Universidade do Minho
Escola de Ciências

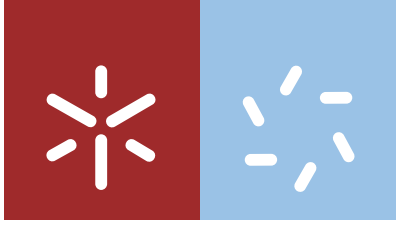
Osvaldo Pedro Candeia Jamba

Resolução Numérica da Equação de Poisson

outubro de 2019

Osvaldo Pedro Candeia Jamba **Resolução Numérica da Equação de Poisson**

UMinho|2019



Universidade do Minho
Escola de Ciências

Osvaldo Pedro Candeia Jamba

Resolução Numérica da Equação de Poisson

Dissertação de Mestrado
Mestrado em Matemática e Computação

Trabalho realizado sob orientação do
Professor Doutor Rui Manuel Silva Ralha
e da
Professora Doutora Carla Maria Alves Ferreira

outubro de 2019

AGRADECIMENTOS

Agradeço a Deus Todo Poderoso, criador de todo conhecimento, pela saúde, por me conduzir, iluminar e conceder força para superar todos os obstáculos, sem nunca perder a esperança, mesmo quando tudo parecia perdido.

Agradeço à minha esposa, meus pais e irmão, pelo apoio incondicional que me deram durante a formação, pois sem eles nada teria sido possível.

Agradeço aos Professores José Espírito Santo, Pedro Patrício, José Bacelar, Rui Ralha, José Valença, Stéphane Clain, Victor Alves e José Pina, pela paciência, atenção e disponibilidade que sempre apresentaram, e a todos os professores que direta ou indiretamente influenciaram a minha formação.

Agradeço de modo especial, ao Professor Rui Ralha e à Professora Carla Ferreira, meus orientadores incansáveis, sem eles nada seria possível. Sou grato pela paciência, disponibilidade e entrega que manifestaram durante a elaboração da dissertação.

A todos os meus colegas que não mediram esforços para me ajudar com as dúvidas que surgiram ao longo da minha formação, o meu muito obrigado.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Título da dissertação: Resolução Numérica da Equação de Poisson.

Esta dissertação tem como objetivo a resolução numérica da equação de Poisson com condições de fronteira de Dirichlet. Como exemplo de aplicação, consideramos o problema da determinação dos valores de equilíbrio da temperatura em pontos do interior de um domínio plano (quadrado). A discretização do problema é feita com a fórmula clássica dos cinco pontos. Para a resolução do sistema de equações, limitar-nos-emos a usar os métodos iterativos de Jacobi, de Gauss-Seidel e SOR (sobre-relaxação sucessiva). Apresentamos a teoria relevante para o problema, em particular no que diz respeito à convergência dos métodos referidos. Desenvolvemos e testámos códigos Matlab que implementam os métodos.

Palavras-Chaves: Equação de Poisson, diferenças finitas, métodos iterativos, convergência.

Abstract

Dissertation Title: Numerical Resolution of Poisson's Equation.

The subject of this dissertation is the numerical resolution of Poisson's equation with Dirichlet's boundary conditions. As an example of application, we consider the problem of determining the steady state values of temperature at points in the interior of a flat (square) domain. The discretization of the problem is carried out with the classic five points formula. For the solution of the linear system of equations, we limit ourselves to using the iterative methods of Jacobi, Gauss-Seidel and SOR (successive over-relaxation). We present the theory relevant to the problem, in particular with regard to the convergence of the referred methods. We have developed and tested Matlab codes that implement the methods.

Keywords: Poisson's equation, finite differences, iterative methods, convergence.

Conteúdo

Lista de Tabelas

Lista de Figuras

Introdução

A equação de Poisson é uma equação diferencial às derivadas parciais que ocorre com frequência em muitas áreas, por exemplo em eletrostática, engenharia mecânica e física teórica. Na impossibilidade de resolver exatamente a equação por métodos da análise matemática, a alternativa é calcular numericamente a solução de cada problema concreto e tal implica a resolução de um sistema de equações lineares.

Os métodos para resolver sistemas de equações lineares dividem-se em duas categorias: *métodos diretos* e *métodos iterativos*. Os primeiros, como é o caso do método de eliminação de Gauss, permitem obter a solução do sistema após um número finito de operações aritméticas e são, em geral, usados nos casos em que a matriz é densa, isto é, a percentagem de entradas iguais a zero é baixa. A solução produzida por um método direto só não será exata por culpa dos erros de arredondamento. Os métodos iterativos produzem uma sucessão de aproximações que, idealmente, converge para um limite que é a solução procurada. Por outras palavras, sempre que tomamos como solução uma aproximação calculada por um método iterativo estamos a cometer também um erro de truncatura e este erro não existe quando se usa um método direto. Tal não implica que os métodos diretos produzam melhores aproximações do que os métodos iterativos. Estes são capazes de produzir aproximações com erros pequenos, sempre dependente da precisão da aritmética usada. Além disto, são mais flexíveis já que o critério de paragem é fixado em função da qualidade que se deseja para aproximação a calcular. Os métodos iterativos são muito populares para os sistemas de grande dimensão que apresentam elevada percentagem de coeficientes nulos (designamos estes sistemas/matrizes por esparsos/esparsas, tradução do inglês *sparse*). A razão é a seguinte: ao contrário do que acontece nos métodos diretos, em que a matriz do sistema sofre sucessivas alterações, esta não é alterada nos métodos iterativos. Na verdade, podemos neste caso pensar na matriz, digamos A, como uma "caixa preta" que recebe um vetor $\boldsymbol{x}$ e produz o vector $A * \boldsymbol{x}$, uma vez que a multiplicação matriz-vetor é a forma como a matriz é usada no contexto dos métodos iterativos.

O armazenamento de matrizes esparsas, "grandes", na forma usual de um "array" de duas dimensões requer quantidades elevadas de memória que serão ocupadas com zeros. É certo que existem formas de *armazenamento compacto* de matrizes que podem ser usadas para armazenar apenas as entradas não nulas e que têm flexibilidade para guardar mais

entradas não nulas, à medida que elas vão sendo produzidas por um método direto (na terminologia inglesa, isto costuma designar-se por *fill-in*). O grau, maior ou menor, de *fill-in* que ocorre depende da forma como as entradas nulas estão distribuídas (o padrão de esparsidade) e também da forma como a transformação da matriz é feita. Por exemplo, no caso de uma matriz em que as entradas não nulas estão confinadas a uma "banda" de $2n+1$ diagonais, a diagonal principal e n diagonais imediatamente acima e abaixo desta, o método de eliminação de Gauss sem pivotação não provocará o aparecimento de qualquer entrada não nula fora daquela banda. Mesmo nesta situação poderá ser mais vantajoso usar métodos iterativos se n for grande e dentro da banda a maioria das entradas forem nulas. A vantagem é de poder resolver problemas de grande dimensão sem necessidade de armazenar grandes quantidades de dados (muitas vezes, isto significa poder resolver problemas de maior dimensão com a memória disponível) e também reduzir o tempo de computação.

O sistema de equações que resulta da discretização da equação de Poisson com fórmulas de diferenças finitas é esparso. No caso do problema a uma única dimensão, a matriz do sistema é tridiagonal o que dificilmente justifica a utilização de métodos iterativos. Mas já se justifica no caso do número de dimensões ser dois (usaremos as siglas 1D e 2D para nos referirmos ao problema a uma e a duas dimensões, respetivamente). Mas começaremos por analisar o caso 1D já que, como veremos, a matriz para o caso 2D é obtida a partir da matriz em 1D à custa de somas de Kronecker.

Esta dissertação é pois uma incursão no vastíssimo tema dos métodos iterativos para sistemas de equações lineares no contexto da resolução numérica da equação de Poisson. No Capítulo 1 apresentamos uma compilação de resultados básicos que ajudam à sustentação teórica do que se faz nos capítulos seguintes, em particular, no que diz respeito à análise da convergência. No Capítulo 2 apresentamos os métodos clássicos de Jacobi, Gauss-Seidel e SOR (sobre-relaxação sucessiva). Em particular, apresentamos a dedução da matriz de iteração de cada um daqueles métodos bem como a análise dos respectivos raios espectrais daquelas matrizes.

No Capítulo 3, começamos por apresentar um problema concreto cujo modelo matemático é a equação de Poisson: a determinação dos valores de equilíbrio da temperatura em cada ponto interior de uma placa sobre cujos bordos se aplicam fontes de calor de valor constante (condições de fronteira do tipo Dirichlet). Fazemos, em seguida, a derivação das relações das incógnitas (temperatura em cada um dos pontos) correspondentes a pontos vizinhos nas malhas, que resultam da discretização da equação usando diferenças centradas para aproximar as derivadas de segunda ordem presentes na equação.

No Capítulo 4, apresentamos resultados numéricos de testes realizados com os códigos que desenvolvemos no Matlab. Os resultados correspondem a dois problemas que diferem apenas nas condições de fronteira. Num caso a solução exata é conhecida, no outro não. Para cada método, desenvolvemos dois códigos, um que armazena as matrizes de forma

explícita e que, portanto, não tira partido da estrutura especial exibida e outro que não requer o armazenamento de qualquer matriz e simplesmente implementa cada produto matriz-vetor em termos das relações conhecidas entre as incógnitas. Comparámos tempos de execução e número de iterações realizadas. No caso do SOR, quisemos também testar experimentalmente o que a teoria indica relativamente ao valor ótimo do parâmetro de relaxação.Também comparámos a convergência do SOR para duas ordenações diferentes dos nós da malha: a ordenação *natural* e a ordenação *Red-Black*, esta última de interesse para a computação paralela. Finalmente, algumas conclusões são apresentadas no último capítulo.

Capítulo 1

Conceitos básicos e resultados preliminares

1.1 Normas

As normas de vetores e normas de matrizes são ferramentas imprescindíveis no estudo dos algoritmos da álgebra linear numérica. De uma maneira informal, estas normas são usadas para medir globalmente o tamanho, em valor absoluto, das entradas dos vetores e matrizes.

1.1.1 Normas vetoriais

Uma norma vetorial é uma generalização do módulo de um escalar, é um número real não negativo que representa o *comprimento* do vetor. Utilizaremos apenas as normas-p, com p um número inteiro, que são definidas em $\mathbb{R}^n$ da seguinte forma

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \qquad \text{com} \qquad 1 \leqslant p < \infty.$$

O caso em que $p = 2$ conduz à *norma euclidiana*

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}. \tag{1.1}$$

Outras normas muito utilizadas correspondem ao caso em que $p = 1$

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|, \tag{1.2}$$

que é conhecida como a *norma da soma*, e ao caso em que $p = \infty$

$$\|\boldsymbol{x}\|_\infty = \lim_{p\to\infty} \|\boldsymbol{x}\|_p = \max_{1\leqslant i\leqslant n} |\boldsymbol{x}_i|, \tag{1.3}$$

por vezes, designada como a *norma do máximo*.

Para quaisquer vetores $\boldsymbol{x}$ e $\boldsymbol{y}$ e para qualquer escalar α, uma norma vetorial respeitam as seguinte condições

$$\begin{aligned} \|\alpha\boldsymbol{x}\| &= |\alpha|\|\boldsymbol{x}\|, && \text{(homogeneidade)} \\ \|\boldsymbol{x}+\boldsymbol{y}\| &\leqslant \|\boldsymbol{x}\| + \|\boldsymbol{y}\| && \text{(desigualdade triangular)} \\ \|\boldsymbol{x}\| &\geqslant 0 && \text{(positividade)} \\ \|\boldsymbol{x}\| = 0 &\Longrightarrow \boldsymbol{x} = 0 && \text{(definidade).} \end{aligned}$$

Duas normas $\|.\|_a$ e $\|.\|_b$, dizem-se *equivalentes* se existirem constantes $0 < c_1,\, c_2 < \infty$ tais que

$$c_1\|\boldsymbol{x}\|_a \leqslant \|\boldsymbol{x}\|_b \leqslant c_2\|\boldsymbol{x}\|_a,$$

para qualquer vetor $\boldsymbol{x}$. É de enfatizar que todas as normas em espaços lineares de dimensão finita são equivalentes [9, p. 792, Teorema A.3.1.]. Em particular, tem-se

$$\|\boldsymbol{x}\|_2 \leqslant \|\boldsymbol{x}\|_1 \leqslant \sqrt{n}\|\boldsymbol{x}\|_2. \tag{1.4}$$

Observe-se que, para duas normas distintas $\|.\|_a$ e $\|.\|_b$, podemos ter

$$\|\boldsymbol{x}\|_a < \|\boldsymbol{y}\|_a \qquad \text{e} \qquad \|\boldsymbol{x}\|_b > \|\boldsymbol{y}\|_b. \tag{1.5}$$

Por exemplo, para $\boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ e $\boldsymbol{y} = \begin{pmatrix} \sqrt{3}/2 \\ \sqrt{3}/2 \end{pmatrix}$ temos $\|\boldsymbol{x}\|_\infty > \|\boldsymbol{y}\|_\infty$ e $\|\boldsymbol{x}\|_2 < \|\boldsymbol{y}\|_2$.

1.1.2 Normas matriciais

Dadas duas matrizes $A, B \in \mathbb{C}^{n\times n}$ e um escalar $\alpha \in \mathbb{C}$, para qualquer norma matricial verificam-se as propriedades

$$\begin{aligned} \|\alpha A\| &= |\alpha|.\|A\| \\ \|A+B\| &\leqslant \|A\| + \|B\| \\ \|A\| &\geqslant 0 \\ \|A\| = 0 &\Longrightarrow A = 0 \\ \|AB\| &\leqslant \|A\|\cdot\|B\| && \text{(submultiplicatividade)} \end{aligned}$$

As primeiras quatro propriedades são as mesmas de uma norma vetorial, apenas se acrescenta a propriedade de submultiplicatividade.

Uma classe importante de normas de matrizes são as normas *induzidas* pelas normas vetoriais e que se definem por

$$\|A\| = \max_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|},$$

o que é equivalente a

$$\|A\| = \max_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|. \tag{1.6}$$

Em geral o cálculo de uma norma matricial não é fácil, mas existem algumas exceções com interesse prático.

Temos, por exemplo, a norma matricial $\|\cdot\|_\infty$ definida por

$$\|A\|_\infty = \max_{1\leqslant i\leqslant n} \left(\sum_{j=1}^{n} |a_{ij}| \right) \tag{1.7}$$

que corresponde ao máximo das somas por linhas (em valor absoluto) e é induzida pela norma vetorial (1.3). De facto, usando a definição (1.6), se tomarmos um vetor $\boldsymbol{x}$ tal que $\|\boldsymbol{x}\|_\infty = 1$ significa que $|x_i| \leqslant 1$, $i = 1, \ldots, n$, e existe pelo menos um índice i tal que $|x_i| = 1$. Temos, então, que

$$\|A\boldsymbol{x}\|_\infty = \max_i \left| \sum_{j=1}^{n} a_{ij}x_j \right| \leqslant \max_i \sum_{j=1}^{n} |a_{ij}||x_j| \leqslant \max_i \sum_{j=1}^{n} |a_{ij}|$$

e basta agora exibir um vetor $\boldsymbol{x}$ para o qual se tem a igualdade para chegarmos à definição (1.7). Seja, então, k tal que

$$\max_i \sum_{j=1}^{n} |a_{ij}| = \sum_{j=1}^{n} |a_{kj}|$$

e seja $\boldsymbol{x}$ definido por

$$x_j = sign(a_{kj}) = \begin{cases} 1, & \text{se } a_{kj} > 0 \\ 0, & \text{se } a_{kj} = 0 \\ -1, & \text{se } a_{kj} < 0 \end{cases}.$$

Neste caso vem

$$\left| \sum_{j=1}^{n} a_{kj}x_j \right| = \sum_{j=1}^{n} |a_{kj}||x_j| = \sum_{j=1}^{n} |a_{kj}|$$

e concluímos que

$$\|A\boldsymbol{x}\|_\infty = \sum_{j=1}^{n} |a_{kj}|.$$

Temos também a norma matricial $\|\cdot\|_1$ definida por

$$\|A\|_1 = \max_{1\leqslant j\leqslant n} \left(\sum_{i=1}^{n} |a_{ij}| \right) \tag{1.8}$$

que corresponde ao máximo das somas por colunas (em valor absoluto) e é induzida pela norma vetorial (1.2). Com efeito, analogamente ao caso anterior, para um vetor $\boldsymbol{x}$ tal que $\|x\|_1 = 1$, obtemos

$$\begin{aligned}
\|A\boldsymbol{x}\|_1 &= \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij}x_j \right| = \sum_{i=1}^{n} |a_{i1}x_1 + \cdots + a_{in}x_n| \\
&\leqslant \sum_{i=1}^{n} \left(|a_{i1}||x_1| + \cdots + |a_{in}||x_n| \right) \\
&= |x_1| \sum_{i=1}^{n} |a_{i1}| + \cdots + |x_n| \sum_{i=1}^{n} |a_{in}| \\
&\leqslant \left(\max_j \sum_{i=1}^{n} |a_{ij}| \right) \left(|x_1| + \cdots + |x_n| \right) = \max_j \sum_{i=1}^{n} |a_{ij}|.
\end{aligned}$$

Se

$$\max_j \sum_{i=1}^{n} |a_{ij}| = \sum_{i=1}^{n} |a_{ik}|$$

e considerarmos $\boldsymbol{x}$ tal que $x_k = 1$ e $x_j = 0$, $j \neq k$, obtemos igualdades na relações acima e o resultado pretendido.

Outra norma também frequentemente usada na prática (não induzida por uma norma vetorial) é a *norma de Frobenius* que se define da forma

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

e que é igual à norma euclidiana para vetores, se encararmos a matriz A como um vetor em $\mathbb{R}^{n\times n}$.

Uma norma matricial diz-se *compatível* com uma norma de vetor se para qualquer matriz $A \in \mathbb{C}^{n\times n}$ e qualquer vetor $\boldsymbol{x} \in \mathbb{C}^n$ se tem

$$\|A\boldsymbol{x}\| \leqslant \|A\| \cdot \|\boldsymbol{x}\|. \tag{1.9}$$

As normas matriciais apresentadas em (1.8) e em (1.7) são compatíveis, respetivamente, com as normas vetoriais dadas em (1.2) e (1.3).

Sempre que nada seja dito em contrário, as normas de matrizes serão sempre entendidas como normas compatíveis com alguma norma vetorial, que é o que acontece sempre no caso de uma norma vetorial e a correspondente norma matricial induzida.

1.2 Valores e vetores próprios

Para estudar a convergência de métodos iterativos precisamos de relembrar algumas definições e propriedades relacionadas com valores e vetores próprios.

Definição 1.2.1. *Seja $A \in \mathbb{C}^{n\times n}$. Um vetor $\boldsymbol{v} \in \mathbb{C}^n$, $\boldsymbol{v} \neq \mathbf{0}$, é dito vetor próprio de A, se existir um número $\lambda \in \mathbb{C}$ tal que $A\boldsymbol{v} = \lambda\boldsymbol{v}$. O escalar λ é denominado valor próprio de A associado ao vetor próprio $\boldsymbol{v}$.*

Definição 1.2.2. *O raio espectral da matriz A denotado por $\rho(A)$ é definido como o máximo dos valores próprios de A em valor absoluto, ou seja,*

$$\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda| \tag{1.10}$$

onde $\sigma(A)$ designa o conjunto dos valores próprios de A, também denominado de espectro de A.

