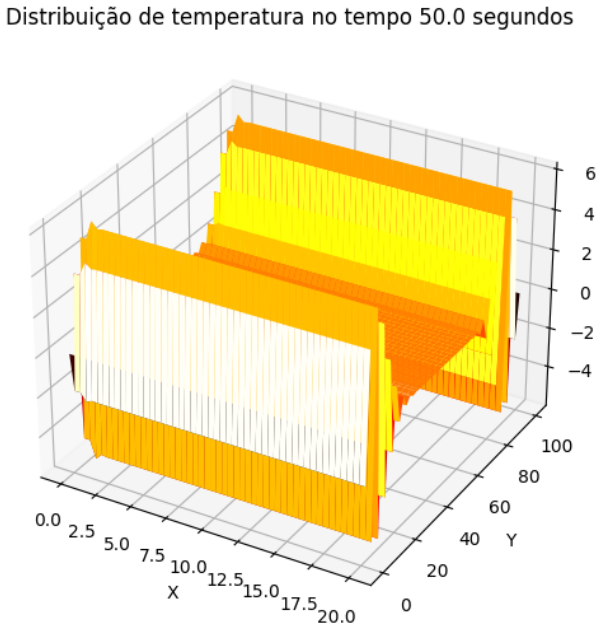

Figura 18: Plotagem do gráfico 3D no tempo de 50 segundos.

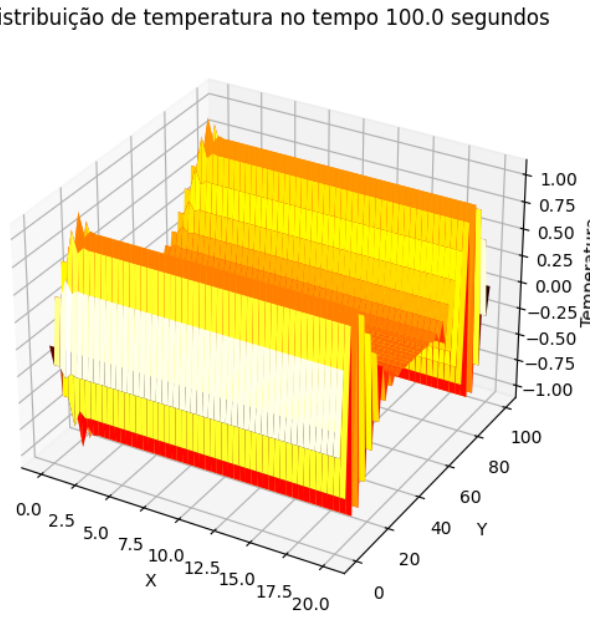

Figura 19: Plotagem do gráfico 3D no tempo de 100 segundos.

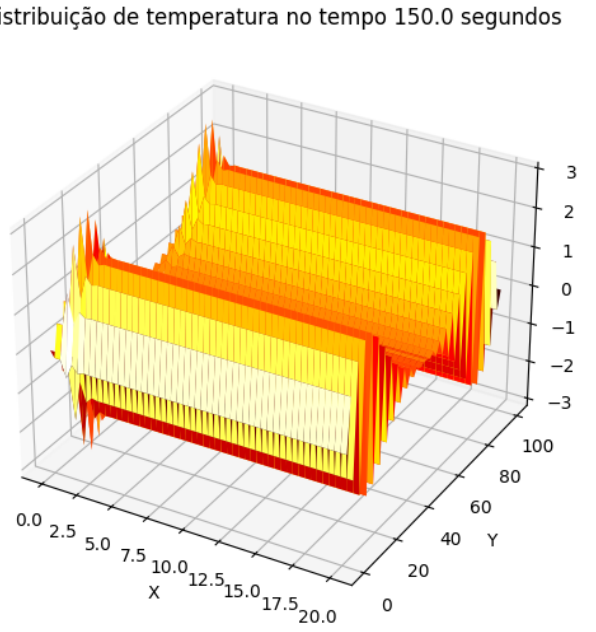

Figura 20: Plotagem do gráfico 3D no tempo de 150 segundos.

