Resolução numérica da equação de Black-Scholes em uma e duas dimensões

Cassiano Reinert Novais dos Santos

Trabalho de Conclusão de Curso apresentado
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Bacharel em Matemática Aplicada e Computacional

Curso: Bacharelado em Matemática Aplicada e Computacional com habilitação em Estatística Econômica

Orientador: Prof. Dr. Pedro da Silva Peixoto

Coorientador: Prof. Dr. Alexandre Roma

São Paulo, abril de 2014

Resolução numérica da equação de Black-Scholes em uma e duas dimensões

Agradecimentos

A incluir.

Resumo

Este trabalho de conclusão de curso tem por objetivo a resolução numérica da Equação de Black-Scholes em uma e duas dimensões. Empregamos o método de Euler Implícito para discretizar a variável temporal e diferenças centradas para a variável(is) espacial(is) da equação do calor, obtida através de mudanças de variáveis da Equação de Black-Scholes. O método de Euler Implícito produz um sistema linear para cada instante de tempo, que é resolvido através de um método direto para a equação do calor em uma dimensão e de métodos indiretos tanto para uma quanto para duas dimensões. Empregamos métodos simples de paralelismo para resolver os sistemas lineares nas duas dimensões.

Palavras-chave: métodos numéricos, equação de Black-Sholes, computação paralela.

Abstract

This course conclusion work aims to numerically solve the Black-Scholes equation in one and two dimensions. We employ the Backward Euler method to discretize the temporal variable and centered differences for the space variable(s) for Heat Equation, obtained through change variables of the Black-Scholes equation. The Implicit Euler method produces a linear system for each time instant, that is solved by a direct method to the Heat Equation in one dimension and indirect methods for both one and two two dimensions. We employ simple parallelism methods to solve the linear systems in two dimensions.

Keywords: numerical methods, Black-Scholes equation, paralell computing.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Opções são um tipo de derivativo, um instrumento financeiro que, como o nome diz, deriva seu valor de um ativo adjacente que pode ser uma ação, moeda, taxa de juros, *commodity* etc. Uma opção estabelece um contrato que dá ao comprador o direito de comprar ou vender o ativo adjacente a um preço previamente definido numa data especificada ou até essa data. São muito usadas no mercado financeiro como instrumento de *hedge* - um seguro contra movimentos indesejáveis de preços.

Em 1973, Fischer Black e Myron Scholes publicaram o artigo "The Pricing of Options and Corporate Liabilities" no periódico *Journal of Political Economy* que definia um modelo matemático de precificação de opções.

Este modelo foi adotado rapidamente, e se tornou eventualmente o mais famoso, e mais utilizado em finanças. Sua relevância como modelo de precificação de derivativos ficou consagrada após Miryon Scholes, Robert Merton e Fischel Black receberem o prêmio Nobel de Economia em 1997. A partir daí, o mercado de opções cresceu rapidamente e a aplicação de matemática sofisticada na área financeira passou a se tornar intensa, levando ao desenvolvimento de novos campos em matemática e engenharia financeira.

A equação de Black-Scholes pode ser transformada em uma equação diferencial parcial parabólica cuja solução análitica é conhecida porém seu cálculo envolve o cálculo de integrais impróprias e, por isso, métodos numéricos de resolução da equação parabólica podem ser mais eficientes. Por outro lado, a equação de Black-Scholes com solução analítica se refere a uma classe de opções bastante específica, e variações da equação para abarcar outros tipos de opções e arranjos desembocam em modelos que não têm solução analítica ou são de difícil obtenção.

Métodos numéricos também são uma boa maneira de resolver a equação de Black-Scholes quando esta se refere a um portfólio que contém duas ou mais opções. Nesse caso a equação parabólica derivada contém duas ou mais variavéis espaciais e soluções exatas se tornam progressivamente difícies.

Complementar ao uso de métodos numéricos, a modelagem de problemas financeiros através de equações diferenciais parciais tem sido favorecida pela cada vez mais ampla aplicação de algoritmos que são executados paralelamente em processadores multi-core e/ou *clusters* de processadores. Para isso, são usadas tecnologias que estão disponíveis gratuitamente como aplicativos FOSS (software livre e de código aberto, na sigla em inglês).

Em particular, a tecnologia OpenMP - voltada para soluções de paralelização em ambiente de memória compartilhada - nascida de um consórcio entre grandes empresas de tecnologia da informação - torna relativamente simples a paralelização de algoritmos oriundos de métodos numéricos.

1.1 Objetivos

Este trabalho de conclusão de curso tem por objetivo analisar a adequação de métodos numéricos na resolução da equação de Black-Scholes em uma e duas dimensões, bem como a possibilidade de implementações desses métodos em ambiente de paralelização com memória compartilhada.

1.2 Organização do Trabalho

No Capítulo 2 apresentamos o modelo matemático que serve de motivação para a resolução numérica de uma equação parabólica. Segue-se a descrição e análise do método numérico escolhido para resolução dessa equação. O capítulo 4 apresenta os métodos empregados para resolução de sistemas lineares.

O Capítulo 5 apresenta a Inteface de Programação de Aplicativos (API, na sigla em inglês) *OpenMP*, uma API voltada para programação de aplicações que rodam em paralelo.

Finalmente, no capítulo 6 mostramos os resultados obtidos para a equação do calor em uma e duas dimensões.

No último capítulo apresentamos as conclusões.

Capítulo 2

Modelo matemático

No estudo de equações a derivadas parciais é muito comum a modelagem de problemas oriundos da Física. A própria classificação destas equações definem problemas típicos das ciências físicas.

- Equação elíptica: Equação de Poisson, aplicada ao estudo do fenômeno da distribuição do calor em uma barra, superfície ou volume;
- Equação parabólica: Equação de Fourier, que descreve a propagação do calor em sólidos, ao longo do tempo;
- Equação hiperbólica: Equação da onda, descrevendo a propagação de diversos tipos de ondas em diversos meios materiais ou não, estudada em acústica, eletromagnetismo, e dinâmica dos fluidos.

Apesar de serem introduzidas geralmente envolvendo problemas das ciências físicas, as equações a derivadas parciais também são usadas para modelar problemas de outras naturezas. Na teoria de probabilidade aplicada em finanças, o modelo de Black e Scholes [eMS73], é uma equação diferencial que envolve a determinação teórica do preço de opções européias. Este modelo é usado extensamente por fundos de investimentos, bancos e outras instituições financeiras, a fim de determinar preços de ações no sentido de "eliminar o risco" de perda. Scholes recebeu o prêmio Nobel de Economia em 1997 junto com Robert C. Merton, que expandiu o entendimento do modelo matemático de precificação de opções.

É usada para determinar preço de opções de compra (*call*), e opções de venda (*put*). No caso de uma opção de compra, o emissor da opção vende o direito de comprar a ação adjacente pelo preço contratado caso o comprador da opção deseje comprar a ação. No caso de uma opção de venda, o emissor da opção vende o direito de vender a ação. Portanto ele é obrigado a comprar a ação caso o comprador da opção deseje vendê-la. [1]

O preço de uma opção é uma função de 5 variáveis: preço da ação adjacente, preço de exercício da opção (*strike price*), tempo até a expiração do exercício da opção, volatilidade da ação adjacente, e a taxa de juros livre de risco.

Opções são classificadas em famílias com respeito à data de sua expiração. A grande maioria classificam-se ou como opções européias ou como opções americanas. Se a opção é européia, ela pode ser exercida somente na data de expiração. Se a opção é americana, ela pode ser exercida em qualquer momento antes da data de expiração. [2]

Uma solução $V(S,t)$ para a equação de Black-Scholes mostra o preço da opção que deve ser exercido a fim de estabelecer um *hedge*, [3] ou seja, o valor em que se deve comprar ou vender a ação adjacente, para evitar perdas substanciais.

[1] http://pt.wikipedia.org/wiki/Mercado_de_opções. Página visitada em abril de 2014

[2] http://en.wikipedia.org/wiki/European_call. Página visitada em abril de 2014

[3] http://pt.wikipedia.org/wiki/Cobertura_(finanças). Página visitada em maio de 2014. *Hedge* - cobertura em inglês - é uma palavra inglesa usada em finanças para designar "um instrumento que visa proteger operações financeiras contra o risco de grandes variações de preço de determinado ativo"

O modelo descreve uma equação diferencial que pode ser reduzida à uma equação de difusão, cuja solução numérica pode ser obtida através da discretização pelo Método de diferenças finitas. Nesse trabalho iremos usar um método implícito para obter aproximações para equação de Black-Scholes.

A equação de Black-Scholes, definida em [eMS73] é

$$\begin{aligned} \frac{\partial V}{\partial t} + \sigma^2 S^2 \frac{\partial^2}{\partial^2 S^2} \ + rS\frac{\partial V}{\partial S} - rV = 0, 0 \leq S, 0 \leq t \leq T, \\ V(S,T) = \max\{S-K,0\}, \quad 0 \leq S, \\ V(0,t) = 0, \quad 0 \leq t \leq T, \\ V(S,t) \to S, \quad S \to \infty. \end{aligned} \tag{2.1}$$

onde
S - preço da ação,
$V(S,t)$ - preço da opção em função do tempo e do preço da ação,
r - taxa de juros livre de risco anualizada,
σ - "volatilidade dos retornos da ação"

No caso de uma opção de compra européia, as condições inicial e de contorno são, respectivamente,

$$\begin{aligned} V(S,T) = \max\{S-K,0\}, \quad 0 \leq S, \\ V(0,t) = 0, \quad 0 \leq t \leq T, \\ V(S,t) \to S, \quad S \to \infty. \end{aligned} \tag{2.2}$$

A condição final $V(S,T)$ indica que o preço da opção valerá a diferença entre o preço do ativo adjacente, e o preço de exercício K, caso essa difença seja positiva.