Lema 1.1. *Seja $\rho(A)$ o raio espetral da matriz A. Então, para qualquer norma matricial induzida,*

$$\rho(A) \leqslant \|A\|.$$

Demonstração. Seja λ um valor próprio de A e $\boldsymbol{v}$ um vector próprio associado a λ. Por definição $A\boldsymbol{v} = \lambda\boldsymbol{v}$, $\boldsymbol{v} \neq \mathbf{0}$, e vem

$$|\lambda|\|\boldsymbol{v}\| = \|\lambda\boldsymbol{v}\| = \|A\boldsymbol{v}\| \leqslant \|A\|\|\boldsymbol{v}\|,$$

ou seja,

$$|\lambda|\|\boldsymbol{v}\| \leqslant \|A\|\|\boldsymbol{v}\|.$$

Como $\|\boldsymbol{v}\| \neq 0$, dividindo ambos os membros por $\|\boldsymbol{v}\|$, obtemos

$$|\lambda| \leqslant \|A\|$$

para qualquer valor próprio λ de A e, portanto, $\rho(A) \leqslant \|A\|$. □

A seguir mencionamos algumas propriedades dos valores próprios.

- Os valores próprios de A são as soluções da equação $\det(A-\lambda I) = 0$, designada equação característica, uma vez que a equação $A\boldsymbol{v} = \lambda\boldsymbol{v}$ equivale ao sistema homogéneo $(A - \lambda I)\boldsymbol{v} = \mathbf{0}$ que apenas tem solução $\boldsymbol{v} \neq \mathbf{0}$ se $\det(A - \lambda I) = 0$.

- Ao polinómio $p(\lambda) = \det(A-\lambda I) = (-1)^n\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$ de grau n chamamos polinómio característico e, de acordo com o Teorema Fundamental da Álgebra, p tem exatamente n zeros (possivelmente não todos distintos) $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$. Isto é, o polinómio p pode escrever-se na forma $p(\lambda) = (-1)^n(\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n)$. Os valores próprios de A são, portanto, os zeros de p.

- O determinante de A é igual ao ao produto dos valores próprios de A. De facto, $\det(A) = p(0) = \lambda_1 \cdots \lambda_n$.

- O *traço* de A é igual à soma dos valores próprios de A, ou seja, $\mathrm{tr}(A) = a_{11} + a_{22} + \cdots + a_{nn} = \lambda_1 + \cdots + \lambda_n$.

- Se λ é um valor próprio de A, então $\lambda - \alpha$ é um valor próprio de $A - \alpha I$, uma vez que $(A - \alpha I)\boldsymbol{v} = A\boldsymbol{v} - \alpha\boldsymbol{v} = (\lambda - \alpha)\boldsymbol{v}$.

- Sendo λ um valor próprio de A, então λ^p é um valor próprio de A^p, com $p \in \mathbb{N}$, pois de $A\boldsymbol{v} = \lambda\boldsymbol{v}$ segue que $A^p\boldsymbol{v} = \lambda^p\boldsymbol{v}$.

- Se A tem valores próprios todos diferentes de zero, os valores próprios da inversa de A serão iguais aos inversos aritméticos dos valores próprios de A. De facto, de $A\boldsymbol{v} = \lambda\boldsymbol{v}$ resulta $A^{-1}\boldsymbol{v} = \dfrac{1}{\lambda}\boldsymbol{v}$.

- As *transformações de semelhança* preservam os valores próprios. De facto, para uma dada matriz invertível B, de $A\boldsymbol{v} = \lambda\boldsymbol{v}$ obtemos $(B \cdot A \cdot B^{-1})B\boldsymbol{v} = \lambda B\boldsymbol{v}$, ou seja, λ é valor próprio da matriz $B \cdot A \cdot B^{-1}$ com vetor próprio $B\boldsymbol{v}$.

A norma matricial $\|.\|_2$ é definida por

$$\|A\|_2 = \sqrt{\rho(A^T A)}.$$

Se A for uma matriz real simétrica, ou seja, $A = A^T$, então $A^T A = A^2$ e a norma matricial $\|.\|_2$ é dada por

$$\|A\|_2 = \rho(A). \tag{1.11}$$

Algumas classes de matrizes, como por exemplo as matrizes *definidas positivas*, aparecem frequentemente associadas a aplicações. Uma matriz real A de ordem $n \times n$ simétrica diz-se *definida positiva (negativa)* se $\boldsymbol{x}^T A\boldsymbol{x} > 0$ ($\boldsymbol{x}^T A\boldsymbol{x} < 0$) para todos os vetores $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq \boldsymbol{0}$, onde $\boldsymbol{x}^T$ denota o vetor $\boldsymbol{x}$ transposto. Uma caracterização das matrizes definidas positivas (negativas) é a de que todos os valores próprios de A são positivos (negativos), ou seja, A é definida positiva (negativa) se e só se $\lambda > 0$ ($\lambda < 0$), para todo o $\lambda \in \sigma(A)$.

1.3 Número de condição

O *número de condição* de uma matriz A relativamente à norma $\|.\|$ é definido por

$$\mathrm{cond}(A) = \|A\| \cdot \|A^{-1}\|.$$

Por convenção, $\text{cond}(A) = \infty$ se A for singular. Se $\text{cond}(A)$ for um valor elevado, então pequenas perturbações nos dados dos problema, isto é, nas entradas da matriz A e/ou do vetor $\boldsymbol{b}$, induzirão erros muito maiores na solução $\boldsymbol{x}$ calculada. Neste caso o sistema $A\boldsymbol{x} = \boldsymbol{b}$ diz-se *mal condicionado*. Analisaremos com detalhe em que medida uma perturbação $\boldsymbol{\delta b}$ no vetor $\boldsymbol{b}$ dos termos independentes altera a solução $\boldsymbol{x}$ do sistema. De

$$A(\boldsymbol{x} + \boldsymbol{\delta x}) = \boldsymbol{b} + \boldsymbol{\delta b}$$

resulta

$$A\boldsymbol{\delta x} = \boldsymbol{\delta b}.$$

Como A é invertível, temos

$$\boldsymbol{\delta x} = A^{-1}\boldsymbol{\delta b} \tag{1.12}$$

e

$$\|\boldsymbol{\delta x}\| \leqslant \|A^{-1}\| \cdot \|\boldsymbol{\delta b}\|. \tag{1.13}$$

A desigualdade anterior estabelece um majorante para o erro absoluto na solução em função do erro absoluto em $\boldsymbol{b}$. De $A\boldsymbol{x} = \boldsymbol{b}$ vem

$$\frac{1}{\|\boldsymbol{x}\|} \leqslant \frac{\|A\|}{\|\boldsymbol{b}\|} \tag{1.14}$$

e de (1.13) segue

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leqslant \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|}. \tag{1.15}$$

A relação anterior mostra que os erros relativos na solução estão relacionados com as perturbações relativas em $\boldsymbol{\delta b}$ por intermédio de $\text{cond}(A)$.

No caso geral de pertubações tanto em A como em $\boldsymbol{b}$, e assumindo que $\|\delta A\| < 1/\|A^{-1}\|$, tem-se (ver teorema 3.1.1 em [13])

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\text{cond}(A)}{\left(1 - \text{cond}(A)\frac{\|\delta A\|}{\|A\|}\right)} \left(\frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|} + \frac{\|\delta A\|}{\|A\|}\right). \tag{1.16}$$

No caso de A ser simétrica com valores próprios todos diferentes de zero, resulta de (1.11) que o número de condição relativamente à $\|\cdot\|_2$ é dado por

$$\text{cond}(A) = \rho(A)\rho(A^{-1}).$$

Se λ_1 e λ_n forem os valores próprios de menor e maior valor absoluto, respetivamente, então

$$\text{cond}(A) = \frac{|\lambda_n|}{|\lambda_1|}. \tag{1.17}$$

1.4 Métodos iterativos clássicos

Considere o sistema de equações lineares

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1.18}$$

onde $A = (a_{ij})$, $i, j = 1, \ldots, n$, com $\det(A) \neq 0$. Um método de resolução do sistema (1.18) diz-se que é um método iterativo quando fornece uma sequência de soluções aproximadas em que cada solução aproximada é obtida a partir da solução anterior pela aplicação dos mesmos procedimentos. Da equação (1.18) obtemos uma equação equivalente da forma

$$\boldsymbol{x} = G\boldsymbol{x} + \boldsymbol{c} \tag{1.19}$$

e definimos um processo iterativo com a expressão geral

$$\boldsymbol{x}_{k+1} = G\boldsymbol{x}_k + \boldsymbol{c}, \quad k = 0, 1, \ldots, \tag{1.20}$$

onde G é uma matriz $n \times n$, designada *matriz de iteração*, e $\boldsymbol{c}$ é um vetor $n \times 1$. Assim, partindo de uma aproximação inicial $\boldsymbol{x}_0$ para a solução exata $\boldsymbol{x}^*$ do sistema (1.18), com a fórmula (1.20), geramos uma sequência de aproximações $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ que naturalmente se pretende que seja convergente para $\boldsymbol{x}^*$, isto é,

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}^* \qquad \text{ou} \qquad \lim_{k\to\infty} \|\boldsymbol{e}_k\| = 0$$

com

$$\boldsymbol{e}_k = \boldsymbol{x}^* - \boldsymbol{x}_k. \tag{1.21}$$

Se G e $\boldsymbol{c}$ não dependerem de k diz-se que o método iterativo é um *método estacionário*, caso contrário, diz-se que o método é *não-estacionário*.

1.4.1 Convergência

Nesta secção vamos discutir condições necessárias e suficientes para a convergência de métodos iterativos estacionários. Partindo da expressão (1.19), observe-se que estes métodos são métodos iterativos de ponto fixo.

Definição 1.4.1. *Um método iterativo definido por (1.20) diz-se convergente se, para qualquer estimativa $\boldsymbol{x}_0$, a sucessão $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ... convergir para um limite independente de $\boldsymbol{x}_0$, ou seja, para qualquer $\boldsymbol{e}_0$,*

$$\lim_{k\to\infty} \|\boldsymbol{e}_k\| = 0.$$

O erro $\boldsymbol{e}_{k+1}$ na iteração $k+1$ pode expressar-se em função do erro $\boldsymbol{e}_k$ na iteração k. Se a (1.19) com $\boldsymbol{x} = \boldsymbol{x}^*$ subtraírmos (1.20) membro a membro obtemos

$$\boldsymbol{x}^* - \boldsymbol{x}_{k+1} = G(\boldsymbol{x}^* - \boldsymbol{x}_k)$$

ou seja,

$$\boldsymbol{e}_{k+1} = G\boldsymbol{e}_k. \tag{1.22}$$

Aplicando esta expressão sucessivamente obtemos

$$\boldsymbol{e}_{k+1} = G\boldsymbol{e}_k = G^2\boldsymbol{e}_{k-1} = G^3\boldsymbol{e}_{k-2} = \cdots = G^{k+1}\boldsymbol{e}_0.$$

Podemos então escrever para $k = 1, 2, \ldots$

$$\boldsymbol{e}_k = G^k\boldsymbol{e}_0. \tag{1.23}$$

O Teorema seguinte apresenta uma condição necessária e suficiente de convergência para estes métodos iterativos.

Teorema 1.1. *O método iterativo dado pela expressão (1.20) é convergente se e só se*

$$\lim_{k\to\infty} \|G^k\| = 0. \tag{1.24}$$

Demonstração. Vamos primeiro provar que a condição $\lim_{k\to\infty} \|G^k\| = 0$ é suficiente. Aplicando a norma a ambos os membros da expressão (1.23), vem

$$0 \leqslant \|\boldsymbol{e}_k\| = \|G^k\boldsymbol{e}_0\| \leqslant \|G^k\|\|\boldsymbol{e}_0\|.$$

Usando a hipótese de que $\lim_{k\to\infty} \|G^k\| = 0$, temos

$$0 \leqslant \lim_{k\to\infty} \|\boldsymbol{e}_k\| \leqslant \lim_{k\to\infty} \|G^k\|\|\boldsymbol{e}_0\| = \|\boldsymbol{e}_0\| \lim_{k\to\infty} \|G^k\| = 0.$$

Portanto, não há dúvidas de que

$$\lim_{k\to\infty} \|\boldsymbol{e}_k\| = 0,$$

qualquer que seja $\boldsymbol{e}_0$, ou seja, o método é convergente.

Vamos agora provar que a condição (1.24) é necessária. Supondo então que o método é convergente, temos

$$\lim_{k\to\infty} \|G^k\boldsymbol{y}\| = \lim_{k\to\infty} \|\boldsymbol{e}_k\| = 0$$

para qualquer vetor $\boldsymbol{y}(= \boldsymbol{x}_0)$. Pela definição de norma matricial temos que

$$\|G^k\| = \max_{\boldsymbol{y}\neq 0} \frac{\|G^k\boldsymbol{y}\|}{\|\boldsymbol{y}\|}.$$

Como

$$\lim_{k\to\infty} \|G^k \boldsymbol{y}\| = 0,$$

podemos também afirmar que

$$\lim_{k\to\infty} \|G^k\| = 0,$$

como pretendíamos concluir. □

Com um ligeiro abuso de linguagem, diz-se que uma matriz que satisfaça a condição (1.24) é uma *matriz convergente*. Esta condição é de verificação difícil na prática, pelo que é útil dispor de outras condições de convergência, ainda que apenas suficientes.

Teorema 1.2. *Se em alguma norma*

$$\|G\| < 1 \tag{1.25}$$

então o método iterativo (1.20) *é convergente.*

Demonstração. Se $\|G\| < 1$, então

$$\lim_{k\to\infty} \|G\|^k = 0.$$

Como

$$0 \leqslant \|G^k\| \leqslant \|G\|^k,$$

sabemos também que

$$\lim_{k\to\infty} \|G^k\| = 0$$

e, pelo Teorema 1.1, concluímos que o método iterativo (1.20) é convergente. □

A condição de convergência (1.24) pode exprimir-se em termos dos valores próprios da matriz G de acordo com o teorema que vamos apresentar a seguir.

Teorema 1.3. *A matriz G é convergente se e só se $\rho(G) < 1$, onde $\rho(G)$ denota o raio espectral da matriz G.*

Demonstração. Comecemos por demonstrar a necessidade. Considere-se que G é convergente, isto é, $\lim_{k\to\infty} \|G^k\| = 0$, e seja $(\lambda, \boldsymbol{y})$ um par próprio qualquer de G. Como

$$G^k \boldsymbol{y} = G^{k-1}(G\boldsymbol{y}) = G^{k-1}(\lambda \boldsymbol{y}) = \lambda G^{k-1}\boldsymbol{y} = \ldots = \lambda^{k-1} G\boldsymbol{y} = \lambda^k \boldsymbol{y},$$

vem que

$$\lim_{k\to\infty} \|G^k \boldsymbol{y}\| = \lim_{k\to\infty} \|\lambda^k \boldsymbol{y}\| = \|\boldsymbol{y}\| \lim_{k\to\infty} |\lambda|^k.$$

Como $\|G^k \boldsymbol{y}\| \leqslant \|G^k\|\|\boldsymbol{y}\|$ e a matriz G é convergente, segue que $\lim_{k\to\infty} \|G^k \boldsymbol{y}\| = 0$ e, portanto, também

$$\lim_{k\to\infty} |\lambda|^k = 0.$$

Isto implica que $|\lambda| < 1$, para qualquer valor próprio λ, e temos então que $\rho(G) < 1$.

Para demonstrar a suficiência vamos utilizar o resultado apresentado no Problema 9.10.8 em [9, pg. 549] que afirma que para qualquer $\varepsilon > 0$, existe uma norma $\|.\|_\varepsilon$ tal que

$$\rho(G) \leqslant \|G\|_\varepsilon \leqslant \rho(G) + \varepsilon.$$

Supondo então que $\rho(G) < 1$ e tomando $\varepsilon < 1 - \rho(G)$, podemos obter uma norma $\|.\|_\varepsilon$ tal que $\|G\|_\varepsilon < 1$. Logo, pelo Teorema 1.2 concluímos que a matriz G é convergente. □

Em (1.22) temos expressa uma relação linear entre os erros e_k e e_{k+1} em iterações consecutivas. Desta expressão podemos escrever

$$\|\boldsymbol{e}_{k+1}\| \leqslant \|G\|\|\boldsymbol{e}_k\|, \tag{1.26}$$

ou ainda,

$$\frac{\|\boldsymbol{e}_{k+1}\|}{\|\boldsymbol{e}_k\|^p} \leqslant \|G\|$$

com $p = 1$. Dizemos, neste caso, que a *razão de convergência* é $p = 1$, admitindo que $\|G\| < 1$. É facil perceber que quando mais próximo de zero for $\|G\|$, mais fácil e rápida é a convergência do método. De (1.23) podemos também escrever

$$\|\boldsymbol{e}_k\| \leqslant \|G\|^k \|\boldsymbol{e}_0\| \tag{1.27}$$

o que nos diz que em k iterações o erro inicial será pelos menos multiplicado por $\|G\|^k$.

No caso de G ser uma matriz simétrica, $\|G\|_2 = \rho(G)$ e (1.26) dá lugar a

$$\|\boldsymbol{e}_{k+1}\|_2 \leqslant \rho(G)\|\boldsymbol{e}_k\|_2. \tag{1.28}$$

Isto leva-nos a definir o conceito de *taxa de convergência*.

Definição 1.4.2. *A taxa de convergência do processo iterativo (1.20) é definida por*

$$R = -\log_{10} \rho(G).$$

Observe-se que a taxa de convergência assim definida indica-nos o ganho em número de casas decimais na solução por iteração, uma vez que

$$\log_{10} \frac{\|\boldsymbol{e}_{k+1}\|_2}{\|\boldsymbol{e}_k\|_2} \leqslant \log_{10} \rho(G)$$

o que equivale a ter

$$\log_{10} \|\boldsymbol{e}_k\|_2 - \log_{10} \|\boldsymbol{e}_{k+1}\|_2 \geqslant -\log_{10} \rho(G).$$

Quanto mais pequeno for $\rho(G)$, maior é a taxa de convergência, isto é, maior é o número de casas decimais corretas obtidas por iteração.