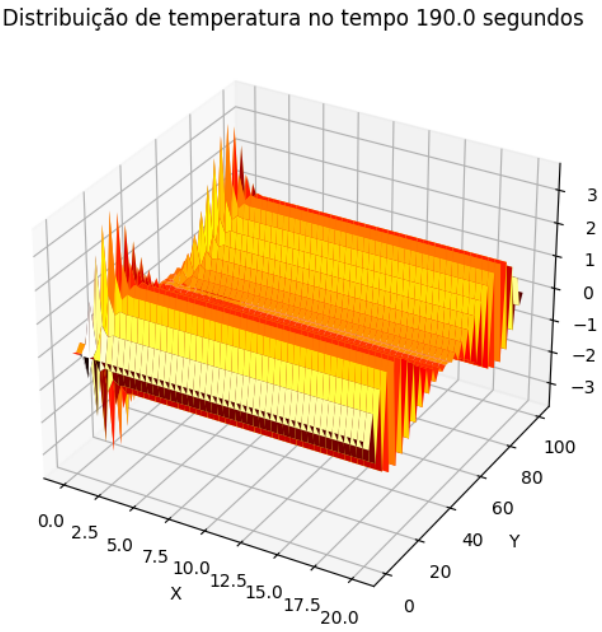

Figura 21: Plotagem do gráfico 3D no tempo de 190 segundos.

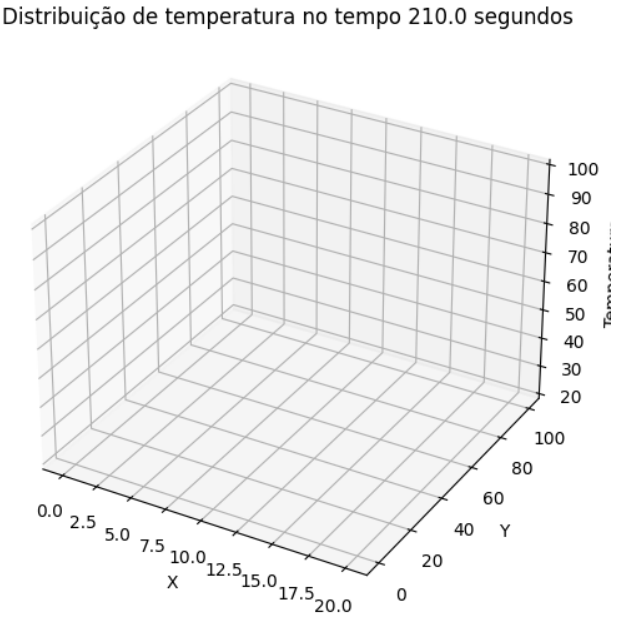

Figura 22: Plotagem do gráfico 3D no tempo de 210 segundos, quando o calor acaba

Nesse exercício, o domínio bidimensional é discretizado em uma grade retangular com Nx pontos na direção x e Ny pontos na direção y. O tamanho do passo na direção x é hx e na direção y é hy. Além disso, o parâmetro alpha é uma constante relacionada à difusividade térmica do material, dt é o intervalo de tempo e k é a condutividade térmica e a matriz do sistema é construída a partir de blocos diagonais e fora da diagonal. Os blocos diagonais contêm os termos relacionados aos pontos centrais da grade, enquanto os blocos fora da diagonal contêm os termos relacionados aos pontos vizinhos na direção x e y. A

matriz final é montada combinando os blocos diagonais e fora da diagonal de acordo com a estrutura específica do método de Laplace-Fourier. Então, faz-se uso da função plot.spy() é usada para visualizar a estrutura esparsa da matriz, mostrando os pontos onde os elementos são diferentes de zero.