A equação de Black-Scholes é uma equação a derivadas parciais na variável dependente V, e nas variáveis independentes t e S. É uma equação diferencial parcial de segunda ordem com coeficientes variáveis. Esta equação pode ser transformada em uma equação de difusão, numa primeira aproximação, através das seguintes mudanças de variáveis.

$$\begin{aligned} &S = e^x, \\ &t = T - \frac{2\tau}{\sigma^2}, \\ &V(S,t) = v(x,\tau) = v(\ln(S), \frac{\sigma^2}{2}(T-t)). \end{aligned} \tag{2.3}$$

Calculando as derivadas parciais $\frac{\partial V}{\partial t}, \frac{\partial V}{\partial S}, \frac{\partial^2 V}{\partial S^2}$, expressas em função das derivadas de v em relação a x e τ, e substituindo as expressões obtidas na equação de Black-Scholes obtemos [4]

$$\begin{aligned} &\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2} + (\frac{2r}{\sigma^2} - 1)\frac{\partial v}{\partial x} - \frac{2r}{\sigma^2} v, -\infty \leq x \leq \infty, 0 \leq t \leq \frac{\sigma^2}{2} T \\ &v(x,0) = V(e^2, T) = f(e^x), -\infty \leq x \leq \infty \end{aligned} \tag{2.4}$$

A fim de eliminar as parcelas da derivada parcial de v em relação a x e da função v no segundo termo da equação 2.4, fazemos $k = \frac{2r}{\sigma^2}, t = \tau, v(x,t) = \mathrm{e}^{\alpha x + \beta t} u(x,t) = \phi u$. As derivadas parciais de de v em relação a x e t serão dadas por

[4] Para demonstração, ver http://www.math.tamu.edu/ stecher/425/Sp12/blackScholesHeatEquation.pdf, e http://en.wikipedia.org/wiki/Black-Scholes_model

$$\begin{aligned}\frac{\partial v}{\partial t} &= \beta\phi u + \phi\frac{\partial u}{\partial t}\\ \frac{\partial v}{\partial x} &= \alpha\phi u + \phi\frac{\partial u}{\partial x}\\ \frac{\partial^2 v}{\partial x^2} &= \alpha^2\phi u + 2\alpha\phi\frac{\partial u}{\partial x} + \phi\frac{\partial^2 u}{\partial x^2}\end{aligned} \tag{2.5}$$

Fazendo ainda

$$\begin{aligned}\alpha &= \frac{-1}{2}(\kappa-1) = \frac{\sigma^2-2r}{2\sigma^2},\\ \beta &= \frac{-1}{4}(\kappa+1)^2 = -\frac{\sigma^2-2r}{2\sigma^2}^2,\end{aligned} \tag{2.6}$$

obtemos a equação de difusão na variável dependente u e nas variáveis independentes x e t, dada por

$$\begin{aligned}&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, -\infty < x < \infty, 0 \le t \le \frac{\sigma^2}{2}T\\ &u(x,0) = e^{-\alpha x}v(x,0) = e^{-\alpha x}f(e^x), -\infty < x < \infty\end{aligned} \tag{2.7}$$

Se temos uma opção de compra européia, com preço de exercício E, então $f(x) = max(x-E,0)$. A solução da equação de difusão com condição inicial dada em 2.7 possui uma forma fechada, dada por

$$u(x,t) = \frac{1}{\sqrt{4\pi t}}\int_{-\infty}^{\infty} u(\xi,0)e^{-\frac{(x-\xi)^2}{4t}}d\xi. \tag{2.8}$$

Porém essa fórmula é válida apenas para precificação de opções de compra européias [5]. Em geral, é difícil obter uma solução analítica para equação de Black-Scholes aplicada a opções genéricas, por exemplo, no caso de opções americanas e asiáticas, bem como outros tipos de opções exóticas, cujos padrões de retornos apresentam configurações mais complexas. [6]. Nese caso as equações de Black-Scholes não possuem soluções analíticas e, portanto, precisam ser resolvidas numericamente.

Outro ponto a favor da obtenção de soluções numéricas é o fato de que a determinação da solução analítica envolve o cálculo de duas integrais impróprias para cada valor de S e t, o que é feito através da implementação de métodos numéricos custosos.

Feita a transformação da Equação de Black-Scholes em equação do calor, podemos implementar um método de resolução numérica.

No caso da equação de Black-Scholes em duas dimensões torna-se mais difícil encontrar uma solução analítica, e métodos numéricos devem ser empregados para se encontrar aproximações.

2.1 Black-Scholes 2D

O problema unidimensional da equação de Black-Scholes se tranforma em um problema multi-dimensional quando uma opção está atrelada a mais de um ativo adjacente.

No caso em que a opção está ligada a dois ativos a Equação de Black-Scholes é dada por

$$\begin{aligned}\frac{\partial V}{\partial t} + \sigma_1^2 S_1^2\frac{\partial^2 V}{\partial^2 S_1^2} + \sigma_2^2 S_2^2\frac{\partial^2 V}{\partial^2 S_2^2} + \rho\sigma_1\sigma_2 S_1 S_2\frac{\partial V}{\partial S_1 S_2} + rS_1\frac{\partial V}{\partial S_1} + rS_2\frac{\partial V}{\partial S_2} - rV = 0,\\ 0 \le S_1, 0 \le S_2, 0 \le t \le T,\end{aligned} \tag{2.9}$$

[5] en.wikipedia.org/wiki/Black?Scholes_equation
[6] http://en.wikipedia.org/wiki/Exotic_option

S_1 e S_2 representam os dois ativos adjacentes; σ_1 e σ_2 são as volatilidades dois dois ativos; ρ é a correlação entre os dois ativos.

A equação 2.9 também tem uma solução analítica para opções de compra européias. Ver, por exemplo, [eYK13]. Porém, como no caso unidimensional, para opções com data de expiração variável, opções exóticas etc, não é possível ou é muito custoso encontrar uma solução analítica e, portanto, métodos numéricos devem empregados.

Para facilitar o uso de métodos numéricos podemos simplificar a equação 2.9 de maneira a obter uma equação de difusão simples, em duas dimensões no espaço. Para os detalhes da transformação desta equação na forma padrão, ver [eCS98].

$$\frac{\partial u}{\partial t} = \alpha^2(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{2.10}$$

Dessa forma, recaímos novamente na Equação do Calor, porém agora em duas dimensões.

Neste trabalho também empregamos o Método de Euler implícito no tempo e diferenças centradas no espaço para resolver a Equação do calor 2D, e estudaremos dois métodos iterativos para resolver o sistema linear que surge da discretização por diferenças finitas desta equação, o Método de Jacobi e o Método de Gauss-Seidel. Apesar de o segundo ser mais eficiente computacionalmente, o Método de Jacobi, como veremos é altamente paralelizável em cada iteração e por isso podemos empregar técnicas simples de programação em paralelo para aumentar sua eficiência.

Capítulo 3

Método numérico

A equação do calor é uma equação diferencial parcial (EDP) parabólica nas variáveis espaço, x, e tempo, t - de segunda ordem, linear, e de coeficientes constantes. Esta equação modela a distribuição do calor em uma barra - ou numa placa, no caso bidimensional - ao longo de um intervalo de tempo.

Uma das formas de se obter uma solução numérica para a equação do calor é através da discretização da equação 2.7. A discretização no espaço é obtida tomando-se $n+1$ pontos $x_0, x_1, ..., x_n$ divididos em n intervalos de comprimento $h = \frac{x_n - x_0}{n}$. Os pontos da barra construídos dessa forma são definidos como $x_i = x_0 + ih, i = 0, 1, 2, ..., n$.

De forma idêntica discretizamos o tempo, obtendo m pontos $t_0, t_1, t_2, ..., t_m$ divididos em intervalos de tamanho fixo igual a $\Delta t = \frac{t_m - t_0}{m}$, e pontos $t_j = t_0 + j\Delta t, j = 0, 1, 2, ..., m$.

Queremos aproximar a solução da equação do calor, $u(t_i, x_i) \approx u_{ij}$, nos pontos (t_i, x_i) da malha.

Podemos usar métodos explícitos ou implícitos para obter a aproximação da solução exata. Ambos possuem vantagens e desvantagens. Uma das boas vantagens de se trabalhar com métodos implícitos é que ele é incondicionalmente estável - não há limite para o tamanho do passo. Uma desvantagem é que em geral esses métodos são mais lentos pois é necessário resolver um sistema linear.