1.4.2 Decomposição da matriz A

Até agora não dissemos como se obtém a matriz G. Uma forma usual de construir esta matriz é a partir de uma decomposição aditiva da matriz A em duas matrizes M e N tais que

$$A = M - N \tag{1.29}$$

onde M é invertível. Ver [9, Secção 8.1.2]. Substituindo (1.29) em (1.18), dá lugar a

$$M\boldsymbol{x} = N\boldsymbol{x} + \boldsymbol{b} \tag{1.30}$$

e daqui resulta a fórmula iterativa

$$M\boldsymbol{x}_{k+1} = N\boldsymbol{x}_k + \boldsymbol{b} \tag{1.31}$$

que se pode escrever na forma

$$\boldsymbol{x}_{k+1} = M^{-1}N\boldsymbol{x}_k + M^{-1}\boldsymbol{b}, \qquad k = 0, 1, \ldots \tag{1.32}$$

Comparando (1.32) com (1.20) podemos fazer as seguintes identificações

$$G = M^{-1}N \qquad \text{e} \qquad \boldsymbol{c} = M^{-1}\boldsymbol{b}.$$

Observe-se que estes métodos se podem ainda escrever na forma

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + M^{-1}\boldsymbol{r}_k \tag{1.33}$$

em que $\boldsymbol{r}_k$ é o *resíduo* na iteração k, definido por

$$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k. \tag{1.34}$$

De facto, multiplicando ambos os membros da expressão (1.29) por M^{-1} temos

$$M^{-1}A = I - M^{-1}N$$

que equivale a escrever

$$M^{-1}N = I - M^{-1}A. \tag{1.35}$$

Substituindo (1.35) em (1.32) vem

$$\boldsymbol{x}_{k+1} = (I - M^{-1}A)\boldsymbol{x}_k + M^{-1}\boldsymbol{b},$$

obtendo-se a expressão em (1.33),

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + M^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k).$$

Definindo $\boldsymbol{h}_k = M^{-1}\boldsymbol{r}_k$, podemos escrever esta expressão como

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h}_k$$

que exprime a aproximação $\boldsymbol{x}_{k+1}$ como uma correção da aproximação $\boldsymbol{x}_k$.

Vamos agora mostrar que estes métodos iterativos, baseados na decomposição aditiva da matriz A, quando convergem, são equivalentes à construção de uma sucessão de *inversas aproximadas* da matriz A.

Lema 1.2. *As iterações geradas pela expressão (1.32) verificam*

$$\boldsymbol{x}_k = G^k\boldsymbol{x}_0 + W_{k-1}\boldsymbol{b} \tag{1.36}$$

com

$$W_{k-1} = \left(\sum_{j=0}^{k-1} G^j\right) M^{-1}.$$

Demonstração. Usando repetidamente

$$\boldsymbol{x}_k = G\boldsymbol{x}_{k-1} + M^{-1}\boldsymbol{b}$$

obtemos

$$\begin{aligned}
\boldsymbol{x}_k &= G\left(G\boldsymbol{x}_{k-2} + M^{-1}\boldsymbol{b}\right) + M^{-1}\boldsymbol{b} = G^2\boldsymbol{x}_{k-2} + (G+I)\,M^{-1}\boldsymbol{b} \\
&= G^2\left(G\boldsymbol{x}_{k-3} + M^{-1}\boldsymbol{b}\right) + (G+I)\,M^{-1}\boldsymbol{b} \\
&= G^3\boldsymbol{x}_{k-3} + \left(G^2+G+I\right)M^{-1}\boldsymbol{b} \\
&= \ldots \\
&= G^k\boldsymbol{x}_0 + \left(G^{k-1}+G^{k-2}+\ldots+G+I\right)M^{-1}\boldsymbol{b} \\
&= G^k\boldsymbol{x}_0 + \left(\sum_{j=0}^{k-1} G^j\right)M^{-1}\boldsymbol{b} = G^k\boldsymbol{x}_0 + W_{k-1}\boldsymbol{b}. \quad \square
\end{aligned}$$

Quando o processo iterativo é convergente, a sucessão

$$W_k = \left(\sum_{j=0}^{k} G^j\right) M^{-1}, \tag{1.37}$$

é uma sucessão que converge para a inversa de A, ou seja, a matriz W_k pode considerar-se uma aproximação para A^{-1}.

Definição 1.4.3. *Diz-se que W é uma inversa aproximada de A se $\|I - WA\| < 1$.*

Lema 1.3. *Seja B uma matriz tal que $\|B\| < 1$. Neste caso é válida a série de Neumann*

$$\sum_{j=0}^{\infty} B^j = I + B + B^2 + B^3 + \ldots = (I - B)^{-1}$$

e podemos escrever

$$\sum_{j=0}^{k} B^j \approx (I - B)^{-1}.$$

Demonstração.
Para qualquer matriz B temos

$$\left\| I - \left(\sum_{j=0}^{k} B^j \right) (I - B) \right\| = \|B^{k+1}\| \leqslant \|B\|^{k+1}. \tag{1.38}$$

Quando $\|B\| < 1$, $\lim_{k\to\infty} \|B^{k+1}\| = 0$ e temos também que

$$\lim_{k\to\infty} \left\| I - \left(\sum_{j=0}^{k} B^j \right) (I - B) \right\| = 0.$$

Ou seja,

$$I - \left(\sum_{j=0}^{\infty} B^j \right) (I - B) = O$$

ou ainda,

$$\sum_{j=0}^{\infty} B^j = (I - B)^{-1}. \quad \square$$

Observe-se que, sendo $\|B\| < 1$, de (1.38), temos

$$\left\| I - \left(\sum_{j=0}^{k} B^j \right) (I - B) \right\| < 1,$$

o que, de acordo com a definição 1.4.3, significa que $\sum_{j=0}^{k} B^j$ é uma inversa aproximada de $(I - B)$. $\square$

Aplicando este teorema ao caso em que B é a matriz de iteração G e usando o resultado em (1.35),

$$G = I - M^{-1}A,$$

podemos concluir que, no caso em que $\|G\| < 1$, o método iterativo é convergente e a sucessão W_k converge para

$$(I - G)^{-1} M^{-1} = \left(M^{-1}A\right)^{-1} M^{-1} = A^{-1}.$$

1.4.3 Critérios de paragem

Quando utilizamos um método iterativo devemos ter em conta que as iterações não podem decorrer infinitamente, precisamos de um número finito de iterações e para tal é necessário definirmos um critério de paragem. Nesta subsecção vamos abordar de forma breve alguns dos critérios mais utilizados na prática [9].

A paragem do processo iterativo deve verificar-se quando a aproximação calculada estiver próxima da solução exata. Uma maneira de conseguir isto, consiste em estimar o erro absoluto ou o erro relativo da aproximação obtida ao fim de k iterações. A partir de duas iterações consecutivas $\boldsymbol{x}_{k+1}$ e $\boldsymbol{x}_k$ e dada uma tolerância absoluta $\tau > 0$ para a precisão desejada para os resultados, podemos definir uma expressão para o critério de paragem da seguinte forma

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \tau \tag{1.39}$$

ou, se $\|\boldsymbol{x}_{k+1}\| \neq 0$, e dada uma tolerância relativa $\tau' > 0$, uma expressão da forma

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|}{\|\boldsymbol{x}_{k+1}\|} < \tau'. \tag{1.40}$$

Juntando (1.39) e (1.40) da seguinte forma

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \tau + \tau'\|\boldsymbol{x}_{k+1}\|$$

permite-nos obter uma expressão que combina os dois critérios. Se $\|\boldsymbol{x}_{k+1}\|$ for pequeno, prevalece o critério da tolerância absoluta. Se $\|\boldsymbol{x}_{k+1}\|$ for grande, é o critério da tolerância relativa que domina. No caso em que tomamos $\tau = \tau'$ ficamos com o critério

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \tau\big(1 + \|\boldsymbol{x}_{k+1}\|\big). \tag{1.41}$$

Os valores dos parâmetros τ e τ' devem ser fixados com cuidado, tendo nomeadamente em atenção que nem (1.39) nem (1.40) garantem, em geral, que o erro absoluto $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|$ esteja abaixo de τ, ou o erro relativo $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|/\|\boldsymbol{x}^*\|$ abaixo de τ'.

Para obter uma estimativa para o erro em $\boldsymbol{x}_{k+1}$ usamos a relação

$$\begin{aligned}\boldsymbol{x}_{k+1} - \boldsymbol{x}^* &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+2} + \boldsymbol{x}_{k+2} - \boldsymbol{x}^* = G\boldsymbol{x}_k - G\boldsymbol{x}_{k+1} + G\boldsymbol{x}_{k+1} - G\boldsymbol{x}^* \\ &= G(\boldsymbol{x}_k - \boldsymbol{x}_{k+1}) + G(\boldsymbol{x}_{k+1} - \boldsymbol{x}^*).\end{aligned}$$

Aplicando normas a ambos os membros, temos

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leqslant \|G\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\| + \|G\|\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|$$

donde resulta

$$\big(1 - \|G\|\big)\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leqslant \|G\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|$$

e, se $\|G\| < 1$,

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leqslant \frac{\|G\|}{1 - \|G\|}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|. \tag{1.42}$$

Quando se tem

$$\frac{\|G\|}{1 - \|G\|} \leqslant 1$$

e (1.39) se verifica, então também

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| < \tau. \tag{1.43}$$

Ou seja, apenas quando $\|G\| \leqslant 0.5$ o critério (1.39) nos garante que o erro absoluto em $\boldsymbol{x}_{k+1}$ é também inferior à tolerância τ.

Quando ao erro relativo em $\boldsymbol{x}_{k+1}$, se multiplicarmos ambos os membros em (1.42) por $1/\|\boldsymbol{x}^*\|$, obtemos a expressão

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|} \leqslant \frac{\|G\|}{1 - \|G\|}\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|}{\|\boldsymbol{x}^*\|} = \frac{\|G\|}{1 - \|G\|}\frac{\|\boldsymbol{x}_{k+1}\|}{\|\boldsymbol{x}^*\|}\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|}{\|\boldsymbol{x}_{k+1}\|}.$$

No caso em que $\|G\| \leqslant 0.5$ e $\dfrac{\|\boldsymbol{x}_{k+1}\|}{\|\boldsymbol{x}^*\|} \leqslant 1$, estando o critério (1.40) satisfeito, podemos também dizer que

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|} < \tau'. \tag{1.44}$$

Outro cuidado a ter em conta é que o valor τ não deve ser muito próximo de zero, uma vez que poderemos estar a exigir para a solução uma precisão que está para além daquela que os erros de arredondamento permitem obter, levando a que o critério de paragem não possa ser satisfeito. De facto, a partir de um certo número de iterações, os resultados deixam de melhorar, estabelecendo-se assim um limite natural ao processo iterativo.

Outro tipo de medida da qualidade da solução é representada pelo resíduo (1.34). Mais precisamente, podemos parar o processo iterativo na iteração k quando

$$\frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{b}\|} < \tau'. \tag{1.45}$$

Interpretando o resíduo como uma perturbação $\boldsymbol{\delta b}$, de (1.15) resulta, então,

$$\frac{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|} < \text{cond}(A) \cdot \tau' \tag{1.46}$$

e, se o número de condição da matriz A for pequeno, teremos obtido uma aproximação $\boldsymbol{x}_k$ com a precisão relativa desejada.

1.4.4 Algoritmo geral para métodos iterativos

Algorithm 1 Algoritmo geral para métodos iterativos

Input: matriz A e vetor $\boldsymbol{b}$,
tolerâncias τ e τ',
máximo número de iterações k_{max},
aproximação inicial $\boldsymbol{x}_0$
Output: aproximação $\boldsymbol{x}_k$ para $A\boldsymbol{x} = \boldsymbol{b}$

$k = 0$ ▷ (inicialização do contador de iterações)
`test`$=$*false* ▷ (inicialização da variável booleana `test`)
while $\big($`test`$==$*false* **and** $k \leqslant k_{max}\big)$ **do**
  $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$
  $M\boldsymbol{h}_k = \boldsymbol{r}_k$ ▷ (resolver por métodos diretos)
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h}_k$
  `test` $= \big(\|\boldsymbol{h}_k\| < \tau + \tau'\|\boldsymbol{x}_{k+1}\|\big)$
  $k = k + 1$
end while

if `test`$==$*true* **then**
  Convergência em k iterações para a precisão desejada.
else
  Teste de convergência não satisfeito em k_{max} iterações.
end if

Capítulo 2

Métodos de Jacobi, Gauss-Seidel e SOR

Neste capítulo descrevemos os métodos iterativos clássicos de resolução de um sistema de equações lineares. Estes métodos são apropriados para sistemas de grande dimensão cuja matriz dos coeficientes é esparsa e apresentam vantagens sobre os métodos diretos, quer em termos de velocidade de convergência quer em termos de exigências de armazenamento de dados.

2.1 Método de Jacobi

Considerando a matriz dos coeficientes do sistema $A\boldsymbol{x} = \boldsymbol{b}$, $A = (a_{ij})$, $i, j = 1, \ldots, n$, definam-se $D = (d_{ij})$, uma matriz diagonal, $L = (l_{ij})$, uma matriz estritamente triangular inferior e $U = (u_{ij})$, uma matriz estritamente triangular superior, tais que

$$d_{ij} = \begin{cases} a_{ij} & \text{se} \quad i = j \\ 0 & \text{se} \quad i \neq j \end{cases}, \qquad l_{ij} = \begin{cases} a_{ij} & \text{se} \quad i > j \\ 0 & \text{se} \quad i \leqslant j \end{cases}, \qquad u_{ij} = \begin{cases} a_{ij} & \text{se} \quad i < j \\ 0 & \text{se} \quad i \geqslant j \end{cases}, \tag{2.1}$$

ou seja, de forma que

$$A = D + (L + U). \tag{2.2}$$

De acordo com a exposição apresentada no Capítulo 1, para o método de Jacobi escolhemos

$$M = D, \qquad N = D - A = -(L + U). \tag{2.3}$$

Nesta decomposição vamos supor que D é uma matriz invertível, o que implica que os elementos d_{ii}, $i = 1, \ldots, n$, sejam todos diferentes de zero. Caso isto não aconteça, se A for invertível é sempre possível reordenar as equações do sistema, ou seja, trocar linhas na matriz ampliada do sistema $(A|\, \boldsymbol{b})$, por forma a que aquela condição se verifique. [1]

[1] Este resultado pode obter-se a partir da fórmula de Leibniz para o cálculo do determinante de uma matriz A de ordem n. Esta fórmula expressa o determinante de A como uma soma dos produtos dos elementos da diagonal principal de cada uma das $n!$ permutações de linhas da matriz A.

Substituindo (2.3) em (1.31) no Capítulo 1,

$$M\boldsymbol{x}_{k+1} = N\boldsymbol{x}_k + \boldsymbol{b}, \tag{2.4}$$

obtemos

$$\boldsymbol{x}_{k+1} = D^{-1}\left[-(L+U)\boldsymbol{x}_k + \boldsymbol{b}\right], \qquad k = 0, 1, \ldots \tag{2.5}$$

e a matriz de iteração é

$$G_J = -D^{-1}(L+U). \tag{2.6}$$

As componentes [2] da iteração $\boldsymbol{x}_{k+1}$ em (2.5) podem escrever-se na forma

$$x_i^{(k+1)} = \left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right)/a_{ii}, \qquad i = 1, 2, \ldots, n \tag{2.7}$$

ou

$$x_i^{(k+1)} = x_i^{(k)} + \left(b_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)}\right)/a_{ii}, \qquad i = 1, 2, \ldots, n. \tag{2.8}$$

A observação destas expressões permite confirmar a afirmação feita anteriormente de que é relativamente fácil nos métodos iterativos tirar partido da esparsidade da matriz A. De facto, se um dado elemento a_{ij} for nulo, o respetivo termo nas expressões acima pode ser simplesmente suprimido. Para tal, basta que a estrutura da esparsidade de A seja convenientemente representada em computador [9].

2.1.1 Convergência do método de Jacobi

A partir do Teorema (1.2) podemos concluir que se

$$\|G_J\|_\infty = \|D^{-1}(L+D)\|_\infty < 1 \tag{2.9}$$

então o método de Jacobi converge qualquer que seja $\boldsymbol{x}_0$.Vamos apresentar de seguida um resultado que garante que esta condição é satisfeita.

Definição 2.1.1. *Seja n um inteiro. Uma matriz $A = (a_{ij})_{n\times n}$ diz-se* ***estritamente diagonal dominante*** *por linhas se*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \qquad i = 1, 2, ..., n, \tag{2.10}$$

[2] Quando for necessário invocar componentes de vetores, usaremos a notação $x_i^{(k)}$ para designar a componente i do vetor $\boldsymbol{x}_k$

e diz-se ***fracamente diagonal dominante*** *por linhas se*

$$|a_{ii}| \geqslant \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \qquad i = 1, 2, ..., n, \tag{2.11}$$

e vale a desigualdade estrita para alguma linha i.

Teorema 2.1. *Se a matriz dos coeficientes do sistema $A\boldsymbol{x} = \boldsymbol{b}$ de n equações em n incógnitas é estritamente diagonal dominante por linhas, então o método de Jacobi é convergente.*

Demonstração. De acordo com a definição 2.1.1, se $A = (a_{ij})$ é uma matriz estritamente diagonal dominante por linhas, obtemos

$$1 > \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right|, \qquad i = 1, 2, ..., n, \tag{2.12}$$

e, consequentemente, concluimos que

$$\|G_J\|_\infty = \|D^{-1}(L+U)\|_\infty = \max_{1\leqslant i\leqslant n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1. \quad \square$$

2.2 Método de Gauss-Seidel

Começaremos com uma pequena explicação do funcionamento do método. Já sabemos que o método de Jacobi utiliza os valores $x_i^{(k)}$ da k-ésima iteração para obter os valores $x_i^{(k+1)}$ da iteração seguinte.

Assumindo que estamos a atualizar os valores das componentes pela ordem natural, no cálculo de $x_i^{(k+1)}$ podem ser usados os valores $x_1^{(k+1)}, ..., x_{i-1}^{(k+1)}$. Esta é a diferença do método Gauss-Saidel relativamente ao método de Jacobi.