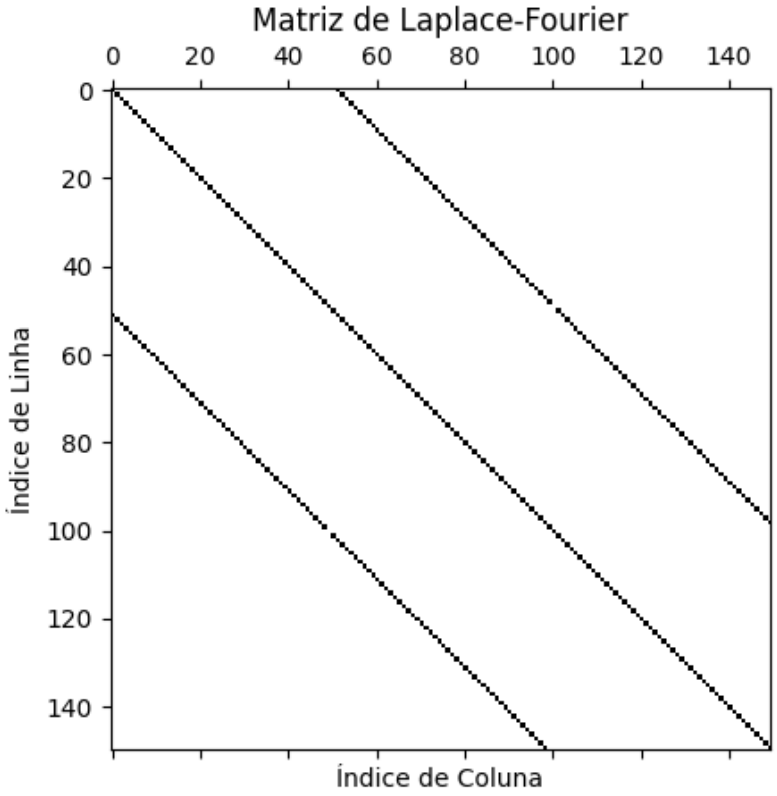

Figura 23: Representação gráfica da matriz de Laplace-Fourier

Para solucionar o Bloco de Base, a função calcula o lado direito da equação usando o método de Fourier. O calor é gerado uniformemente à direita do domínio. As funções representam as condições de contorno de Bloco de Base (BB) para as bordas inferior e superior do domínio, respectivamente. Uma nova função calcula o lado direito da equação usando as condições de contorno de Bloco de Base (BB). O calor é aplicado nas bordas inferior e superior do domínio de acordo com as funções de contorno definidas, então o código calcula o lado direito da equação usando ambas as abordagens e plota os resultados para comparação.

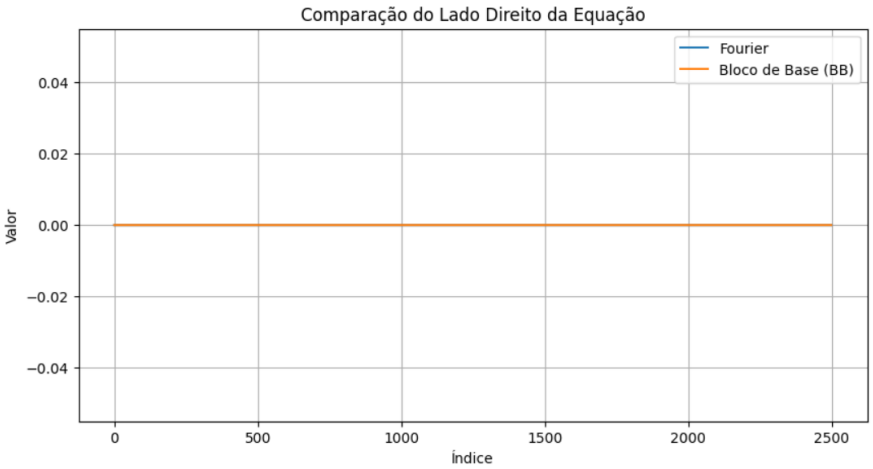

Figura 24: Comparação do lado direito da equação, numa correlação índice x valor.

5 Conclusão

Para desenvolvimento dos códigos que fizeram a resolução das atividades, foi encontrado quatro questões que não encontraram uma resolução. Por essa falta de resolução, não foram inseridos no artigo, mas devem ser citados a não eficiência nessa resolução por parte do aluno.

Porém, mesmo com essas dificuldades, acreditasse que tenha sido um excelente aprendizado para o aluno. Apartir desse projeto, ele experimentou a aplicação real dos seus conhecimentos dentro de um projeto, onde foi realizado a maioria do trabalho que lhe foi conferido.

Dos resultados obtidos, pode-se destacar que ocorreu um bom entendimento do assunto e um bom conhecimento em demonstrá-lo por meio de plotagens gráficas, umas vez que, o gráfico em si não é o objetivo, mas a resolução do problema real por meio de linhas de algoritmos e essas plotagens demonstram apenas a segmentação de como o computador vai realizando a função ou como um problema se comporta num gráfico.

Referências

[Danaila et al., 2007] Danaila, I., Joly, P., Kaber, S. M., and Postel, M. (2007). *An introduction to scientific computing: Twelve computational projects solved with MATLAB*. Springer.