Por ser incondicionalmente estável, e portanto, não haver restrição sobre o tamanho do passo na variável tempo, vamos adotar o método de Euler implícito para encontrar a aproximação para a equação do calor. Os métodos implícitos, em geral, produzem sistemas lineares que podem ser resolvidos por métodos diretos ou indiretos.

Com os intervalos de discretização Δt, no tempo, e h, no espaço, definidos anteriormente, a discretização da equação de difusão 2.6 no ponto (t_i, x_i), pelo método de Euller Implícito fornece

$$\begin{aligned} \frac{u_i^{k+1} - u_i^k}{\Delta t} &= \frac{1}{h^2}(u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}) \\ \Rightarrow u_i^{k+1} &= \frac{1}{(1+2\lambda)}u_i^k + \frac{\lambda}{(1+2\lambda)}(u_{i-1}^{k+1} + u_{i+1}^{k+1}), \end{aligned} \tag{3.1}$$

onde $\lambda = \frac{\Delta t}{h^2}$

A obtenção de solução única para a EDP implica o estabelecimento de condição de contorno e condição inicial que serão dadas respectivamente por

$$u(t, \alpha) = f_A(t), \quad u(\beta, t) = f_B(t), \quad u(0, x) = g(x). \tag{3.2}$$

A obtenção da aproximação u_{ij} no ponto (x_i, t_{j+1}) envolve as aproximações nos pontos (x_{i-1}, t_{j+1}), (x_{i+1}, t_{j+1}) e (x_i, t_j), como mostar a Figura 3.1.

Definidos discretização no tempo e espaço, método numérico, e condições inicial e de contorno, para cada instante t_k, $k = 0, 1, ..., T$, resolvemos o sistema linear $Au = b$, com $u^T = [u_1^{k+1}, u_2^{k+1}, ...u_{n-1}^{k+1}]$,

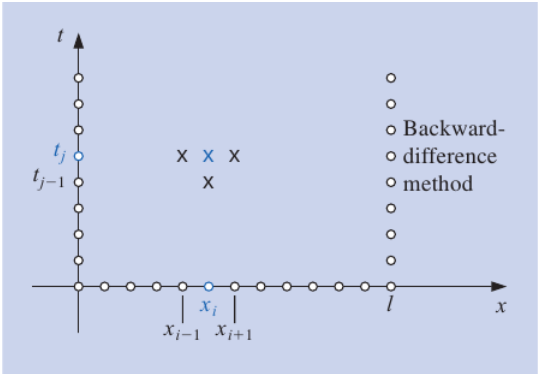


Figura 3.1: O Método de Euler Implícito para a equação do calor. Fonte: [eDF10]

$$A = \begin{bmatrix} 1+2\lambda & -\lambda & & & \\ -\lambda & 1+2\lambda & -\lambda & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda & 1+2\lambda & -\lambda \\ & & & -\lambda & 1+2\lambda \end{bmatrix} \quad , b = \begin{bmatrix} u_1^k + \lambda u_0^{k+1} \\ u_2^k \\ \vdots \\ u_{n-1}^k \\ u_{n-1}^k + \lambda u_n^{k+1} \end{bmatrix}$$

3.1 Erro de discretização local

Para obter o erro de discretização local partimos da equação de discretização 3.11 e substituímos as soluções aproximadas u_i^k pelas soluções reais $u(t,x)$, no que teremos

$$\frac{u(t+\Delta t,x)-u(t,x)}{\Delta t} \approx \frac{1}{h^2}(u(t+\Delta t,x-h)-2u(t+\Delta t,x)+u(t+\Delta t,x+h)) \tag{3.3}$$

Definimos o erro de discretização local como

$$\tau(t,x) = \frac{u(t+\Delta t,x)-u(t,x)}{\Delta t} - \frac{1}{h^2}(u(t+\Delta t,x-h)-2u(t+\Delta t,x)+u(t+\Delta t,x+h)) \tag{3.4}$$

Supondo $u \in C^4$, desenvolvemos as séries de Taylor dos termos $u(t+\Delta t,x)$, $u(t+\Delta t,x-h)$, $u(t+\Delta t,x+h)$. Usando o fato de que $u_{tt} = u_{txx} = u_{xxxx}$, obtemos

$$\begin{aligned} \tau(t,x) &= [u_t + \frac{\Delta t}{2!}u_{tt} + \frac{\Delta t^2}{3!}u_{ttt}...]+ \\ &- [u_{xx} + \frac{1}{12}u_{xxxx} + ...]+ \\ &- [\Delta t u_{txx} + \frac{\Delta t h^2}{12}u_{txxxx} + ...] + ... \\ &= (-\frac{1}{2}\Delta t - \frac{1}{12}h^2)u_{xxxx} + O(\Delta t^2 + \Delta t h^2 + h^4) \end{aligned} \tag{3.5}$$

Portanto, a discretização da Equação de Poisson em uma dimensão, pelo método de Euler implícito no tempo e por diferenças centradas no espaço, gera um erro de discretização local de primeira ordem na variável temporal e de segunda ordem na variável espacial, ou seja o método é de ordem $O(\Delta t + h^2)$.

3.2 Estabilidade do método e ordem de convergência

Um dos caminhos para estudarmos a estabilidade da equação do calor, é fazer a discretização primeiro no espaço para para cada instante de tempo. [1] Esse método é chamado Método de discretização por linhas (MOL na sigla em Inglês). Ele resulta em um sistema de ODE's com cada componente correspondendo à uma solução em determinado ponto da malha, como função do tempo. Obtemos assim, para cada $i = 1, 2, \ldots, n$,

$$u_i'(t) = \frac{1}{h^2}(u_{i-1}(t) - 2u_i(t) + u_{i+1}(t))$$

ou, em forma matricial,

$$\mathbf{u}'(t) = A\mathbf{u}(t) + \mathbf{g}(t),$$

com

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \quad , g(t) = \frac{1}{h^2}\begin{bmatrix} g_0(t) \\ 0 \\ \vdots \\ 0 \\ g_n(t) \end{bmatrix}$$

onde $g_0(t)$ e $g_n(t)$ são as condições de contorno da equação.

Com o problema discretizado dessa forma podemos utilizar a teoria de estabilidade de EDO aplicado ao sistema linear 3.2, usando-se os autovalores da matriz A, que dependem agora da discretização espacial.

Agora, ao aplicar o Método de Euler Implícito $\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta t f(\mathbf{u}^k)$ ao sistema 3.2, resulta na discretização completa - com diferenças centradas no espaço - definida em 3.11. Ou seja, fazemos a aproximação $u_i^k \approx u_i(t_k)$.

Se, para qualquer autovalor μ de A, $\Delta t\mu$ está na região de estabilidade do método de Euler implícito para a ODE, dizemos que o método é estável.

Os autovalores da matriz A são dados por

$$\mu_p = \frac{2}{h^2}\cos(p\pi h) - 1), \quad p = 1, 2, \ldots, n, \quad h = \frac{1}{n}$$

Como μ_p depende de h, o número de autovalores e seus módulos variam com o tamanho do passo na dimensão espacial. Porém, considerando valores fixos para Δt e h, podemos analisar a estabilidade absoluta no sentido da ODE. Desse modo, basta que $\Delta t\mu_p$ esteja na região de estabilidade do método de Euler implícito.

De 3.2, é fácil verificar que $-4/h^2 < \mu_p < 0$ e portanto, é necessário que $-4/h^2$, esteja na região de estabilidade.

Assim, usando o método de Euler implícito para obter a discretização 3.11, obtemos um método absolutamente estável, já que a região de estabilidade do método de Euler implícito para ODE's contém todo o semi-eixo real negativo. Assim, para qualquer escolha de $\Delta t > 0$, o método é estável, embora para Δt muito grande pode não ter acurácia. Em geral podemos tomar $\Delta t = O(h)$ para conseguir uma solução razoável.

Para estudar a convergência do método 3.11, vamos escrevê-lo na forma

$$\mathbf{u}^{k+1} = B\mathbf{u}^k + \mathbf{c}^k$$

com $B = A^{-1}$ [A da discretização do nosso método], e $c = A^{-1}b$, em uma malha com $\frac{1}{n}$ e $c^k \in \mathbb{R}^{n-1}$. Aplicamos a equação a diferenças à solução exata $u(x,t)$ para obter

[1] Esta seção é baseada em [LeV05]

$$\hat{\mathbf{u}}^{k+1} = B\hat{\mathbf{u}}^k + \mathbf{c}^k + \Delta t\boldsymbol{\tau}^k$$

onde

$$\hat{\mathbf{u}}^k = \begin{bmatrix} u(x_1,t_k) \\ u(x_2,t_k) \\ \vdots \\ u(x_{n-1},t_k) \end{bmatrix}, \quad \boldsymbol{\tau}^k = \begin{bmatrix} \tau(x_1,t_k) \\ \tau(x_2,t_k) \\ \vdots \\ \tau(x_{n-1},t_k) \end{bmatrix} \tag{3.6}$$

Subtraindo 3.2 de 3.2 obtemos a equação a diferenças do erro global, $E^k = u^k - \hat{u}^k$,

$$E^{k+1} = BE^k - \Delta t\boldsymbol{\tau}^k$$

e, assim,

$$E^k = B^kE^0 - \Delta t\sum_{n=1}^{k} B^{k-n+1}\boldsymbol{\tau}^{n-2}$$

$$\Rightarrow \|E^k\| \leq \|B^k\|\|E^0\| + \Delta t\sum_{n=1}^{k}\|B^{k-n+1}\|\|\boldsymbol{\tau}^{n-2}\|.$$

Se $\boldsymbol{\tau}^{n-2} \to 0$ em cada passo, ou seja, o método é consistente, e se, $\|B^k\|$ é uniformemente limitada para todo Δt e k tal que $k\Delta t \leq T$ e, portanto, estável, pelo Teorema de Equivalencia de Lax o método converge. [2]

3.3 Verificação da implementação do método de Euler implícito no tempo e diferenças centradas no espaço

Para a validação dos algoritimos implementados neste trabalho utilizaremos a técnica clássica de validação por solução manufaturada. [3]

Vamos apresentar os resultados para um problema de Cauchy de uma equação diferencial ordinária, com método de discretização explícito, porém o raciocínio é válido para problemas que envolvem EDP's.