Consideremos novamente as matrizes D, L e U , definidas em (2.1). No processo iterativo (2.4), a escolha

$$M = D + L \qquad \text{e} \qquad N = -U \tag{2.13}$$

dá-nos o método de Gauss-Seidel. Substituindo em (2.4) obtemos

$$\boldsymbol{x}_{k+1} = (D+L)^{-1}[-U\boldsymbol{x}_k + \boldsymbol{b}], \qquad k = 0, 1, \ldots \tag{2.14}$$

e resulta que a matriz de iteração é dada por

$$G_{GS} = -(D+L)^{-1}U. \tag{2.15}$$

No lugar de (2.14) escrevemos

$$\boldsymbol{x}_{k+1} = D^{-1}[-L\boldsymbol{x}_{k+1} - U\boldsymbol{x}_k + \boldsymbol{b}], \qquad k = 0, 1, \ldots \tag{2.16}$$

e as componentes de $\boldsymbol{x}_{k+1}$ são dadas por

$$x_i^{(k+1)} = \Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big)/a_{ii}, \qquad i = 1, 2, \ldots, n, \tag{2.17}$$

ou

$$x_i^{(k+1)} = x_i^{(k)} + \Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\Big)/a_{ii}, \qquad i = 1, 2, \ldots n. \tag{2.18}$$

2.2.1 Convergência do método de Gauss-Seidel

Do teorema (1.2) conclui-se que se

$$\|G_{GS}\|_\infty = \|(D + L)^{-1}U\|_\infty < 1 \tag{2.19}$$

então o método de Gauss-Seidel converge qualquer que seja $\boldsymbol{x}_0$

Teorema 2.2. *Se a matriz dos coeficientes do sistema $A\boldsymbol{x} = \boldsymbol{b}$ de n equações em n incógnitas é estritamente diagonal dominante por linhas, então o método de Gauss-Seidel é convergente.*

Demonstração. Basta conseguirmos provar a expressão (2.19).
Pela definição de norma temos que

$$\|(D + L)^{-1}U\| = \max_{y\neq 0} \|\boldsymbol{z}\|/\|\boldsymbol{y}\| \quad \text{com} \quad \boldsymbol{z} = (D + L)^{-1}U\boldsymbol{y}$$

mas, sendo assim, estes vetores $\boldsymbol{z}$ e $\boldsymbol{y}$ estão relacionados através do sistema de equações lineares

$$(D + L)\boldsymbol{z} = U\boldsymbol{y}$$

a que corresponde à expressão

$$z_i = \Big(-\sum_{j=1}^{i-1} a_{ij}z_j + \sum_{j=i+1}^{n} a_{ij}y_j\Big)/a_{ii}.$$

Tomando módulos obtemos

$$|z_i| \leqslant \sum_{j=i+1}^{n} \frac{|a_{ij}|}{|a_{ii}|}|y_j| + \sum_{j=1}^{i-1} \frac{|a_{ij}|}{|a_{ii}|}|z_j|, \tag{2.20}$$

e podemos ecrever

$$|z_i| \leqslant \beta_i \|\boldsymbol{y}\|_\infty + \alpha_i \|\boldsymbol{z}\|_\infty \tag{2.21}$$

com

$$\beta_i = \sum_{j=i+1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \qquad \text{e} \qquad \alpha_i = \sum_{j=1}^{i-1} \left|\frac{a_{ij}}{a_{ii}}\right|.$$

Se m é um índice (existe pelo menos um) para o qual se verifica que

$$|z_m| = \|\boldsymbol{z}\|_\infty = \max_{1\leqslant i\leqslant n} |z_i|,$$

então, decorre de (2.21) que

$$\|\boldsymbol{z}\|_\infty \leqslant \beta_m \|\boldsymbol{y}\|_\infty + \alpha_m \|\boldsymbol{z}\|_\infty.$$

Sendo assim, obtemos

$$\|\boldsymbol{z}\|_\infty \leqslant \frac{\beta_m}{1-\alpha_m}\|\boldsymbol{y}\|_\infty \leqslant \max_{1\leqslant i\leqslant n}\left(\frac{\beta_i}{1-\alpha_i}\right)\|\boldsymbol{y}\|_\infty < \|\boldsymbol{y}\|_\infty, \qquad \boldsymbol{y} \neq 0,$$

uma vez que, para matrizes de diagonal estritamente dominante por linhas, é válido que $\beta_i + \alpha_i < 1$, para todo e qualquer i, e portanto

$$\frac{\beta_i}{1-\alpha_i} < 1.$$

Por conseguinte, como $\|\boldsymbol{z}\|_\infty / \|\boldsymbol{y}\|_\infty < 1$, para todo e qualquer $\boldsymbol{y} \neq 0$, fica deste modo provado que

$$\|G_{GS}\|_\infty < 1,$$

conforme pretendíamos. □

Acabámos de ver que quer o método de Gauss-Seidel quer o de Jacobi convergem quando a matriz A possui diagonal principal estritamente dominante por linhas. Além disso, de acordo com Theorem 6.2. em [3, p. 287], temos

$$\|G_{GS}\|_\infty \leqslant \|G_J\|_\infty < 1.$$

No entanto, esta condição de convergência pode ser enfraquecida quando A é uma matriz *irredutível*, como mostra o resultado que apresentamos de seguida.

Definição 2.2.1. *Uma matriz $A = (a_{i,j})_{n\times n}$ é **irredutível** se não existir uma matriz de permutação P tal que*

$$PAP^T = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix} \tag{2.22}$$

onde $m < n$, $B = (b_{ij})_{m\times m}$, $C = (c_{ij})_{m\times(n-m)}$ e $D = (d_{ij})_{(n-m)\times(n-m)}$. Ou seja, a matriz não pode ser decomposta de forma que um conjunto de variáveis fique independente das outras.

Teorema 2.3 ([3], Theorem 6.11.)**.** *Se A é irredutível e fracamente diagonal dominante por linhas, então ambos os métodos de Jacobi e Gauss - Seidel convergem e* $\rho(G_{GS}) < \rho(G_J) < 1$.

Ver ([3], pp. 288-289).

O resultado seguinte fornece-nos uma outra condição suficiente de convergência para o método de Gauss-Seidel.

Teorema 2.4. *Se A é simétrica definida positiva então o método de Gauss-Seidel converge.*

Este resultado é o caso particular em que $\omega = 1$ do Teorema 2.6 que vamos apresentar no final da secção seguinte.

Quanto ao método e Jacobi, no caso da matriz A ser simétrica definida positiva, o método pode ou não convergir.

2.3 O método da sobre-relaxação sucessiva (SOR)

Podemos acelerar a convergência nos dois métodos clássicos apresentados mas, uma vez que o método de Gauss-Seidel converge, em geral, mais rapidamente do que o método de Jacobi, neste trabalho limitar-nos-emos, como é prática usual, a considerar o método da sobre-relaxação sucessiva (SOR) no contexto do método de Gauss-Seidel.
Da equação (1.18) obtemos, com $\omega \neq 0$ (parâmetro de relaxação),

$$\omega A\boldsymbol{x} = \omega \boldsymbol{b} \tag{2.23}$$

onde

$$\begin{aligned} \omega A =& \omega(D + L + U) \\ =& (D + \omega L) - [D - \omega(D + U)]. \end{aligned} \tag{2.24}$$

Substituindo (2.24) em (2.23) temos

$$\Big((D + \omega L) - [D - \omega(D + U)]\Big)\boldsymbol{x} = \omega \boldsymbol{b} \tag{2.25}$$

que é a equação do método de Gauss-Seidel com relaxação. Esta equação dá lugar à fórmula iterativa

$$(D + \omega L)\boldsymbol{x}_{k+1} = [(1 - \omega)D - \omega U]\boldsymbol{x}_k + \omega \boldsymbol{b} \tag{2.26}$$

que se pode escrever na forma

$$\begin{aligned} \boldsymbol{x}_{k+1} &= -\omega D^{-1} L\boldsymbol{x}_{k+1} + D^{-1}[(1 - \omega)D - \omega U]\boldsymbol{x}_k + \omega D^{-1}\boldsymbol{b} \\ &= (1 - \omega)\boldsymbol{x}_k + \omega D^{-1}\left(\boldsymbol{b} - L\boldsymbol{x}_{k+1} - U\boldsymbol{x}_k\right) \end{aligned}$$

ou ainda

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \omega D^{-1}[\boldsymbol{b} - L\boldsymbol{x}_{k+1} - (D+U)\boldsymbol{x}_k].$$

Daqui resulta que a i-ésima componente de $\boldsymbol{x}_{k+1}$ é dada por

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big)/a_{ii}, \qquad i = 1, 2, \ldots, n, \quad (2.27)$$

ou

$$x_i^{(k+1)} = x_i^{(k)} + \omega\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\Big)/a_{ii}, \qquad i = 1, 2, \ldots, n. \quad (2.28)$$

Com $\omega = 1$ resulta a expressão (2.18) do método Gauss-Seidel. Com $\omega > 1$ e $\omega < 1$ têm-se os chamados métodos de *sobre-relaxação* e *sub-relaxação*, respetivamente.

2.3.1 Convergência do método SOR

O valor do parâmetro de relaxação ω que conduz à melhor convergência possível varia com o sistema de equações que se quer resolver. Não há como determinar, *a priori*, esse valor ótimo, exceto para alguns, poucos, casos particulares. Normalmente realizam-se testes numéricos com diversos valores de ω num intervalo de valores admissíveis para verificar qual o melhor valor para o problema em causa.
Para determinar a matriz de iteração correspondente ao método de Gauss-Seidel com relaxação, observe-se de (2.26) que a decomposição da matriz ωA correspondente a este processo iterativo é, então,

$$M = D + \omega L, \qquad N = (1-\omega)D - \omega U.$$

Sendo assim, a matriz de iteração é dada por

$$\begin{aligned} G_{SOR}(\omega) = M^{-1}N &= (D+\omega L)^{-1}\left[(1-\omega)D - \omega U\right] \\ &= \left[D(I+\omega D^{-1}L)\right]^{-1}\left[D\big((1-\omega)I - \omega D^{-1}U\big)\right] \\ &= (I+\omega D^{-1}L)^{-1}D^{-1}D\big((1-\omega)I - \omega D^{-1}U\big) \\ &= (I+\omega D^{-1}L)^{-1}\big((1-\omega)I - \omega D^{-1}U\big). \end{aligned} \quad (2.29)$$

A importância desta expressão da matriz de iteração será evidente na demonstração do resultado clássico que a seguir se apresenta.

Teorema 2.5. *Seja $G_{SOR}(\omega)$ a matriz do método de Gauss-Seidel com relaxação dada pela expressão (2.29). Então*

$$|\omega - 1| \leqslant \rho\big(G_{SOR}(\omega)\big). \tag{2.30}$$

A igualdade verifica-se apenas quando todos os valores próprios de $G_{SOR}(\omega)$ tiverem módulo igual à unidade.

Para haver convergência, o fator de relaxação deve satisfazer a condição necessária

$$0 < \omega < 2. \tag{2.31}$$

Demonstração. Uma vez que $I + \omega D^{-1}L$ é uma matriz triangular inferior de diagonal principal unitária, e $(1-\omega)I - \omega D^{-1}U$ é uma matriz triangular superior cujos elementos na diagonal principal são todos iguais a $(1-\omega)$, tem-se

$$\begin{aligned}
\det\left(G_{SOR}(\omega)\right) &= \det\left[(I + \omega D^{-1}L)^{-1}\big((1-\omega)I - \omega D^{-1}U\big)\right] \\
&= \det\left((I + \omega D^{-1}L)^{-1}\right) \cdot \det\left((1-\omega)I - \omega D^{-1}U\right) \\
&= \det[(1-\omega)I - \omega D^{-1}U] \\
&= (1-\omega)^n.
\end{aligned}$$

Por outro lado, sabemos também que

$$\det\left(G_{SOR}(\omega)\right) = \prod_{k=1}^{n} \mu_k(\omega)$$

onde os μ_k designam os valores próprios de $G_{SOR}(\omega)$. Assim,

$$\prod_{k=1}^{n} \mu_k(\omega) = (1-\omega)^n, \tag{2.32}$$

e, aplicando módulos, obtemos

$$|1-\omega|^n = \prod_{k=1}^{n} |\mu_k(\omega)| \leqslant \left[\max_{1\leqslant k\leqslant n} |\mu_k(\omega)|\right]^n = \left[\rho(G_{SOR}(\omega))\right]^n,$$

donde se conlui que

$$|1-\omega| \leqslant \rho(G_{SOR}(\omega)).$$

A partir do Teorema 1.3 verificamos que a condição

$$0 < \omega < 2$$

é uma condição necessária para haver convergência do método de Gauss-Seidel com relaxação. De facto, no caso de convergência, temos

$$|1-\omega| \leqslant \rho\big(G_{SOR}(\omega)\big) < 1$$

e, portanto,

$$|1-\omega| < 1.$$

□

No entanto, no caso em que A é definida positiva a condição (2.31) é também suficiente.

Teorema 2.6 ([9], Teorema 8.4.4)**.** *Se A for simétrica definida positiva e $0 < \omega < 2$, então o método de Gauss-Seidel com relaxação converge.*

Capítulo 3

Discretização da Equação de Poisson

3.1 Uma aplicação: difusão de calor

Consideremos uma placa de um material condutor em cujo bordo superior se aplica, de uma forma constante no tempo, uma temperatura de 0^0C e que nos restantes bordos se tem uma fonte de calor constante de 100^0C, tal como se ilustra na figura seguinte.

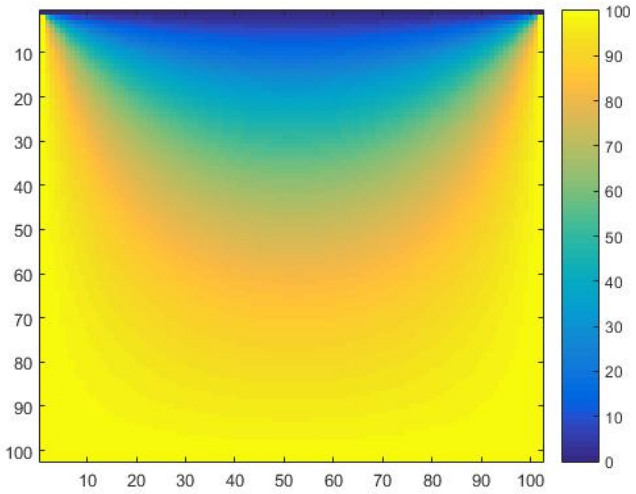


Figura 3.1 Distribuição dos valores da temperatura numa placa.

A temperatura em cada ponto interior da placa variará no tempo em função dos valores da temperatura nos bordos, até atingir um valor que permanecerá constante, ou seja, em equilíbrio, a partir deste momento. Denotaremos por $u(x, y)$ este valor no ponto de coordenadas (x, y) num referencial cujos eixos coincidem com o bordo inferior (abcissas)

e o bordo esquerdo (ordenadas). Nestas condições, o valor $u(x, y)$ é a solução da equação diferencial (ver [6])

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{3.1}$$

definida para $0 \leqslant x, y \leqslant T$, que satisfaz as condições de fronteira

$$u(x, 0) = 100; \qquad u(x, T) = 0 \quad \text{com} \quad 0 \leqslant x \leqslant T$$

$$u(0, y) = 100; \qquad u(T, y) = 100 \quad \text{com} \quad 0 \leqslant y \leqslant T.$$

Assumindo que as faces da nossa placa estão isoladas de tal forma que não existe troca de calor com o meio ambiente, então a equação (3.1), que é a chamada equação de Poisson em duas dimensões, dá lugar a

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{3.2}$$

Esta é a equação de Laplace que é afinal a equação de Poisson no caso particular em que $f(x, y) = 0$.

3.2 Formalização matemática do problema

Dependendo da dimensão do domínio do problema, a incógnita u da equação de Poisson é uma função de 1, 2 ou 3 variáveis. Temos então a seguinte formalização. Seja Ω um conjunto conexo em $\mathbb{R}^d, d = 1, 2$ ou 3 e $\partial\Omega$ a fronteira desse conjunto. A equação de Poisson pode ser escrita como

$$\Delta u = f \qquad \text{em} \qquad \Omega \tag{3.3}$$

onde

$$\Delta u = \frac{d^2 u}{dx^2} \qquad \text{para} \qquad d = 1,$$

$$\Delta u = \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} \qquad \text{para} \qquad d > 1.$$

A incógnita $u : \Omega \to \mathbb{R}$ é uma função de classe C^2 e $f : \Omega \to \mathbb{R}$ uma função conhecida [6]. Em cada problema particular onde ocorre a equação (3.3), a solução a determinar tem que satisfazer as condições de fronteira do problema, que podem ser de diferentes tipos. As chamadas condições de Dirichlet correspondem ao caso em que são conhecidos os valores da função u na fronteira $\partial\Omega$. Este é o caso do exemplo apresentado antes.

3.3 Método das diferenças finitas

A maior parte dos métodos numéricos para resolução de equações diferenciais envolve o uso de aproximações para as derivadas. Estas aproximações são designadas por diferenças finitas e têm por base a discretização do problema. Nesta seçcão vamos descrever estes métodos para a equação de Poisson nos casos de uma e de duas dimensões.

3.3.1 Caso unidimensional

Consideremos o caso mais simples da equação (3.3), o caso unidimensional,

$$\frac{d^2u}{dx^2} = f(x), \qquad a < x < b. \tag{3.4}$$

Começamos por definir $n+2$ pontos igualmente espaçados no intervalo $[a,b]$,

$$x_0 = a, \quad x_i = a + ih, \qquad i = 1, ..., n+1,$$

com $h = \dfrac{b-a}{n+1}$, como mostra a figura seguinte.

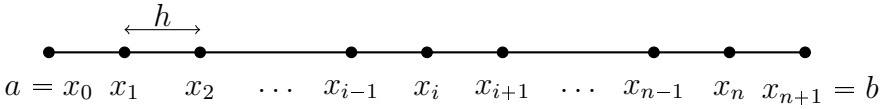


Figura 3.2 Discretização do intervalo $[a,b]$.

Supondo que u é uma função contínua com derivadas contínuas em x_i, consideremos a série de Taylor de u em torno de x_i,

$$u(x_i + \Delta x) = u(x_i) + (\Delta x)u'(x_i) + \frac{(\Delta x)^2}{2!}u''(x_i) + \frac{(\Delta x)^3}{3!}u'''(x_i) + \cdots \tag{3.5}$$

para cada $i = 1, \ldots, n$. Desta expressão, com $\Delta x = h$, resulta

$$u(x_i + h) = u(x_i) + hu'(x_i) + \frac{h^2}{2!}u''(x_i) + \frac{h^3}{3!}u'''(x_i) + \cdots \tag{3.6}$$

e, resolvendo em ordem a $u'(x_i)$, obtemos

$$u'(x_i) = \frac{u(x_i + h) - u(x_i)}{h} + \left[-\frac{h}{2!}u''(x_i) - \frac{h^2}{3!}u'''(x_i) - \cdots\right].$$

Temos assim uma aproximação para a derivada de primeira ordem

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_i)}{h} \tag{3.7}$$

com erro de truncatura local

$$erro_{TL} = -\frac{h}{2!}u''(x_i) - \frac{h^2}{3!}u'''(x_i) - \cdots . \tag{3.8}$$

A expressão (3.7) é conhecida como fórmula de *diferença finita progressiva de primeira ordem*. O erro de truncatura (3.8) tende para zero com h, ou seja, é $O(h)$ e podemos escrever

$$u'(x_i) = \frac{u(x_i + h) - u(x_i)}{h} + O(h). \tag{3.9}$$

Observe-se que $O(h)$ representa a expressão para $erro_{TL}$ e apenas indica como o erro de truncatura varia com o refinamento da malha e não o valor do erro.

De modo análogo podemos obter uma aproximação designada por *diferença finita regressiva de primeira ordem*. Da expansão em série de Taylor (3.5), agora com $\Delta x = -h$, obtemos

$$u(x_i - h) = u(x_i) - hu'(x_i) + \frac{h^2}{2!}u''(x_i) - \frac{h^3}{3!}u'''(x_i) + \cdots \tag{3.10}$$

donde segue

$$u'(x_i) = \frac{u(x_i) - u(x_i - h)}{h} + \left[\frac{h}{2!}u''(x_i) - \frac{h^2}{3!}u'''(x_i) - \cdots\right]$$

e a aproximação

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h}$$

também com erro de truncatura local $O(h)$.