O próximo teorema [Bul92], mostra que podemos escrever a solução numérica de um método de passo único aplicado a um problema de Cauchy como uma expansão em potências do tamanho do passo.

Teorema 3.3.1. *Seja o problema de Cauchy*

$$\begin{cases} \frac{d}{dt}y(t) &= f(t,y(t)) \\ y(t_0) &= y_0 \end{cases} \tag{3.7}$$

com solução numérica dada pelo método de passo único $\eta_{k+1} = \eta_k + \Delta t\Phi(t_k,\eta_k,\Delta t)$, e seja uma função $f(t,y)$, com $N+2$ derivadas parciais com relação a y, contínuas e limitadas no intevalo $(t,y); t_0 \leq t \leq T, y \in \Re$,

existe a expansão em potências de Δt da solução numérica $\eta(t,\Delta t)$, na forma

$$\eta(t,\Delta t) = y(t) + \Delta t^p e_p(t) + \Delta t^{p+1}e_{p+1}(t) + \cdots + \Delta t^N e_N(t) + \Delta t^{N+1}E_{N+1}(t,\Delta t), \tag{3.8}$$

com $e_j(t_0) = 0, j = p, p+1, \ldots,$ válida para todo $t \in [t_0,T]$ e todo $\Delta t = \frac{t-t_0}{n}$, $n = 1,2,\ldots$.

[2] Para uma apresentação detalhada do Teorema de Equivalência de Lax ver [eKWM67]

[3] Esta seção é baseada em notas de aula do Prof. Alexandre Roma para a disciplina MAP 5725, do IME-USP, de fevereiro de 2012

De 3.8, e subtraindo $y(t)$ dos dois lados da equação obtemos uma expressão do erro global de discretização no instante t como uma expansão sem série assintótica

$$-e(t,\Delta t) = \eta(t,\Delta t) - y(t) = \sum_{j=p}^{N} \Delta t^j e_j(t) + \Delta t^{N+1} E_{N+1}(t,\Delta t) \approx e_p(t)\Delta t^p \tag{3.9}$$

Para determinar a ordem p do método, partimos do fato de conhecermos o erro de discretização global, já que em 3.8 é dada a solução exata $y(t)$ do Problema de Cauchy. Dessa forma, determinamos as aproximações numéricas $\eta(t,\Delta t)$ e $\eta(t,\frac{\Delta t}{2})$ com passos Δt e $\frac{\Delta t}{2}$, respectivamente. Temos que

$$\begin{aligned} \left|\frac{\eta(t,\Delta t) - y(t)}{\eta(t,\frac{\Delta t}{2} - y(t)}\right| \approx \left|\frac{e_{p^*}(t)\Delta t^{p^*}}{e_{p^*}(t)(\frac{\Delta t}{2})^{p^*}}\right| = 2^{p^*} \\ \Rightarrow p^* \approx log_2\left(\frac{\eta(t,\Delta t) - y(t)}{\eta(t,\frac{\Delta t}{2} - y(t)}\right). \end{aligned} \tag{3.10}$$

Esse último resultado vale tanto para métodos explícitos como para métodos implícitos. Na notação que estamos usando, basta substituir as expressões da aproximação numérica e da solução exata por $u_ij(\Delta t)$ e $u(x,t,\Delta t)$, respectivamente.

Portanto, ao executar essa estratégia para tamanhos de passo no tempo sucessivamente menores, $\Delta t, \frac{\Delta t}{2}, \frac{\Delta t}{4}, \ldots$, obtemos uma sequência $p_1^*, p_2^*, p_3^*, \ldots$, que converge para a ordem do método em questão.

Em nossa discretização, esperamos obter ordem um para escolhas de Δt, tal que $\Delta t \approx h$ e ordem dois para $\Delta t \approx h^2$. Assim, dado que estamos escolhendo uma solução manufaturada e o método usado é incondicionalmente estável, temos a liberdadde e escolher Δt próximo de h^2 para que tenhamos máxima ordem de convergência possível.

No capítulo 6 mostramos as ordens de convergência obitdas para algumas escolhas de Δt usando a técnica descrita nesta seção.

3.4 Discretização da Equação do Calor 2D

A discretização por diferenças finitas da equação do calor 2D, $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x^2} + \frac{\partial u}{\partial y^2}$, usando Euler implícito é semelhante à discretização em uma dimensão. Acrescentamos a segunda dimensão no espaço e obteremos um sistema linear diagonalmente dominante, porém com uma matriz de coeficientes bloco tridiagonal.

$$\begin{aligned} \frac{u_{i,j}^{k+1} - u_{i,j}^k}{\Delta t} &= \frac{1}{\Delta x^2}(u_{i-1,j}^{k+1} - 2u_{i,j}^{k+1} + u_{i+1,j}^{k+1}) + \frac{1}{\Delta y^2}(u_{i,j-1}^{k+1} - 2u_{i,j}^{k+1} + u_{i,j+1}^{k+1}) \\ \Rightarrow u_{ij}^{k+1} &= \frac{1}{(1+2\lambda_1+\lambda_2)}u_{ij}^k + \frac{\lambda_1}{(1+2\lambda_1+2\lambda_2)}(u_{i-1,j}^{k+1} + u_{i+1,j}^{k+1}) + \frac{\lambda_2}{(1+2\lambda_1+2\lambda_2)}(u_{i,j-1}^{k+1} + u_{i,j+1}^{k+1}), \\ \Rightarrow u_{ij}^{k+1} &= \frac{1}{(\beta}u_{ij}^k + \frac{\lambda_1}{\beta}(u_{i-1,j}^{k+1} + u_{i+1,j}^{k+1}) + \frac{\lambda_2}{\beta}(u_{i,j-1}^{k+1} + u_{i,j+1}^{k+1}), \\ & i = 1,2,\ldots,n-1,\ j = 1,2,\ldots,m-1, \\ & \lambda_1 = \frac{\Delta t}{\Delta x^2},\ \lambda_2 = \frac{\Delta t}{\Delta y^2},\ \beta = 1 + 2\lambda_1 + 2\lambda_2. \end{aligned} \tag{3.11}$$

Podemos escrever este conjunto de equações na forma matricial $Au = b$ com
$u^T = [u_{11}^{k+1}, u_{21}^{k+1}, \ldots, u_{n-1,1}^{k+1}, u_{12}^{k+1}, u_{22}^{k+1}, \ldots, u_{n-1,2}^{k+1}, \ldots, u_{1,m-1}^{k+1}, u_{2,m-2}^{k+1}, \ldots, u_{n-1,m-1}^{k+1}]$, e

$$A = \begin{bmatrix} \boldsymbol{B_1} & \boldsymbol{B_2} & & & \\ \boldsymbol{B_2} & \boldsymbol{B_1} & \boldsymbol{B_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{B_2} & \boldsymbol{B_1} & \boldsymbol{B_2} \\ & & & \boldsymbol{B_2} & \boldsymbol{B_1} \end{bmatrix} \quad , b = \begin{bmatrix} \boldsymbol{r_1} \\ \boldsymbol{r_2} \\ \vdots \\ \boldsymbol{r_{n-2}} \\ \boldsymbol{r_{n-1},} \end{bmatrix}$$

onde

$$\boldsymbol{B_1} = \begin{bmatrix} \beta & -\lambda_1 & & & \\ -\lambda_1 & \beta & -\lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda_1 & \beta & -\lambda_1 \\ & & & -\lambda_1 & \beta \end{bmatrix} \quad , \boldsymbol{B_2} = \begin{bmatrix} -\lambda_2 & & \\ & \ddots & \\ & & -\lambda_2 \end{bmatrix}$$

e

$$\boldsymbol{r_1} = \begin{bmatrix} u_{11}^k + \lambda_1 u_{01}^{k+1} + \lambda_2^{k+1} \\ u_{21}^k + \lambda_2 u_{20}^{k+1} \\ \vdots \\ u_{n-1,1}^k + \lambda_1 u_{n1}^{k+1} + \lambda_2 u_{n-1,0}^{k+1} \end{bmatrix}, \boldsymbol{r_2} = \begin{bmatrix} u_{12}^k + \lambda_1 u_{02}^{k+1} \\ u_{22}^k \\ \vdots \\ u_{n-1,2}^k + \lambda_1 u_{n2}^{k+1} \end{bmatrix},$$

$$\boldsymbol{r_3} = \begin{bmatrix} u_{1,m-2}^k + \lambda_1 u_{0,m-2}^{k+1} \\ u_{2,m-2}^k \\ \vdots \\ u_{n-1,m-2}^k + \lambda_1 u_{n,m-2}^{k+1} \end{bmatrix}, \boldsymbol{r_4} = \begin{bmatrix} u_{1,m-1}^k + \lambda_1 u_{0,m-1}^{k+1} + \lambda_2 u_{1m}^{k+1} \\ u_{2,m-2}^k + \lambda_2 u_{2m}^{k+1} \\ \vdots \\ u_{n-1,m-1}^k + \lambda_1 u_{n,m-1}^{k+1} + \lambda_2 u_{n-1,m}^{k+1} \end{bmatrix}.$$

O erro de discretização local do Método de Euler implícito aplicado à equação do calor 2D acima é obtido de forma semelhante à 3.5, apenas acrescentando a segunda variável espacial y. Assim, também em duas dimensões obtemos ordem de consistência $O(\Delta t + \Delta x^2 + \Delta y^2)$.