Uma fórmula de diferenças finitas de segunda ordem para aproximar a primeira derivada de u pode obter-se subtraindo a expressão (3.10) de (3.6). Assim,

$$u(x_i + h) - u(x_i - h) = 2hu'(x_i) + \frac{2h^3}{3!}u'''(x_i) + \cdots$$

e temos

$$u'(x_i) = \frac{u(x_i + h) - u(x_i - h)}{2h} + \left[-\frac{h^2}{3!}u'''(x_i) - \cdots\right].$$

À aproximação

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2h},$$

cujo erro de truncatura é $O(h^2)$, chamamos *diferença finita centrada de segunda ordem*.

Estas técnicas podem generalizar-se facilmente para o cálculo de derivadas de ordem superior. Vejamos o caso da segunda derivada. Se adicionarmos (3.6) e (3.10) obtemos

$$u(x_i + h) + u(x_i - h) = 2u(x_i) + \frac{2h^2}{2!}u''(x_i) + \frac{2h^4}{4!}u^{(IV)}(x_i) + \cdots$$

e, portanto,

$$u''(x_i) = \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} + \left[-\frac{h^2}{12} u^{(IV)}(x_i) - \cdots \right] \tag{3.11}$$

donde segue a aproximação para a segunda derivada

$$u''(x_i) \approx \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} \tag{3.12}$$

com erro de truncatura $O(h^2)$. Vamos agora considerar a discretização da equação (3.4) com as condições de fronteira de Dirichlet

$$u(a) = \alpha \qquad \text{e} \qquad u(b) = \beta. \tag{3.13}$$

Para cada ponto $x_i = a + ih$, $i = 0, \ldots, n+1$, usaremos as abreviaturas

$$f_i = f(x_i) \qquad \text{e} \qquad u_i = u(x_i).$$

Se usarmos a aproximação para a segunda derivada dada por (3.12) em cada ponto x_i e ignorarmos os erros de truncatura, podemos reescrever a equação (3.4) em $x = x_i$ na forma

$$u_{i-1} - 2u_i + u_{i+1} = h^2 f_i \tag{3.14}$$

com $i = 1, \ldots, n$. E introduzindo as condições de fronteira $u_0 = \alpha$ e $u_{n+1} = \beta$, ficamos com o sistema de n equações lineares

$$\begin{aligned}
-2u_1 + u_2 &= h^2 f_1 - \alpha \\
u_1 - 2u_2 + u_3 &= h^2 f_2 \\
&\vdots \\
u_{n-2} - 2u_{n-1} + u_n &= h^2 f_{n-1} \\
u_{n-1} - 2u_n &= h^2 f_n - \beta.
\end{aligned} \tag{3.15}$$

Usando a formulação matricial temos o sistema

$$A\boldsymbol{u} = \boldsymbol{b}$$

com

$$A = \begin{pmatrix}
-2 & 1 & 0 & \cdots & 0 & 0 & 0 \\
1 & -2 & 1 & \cdots & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 & -2 & 1 \\
0 & 0 & 0 & \cdots & 0 & 1 & -2
\end{pmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} \quad \text{e} \quad \boldsymbol{b} = \begin{pmatrix} h^2 f_1 - \alpha \\ h^2 f_2 \\ \vdots \\ \vdots \\ \vdots \\ h^2 f_{n-1} \\ h^2 f_n - \beta \end{pmatrix}. \tag{3.16}$$

Assim, com a resolução desta equação matricial obtemos uma solução aproximada $\boldsymbol{u} = (u_1, \ldots, u_n)$ do problema expresso em (3.4) e (3.13) nos pontos x_i, $i = 1, \ldots, n$, do intervalo $[a, b]$.

3.3.2 Caso bidimensional

Vamos agora considerar a equação (3.3) em duas dimensões

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \qquad a < x < b, \quad c < y < d, \tag{3.17}$$

com condições de Dirichlet na fronteira do domínio

$$\begin{aligned} u(x, c) &= g(x, c), \quad u(x, d) = g(x, d), \\ u(a, y) &= g(a, y), \quad u(b, y) = g(b, y). \end{aligned} \tag{3.18}$$

Discretizamos o problema definindo uma malha de $(n + 2) \times (n + 2)$ pontos no retângulo $[a, b] \times [c, d]$, com espaçamento horizontal $\Delta x = \frac{b-a}{n+1}$ e espaçamento vertical $\Delta y = \frac{d-c}{n+1}$, como mostra a figura seguinte.

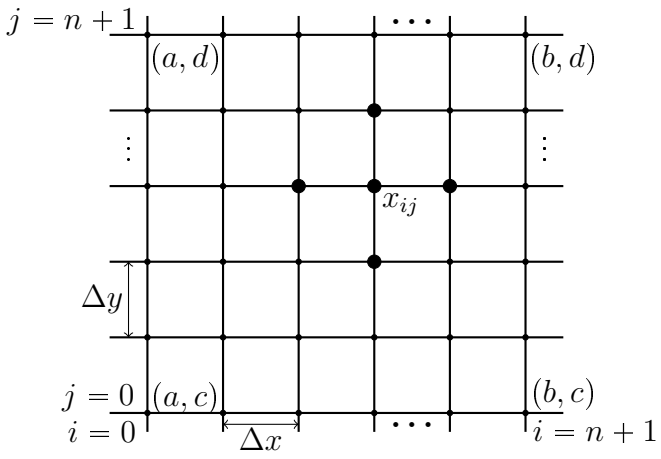


Figura 3.3 Discretização do domínio $[a, b] \times [c, d]$

Neste trabalho vamos considerar $\Delta x = \Delta y = h$, ou seja, uma malha bidimensional com espaçamento uniforme. Assim, os pontos da malha são

$$(x_i, y_j) = (a + ih, b + jh), \quad i, j = 0, ..., n + 1.$$

Usando as notações

$$u_{ij} = u(x_i, y_j), \quad f_{ij} = f(x_i, y_j), \quad g_{ij} = g(x_i, y_j)$$

e a aproximação de diferenças finitas (3.12) para as derivadas parciais de segunda ordem em cada ponto (x_i, y_j), temos

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} \tag{3.19}$$

e

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j} \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2}. \tag{3.20}$$

Adicionando as expressões (3.19) e (3.20), podemos escrever

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} + \left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j} \approx \frac{u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2} \tag{3.21}$$

que é a chamada *fórmula de cinco pontos*. Ver Figura 3.3.

Desprezando os erros de truncatura, em cada ponto (x_i, y_j) a equação de Poisson (3.17) dá lugar à equação

$$\frac{u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2} = f_{i,j} \tag{3.22}$$

e define, em conjunto com as condições de fronteira (3.18), um sistema de n^2 equações lineares em n^2 incógnitas $u_{i,j}$, $i, j = 1, \ldots, n$,

$$\begin{aligned}
u_{1,0} + u_{0,1} - 4u_{1,1} + u_{2,1} + u_{1,2} &= h^2 f_{1,1} \\
u_{2,0} + u_{1,1} - 4u_{2,1} + u_{3,1} + u_{2,2} &= h^2 f_{2,1} \\
&\vdots \\
u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1} &= h^2 f_{i,j} \\
&\vdots \\
u_{n,n-1} + u_{n-1,n} - 4u_{n,n} + u_{n+1,n} + u_{n,n+1} &= h^2 f_{n,n}
\end{aligned} \tag{3.23}$$

Por forma a escrever estas equações numa simples equação matricial, precisamos de considerar as incógnitas $u_{i,j}$ num único vetor coluna de dimensão n^2 e isto requer escolher uma ordenação para as incógnitas. Existem várias ordenações possíveis para as quais estão associadas diferentes matrizes [11]. Vamos, de forma algo arbitrária, escolher a ordem em que os pontos da malha são percorridos linha à linha, da esquerda para a direita e de baixo para cima (dita ordem natural ou lexicográfica). Vejamos o exemplo em que $n = 3$. Neste caso particular definimos

$$\boldsymbol{u}^T = \begin{pmatrix} u_{11} & u_{21} & u_{31} & u_{12} & u_{22} & u_{32} & u_{13} & u_{23} & u_{33} \end{pmatrix}$$

como o vetor das incógnitas. Por simplicidade, vamos fazer a identificação

$$\boldsymbol{u}^T = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & u_9 \end{pmatrix}$$

e, correspondentemente, teremos $f_1, f_2, \ldots, f_9$ para os valores de f. Veja-se a figura seguinte.

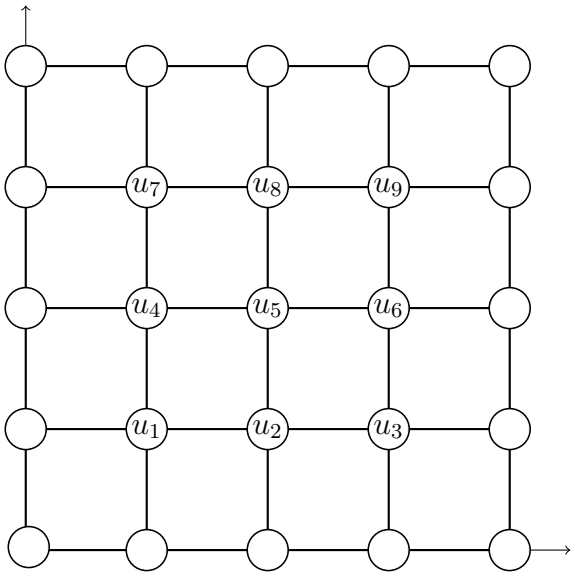


Figura 3.4 Ordenação natural das incógnitas para uma malha bidimensional 5×5

Assim, o sistema de equações (3.23) pode escrever-se na forma

$$\left(\begin{array}{ccc|ccc|ccc} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{array}\right) \left(\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \hline u_4 \\ u_5 \\ u_6 \\ \hline u_7 \\ u_8 \\ u_9 \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \hline b_4 \\ b_5 \\ b_6 \\ \hline b_7 \\ b_8 \\ b_9 \end{array}\right) \tag{3.24}$$

onde

$$\left(\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \hline b_4 \\ b_5 \\ b_6 \\ \hline b_7 \\ b_8 \\ b_9 \end{array}\right) = \left(\begin{array}{c} h^2 f_1 - u_{1,0} - u_{0,1} \\ h^2 f_2 - u_{2,0} \\ h^2 f_3 - u_{3,0} - u_{4,1} \\ \hline h^2 f_4 - u_{0,2} \\ h^2 f_5 \\ h^2 f_6 - u_{4,2} \\ \hline h^2 f_7 - u_{0,3} - u_{1,4} \\ h^2 f_8 - u_{2,4} \\ h^2 f_9 - u_{4,3} - u_{3,4} \end{array}\right).$$

No caso geral, ficamos com um sistema de n^2 equações lineares em n^2 incógnitas representado matricialmente por

$$T\boldsymbol{u} = \boldsymbol{b} \tag{3.25}$$

onde T é a matriz quadrada de ordem n^2, tridiagonal por blocos, dada por

$$T = \begin{pmatrix} B & I & 0 & \cdots & 0 & 0 & 0 \\ I & B & I & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I & B & I \\ 0 & 0 & 0 & \cdots & 0 & I & B \end{pmatrix}, \tag{3.26}$$

sendo I a matriz identidade de ordem n e

$$B = \begin{pmatrix} -4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -4 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -4 \end{pmatrix} \tag{3.27}$$

também de ordem n.

Outra forma de "percorrer" as linhas e colunas da malha é usar a ordenação *Red-Black*. Nesta ordenação, os nós da malha são divididos em dois conjuntos de cores diferentes à semelhança de um tabuleiro de xadrez, conforme se apresenta na Figura 3.5, no caso em que $n = 3$.

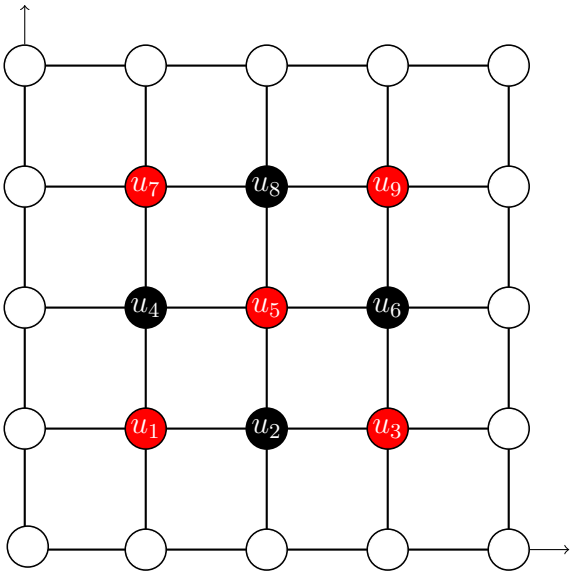


Figura 3.5 Ordenação *Red-Black* das incógnitas para uma malha bidimensional 5×5

Assim, quando usamos um método iterativo para a resolução do sistema, se atualizarmos todos os nós vermelhos primeiro, iremos usar apenas os valores dos nós pretos. De seguida, quando atualizamos os nós pretos, que estão apenas adjacentes aos nós vemelhos, poderemos utilizar apenas os novos valores referentes aos nós vermelhos. No nosso problema, a ordenação *Red-Black* leva a que o sistema de equações (3.23) se escreva na forma

$$\left(\begin{array}{ccccc|cccc} -4 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & -4 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & -4 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & -4 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & -4 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -4 \end{array}\right) \left(\begin{array}{c} u_1 \\ u_3 \\ u_5 \\ u_7 \\ u_9 \\ \hline u_2 \\ u_4 \\ u_6 \\ u_8 \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_3 \\ b_5 \\ b_7 \\ b_9 \\ \hline b_2 \\ b_4 \\ b_6 \\ b_8 \end{array}\right). \tag{3.28}$$

Observe-se que podemos obter a matriz dos coeficientes para a ordenação *Red-Black*,

$$T_{RB} = \left(\begin{array}{ccccc|cccc} -4 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & -4 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & -4 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & -4 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & -4 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -4 \end{array}\right) \tag{3.29}$$

através da transformação de semelhança $PTP^T = PTP^{-1}$ onde T é a matriz dos coeficientes associada à ordenação natural dada em (3.24) e P é a matriz de permutação definida por

$$P = \left(\begin{array}{ccccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right). \tag{3.30}$$

3.4 Produtos de Kronecker

O produto de Kronecker é útil para descrever problemas lineares que envolvem duas ou mais variáveis uma vez que permite representações mais compactas [11].

Definição 3.4.1. *Sejam $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$. O produto de kronecker de A e B é definido como sendo a matriz*

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix} \in \mathbb{R}^{mp\times nq}. \tag{3.31}$$

Definição 3.4.2. *Se $A \in \mathbb{R}^{n\times n}$ e $B \in \mathbb{R}^{m\times m}$, a soma de kronecker de A e B é a matriz definida por*

$$A \oplus B = (I_m \otimes A) + (B \otimes I_n) \in \mathbb{R}^{mn\times mn}.$$

Observe-se que, em geral, $A \oplus B \neq B \oplus A$.

Vejamos como podemos representar a matriz T em (3.25), matriz associada à equação de Poisson em duas dimensões, usando estas duas operações de Kronecker envolvendo a matriz identidade I de ordem n e a matriz A em (3.16), matriz associada à equação de Poisson unidimensional. Considerem-se os produtos de Kronecker

$$I \otimes A = \begin{pmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{pmatrix}$$

e

$$A \otimes I = \begin{pmatrix} -2I & I & 0 & \cdots & 0 & 0 & 0 \\ I & -2I & I & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I & -2I & I \\ 0 & 0 & 0 & \cdots & 0 & I & -2I \end{pmatrix}$$

e a sua soma

$$(I \otimes A) + (A \otimes I) = \begin{pmatrix} A-2I & I & 0 & \cdots & 0 & 0 & 0 \\ I & A-2I & I & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I & A-2I & I \\ 0 & 0 & 0 & \cdots & 0 & I & A-2I \end{pmatrix} \tag{3.32}$$

Observe-se que $A - 2I = B$, com B dada em (3.27), e que, portanto,

$$T = (I \otimes A) + (A \otimes I).$$

Atendendo à definição 3.4.2, podemos escrever

$$T = A \oplus A, \tag{3.33}$$

ou seja, a matriz T da representação matricial do problema bidimensional pode escrever-se a partir da matriz A associada ao problema unidimensional usando produtos de Kronecker.

3.5 Existência de solução e condicionamento do sistema

Teorema 3.1 ([7], Theorem 2.2)**.** *Os valores próprios de uma matriz tridiagonal da forma*

$$M = \begin{pmatrix} b & c & 0 & \cdots & 0 & 0 & 0 \\ a & b & c & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a & b & c \\ 0 & 0 & 0 & \cdots & 0 & a & b \end{pmatrix} \tag{3.34}$$

são dados por

$$\lambda_k = b + 2\sqrt{ac} * \cos\frac{k\pi}{n+1}, \quad k = 1, \ldots, n. \tag{3.35}$$

Com $a = c = 1$ e $b = -2$ em (3.34), tem-se a matriz A dada por (3.16) cujos valores próprios são então

$$\lambda_k = -2 + 2\cos\frac{k\pi}{n+1}, \quad k = 1, \ldots, n. \tag{3.36}$$

É fácil concluir que os valores próprios satisfazem $-4 < \lambda_k < 0$, para $k = 1, \ldots, n$, e o menor e o maior (em valor absoluto) são, respetivamente,

$$\lambda_1 = -2 + 2\cos\frac{\pi}{n+1} \tag{3.37}$$

e

$$\lambda_n = -2 + 2\cos\frac{n\pi}{n+1}. \tag{3.38}$$

Podemos agora concluir que a equação de Poisson unidimensional tem uma e uma só solução (independentemente das condições de Dirichlet fixadas) uma vez que a matriz do correspondente sistema tem valores próprios todos diferentes de zero. Porém, o número de condição dado em (1.17) do sistema em (3.16) cresce com a dimensão n, isto é, o sistema é mal condicionado para valores de n grandes. Atendendo a que, para n grande,

$$\lambda_n \approx -2 + 2\cos\pi = -4 \qquad \text{e} \qquad \lambda_1 \approx -2 + 2\left(1 - \frac{(\pi/n+1)^2}{2}\right) = -\frac{\pi^2}{(n+1)^2}$$

obtemos uma estimativa para o número de condição dada por

$$\frac{\lambda_n}{\lambda_1} \approx \frac{4}{\pi^2}(n+1)^2. \tag{3.39}$$

Por exemplo, para $n = 10^5$, temos $\dfrac{\lambda_n}{\lambda_1} \approx 4 \times 10^9$ e o sistema é mal condicionado. De acordo com (1.46), para garantir que uma aproximação $\boldsymbol{x}_k$ tem um erro relativo inferior a 10^{-3}, por exemplo, terá de ser $\|A\boldsymbol{x}_k - \boldsymbol{b}\|/\|\boldsymbol{b}\| < 10^{-12}$.

Para a determinação dos valores próprios da matriz do sistema (3.25) associado ao problema bidimensional temos o teorema seguinte.