Também pode-se mostrar que o Método de Euler é incondicionalmente estável e convergente, com ordem de convergência igual à ordem de consistência.

Resolveremos o sistema linear $A\mathbf{u} = \mathbf{b}$ aplicando os métodos de Jacobi e de Gauss-Seidel. Vamos compar o Método de Jacobi sequencial com sua implementação em paralelo.

Capítulo 4

Métodos de resolução de sistemas lineares

A resolução do sistema linear $Au = b$ pode ser obtida através de métodos diretos e indiretos. A depender da matriz A envolvida na resolução do sistema, uma das duas classes de método se adequará melhor à tarefa.

Aqui resolvemos o sistema que surge do Método de Euler implícito aplicado à equação do calor em uma dimensão, por um método direto - Fatorização de Crout - e por dois métodos indiretos - Jacobi e Gauss-Seidel.

Métodos indiretos de resolução do problema são mais facilmente paralelizáveis. Em particular, como veremos mais adiante, o Método de Jacobi possui alto grau de paralelização. Por isso utilizamos esse método para a equação de Black-Scholes tanto em uma como em duas dimensões.

4.1 Um método eficiente para resolução de sistemas tridiagonais

Métodos diretos de resolução de sistemas lineares têm a vantagem de fornecerem a solução exata a menos de problemas de arredondamento na aritmética envolvida. Os vários métodos existentes em geral se utilizam de alguma variação da elinação gaussiana e de alguma forma de fatorização da matriz A. Esta última aumenta consideravelmente a eficiência do algoritimo se é uma fatorização LU. [1]

Definição 4.1.1. *Uma matriz $n \times n$ é chamada **matriz** banda se existem inteiros p e q, $1 < p$, $q < n$, tais que $a_{ij} = 0$ se $p \leq i - j$ ou $q \leq i - j$. O **tamanho da banda** de uma matriz banda é definido como $w = p + q - 1$.*

No nosso sistema linear oriundo da discretização implícita adotada, estamos lidando com uma matriz de coeficientes que é tridiagonal. Matrizes tridiagonais ocorrem quando $p = q = 2$, ou seja, o tamanho da banda é igual a 3 - diagonal principal, diagonal acima e diagonal abaixo são diferentes de 0.

Um algoritimo eficiente para resolver sistemas lineares cuja matriz é tridiagonal e definida positiva utiliza a Decomposição de Crout, um tipo especial de decomposição LU.

Dadas as matrizes

$$L = \begin{bmatrix} l_{11} & 0 & \dots & \dots & 0 \\ l_{21} & l_{22} & \ddots & & \vdots \\ 0 & \ddots & \ddots & & \\ \vdots & \ddots & & & 0 \\ 0 & \dots & 0 & l_{n,n-1} & l_{nn} \end{bmatrix} , U = \begin{bmatrix} u_{11} & 0 & \dots & \dots & 0 \\ u_{21} & u_{22} & \ddots & & \vdots \\ 0 & \ddots & \ddots & & \\ \vdots & \ddots & & & 0 \\ 0 & \dots & 0 & u_{n,n-1} & u_{nn} \end{bmatrix}$$

[1] Esta seção e as próximas duas são basedadas em [eDF10].

As entradas da matriz A considerando as matrizes L e U acima, com exceção das entradas nulas são dadas por

$$\begin{aligned} a_{11} &= l_{11}; \\ a_{i,i-1} &= l_{i,i-1}, \quad i = 2,3,\ldots,n; \\ a_{ii} &= l_{i,i-1}u_{i-1,i} + l_{ii}, \quad i = 2,3,\ldots,n; \\ a_{i,i+1} &= l_{ii}u_{i,i+1}, \quad i = 1,2,\ldots,n-1. \end{aligned} \tag{4.1}$$

Este sistema de equações é resolvido aplicando-se um algoritimo que usa a Fatorização de Crout. O algoritimo envolve $(5n-4)$ multiplicações/ divisões e $(3n-3)$ adições/subtrações, o que representa uma vantagem significativa para métodos de fatorização LU que não levam em conta o fato da matriz A ser tridiagonal.

4.2 Os método de Jacobi e Gauss-Seidel

Os métodos iterativos são usados na prática quando precisamos resolver sistemas lineares cuja ordem da matriz de coeficientes é grande. Essa necessidade surge quando precisamos obter aproximações melhores da solução exata via um método numérico dado. A equação do calor modela a distribuição de calor em uma barra, e os pontos em que estamos calculando aproximações à solução dessa equação formam uma malha que, à medida em que aumentamos esse número de pontos, e a malha fica mais fina, a ordem da matriz de coeficientes aumenta. Para uma determinada ordem desta matriz, os algoritmos diretos passam a ser menos eficientes que os indiretos.

Existem muitos métodos iterativos eficientes para resolução de sistemas lineares. Exemplos são o algoritmo de Gauss-Seidel, SOR, Gradientes Conjugados etc, porém, especificamente para o caso de uma matriz tridiagonal, os métodos de Jacobi e Gauss-Seidel são bastante apropriados, sendo que na prática Gauss-Seidel - um aprimoramento de Jacobi - é o mais utilizado.

Como neste trabalho iremos empregar técnicas de paralelização de algoritimos numéricos, o algoritimo de Jacobi será utilizado pois entre os algoritimos iterativos, é o que apresenta maior facilidade de paralelização. Implementamos o Método de Gauss-Seidel para fazer um comparativo entre os três métodos - Jacobi, Gauss-Seidel e Fatorização de Crout - na resolução sequencial.

Em geral os métodos iterativos de solução de Sistemas Lineares transformam o sistema $A\mathbf{x} = \mathbf{b}$ em $\mathbf{x} = T\mathbf{x} + \mathbf{c}$. O processo inicia com um "chute" inicial $\mathbf{x}^{(0)}$ e calcula-se uma sequência de aproximações $\mathbf{x}^{(k)}$, dadas por

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}, \quad k = 1,2,... \tag{4.2}$$

O Método de Jacobi é obtido parcelando a matriz $A = [a_{i,j}], j = 1,2,...,n$ em três matrizes na forma $A = D - L - U$, onde D, L, U são dadas por

$$A = \begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & a_{n,n} \end{bmatrix} - \begin{bmatrix} 0 & \ldots & \ldots & 0 \\ -a_{21} & \ddots & & \vdots \\ \vdots & \ddots & & \\ -a_{n1} & \ldots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \ldots & -a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ & & & -a_{n-1,n} \\ 0 & \ldots & \ldots & 0 \end{bmatrix}$$

De $A\mathbf{x} = \mathbf{b} \Leftrightarrow (D - L - U)\mathbf{x} = \mathbf{b}$, obtemos

$$D\mathbf{x} = (L+U)\mathbf{x} + \mathbf{b} \Rightarrow \mathbf{x} \quad = D^{-1}(L+U)\mathbf{x} + D^{-1}\mathbf{b}, \quad seexisteD^{-1}. \tag{4.3}$$

No método de Euler implícito, onde obtemos os elementos das diagonais na forma $(1+2\lambda)$, é garantida a existência de D^{-1}.

Portanto, o método de Jacobi em forma matricial é dado porém

$$\mathbf{x}^{(k)} = D^{-1}(L+U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b} \tag{4.4}$$

que está na forma 4.2, com $T = D^{-1}(L+U)$ e $\mathbf{c} = D^{-1}\mathbf{b}$.

Esta forma matricial do Método de Jacobi é usada para fins teóricos. Em particular, há um Teorema que postula a convergência da sequência $\mathbf{x}^{(k)}{}_{k=0}^{\infty}$ do Método de Jacobi e de Gauss-Seidel para a solução única do sistema $A\mathbf{x} = \mathbf{b}$, para qualquer escolha de $\mathbf{x}^{(0)}$. [2]

Na prática a resolução computacional do método de Jacobi se faz através da iteração

$$x_i^k = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1\\ j\neq i}}^{n}(-a_{ij}x_j^{k-1}) + b_i\right], \quad i = 1, 2, \ldots, n. \tag{4.5}$$

O Método de Gauss-Seidel constitui um aprimoramento do Método de Jacobi. Parte do fato de que em 4.5 os pontos $x_1^k, \ldots, x_{i-1}^k$ para $i > 1$, já foram calculados e, portanto, devem ser melhores aproximações em relação à iteração anterior. Desse modo podemos usá-los para calcular a componente x_i^k. Assim, a forma computacional do Método de Gauss-Seidel é dada portanto

$$x_i^k = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}(a_{ij}x_j^k) - \sum_{j=i+1}^{n}(a_{ij}x_j^{k-1}) + b_i\right], \quad i = 1, 2, \ldots, n. \tag{4.6}$$

Para escrever a forma matricial, partimos de 4.6, e multiplicamos ambos os lados das equações por a_{ii} para cada $i = 1, 2, \ldots, n$. Deixando os termos da iteração k do lado esquerdo e os termos da iteração $k-1$ do lado direito, obtemos o sistema de equações

$$\begin{aligned} a_{11}x_1^k &= -a_{12}x_2^{k-1} - a_{13}x_3^{k-1} + \cdots - a_{1n}x_n^{k-1} + b_1, \\ a_{21}x_1^k + a_{22}x_2^k &= -a_{23}x_3^{k-1} + \cdots - a_{2n}x_n^{k-1} + b_2, \\ &\vdots \\ a_{n1}x_1^k + a_{n2}x_2^k + \cdots + a_{nn}x_n^k &= b_n. \end{aligned} \tag{4.7}$$

Tomando D, L e U, definidas em 4.2, escrevemos

$$\begin{aligned} (D-L)\mathbf{x}^{(k)} &= U\mathbf{x}^{(k-1)} + \mathbf{b} \\ \Rightarrow \mathbf{x}^{(k)} &= (D-L)^{-1}U\mathbf{x}^{(k-1)} + (D-L)^{-1}\mathbf{b}, \quad k = 1, 2, \ldots \\ &\text{ou} \\ \mathbf{x}^{(k)} &= T_g\mathbf{x}^{(k-1)} + \mathbf{c}_g. \end{aligned} \tag{4.8}$$

com $T_g = (D-L)^{-1}$ e $\mathbf{c}_g = (D-L)^{-1}\mathbf{b}$.

Esta forma é válida uma vez que na discretização por nós adotada gera um sistema linear com as entradas a_{ii}, $i = 1, 2, \ldots, n$, não nulas, e portanto $(D-L)$ é não singular.

4.3 Convergência dos métodos de Jacobi e Gauss-Seidel

Os métodos iterativos devem definir algum critério de parada para a aproximação que geram em relação à solução real do sistema linear. Em geral calcula-se alguma norma da diferença entre o vetor $\mathbf{x}^{(k+1)}$ e o vetor $\mathbf{x}^{(k)}$ - entre duas iterações consecutivas - em termos absolutos ou relativos. Ou seja calculamos, para cada iteração, $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty$ ou, $\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty}{\|\mathbf{x}^{(k+1)}\|_\infty}$, e comparamos com um valor pré-estabelecido para essa diferença tão menor quanto mais desejamos que a solução aproximada pelo método de discretização esteja próxima da solução exata.

[2] Ver [eDF10], p. 459

Em nossos algoritmos usaremos a norma infinito absoluta da diferença entre dois vetores aproximação de iterações consecutivas. E como queremos alcançar uma aproximação da solução exata tão próxima quanto possível, estabelecemos como tolerância entre duas iterações consecutivas um valor da ordem próxima do erro de máquina para o tipo *double* da Linguagem C, 10^{-14}.

A análise de convergência de métodos iterativos usa a definição matricial genérica desses métodos dada em 4.2. Os resultados que apresentamos aqui fazem uso de dois conceitos importantes. Raio espectral e norma de uma matriz quadrada.

Definição 4.3.1. *O raio espectral de uma matriz A é definido como*

$$\rho(A) = \max|\lambda|, \quad \textit{onde } \lambda \textit{ é um autovalor da matriz } A. \tag{4.9}$$

Definição 4.3.2. *Seja o conjunto de todas as matrizes $n \times n$. A norma de uma matriz é uma função $\|.\| : \mathbb{R}^{n\times n} \to \mathbb{R}$, definida neste conjunto e que satisfaz, para todas as matrizes $n \times n$ A, B, e todo $\alpha \in \mathbb{R}$:*

- $\|A\| \geq 0$;
- $\|A\| = 0, \quad$ *se e somente* $\quad A = 0$;
- $\|\alpha A\| = |\alpha|\|A\|$;
- $\|A + B\| \leq \|A\| + \|B\|$;
- $\|AB\| \leq \|A\|\|B\|$.

Para calcular a norma de uma matriz usamos o seguinte resultado, que permite calcular a norma de uma matriz a partir da norma do vetor $\|A\mathbf{x}\|$, e que chamamos de norma induzida (ou natural) da matriz.

Teorema 4.3.3. *Seja a norma vetorial $\|.\|$ em $\mathbb{R}$. A norma induzida da matriz A é dada por*

$$\|A\| = \max_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\| \tag{4.10}$$

Os próximos dois resultados são os subsídios principais para a definição de convergência de um método iterativo.

Teorema 4.3.4. *As seguintes afirmações são equivalentes.*

1. *A é uma matriz convergente*
2. $\lim_{n\to\infty} \|A^n\| = 0, \quad$ *para alguma norma natural.*
3. $\lim_{n\to\infty} \|A^n\| = 0, \quad$ *para todas as normas naturais.*
4. $\rho(A) < 1$.
5. $\lim_{n\to\infty} A^n\boldsymbol{x} = 0, \quad$ *para todo x.*

Ver [eHBK66], para a prova desse teorema.

Lema 4.3.5. *Se o raio espectral da matriz T satisfaz $\rho(T) < 1$, então $(I - T)^{-1}$ existe, e*

$$(I - T)^{-1} = I + T + T^2 + \cdots = \sum_{j=0}^{\infty} T^j. \tag{4.11}$$

O próximo resultado define a condição para que um método iterativo seja convergente.

Teorema 4.3.6. *Para qualquer $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, a sequencia $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ definida por*

$$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}, \quad k = 1, 2, ... \tag{4.12}$$

converge para a única solução $x = Tx + c$ se, e somente se, $\rho(T) < 1$.

Demonstração. Primeiro, suponha que $\rho(1) < 1$. Então,

$$\begin{aligned} \mathbf{x}^{(k)} &= T\mathbf{x}^{(k-1)} + \mathbf{c} \\ &= T(T\mathbf{x}^{(k-2)} + \mathbf{c}) + \mathbf{c} \\ &= T^2\mathbf{x}^{(k-2)} + (T + I)\mathbf{c} \\ &\vdots \\ &= T^k\mathbf{x}^{(0)} + (T^{k-1} + \cdots + T + I)\mathbf{c}. \end{aligned} \tag{4.13}$$

Como $\rho(T) < 1$, pelo Teorema 4.3.4, T é convergente, e

$$\lim_{k\to\infty} T^k\mathbf{x}^{(0)} = 0. \tag{4.14}$$

Pelo Lema 4.3.5,

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \lim_{k\to\infty} T^k\mathbf{x}^{(0)} + (\sum_{j=0}^{\infty} T^j)\mathbf{c} = \mathbf{0} + (I - T)^{-1}\mathbf{c} = (I - T)^{-1}\mathbf{c}.$$

Como $\mathbf{x} = T\mathbf{x} + \mathbf{c} \Rightarrow \mathbf{x} = (I - T)^{-1}\mathbf{c}$, a sequencia $\{\mathbf{x}^{(k)}\}$ converge para o vetor solução $\mathbf{x}$.

Para provar a volta, considere $z \in \mathbb{R}^n$, e $\mathbf{x}$ a solução única de $\mathbf{x} = T\mathbf{x} + \mathbf{c}$. Defina $\mathbf{x}^{(0)} = \mathbf{x} - \mathbf{z}$, e $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}, k \geq 1$. Então $\{\mathbf{x}^{(k)}\}$ converge para $\mathbf{x}$. Temos também que,

$$\begin{aligned} \mathbf{x} - \mathbf{x}^{(k)} &= (T\mathbf{x} + \mathbf{c}) - (T\mathbf{x}^{(k-1)} + \mathbf{c}) \\ &= T(\mathbf{x} - \mathbf{x}^{(k-1)}) \\ &= T^2(\mathbf{x} - \mathbf{x}^{(k-2)}) \\ &\vdots \\ &= T^k(\mathbf{x} - \mathbf{x}^{(0)}) \\ &= T^k\mathbf{z}. \end{aligned}$$

Assim, $\lim_{k\to\infty} T^k(\mathbf{x} - \mathbf{x}^{(0)}) = \lim_{k\to\infty} T^k(\mathbf{x} - \mathbf{x}^{(k)}) = \mathbf{0}$.