Teorema 3.2 ([3], Capítulo 6)**.** *Se $A \in \mathbb{R}^{n\times n}$ tem valores próprios λ_i e $B \in \mathbb{R}^{m\times m}$ tem valores próprios μ_j, então a soma de kronecker $A \oplus B$, dada pela definição 3.4.2, tem valores próprios*

$$\lambda_1 + \mu_1, \ldots, \lambda_1 + \mu_m, \lambda_2 + \mu_1, \ldots, \lambda_2 + \mu_m, \ldots, \lambda_n + \mu_1, \ldots, \lambda_n + \mu_m.$$

Uma vez que $T = A \oplus A$, os valores próprios de T são dados por

$$\lambda_{i,j} = -4 + 2\cos\frac{i\pi}{n+1} + 2\cos\frac{j\pi}{n+1}, \quad i,j = 1,\ldots,n, \tag{3.40}$$

e satisfazem $-8 < \lambda_{i,j} < 0$, para $i,j = 1,\ldots,n$. Em valor absoluto,

$$\lambda_{1,1} = -4 + 4\cos\frac{\pi}{n+1} \tag{3.41}$$

é o menor valor próprio e

$$\lambda_{n,n} = -4 + 4\cos\frac{n\pi}{n+1} \tag{3.42}$$

é o maior valor próprio. Todos os valores próprios são não nulos e quando n aumenta $\lambda_{1,1}$ tende para zero e $\lambda_{n,n}$ para -8. Assim, tal como no caso unidimensional, a equação de Poisson bidimensional, traduzida no sistema dado em (3.25), tem uma e uma só solução e o número de condição em (1.17) aumenta com a dimensão n, sendo também válida a aproximação obtida em (3.39) para $\dfrac{\lambda_{n,n}}{\lambda_{1,1}}$ quando n é grande.

3.6 Raio espectral das matrizes de iteração

Nesta secção vamos apresentar os raios espectrais das matrizes de iteração dos métodos de Jacobi, Gauss-Seidel e SOR para o problema de Poisson unidimensional e bidimensional.

3.6.1 Caso unidimensional

Seja A a matriz dos coeficientes dada em (3.16) e recorde-se a decomposição $A = D+(L+U)$ apresentada em (2.2),

$$A = \begin{pmatrix} -2 & & & \cdots & 0 \\ & -2 & & & \\ & & \ddots & & \\ & & & -2 & \\ 0 & \cdots & & & -2 \end{pmatrix} + \begin{pmatrix} 0 & 1 & & \cdots & 0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 0 & \cdots & & 1 & 0 \end{pmatrix}. \tag{3.43}$$

A matriz de iteração do método de Jacobi é a matriz

$$G_J = -D^{-1}(L+U) = \frac{1}{2}\begin{pmatrix} 0 & 1 & & \cdots & 0 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 0 & \cdots & & 1 & 0 \end{pmatrix}$$

De acordo com o Teorema 3.1, os valores próprios da matriz G_J são dados por

$$\lambda_k = \cos\frac{k\pi}{n+1}, \qquad k = 1, \ldots, n, \tag{3.44}$$

sendo que o maior valor próprio em módulo se obtém quando $k = 1$ (e $k = n$). Ou seja, o raio espectral de G_J é

$$\rho(G_J) = \cos\frac{\pi}{n+1} \simeq 1 - \frac{\pi^2}{2(n+1)^2}. \tag{3.45}$$

Para a determinação dos raios espectrais das matrizes G_{GS} e G_{SOR}, associadas aos métodos de Gauss-Seidel e SOR, respetivamente, podemos usar o teorema que apresentamos a seguir. Este teorema exibe uma relação entre os raios espectrais $\rho(G_{GS})$ e $\rho(G_{SOR})$ e o raio espectral $\rho(G_J)$ da matriz do método de Jacobi.

Teorema 3.3 ([2], Theorem 5.6.3)**.** *Seja A uma matriz real simétrica, definida positiva e com a forma tridiagonal por blocos*

$$A = \begin{pmatrix} D_1 & U_1 & 0 & \cdots & 0 & 0 & 0 \\ L_2 & D_2 & U_2 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & L_{n-1} & D_{n-1} & U_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & L_n & D_n \end{pmatrix}, \tag{3.46}$$

onde D_i são submatrizes diagonais. Então

$$\rho(G_{GS}) = \rho^2(G_J),$$

onde G_J e G_{GS} são as matrizes de iteração dos métodos de Jacobi e Gauss-Seidel correspondentes à matriz A.

O fator de relaxação ótimo ω_{opt} no método SOR é dado por

$$\omega_{opt} = \frac{2}{1 + \big(1 - \rho(G_{GS})\big)^{1/2}}, \quad \rho(G_{GS}) < 1,$$

a que corresponde o raio espectral

$$\rho(G_{SOR}\big(\omega_{opt})\big) = \omega_{opt} - 1.$$

Observe-se que a matriz A em (3.43) tem a forma tridiagonal por blocos (3.46) mas é simétrica definida negativa (todos os valores próprios, dados em (3.36), são negativos). No entanto, o Teorema 3.3 aplica-se igualmente uma vez que $-A$ e A têm as mesmas matrizes de iteração G_J, G_{GS} e G_{SOR}. Assim, a partir do Teorema 3.3 e de (3.45), podemos concluir que o raio espectral da matriz de iteração de Gauss-Seidel é

$$\rho(G_{GS}) = \cos^2 \frac{\pi}{n+1}. \tag{3.47}$$

Uma vez que $\rho(G_{GS}) < \rho(G_J) < 1$, podemos garantir que os métodos são convergentes e que o método de Gauss-Seidel possui uma taxa de convergência mais elevada do que o método de Jacobi. Mais precisamente, ao fim de k iterações, para o método de Jacobi dado G_J ser uma matriz real simétrica e $\|G_J\|_2 = \rho(G_J)$, a partir de (1.27) obtemos

$$\|e_k\|_2 \leqslant [\rho(G_J)]^k \|e_0\|_2. \tag{3.48}$$

Para o método de Gauss-Seidel não se tem $\rho(G_{GS}) = \|G_{GS}\|_2$ porque G_{GS} não é simétrica, mas podemos escrever $\|G_{GS}\|_2 = \theta \cdot \rho(G_{GS})$ com $\theta \geqslant 1$ e

$$\|e_k\|_2 \leqslant [\theta \cdot \rho(G_{GS})]^k \|e_0\|_2 = \theta^k [\rho(G_J)]^{2k} \|e_0\|_2,$$

uma vez que $\rho(G_{GS}) = \rho^2(G_J)$. Ou seja, uma iteração do método de Gauss-Seidel faz decrescer o erro tanto quanto duas iterações do método de Jacobi. Por outras palavras, no método de Gauss-Seidel o número de iterações necessárias para se obter uma dada precisão na solução será aproximadamente metade do número de iterações que o método de Jacobi exige.

Do Teorema 3.3 obtemos também que o fator de relaxação ótimo ω_{opt} no método SOR para o problema de Poisson é

$$\omega_{opt} = \frac{2}{1 + \left(1 - \cos^2\left(\frac{\pi}{n+1}\right)\right)^{1/2}} = \frac{2}{1 + \sin\left(\frac{\pi}{n+1}\right)} \tag{3.49}$$

com o correspondente raio espectral

$$\rho\big(G_{SOR}(\omega_{opt})\big) = \frac{2}{1 + \sin\left(\frac{\pi}{n+1}\right)} - 1 = \frac{1 - \sin\left(\frac{\pi}{n+1}\right)}{1 + \sin\left(\frac{\pi}{n+1}\right)} \simeq 1 - \frac{2\pi}{n+1},$$

para n grande, uma vez que, para x perto de zero,

$$\frac{1 - \sin x}{1 + \sin x} \simeq \frac{1 - x}{1 + x} \simeq 1 - 2x.$$

3.6.2 Caso bimensional

Consideremos a matriz T dada em (3.26) para o problema de Poisson bidimensional correspondente à ordenação lexicográfica dos nós da malha. É fácil calcular o raio espectral da correspondente matriz de iteração G_J do método de Jacobi, uma vez que

$$T = -4I + (T + 4I) = D + (L + U)$$

e, portanto,

$$G_J = -(-4I)^{-1}(T + 4I) = \frac{1}{4}T + I.$$

Assim, conhecidos os valores próprios $\lambda_{i,j}$ de T, dados em (3.40), os valores próprios de G_J são

$$\frac{1}{4}\lambda_{i,j} + 1 = \frac{1}{2}\left(\cos\frac{i\pi}{n+1} + \cos\frac{j\pi}{n+1}\right), \quad i, j = 1, \ldots, n.$$

O maior valor próprio em módulo ocorre quando $i = j = 1$ (e quando $i = j = n$), ou seja, o raio espectral de G_J é

$$\rho(G_J) = \frac{1}{2}\left(\cos\frac{\pi}{n+1} + \cos\frac{\pi}{n+1}\right) = \cos\frac{\pi}{n+1}, \tag{3.50}$$

o mesmo que se obteve para o caso unidimensional.

Vamos agora observar que para a matriz T_{RB} da ordenação *Red-Black* (veja-se o exemplo em (3.29)), o raio espectral da correspondente matriz de iteração do método de Jacobi $G_{J_{RB}}$ é também igual ao raio espectral (3.50) da matriz G_J para a ordenação lexicográfica. De

facto, T_{RB} é uma matriz semelhante à matriz T e, para uma dada matriz de permutação P, temos

$$T_{RB} = -4I + (T_{RB} + 4I) = -4I + \left(PTP^{-1} + 4I\right).$$

A correspondente matriz de iteração do método de Jacobi é

$$G_{J_{RB}} = -(-4I)^{-1}\left(PTP^{-1} + 4I\right) = \frac{1}{4}PTP^{-1} + I$$

e, como os valores próprios da matriz PTP^{-1} são iguais aos da matriz T, temos também

$$\rho\left(G_{J_{RB}}\right) = \cos\frac{\pi}{n+1}.$$

A matriz T_{RB} é uma matriz simétrica, os seus valores próprios (iguais aos da matriz T) são todos negativos (e, portanto, $-T_{RB}$ é definida positiva) e é uma matriz constituída por quatro blocos que respeitam a forma em (3.46). Ou seja, T_{RB} cumpre as condições de aplicação do Teorema 3.3. Temos, então, tal como para o caso unidimensional, o raio espectral da matriz de iteração do método de Gauss-Seidel $G_{GS_{RB}}$ dado por

$$\rho\left(G_{GS_{RB}}\right) = \rho^2\left(G_{J_{RB}}\right) = \cos^2\frac{\pi}{n+1}$$

e, para a matriz de iteração $G_{SOR_{RB}}$ do método SOR,

$$\rho\big(G_{SOR_{RB}}(\omega_{opt})\big) \simeq 1 - \frac{2\pi}{n+1}$$

para

$$\omega_{opt} = \frac{2}{1 + \sin\left(\frac{\pi}{n+1}\right)}.$$

A matriz T não se encontra na forma (3.46) e, por isso, o Teorema 3.3 não pode ser aplicado. As fórmulas simples do Teorema 3.3, que relacionam os raios espectrais das matrizes G_J, G_{GS} e $G_{SOR}(w)$, permitindo uma comparação da rapidez de convergência dos métodos, exigem um determinado padrão de zeros na matriz do sistema. Este padrão consegue-se com a ordenação *Red-Black* mas não com a ordenação lexicográfica.

Capítulo 4

Experiências numéricas

Implementámos os nossos algoritmos no Matlab (R2018b) num computador LAPTOP-REDGJNFT Intel(R) Celeron(R) CPU N2840 @ 2.16GHz, RAM 4.00 GB, no sistema operativo Windows 10 Home de 64 bits.

Desenvolvemos os códigos `Jacobi`, `Gauss_Seidel` e `SOR` que são implementações gerais no sentido em que podem ser usados para qualquer sistema $A\boldsymbol{x} = \boldsymbol{b}$, dados A e $\boldsymbol{b}$ de forma explícita. A matriz A e o vetor $\boldsymbol{b}$ são, neste caso, construídos da forma descrita no capítulo anterior (código `matriz_vetor` no Anexo A). Também desenvolvemos os códigos `Jacobi_Poison`, `Gauss_Seidel_Poison` e `SOR_Poison` especializados para o nosso problema e que não requerem a matriz A, uma vez que as expressões (2.7), (2.17) e (2.27), se traduzem, neste caso particular e para a ordem natural ou lexicográfica, simplesmente por

$$w_{i,j} = \left(u_{i,j-1} + u_{i-1,j} + u_{i+1,j} + u_{i,j+1}\right)/4, \tag{4.1}$$

$$w_{i,j} = \left(w_{i,j-1} + w_{i-1,j} + u_{i+1,j} + u_{i,j+1}\right)/4 \tag{4.2}$$

e

$$w_{i,j} = (1-\omega)u_{i,j} + \omega\left(w_{i,j-1} + w_{i-1,j} + u_{i+1,j} + u_{i,j+1}\right)/4 \tag{4.3}$$

respetivamente, onde $u_{i,j}$ e $w_{i,j}$ representam os valores $u_{i,j}^{(k)}$ e $u_{i,j}^{(k+1)}$. Neste capítulo vamos-nos referir a estes códigos como Jacobi, Gauss-Seidel, SOR, Jacobi-P, Gauss-Seidel-P e SOR-P, respetivamente.

A escolha da ordenação *Red-Black* leva a duas passagens pela malha. Durante a primeira passagem atualizamos metade das incógnitas utilizando valores da iteração anterior e na segunda passagem atualizamos a restante metade das incógnitas usando já os novos valores obtidos na primeira passagem da iteração atual. No caso do método de SOR a expressão (2.27), para cada uma destas passagens, traduz-se então por

$$w_{i,j} = (1-\omega)u_{i,j} + \omega\left(u_{i,j-1} + u_{i-1,j} + u_{i+1,j} + u_{i,j+1}\right)/4, \quad \text{se } i+j \text{ é par}, \tag{4.4}$$

para a primeira passagem, e

$$w_{i,j} = (1-\omega)u_{i,j} + \omega\left(w_{i,j-1} + w_{i-1,j} + w_{i+1,j} + w_{i,j+1}\right)/4, \qquad \text{se } i+j \text{ é ímpar}, \tag{4.5}$$

para a segunda passagem. No Matlab este método está implementado no código `SOR_RB` (também apresentado no anexo A) e que passamos a designar por SOR-RB.

4.1 Problema I

O nosso primeiro exemplo de aplicação consiste em determinar uma solução aproximada da equação (3.2) com $0 < x, y < 1$ e os seguintes valores de fronteira

$$\begin{aligned} u(x,0) &= 0, \quad u(x,1) = 100x, \qquad 0 \leqslant x \leqslant 1; \\ u(0,y) &= 0, \quad u(1,y) = 100y, \qquad 0 \leqslant y \leqslant 1. \end{aligned}$$

cuja a solução exata é $u(x,y) = 100xy$.

Na Tabela 4.1 apresentamos o número k de iterações necessárias para cumprir o critério de paragem (1.41) com $tol = 10^{-5}$ em cada um dos códigos Jacobi, Gauss-Seidel e SOR (com $\omega = 1.7$) e as correspondentes versões que não requerem o armazenamento da matriz do sistema ("matrix-free" methods).

Códigos implementados	k	$\rho(G)$	Erro relativo
Jacobi	156	0.9511	1.96e-004
Gauss-Seidel	88	0.9045	9.47e-005
SOR	40	0.7000	2.27e-006
Jacobi-P	156	0.9511	1.96e-004
Gauss-Seidel-P	88	0.9045	9.47e-005
SOR-P	40	0.7000	2.27e-006

Tabela 4.1 Número de iterações, raio espectral da matriz de iteração e erro relativo.

É importante enfatizar que a solução do sistema corresponde exatamente ao valor da solução exata em cada ponto (x_i, y_j) da malha de discretização definida. Isto acontece por serem nulos os erros de truncatura, uma vez que a solução exata $u(x,y) = 100xy$ tem derivadas parciais de quarta ordem (e de ordem superior) nulas.

Na última coluna da tabela registam-se os erros relativos $\|\boldsymbol{x}_k - \boldsymbol{u}^*\|/\|\boldsymbol{u}^*\|$, onde $\boldsymbol{u}^*$ é a solução exata e $\boldsymbol{x}_k$ é a solução aproximada obtida na última iteração efetuada. Podemos observar que apenas no caso do SOR o erro é menor do que a tolerância usada

($tol = 10^{-5}$), embora seja $\rho(G_{SOR}) > 0.5$ (antes vimos que se $\rho < 0.5$ tem-se necessariamente $\|\boldsymbol{x}_{k+1} - \boldsymbol{u}^*\| \leqslant \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|$).

Neste exemplo inicial usamos $h = 0.1$ a que corresponde uma malha de $n + 2 = 11$ pontos em cada direção, ou seja, um sistema com $n^2 = 81$ incógnitas (os valores de u nos pontos interiores). Como é esperado, o método de Jacobi é o que tem convergência mais lenta e o método SOR, com o fator de relaxação $\omega = 1.7$, é o mais rápido. Enfatizamos que os códigos Jacobi e Jacobi-P requerem exatamente o mesmo número de iterações para produzir a mesma aproximação. O mesmo é verdade para os códigos Gauss-Seidel e Gauss-Seidel-P, SOR e SOR-P. O que difere muito são os tempos de execução, como se pode verificar na Tabela 4.2 onde estão registados os tempos de execução (em segundos) para diferentes valores de n.

n	Jacobi	Gauss-Seidel	SOR	Jacobi-P	Gauss-Seidel-P	SOR-P
25	2.5885	1.8547	0.4634	0.0129	0.0097	0.0026
50	137.4857	85.2456	19.0328	0.1033	0.0683	0.0200
100	8.4825e+03	7.6181e+03	1.1363e+03	1.0257	0.6759	0.2212
150	——	——	1.4704e+04	3.6259	2.5036	0.7976
200	——	——	——	8.0136	5.9109	1.9889
250	——	——	——	14.6332	11.5445	4.2723
300	——	——	——	23.9418	19.4556	8.1054

Tabela 4.2 Tempo de execução (em segundos) dos diferentes códigos.

Uma vez mais, fica clara a superioridade do método SOR (com $\omega = 1.7$) relativamente aos outros métodos aqui testados. Mas a conclusão principal é a de que os códigos que usam a matriz do sistema de forma explícita são extremamente ineficientes. Por exemplo, para $n = 150$ (isto corresponde a um sistema com 22500 equações), o código SOR usou cerca de 4 horas enquanto que o correspondente SOR-P terminou ao fim de 0.8 segundos. Em desfavor dos códigos Jacobi, Gauss-Seidel e SOR, acresce a necessidade do armazenamento explícito das n^2 entradas da matriz. Para $n = 300$, isto corresponde a ocupar cerca de 700Mbytes de memória do computador para armazenar uma matriz. Nas versões "P" dos mesmos códigos, este custo não existe.

Fica bem clara a importância de tirar partido da estrutura especial da matriz dos sistema que ocorre na resolução numérica da equação de Poisson. A partir daqui, não usaremos mais os códigos "gerais" Jacobi, Gauss-Seidel e SOR.

Na Figura 4.1 apresentamos um gráfico com o número de iterações requeridas por cada um dos códigos Jacobi-P, Gauss-Seidel-P e SOR-P ($w = 1.7$), com $n = 9$, e para valores de tol de 10^{-1} até 10^{-15}.

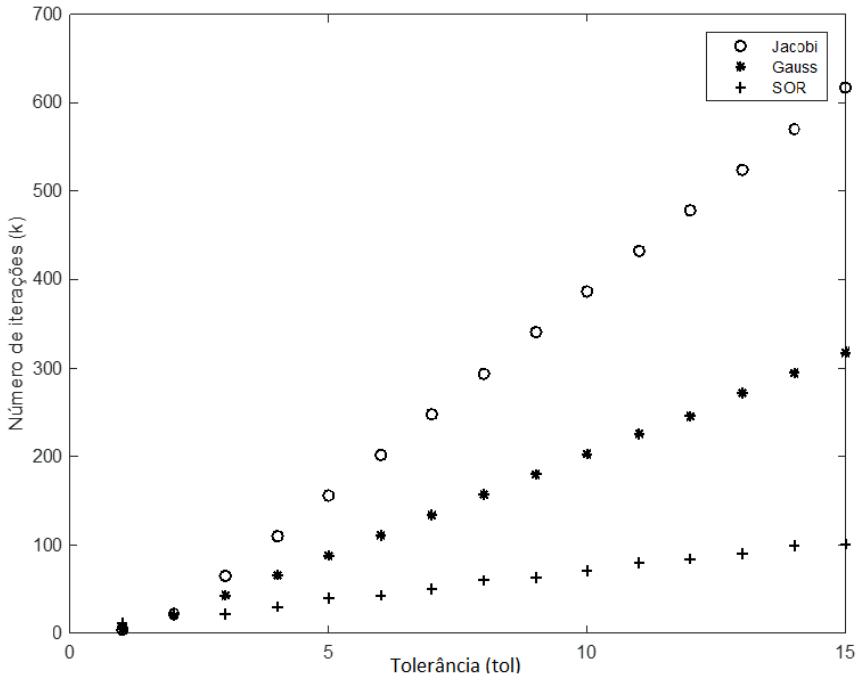


Figura 4.1 Número de iterações em função de $tol = 10^{-t}$, $t = 1, 2, \ldots 15$.

A partir da representação gráfica dos três métodos, podemos observar mais uma vez que, para os mesmos valores de tol, no gráfico de SOR os valores para o número de iterações são significativamente menores do que com os outros dois gráficos.

Na Figura 4.2 apresentamos, usando um mapa de cores, as soluções exatas para os valores da temperatura em equilíbrio nos pontos (x_i, y_j) da malha com $n = 100$.

4.2 Problema II

Determinaremos uma solução aproximada da mesma equação (3.2) mas alterando os valores de fronteira que vão ser os que correspondem ao caso ilustrado na Figura 3.1, isto é,

$$\begin{aligned} u(x,0) = 100, \quad u(x,1) = 0, \qquad 0 \leqslant x \leqslant 1; \\ u(0,y) = 100, \quad u(1,y) = 100, \qquad 0 \leqslant y \leqslant 1. \end{aligned}$$

Neste caso a solução exata não é conhecida pelo que, ao contrário do que se fez no exemplo anterior, não será possível determinar exatamente os erros das aproximações que vamos calcular. Este exemplo representa os casos em que é de facto necessária a resolução numérica do problema e tudo o que sabemos sobre os erros de truncatura é que são da ordem de grandeza de h^2.

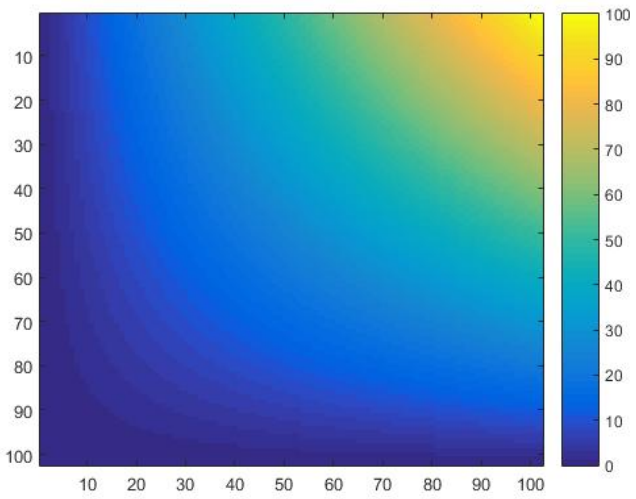


Figura 4.2 Valores de equilíbrio para a temperatura na placa.

Usaremos os códigos Jacobi-P, Gauss-Seidel-P, SOR-P e SOR-RB com os parâmetros de entrada $n = 6$, $tol = 10^{-4}$, $kmax = 10^4$, e ainda $\omega = 1.7$, no caso de SOR-P, e o valor ótimo expresso em (3.49) no caso de SOR-RB.

Nas Tabelas 4.3-4.6 apresentam-se as aproximações obtidas com os diferentes códigos nos 36 pontos interiores da malha definida, bem como o número de iterações k requeridas para cumprir o critério de paragem definido por (1.41). Observamos que, de uma maneira geral, as aproximações em cada ponto da malha coincidem em quatro algarismos, de acordo com o esperado.

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100.0	52.27	35.08	29.15	29.15	35.08	52.27	100.0
100.0	74.00	58.93	52.39	52.39	58.93	74.00	100.0
100.0	84.81	74.26	69.11	69.11	74.26	84.81	100.0
100.0	90.98	84.23	80.72	80.72	84.23	90.98	100.0
100.0	94.91	90.96	88.86	88.86	90.96	94.91	100.0
100.0	97.70	95.89	94.91	94.91	95.89	97.70	100.0
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Tabela 4.3 Resultados obtidos com Jacobi-P (k= 69).

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100.0	52.28	35.10	29.18	29.18	35.11	52.28	100.0
100.0	74.01	58.96	52.43	52.43	58.97	74.02	100.0
100.0	84.83	74.30	69.16	69.17	74.31	84.83	100.0
100.0	91.00	84.26	80.77	80.77	84.27	91.01	100.0
100.0	94.92	90.99	88.89	88.89	91.00	94.93	100.0
100.0	97.70	95.90	94.93	94.93	95.91	97.71	100.0
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Tabela 4.4 Resultados obtidos com Gauss-Seidel-P (k= 39).

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100.0	52.29	35.11	29.19	29.19	35.11	52.29	100.0
100.0	74.03	58.98	52.46	52.46	58.98	74.03	100.0
100.0	84.84	74.33	69.20	69.20	74.33	84.84	100.0
100.0	91.02	84.30	80.81	80.80	84.29	91.02	100.0
100.0	94.94	91.02	88.91	88.92	91.02	94.94	100.0
100.0	97.71	95.92	94.95	94.95	95.92	97.71	100.0
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Tabela 4.5 Resultados obtidos com SOR-P e $\omega = 1.7$ (k= 29).

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100.0	52.28	35.11	29.19	29.19	35.11	52.28	100.0
100.0	74.03	58.98	52.45	52.45	58.98	74.03	100.0
100.0	84.84	74.32	69.19	69.19	74.33	84.84	100.0
100.0	91.02	84.29	80.80	80.80	84.29	91.02	100.0
100.0	94.94	91.02	88.92	88.92	91.02	94.94	100.0
100.0	97.71	95.92	94.95	94.95	95.92	97.71	100.0
100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Tabela 4.6 Resultados obtidos com SOR-RB e ω ótimo (k= 14).

Na Tabela 4.6 observamos a importância de podermos determinar o valor ótimo ω_{opt} do parâmetro de relaxação. Comparando o número de iterações usadas por SOR-P e SOR-RB, é negligenciável a diferença, para cada um dos valores de ω testados, embora seja de notar

que, neste exemplo, o número de iterações de SOR-RB nunca seja superior ao de SOR-P.

Variando o parâmetro de relaxação observamos na Tabela 4.7 que, para alguns valores de ω, SOR-RB converge mais rapidamente do que SOR-P. Ainda relativamente aos resultados obtidos com SOR-RB, é de observar que com $\omega = 1.45$ é necessária menos uma iteração do que com $\omega_{opt} = 2/(1 + \sin(\frac{\pi}{7}))$. À primeira vista, isto parece contradizer a teoria já que se tem para os raios espectrais das matrizes de iteração

$$\rho(\omega_{opt}) = 0.3948 < \rho(1.45) = 0.45.$$

Nos nossos códigos usou-se a $\|.\|_\infty$ no critério de paragem que está expresso em (1.41). Que esta condição se cumpra com menos iterações para algum valor de $\omega \neq \omega_{opt}$ do que com o próprio ω_{opt} é uma consequência da observação feita no final da subsecção 1.1.1.

ω	SOR-P	SOR-RB
1.09	33	33
1.15	29	29
$2/(1 + \sin(\frac{\pi}{7}))$	15	14
1.45	15	13
1.65	26	24
1.75	35	34
1.90	93	91
1.98	469	466

Tabela 4.7 Número de iterações dos códigos SOR-P e SOR-RB variando ω.

À medida que se aumenta o número de pontos da malha observamos que na Tabela 4.8, o valor do raio espectral da matriz de iteração vai-se aproximando de 1. Podemos observar que este processo é mais rápido para o raio espectral da matriz de iteração do método de Jacobi, e mais lento para raio espectral da matriz de iteração do método de SOR-RB, conforme mostra a nossa tabela

n	$\rho(G_J)$	$\rho(G_{GS_{RB}})$	$\rho(G_{SOR_{RB}})$
6	0.90097	0.81174	0.39481
10	0.95949	0.92063	0.56039
20	0.98883	0.97779	0.74058
40	0.99707	0.99414	0.85779
60	0.99867	0.99735	0.90208
80	0.99925	0.99850	0.92534
100	0.99952	0.99903	0.93968
200	0.99988	0.99976	0.96922
300	0.99995	0.99989	0.97934
400	0.99997	0.99994	0.98445
500	0.99998	0.99996	0.98754

Tabela 4.8 Raio espectral das matrizes de iteração.

Na Figura 3.1, que apresentámos no início do Capítulo 3, estão representadas, usando um mapa de cores, as soluções para os valores da temperatura em equilíbrio nos pontos da placa, obtidas usando o método SOR_RB com o parâmetro ótimo, $n = 100$ e $tol = 10^{-4}$.

Conclusão

A importância da equação de Poisson resulta da diversidade de problemas que a ela estão associados. No nosso trabalho considerámos a resolução numérica desta equação, em uma e duas dimensões, com condições de fronteira de Dirichlet. Isto levou-nos ao foco central da tese que é o dos métodos iterativos para a solução de sistemas de equações lineares que resultam da discretização de domínios contínuos usando técnicas de diferenças finitas.

Procurámos sintetizar a teoria básica dos métodos de relaxação clássicos, o método de Jacobi, o método de Gauss-Seidel e o método de sobre-relaxação (SOR) dando especial ênfase, como não podia deixar de ser, aos raios espetrais das matrizes de iteração de que depende a convergência. A equação de Poisson discretizada por diferenças centradas para as derivadas de segunda ordem é um excelente caso de estudo para comparar estes métodos uma vez que são conhecidos os raios espectrais das matrizes de iteração para cada um dos métodos. No caso do método SOR também é conhecido o valor óptimo do parâmetro de relaxação.

Desenvolvemos e testámos códigos no Matlab. Para cada algoritmo desenvolvemos dois códigos, um que constrói de forma explícita a matriz dos sistema (no caso bidimensional, usámos somas de Kronecker para este efeito) e outro que, tal como é usual na prática, usa as relações entre as variáveis decorrentes das fórmulas de diferenças finitas, sem necessidade de armazenar matrizes, o que se traduz por grande economia no espaço de memória ocupado pelos dados do problema, para sistemas de grande dimensão. Para além disto, sem surpresa, registámos também grandes reduções nos tempos de execução, isto é, os métodos que na terminologia inglesa se designam por "matrix-free" são computacionalmente eficientes, ao contrário dos que usam as matrizes de forma explícita. Nos nossos testes confirmámos o que a teoria indica: a convergência do método de Gauss-Seidel é duas vezes mais rápida do que a do método de Jacobi e o SOR converge muito mais rapidamente do que o método de Gauss-Seidel se se usar o valor ótimo (ou próximo dele) do parâmetro de relaxação.

No problema a duas dimensões, nas implementações sequenciais destes algoritmos, os nós da malha de discretização são usualmente colocados pela ordem natural ou ordem lexicográfica. No caso dos métodos de Gauss-Seidel e SOR, esta ordenação impossibilita a implementação paralela dos algoritmos. Para ultrapassar esta dificuldade, pode usar-se uma ordenação que, à semelhança das casas de um tabuleiro de xadrez, distribui os nós

da malha por dois subconjuntos, R ("Red") e B ("Black"). A ideia é simples e eficaz: como no esquema resultante da fórmula dos cinco pontos, os nós de cada um destes conjuntos não estão ligados entre si, os valores de R só dependem de 4 valores de B e vice-versa. Assim, em cada iteração, os valores de R podem ser atualizados com os valores de B da iteração anterior para, logo a seguir, se atualizarem os valores de B usando os valores de R já produzidos na iteração em curso.

Uma reordenação dos nós corresponde a uma permutação das linhas e colunas da matriz do sistema mas isto não é verdade para as correspondentes matrizes de iteração, excepto no caso do método de Jacobi. Tal permutação de linhas e colunas é uma transformação de semelhança da matriz do sistema mas isto não garante que o raio espetral da matriz de iteração dos métodos de Gauss-Seidel não se altere quando a ordenação RB é usada. Curiosamente, formando de forma explícita as matrizes de iteração do SOR correspondentes às ordens lexicográfica e RB, obtivemos os mesmos raios espetrais mas não encontrámos na literatura resultados teóricos que expliquem esta verificação experimental.

Bibliografia

[1] Balsa, C., *Capítulo 2 - Sistemas de equações Lineares.*

Disponível em: http://www.ipb.pt/~balsa/teaching/MN08/Sist_Lin.pdf

Acesso em: 21/11/2019.

[2] Dahlquist, G. and Björck, Å. (1974), *Numerical Methods.* Prentice-Hall, Inc., Englewood Cloffs, N.J. .

[3] Demmel, J. W. (1997), *Applied Numerical Linear Algebra.* SIAM, Philadelphia, PA, USA.

[4] Evans, D. J. (1984), *Parallel S.O.R. iterative methods.* Parallel Computing, 1, pp.3-18.

Disponível em: http://users.tem.uoc.gr/~vagelis/Courses/Scientific_Computing/Papers/Parallel_SOR_Evans_84.pdf

Acesso em: 21/11/2019.

[5] Favarato, L. F., Mocha, M. and Camargo, R. S. (2018), *Alternative resolution of engineering problems by applying the finite difference method associated to S.O.R.*. Universidade Federal do Espírito Santo.

[6] Fortuna, A. de O. (2000), *Técnicas computacionais para dinâmica dos fluidos: conceitos básicos e aplicações.* USP, São Paulo.

[7] Kulkarni, D., Schmidt, D. and Tsui, Sze-kai (1999), *Eigenvalues of tridiagonal pseudo-Toeplitz matrices.* Linear Algebra and its Applications, Elsevier, 297, pp.63-80.

Disponível em: https://hal.archives-ouvertes.fr/hal-01461924/document

Acesso em: 21/11/2019.

[8] Melo, E. S. e Santos, P. I. (2013), *Métodos Numéricos: Aplicações em Sistemas Lineares e Aproximações de Funções*. Universidade Federal do Amapá.

Disponível em: http://www2.unifap.br/matematica/files/2017/07/tcc-2013-edelson-santos-Trabalho-de-Conclus%C3%A3o-de-Curso.pdf

Acesso em: 21/11/2019.

[9] Pina, H. (2010), *Métodos Numéricos*. Escolar Editora, Lisboa.

[10] Quarteroni, A. and Saleri, F. (2003), *Scientific Computing with MATLAB*. Springer-Verlag, Berlin.

[11] Rocho, V. R. (2014), *Métodos Iterativos para a Solução da Equação de Poisson*. Instituto de Matemática da Universidade Federal do Rio Grande do Sul.

Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/54737/000852494.pdf?sequence=1

Acesso em: 21/11/2019.

[12] Schmidt, M. (2012), *Aplicações do método de sobre-relaxação sucessiva em sistemas de equações lineares*. Universidade Federal de Santa Maria.

[13] Valença, M. R. (1993), *Métodos Numéricos (3.ª ed.)*. Livraria Minho, Braga.

[14] Yueh, W. C. (2005), *Eigenvalues of several tridiagonal matrices*. Applied Mathematics E-Notes, 5, pp.66-74.

Disponível em: http://www.kurims.kyoto-u.ac.jp/EMIS/journals/AMEN/2005/040903-7.pdf

Acesso em: 21/11/2019.