Como $\mathbf{z}$ foi escolhido arbitrariamente, pelo Teorema 4.3.4, T é convergente e $\rho(T) < 1$. □

Para os métodos de Jacobi e Gauss-Seidel, definidos por $\mathbf{x}^{(k)} = T_j\mathbf{x}^{(k-1)} + \mathbf{c}_j$ e $\mathbf{x}^{(k)} = T_g\mathbf{x}^{(k-1)} + \mathbf{c}_g$, respectivamente, onde $T_j = D^{-1}(L+U)$ e $T_g = (D-L)^{-1}U$, se $\rho(T_j) < 1$ e $\rho(T_g) < 1$ os métodos convergem para a solução $\mathbf{x}$, com $A\mathbf{x} = \mathbf{b}, A = (D - L - U)$.

Isso é garantido pelo próximo resultado.

Teorema 4.3.7. *Se A é estritamente diagonal dominante, então para qualquer escolha de $\boldsymbol{x}^{(0)}$, os métodos de Jacobi e Gauss-Seidel geram sequencias $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$, que convergem para a solução única de $A\boldsymbol{x} = \boldsymbol{b}$.* [3]

A relação entre a velocidade de convergência e o raio espectral para métodos iterativos é determinada a partir do Corolário 1 e do Teorema 4.

[3] Burden, p. 459.

Corolário 4.3.8. *Se $\|T\| < 1$ para toda norma natural de matriz e $\boldsymbol{c}$ é um dado vetor, então a sequencia $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ definida por $\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c}$ converge, para todo $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, para um vetor $\boldsymbol{x} \in \mathbb{R}^n$, com $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$, e os erros estão limitados e dados por:*

1. $\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \leq \|T\|^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}\|$;
2. $\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \leq \frac{\|T\|^k}{1-\|T\|} \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|$.

Teorema 4.3.9. *Para toda matriz A e todo $\epsilon > 0$, existe uma norma natural $\|.\|$, tal que $\rho(A) < \|A\| < \rho(A) + \epsilon$. Portanto, $\rho(A)$ é o ínfimo para as normas naturais sobre A.*

Ver [Ort72] para a prova deste teorema
Com estes dois resultados temos que

$$\|\mathbf{x}^{(k)} - \mathbf{x}\| \approx \rho(T)^k \|\mathbf{x}^{(0)} - \mathbf{x}\|.$$

Portanto, quanto menor $\rho(T) < 1$ mais rápida a convergência do método.

O Teorema 5 nos diz que, se ambos os métodos, de Jacobi e Gauss-Seidel, convergem, o Método de Gauss-Seidel converge mais rapidamente. A prova pode ser encontrada em [You71].

Teorema 4.3.10. *Se $a_{ij} \leq 0$, para todo $i \neq j$ e $a_{ij} > 0, i = 1, 2, \ldots, n$, então uma e apenas uma das afirmações abaixo é verdadeira:*

1. $0 \leq \rho(T_g) < \rho(T_j) < 1$;
2. $1 < \rho(T_j) < \rho(T_g)$;
3. $\rho(T_j) = \rho(T_g) = 0$;
4. $\rho(T_j) = \rho(T_g) = 1$.

4.4 Velocidade de convergência dos métodos de Jacobi e Gauss-Seidel

Para os métodos de Jacobi e Gauss-Seidel aplicados à resolução do sistema linear oriundo da discretização da equação do calor em uma dimensão pelo Método de Euler Implícito no tempo e por diferenças centradas no espaço, é possível determinar os valores de $\rho(T_j)$ e $\rho(T_g)$ considerando-se que, para cada instante de tempo, podemos considerar que estamos resolvendo uma ODE $u'' = f$. Nesse caso temos que [4]

$$\rho(T_j) = \cos(\pi h) \approx 1 - \frac{1}{2}\pi^2 h^2 + O(h^4) < 1, \text{ e}$$
$$\rho(T_g) \approx 1 - \pi^2 h^2 + O(h^4) < 1.$$

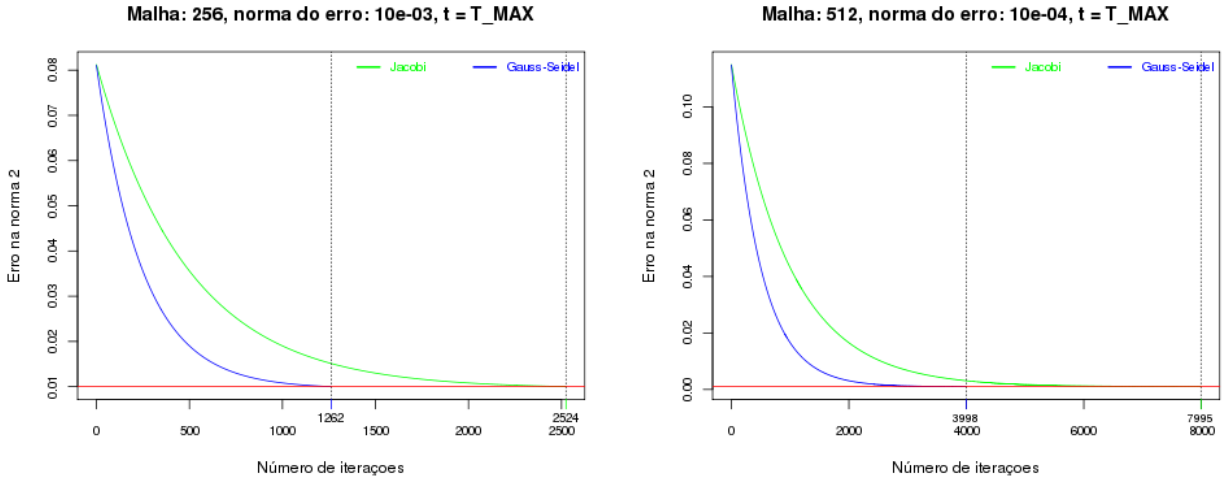
Como $\rho(T_j)$ e $\rho(T_g)$ tendem a 1 quando h tende a 0, quando mais fina a malha de pontos na barra, mais a solução aproximada demora a convergir para uma dada tolerância do erro.

No entanto o método de Gauss-Seidel é melhor que o de Jacobi por um fator de 2 e, assim, tipicamente, o número de iterações necessário para alcançar uma tolerância dada para Gauss-Seidel é metade do necessário para Jacobi.

Pode-se mostrar que essa relação é mantida no caso da análise de convergência para os dois métodos em questão aplicados ao problema bi-dimensional.

Na seção 6.1, onde apresentamos os resultados obtidos usando solução manufaturada, verificamos empiricamente a relação entre número de iterações e a escolha do Método de Jacobi ou Gauss-Seidel. Na figura 4.1 mostramos a relação obtida para as malhas de 256 e 512 pontos. Para esses gráficos usamos $\Delta t = h$, onde Δt é o passo no tempo e h é o passo no espaço.

[4] [LeV05], pp.65, 66.



(a) **(b)**

Figura 4.1: Número de iterações necessário para obter determinado erro

Capítulo 5

OpenMP e a paralelização do Método de Jacobi

OpenMp é uma inteface de programação de aplicações (API na sigla em inglês) que utilizam a capacidade de processamento em paralelo com memória compartilhada existentes em muitas configurações de computadores, e que são atualmente cada vez mais acessíveis ao usuário comum - os computadores com processadores *duo* e *quad cores* invadiram a nova geração de máquinas vendidas no varejo.

Essa tecnologia foi lançada em outubro de 1997, com o nome OpenMP for Fortran 1.0, Em 1998 a API foi extendida para as linguagens C/C++. Foi desenvolvida por consórcio envolvendo Compaq / Digital, Hewlett-Packard Company, Intel Corporation, Sun Microsystems Inc. e o Departamento de Energia dos EUA. [1]

A API é composta de três componentes principais: diretivas para o compilador, bibliotecas de rotinas em tempo de execução, variáveis de ambiente. É portável, sendo implementada para os sistemas operacionais da série Microsoft Windows, e a maioria das plataformas Unix e Unix-like (GNU/Linux, FreeBSD, Apple OS X etc.); e padronizada. [2]

Essas características colocam o padrão OpenMP como o mais amplamente usado quando se trata de paralelização em ambiente de memória compartilhada.

5.1 Conceitos de programação em paralelo e implementação OpenMP

Vamos citar brevemente de alguns conceitos de programação paralela para entender como se dá a paralelização de processos através do OpenMP. [3]

- **Programa sequencial**: especificação de instruções executadas sequencialmente
- **Contexto**: conjunto de registradores e seus valores, pilha de execução, tabelas de administração de memória e de tratamento de sinais etc.
- **Processo**: execução de um programa sequencial.
- **Programação concorrente**: especifiação de diversos programas sequenciais que executam em paralelo (concorrentemente).
- **Multiprocessamento**: existem vários processos compartilhando memória.
- **Processos leves ou *threads***: linhas de execução concorrentes que compartilham o mesmo processo, isto é, são subprocessos. Podemos entender *threads* como funções ou procedimentos dentro de um mesmo programa executando concorrentemente e compartilhando o contexto.