Anexos

Anexo A
Implementação dos métodos no Matlab

Código usado para determinar a solução exata do Problema I.

```
function S=solucao_exata(n)
%
% SOLUÇÃO_EXATA Solução exata para o Problema I em [1].
%
% INPUT:  n - número de pontos interiores da malha em cada direção
%
% OUTPUT: S - matriz com as soluções exatas nos pontos da malha, de
%             ordem N^2, com N=n+2
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
%
N=n+2;
h=1/(N-1);                  %passo da malha
%
%Valores na fronteira da malha
S(1,1:N)=0;                 %fronteira inferior
S(1:N,1)=0;                 %fronteira esquerda
S(1:N,N)=100*(0:h:1);       %fronteira direita
S(N,1:N)=100*(0:h:1);       %fronteira superior
%
%Valores nos pontos interiores da malha
for i=2:N-1
      for j=2:N-1
        x=(i-1)*h;
        y=(j-1)*h;
        S(i,j)=100*x*y;
      end
end
%
S=flipud(S);
end
```

Código para determinar a matriz do sistema e o vetor coluna dos termos independentes do problema de Poisson bidimensional que usamos nos códigos (A.1.1), (A.2.1) e (A.3.1).

```
function [A, b]=matriz_vetor(n)
%
% MATRIZ_VETOR  Matriz do sistema e vetor dos termos idependentes da
%               discretização da equação de Poisson bidimensional.
%
% INPUT:  n - número de pontos interiores da malha em cada direção
%             (n+2 pontos no total, com os pontos da fronteira)
%
% OUTPUT: A - matriz simples do sistema de ordem (n+2)^2
%         b - vetor coluna dos (n+2)^2 termos independentes
%
% Ver Problema I em [1], pag.48.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
%
h=1/(n+1);                  %passo da malha
%
%Valores na fronteira da malha
%
alfa(1:n+2)=0;              %fronteira esquerda
beta(1:n+2)=100*(0:h:1);    %fronteira direita
delta(1:n+2)=100*(0:h:1);   %fronteira superior
gamma(1:n+2)=0;             %fronteira inferior
%
%Primeiro bloco de b
v1=[alfa(2)+gamma(2), gamma(3:n), gamma(n+1)+beta(2)];
%
%Blocos interiores de b
v2=[alfa(3:n); zeros(n-2); beta(3:n)];
v2=v2(:)';
%
%Último bloco de b
v3=[alfa(n+1)+delta(2), delta(3:n), delta(n+1)+beta(n+1)];
%
b=[v1,v2,v3];               %vetor b
%
A=gallery('poisson',n);
A=full(A);                  %matriz de Poisson
end
```

A.1 Códigos de Jacobi

A.1.1 Jacobi

```
function [u, iter]=Jacobi(A, b, tol, kmax)
%
% JACOBI  Método iterativo de Jacobi.
%
% [u, iter]=jacobi(A, b, tol, kmax) determina uma aproximação para
% a solução do sistema Ax=b usando o método iterativo de Jacobi,
% tomando como aproximação  inicial o vector nulo.
%
% INPUT:  A - matriz simples do sistema (ordem n)
%         b - termos independentes (vetor coluna)
%         tol - tolerância para o erro (relativo)
%         kmax - número máximo de iterações
%
% OUTPUT:  u - solução (vetor coluna)
%          iter - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MJ=-inv(D)*(L+U) é a matriz de iteração de
% Jacobi. O método converge sse o raio espectral de MJ < 1
% (consultar Teorema 1.3 em [1], pag.13).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
%
n=length(b);              %comprimento do vetor b
u=zeros(n,1);             %aproximação inicial (vector nulo)
erro=tol+1;               %qualquer valor maior do que tol serve
iter=0;                   %inicialização do contador de iterações
h=zeros(n,1);             %pre-alocar para maior rapidez
%
while erro>tol && iter < kmax
      y=u;                %cópia dos valores da iteração anterior
      for i=1:n
        h(i)=(b(i)-dot(A(i,:),y))/A(i,i);    %fórmula (2.8)
        u(i)=u(i)+h(i);
      end
      erro=norm(h,inf)/(1+norm(u,inf));      %critério de paragem (1.41)
      iter=iter+1;
end
%
if erro<=tol
       disp(['Atingiu-se a precisão desejada em ',num2str(iter),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax'...
             'é insuficiente para permitir alcançar a precisão desejada.']);
end
%
end
```

A.1.2 Jacobi_Poisson

```
function [u,ITER]=Jacobi_Poisson(n, tol, kmax)
%
% JACOBI_POISSON   Método iterativo de Jacobi para a resolução do problema
%                  de Poisson bidimensional.
%
% [u,ITER]=Jacobi_Poisson(n, tol, kmax) determina uma aproximação para a
% solução do sistema Ax=b da discretizaçao do problema de Poisson
% bidimensional (com ordenação lexicográfica para as incógnitas), usando o
% o método iterativo de Jacobi com o vetor nulo como aproximação inicial.
%
% INPUT: n - número de pontos interiores da malha em cada direção
%        tol - tolerância para o erro (relativo)
%        kmax - número máximo de iterações
%
% OUTPUT: u - soluções nos pontos da malha (matriz de ordem N^2, com N=n+2)
%         ITER - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MJ=-inv(D)*(L+U) é a matriz de iteração de
% Jacobi. O método converge sse o raio espectral de MJ < 1
% (consultar Teorema 1.3 em [1], pag.13).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
% Usa-se o índice i para indicar a linha e o índice j para a coluna na
% matriz u. A aproximação da solução no ponto de coordenadas (x_i,y_j)
% encontra-se na entrada (i,j) da matriz u.
%
N=n+2;  %número total de pontos (interiores e de fronteira) em cada direção
%
h=1/(N-1); %passo da malha
%
%Valores de fronteira do Problema I em [1].
u(1,1:N)=0;                 %fronteira inferior
u(1:N,1)=0;                 %fronteira esquerda
u(1:N,N)=100*(0:h:1);       %fronteira direita
u(N,1:N)=100*(0:h:1);       %fronteira superior
%
%Valores de fronteira do Problema II em [1].
%u(1,1:N)=100;              %fronteira inferior
%u(1:N,1)=100;              %fronteira esquerda
%u(1:N,N)=100;              %fronteira direita
%u(N,1:N)=0;                %fronteira superior
%
w=u;
%
%Aproximação inicial para os pontos interiores da malha
u(2:N-1,2:N-1)=0;
%
DIFF=tol+1;                 %qualquer valor maior do que tol serve
ITER=0;                     %inicialização do contador de iterações
```

```
%
while DIFF>tol  &&  ITER<kmax
    for i=2:N-1
        for j=2:N-1
            w(i,j)=(u(i-1,j)+u(i,j-1)+u(i,j+1)+u(i+1,j))/4; %fórmula (4.1)
        end
    end
    %critério de paragem (1.41)
    DIFF=max(max(abs(w-u)))/(1+max(max(abs(w(2:N-1,2:N-1)))));
    u=w;                        %cópia dos valores da iteração atual
    ITER=ITER+1;
end
%
if DIFF<=tol
       disp(['Atingiu-se a precisão desejada em ',num2str(ITER),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax é'...
             'insuficiente para permitir alcançar a precisão desejada.']);
end
%
u=flipud(u);   %matriz com as soluções aproximadas;
               %para obter x na formulação Ax=b, basta vetorizar u,
               %usando x=reshape(u,N^2*N^2,1)
end
```

A.2 Códigos de Gauss-Seidel

A.2.1 Gauss_Sedeil

```
function [u, iter]=Gauss_Seidel(A, b, tol, kmax)
%
% GAUSS_SEIDEL  Método iterativo de Gauss-Seidel.
%
% [u, iter]=gauss_seidel_geral(A,b,tol,kmax) determina uma aproximação
% para a solução do sistema Ax=b usando o método iterativo de Gauss-Seidel,
% tomando como aproximação inicial o vector nulo.
%
% INPUT:  A - matriz simples do sistema (ordem n)
%         b - termos independentes (vetor coluna)
%         tol - tolerância para o erro (relativo)
%         kmax - número máximo de iterações
%
% OUTPUT:  u - solução (vetor coluna)
%          iter - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MGS=-inv(D+L)*U é a matriz de iteração de
% Gauss-Seidel. O método converge sse o raio espectral de MGS < 1.
% (consultar Teorema 1.3 em [1], pag.13).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
%
n=length(b);                %comprimento do vetor b
u=zeros(n,1);               %aproximação inicial (vector nulo)
erro=tol+1;                 %qualquer valor maior do que tol serve
iter=0;                     %inicialização do contador de iterações
h=zeros(n,1);               %pre-alocar para maior rapidez
%
while erro>tol && iter<kmax
    for i=1:n
      h(i)=(b(i)-dot(A(i,1:n),u))/A(i,i);   %fórmula (2.18)
      u(i)=u(i)+h(i);
    end
    erro=norm(h,inf)/(1+norm(u,inf));       %critério de paragem (1.41)
    iter=iter+1;
end
%
if erro>tol
       disp(['Atingiu-se a precisão desejada em ',num2str(iter),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax é insuficiente ',...
             'para permitir alcançar a precisão desejada.']);
end
%
end
```

A.2.2 Gauss_Seidel_Poisson

```
function [u, ITER]=Gauss_Seidel_Poisson(n, tol, kmax)
%
% GAUSS_SEIDEL_POISSON   Método iterativo de Gauss-Seidel para a resolução
%                        do problema de Poisson bidimensional.
%
% [u, ITER]=Gauss_SeideL_particular(n, tol, kmax) determina uma aproximação
% para a solução do sistema Ax=b da discretizaçao do problema de Poisson
% bidimensional (com ordenação lexicográfica para as incógnitas), usando o
% método iterativo de Gauss-Seidel com o vetor nulo como aproximação inicial.
%
% INPUT: n - número de pontos interiores da malha em cada direção
%        tol - tolerância para o erro (relativo)
%        kmax - número máximo de iterações
%
% OUTPUT: u - soluções nos pontos da malha (matriz de ordem N^2, com N=n+2)
%         ITER - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MGS=-inv(D+L)*U é a matriz de iteração de
% Gauss-Seidel. O método converge sse o raio espectral de MGS < 1
% (consultar Teorema 1.3 em [1], pag.13).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
% Usa-se o índice i para indicar a linha e o índice j para a coluna na
% matriz u. A aproximação da solução no ponto de coordenadas (x_i,y_j)
% encontra-se na entrada (i,j) da matriz u.
%
N=n+2; %número total de pontos (interiores e de fronteira) em cada direção
%
h=1/(N-1); %passo da malha
%
%Valores de fronteira do Problema I em [1].
u(1,1:N)=0;                 %fronteira inferior
u(1:N,1)=0;                 %fronteira esquerda
u(1:N,N)=100*(0:h:1);       %fronteira direita
u(N,1:N)=100*(0:h:1);       %fronteira acima
%
%Valores de fronteira do Problema II em [1].
%u(1,1:N)=100;              %fronteira inferior
%u(1:N,1)=100;              %fronteira esquerda
%u(1:N,N)=100;              %fronteira direita
%u(N,1:N)=0;                %fronteira superior
%
w=u;
%
%Aproximação inicial para os pontos interiores da malha
u(2:N-1,2:N-1)=0;
%
DIFF=tol+1;                 %qualquer valor maior do que tol serve
ITER=0;                     %inicialização do contador de iterações
```

```
%
while DIFF>tol &&  ITER<kmax
    for i=2:N-1
        for j=2:N-1
            w(i,j)=(w(i-1,j)+w(i,j-1)+u(i,j+1)+u(i+1,j))/4; %fórmula (4.2)
        end
    end
    %critério de paragem (1.41)
    DIFF=max(max(abs(w-u)))/(1+max(max(abs(w(2:N-1,2:N-1)))));
    u=w;                        %cópia dos valores da iteração atual
    ITER=ITER+1;
end
%
if DIFF<=tol
       disp(['Atingiu-se a precisão desejada em ',num2str(ITER),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax é'...
             'insuficiente para permitir alcançar a precisão desejada.']);
end
%
u=flipud(u); %matriz com as soluções aproximadas;
             %para obter x na formulação Ax=b, basta vetorizar u,
             %usando x=reshape(u,N^2*N^2,1)
end
```

A.3 Códigos de SOR

A.3.1 SOR

```
function [u, iter]=SOR(A, b, tol, kmax, p)
%
% SOR      Método iterativo de SOR.
%
% [u, iter]=SOR(A, b, tol, kmax, p) determina uma aproximação para a
% solução do sistema Ax=b usando o método iterativo de SOR, tomando como
% aproximação inicial o vector nulo.
%
% INPUT:  A - matriz simples do sistema (ordem n)
%         b - termos independentes (vetor coluna)
%         tol - tolerância para o erro (relativo)
%         kmax - número máximo de iterações
%         p - parâmetro de relaxação
%
% OUTPUT:  u - solução (vetor coluna)
%          iter - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MSOR=inv(D+p*L)*[(1-p)*D-p*U] é a matriz
% de iteração de SOR, para um dado valor do parâmetro de relaxação p.
% A condição 0<p<2 é uma condição necessária de convergência
% (consultar Teorema 2.4 em [1], pag.27).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
n=length(b);                %comprimento do vetor b
u=zeros(n,1);               %aproximação inicial (vector nulo)
erro=tol+1;                 %qualquer valor maior do que tol serve
iter=0;                     %inicialização do contador de iterações
h=zeros(n,1);               %pre-alocar para maior rapidez
%
while erro>tol && iter<kmax
    for i=1:n
      h(i)=(b(i)-dot(A(i,1:n),u))/A(i,i);     %fórmula (2.28)
      u(i)=u(i)+p*h(i);
    end
    erro=norm(h,inf)/(1+norm(u,inf));         %critério de paragem (1.41)
    iter=iter+1;
end
%
if erro<=tol
       disp(['Atingiu-se a precisão desejada em ',num2str(iter),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax é'...
             'insuficiente para permitir alcançar a precisão desejada.']);
end
end
```

A.3.2 SOR_Poisson

1. Ordenação lexicográfica

```
function [u, ITER]=SOR_Poisson(n, tol, kmax, p)
%
% SOR_POISSON  Método iterativo de SOR para a resolução do problema de
%              Poisson bidimensional (ordenação lexicográfica).
%
% [u, ITER]=SOR_Poisson(n, tol, kmax, p) determina uma aproximação para
% a solução do sistema Ax=b da discretizaçao do problema de Poisson
% bidimensional (com ordenação lexicográfica para as incógnitas), usando o
% método iterativo de SOR com o vetor nulo como aproximação inicial.
%
% INPUT: n - número de pontos interiores da malha em cada direção
%        tol - tolerância para o erro (relativo)
%        kmax - número máximo de iterações
%        p - parâmetro de relaxação
%
% OUTPUT: u - soluções nos pontos da malha (matriz de ordem N^2, com N=n+2)
%         ITER - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MSOR=inv(D+p*L)*[(1-p)*D-p*U] é a matriz
% de iteração de SOR, para um dado valor p do parâmetro de relaxação.
% A condição 0<p<2 é uma condição necessária de convergência
% (consultar Teorema 2.4 em [1], pag.27).
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
% Usa-se o índice i para indicar a linha e o índice j para a coluna na
% matriz u. A aproximação da solução no ponto de coordenadas (x_i,y_j)
% encontra-se na entrada (i,j) da matriz u.
%
N=n+2; %número total de pontos (interiores e de fronteira) em cada direção
%
h=1/(N-1); %passo da malha
%
%Valores de fronteira do Problema I em [1].
u(1,1:N)=0;                  %fronteira inferior
u(1:N,1)=0;                  %fronteira esquerda
u(1:N,N)=100*(0:h:1);        %fronteira direita
u(N,1:N)=100*(0:h:1);        %fronteira superior
%
%Valores de fronteira do Problema II em [1].
%u(1,1:N)=100;               %fronteira inferior
%u(1:N,1)=100;               %fronteira esquerdo
%u(1:N,N)=100;               %fronteira direito
%u(N,1:N)=0;                 %fronteira superior
%
w=u;
%Aproximação inicial para os pontos interiores da malha
u(2:N-1,2:N-1)=0;
```

```
%
DIFF=tol+1;                     %qualquer valor maior do que tol serve
ITER=0;                         %inicialização do contador de iterações
%
while DIFF>tol && ITER<kmax
    for i=2:N-1
        for j=2:N-1
          %fórmula (4.3)
          w(i,j)=(1-p)*u(i,j)+p*(w(i-1,j)+w(i,j-1)+u(i,j+1)+u(i+1,j))/4;
        end
    end
    %critério de paragem (1.41)
    DIFF=max(max(abs(w-u)))/(1+max(max(abs(w(2:N-1,2:N-1)))));
    u=w;                        %cópia dos valores da iteração atual
    ITER=ITER+1;
end
%
if DIFF<=tol
      disp(['Atingiu-se a precisão desejada em ',num2str(ITER),...
            ' iterações.']);
else
      disp(['O método diverge ou converge mas kmax é'...
            'insuficiente para permitir alcançar a precisão desejada.']);
end
%
u=flipud(u); %matriz com as soluções aproximadas;
             %para obter x na formulação Ax=b, basta vetorizar u,
             %usando x=reshape(u,N^2*N^2,1)
end
```

2. Ordenação "Red-Black"

```
function [u,ITER]=SOR_RB(n, tol, kmax, P)
%
% SOR_RB    Método iterativo de SOR para a resolução do problema de
%           Poisson bidimensional (ordenação "Red-Black").
%
% [u, ITER]=SOR_RB(n, tol, kmax) determina uma aproximação para
% a solução do sistema Ax=b da discretizaçao do problema de Poisson
% bidimensional (com ordenação "Red-Black" para as incógnitas), usando o
% método iterativo de SOR com o parâmetro de relaxação e o vetor nulo
% como aproximação inicial .
%
% INPUT: n - número de pontos interiores da malha em cada direção
%        tol - tolerância para o erro (relativo)
%        kmax - número máximo de iterações
%        P -  Parâmetro de relaxação
%
% OUTPUT: u - soluções nos pontos da malha (matriz de ordem N^2, com N=n+2)
%         ITER - número de iterações
%
% Com A=D+L+U, D diagonal, L e U estritamente triangulares, inferior e
% superior, respetivamente, MSOR=inv(D+P*L)*[(1-P)*D-P*U] é a matriz
% de iteração de SOR, para o valor ótimo P do parâmetro de relaxação.
%
% Consultar o Teorema 3.3 em [1], pag.43-44, e a expressão (3.49) para o
% valor ótimo do parâmatro de relaxação.
%
% Ver Algoritmo geral para métodos iterativos em [1], pag.20.
%
% [1] P. Jamba, "Resolução Numérica da Equação de Poisson".
%     Dissertação de mestrado, UM, 2019.
%
% Usa-se o índice i para indicar a linha e o índice j para a coluna na
% matriz u. A aproximação da solução no ponto de coordenadas (x_i,y_j)
% encontra-se na entrada (i,j) da matriz u.
%
N=n+2;  %número total de pontos (interiores e de fronteira) em cada direção
%
h=1/(N-1);    %passo da malha
%
%Valores de fronteira do Problema I em [1].
%u(1,1:N)=0;                 %fronteira inferior
%u(1:N,1)=0;                 %fronteira esquerda
%u(1:N,N)=100*(0:h:1);       %fronteira direita
%u(N,1:N)=100*(0:h:1);       %fronteira superior
%
%Valores de fronteira do Problema II em [1].
u(1,1:N)=100;               %fronteira inferior
u(1:N,1)=100;               %fronteira esquerda
u(1:N,N)=100;               %fronteira direita
u(N,1:N)=0;                 %fronteira superior
%
w=u;
%
%Aproximação inicial para os pontos interiores da malha
```

```
u(2:N-1,2:N-1)=0;
%
%PARÂMETRO DE RELAXAÇÃO ÓTIMO
%P=2/(1+sin(pi/(n+1)));
%
TOL=tol;
DIFF=TOL+1;                    %qualquer valor maior do que tol serve
ITER=0;                        %inicialização do contador de iterações
%
while DIFF>TOL && ITER<kmax
    %Atualização dos pontos vermelhos
    for i=2:N-1
        for j=2+rem(i,2):2:N-1
            %fórmula (4.4)
            w(i,j)=(1-P)*w(i,j)+P*(u(i-1,j)+u(i,j-1)+u(i,j+1)+u(i+1,j))/4;
        end
    end
    %Atualização dos pontos pretos
    for i=2:N-1
        for j=2+rem(i+1,2):2:N-1
            %fórmula (4.5)
            w(i,j)=(1-P)*w(i,j)+P*(w(i-1,j)+w(i,j-1)+w(i,j+1)+w(i+1,j))/4;
        end
    end
    %critério de paragem (1.41)
    DIFF=max(max(abs(w-u)))/(1+max(max(abs(w(2:N-1,2:N-1)))));
    u=w;                       %cópia dos valores da iteração atual
    ITER=ITER+1;
end
%
if DIFF<=TOL
       disp(['Atingiu-se a precisão desejada em ',num2str(ITER),...
             ' iterações.']);
else
       disp(['O método diverge ou converge mas kmax é'...
             'insuficiente para permitir alcançar a precisão desejada.']);
end
%
u=flipud(u); %matriz com as soluções aproximadas;
             %para obter x na formulação Ax=b, basta vetorizar u,
             %usando x=reshape(u,N^2*N^2,1)
end
```