[1] www2.latech.edu/ box/hapc/openmp.pdf
[2] www2.latech.edu/ box/hapc/openmp.pdf
[3] Estes conceitos foram retirados de Gubi, apostila.pdf

O conceito de *threads* é ilustrado na Figura 5.1. [4]

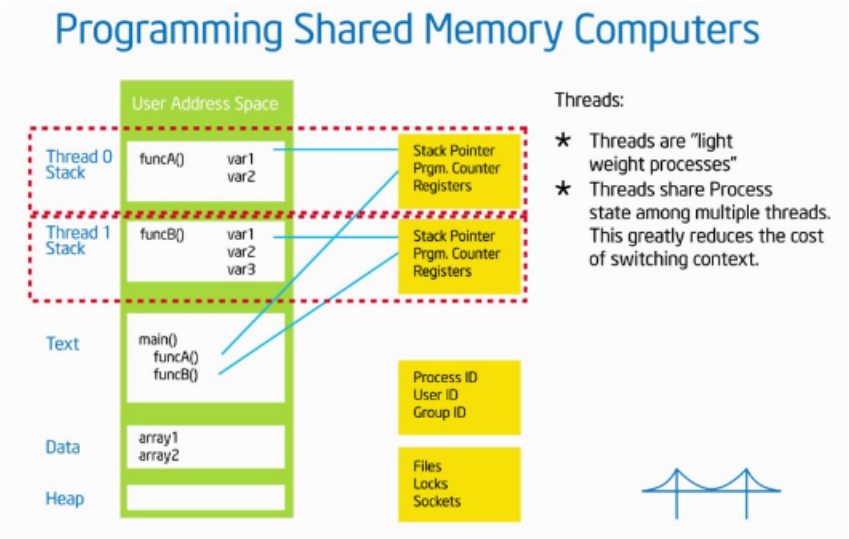


Figura 5.1: *thread* 0 aponta para *Thread 0 Stack*, *thread* 1 aponta para *Thread 1 Stack*. As *threads* 0 e 1 têm suas pilhas de memória privadas, porém compartilham as outras partes da memória: *Text*, *Data* e *Heap*. Fonte: tutorial de treinamento disponibiliado pelo site do projeto *www.openmp.org*

Uma instâcia de um programa é constituída de um processo e várias *threads* que interagem através da escrita/leitura na região de memória compartilhada. O sistema operacional decide quando rodar cada *thread* e também sincroniza a operação das *threads* para garantir a obtenção de resultados corretos. A ordem em que cada *thread* é executada não é determinada a priori, podendo mudar a cada nova execução do programa.

O modelo OpenMP gerencia a utilização de múltiplas threads, sob uma região compartilhada de memória, dentro de um mesmo processo, através de diretivas passadas ao compilador. Essas threads se comunicam através de variáveis compartilhadas.

A forma como essas threads são criadas pelo OpenMP para rodar um bloco de um processo em paralelo e como após a execucação desse bloco o programa volta a ser executado sequencialmente é conhecido como Modelo *Fork/Join*. Este é o modelo fundamental de criação de *threads* do OpenMP.

Em termos básicos, ao iniciar a execução de um programa, é criada uma *thread*, designada como *thread* mestre. Em algum ponto do programa pode ser desejável (e possível) criar *threads* adicionais. Neste ponto ocorre uma bifurcação (*fork*), em que um bloco de programa pode ser processado independentemente por duas ou mais *threads*. Cada *thread* recebe um identificador único, iniciando com *id 0*, para a *thread* mestre, *id 1*, *id 2* etc, para as demais. Ao finalizar o processamento em paralelo daquele bloco, o programa junta (*join*) as *threads* novamente em uma só, voltando a execucação novamente para a *thread* mestre. Isso pode ocorrer diversas vezes ao longo da execução do programa, e mesmo após iniciar a execução de um bloco em paralelo, com diversas *threads* trabalhando ao mesmo tempo, pode-se bifurcar uma ou mais delas também em novas *threads*. A Figura 5.2 ilustra o modelo *fork/join*.

Em um programa em Linguagem C, para definir o número de *threads* a ser usado em um bloco de programa e informar ao compilador o bloco de execução em paralelo utiliza-se respectivamente as instruções `omp_set_num_threads(n)` e `#pragma omp parallel`. Um exemplo simples ilustra a execução de uma função em paralelo em um programa em C.

```
1  double A[1000];
2  omp_set_num_threads(n);
3  #pragma omp parallel
4  {
5      ind ID = omp_get_thread_num();
6      pooh(ID, A);
```

[4] Esta seção é baseada no tutorial de treinamento em OpenMP, disponibilizado no site do projeto *www.openmp.org*

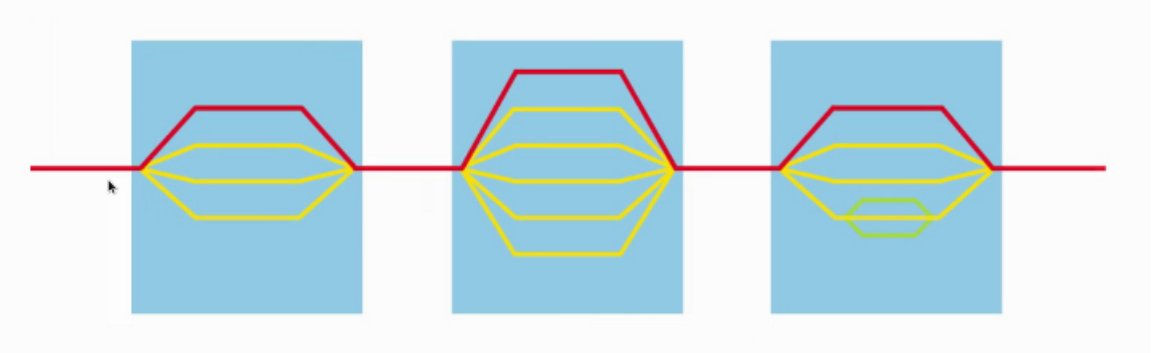


Figura 5.2: Representação visual do Modelo fork/join. Fonte: tutorial de treinamento disponibiliado pelo site do projeto *www.openmp.org*

```
7  }
8  printf("all done\n");
```

Neste código, `omp_set_num_threads(n)` solicita a criação de `n` *threads* e a diretiva `#pragma omp parallel` define o código que será executado em paralelo entre `{` e `}`. Veja a Figura 5.3.

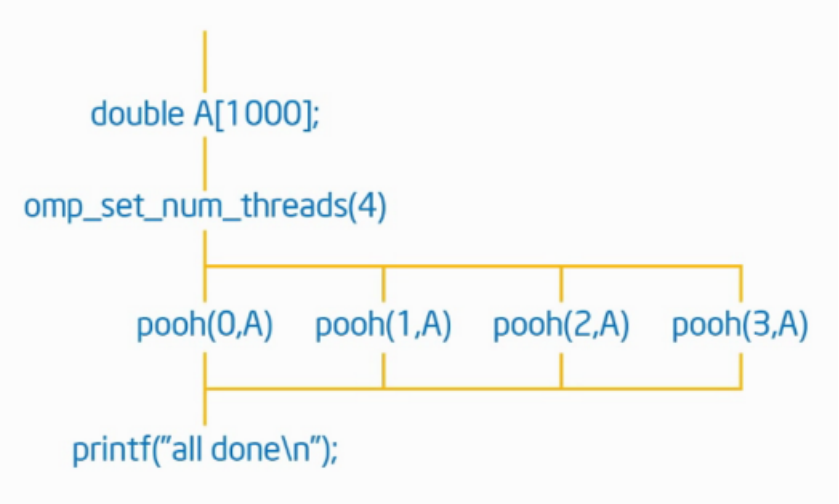


Figura 5.3: Representação visual do Modelo fork/join. Fonte: tutorial de treinamento disponibiliado pelo site do projeto *www.openmp.org*

Na versão paralela do programa que aproxima a solução da equação do calor em 1D e 2D, vamos usar uma construção baseada no paradigma *worksharing*. No paradigma SPMD, todas as *threads* rodam o mesmo código de forma redundante. Porém, como é nosso caso, muitas vezes estamos interessados em dividir partes de um bloco de código de modo que cada uma seja processada independentemente por cada *thread*. Este modo de paralelismo é conhecido como **worksharing**, e o subconjunto desse paradigma que utilizaremos em nosso programa é o *loop construct*.

A diretiva OpenMP para dividir a execução de um laço `for` em C é `#omp parallel for`. Esse objetivo é atingido simplesmente inserindo essa diretiva imediatamente antes de um laço `for`. Dessa forma a paralelização do algoritimo de Jacobi para resolver o sistema linear se dará na barra, no caso 1D, e em cada reta horizontal contendo os pontos discretizados no caso 2D

Capítulo 6

Resultados

6.1 Equação do Calor 1D

O sistema tridiagonal oriundo da discretização adotada favorece a implementação de algoritimos diretos de resolução do sistema linear obtido. As implementações da resolução por métodos indiretos mostrou-se não frutífera dada a alta ineficiência.

Isso ocorre porque para se obter valores razoáveis para a norma do erro no caso dos métodos de Jacobi e Gauss-Seidel, o número de iterações cresce muito rapidamente conforme a malha de pontos aumenta.

Para os resultados a seguir foi utilizado um problema com solução analítica conhecida, dada por

$$u(x,t) = exp(-\pi^2 t)\sin(\pi x) \tag{6.1}$$

no domínio $[0,1] \times [0,0.5]$. Dessa forma o problema que estamos resolvendo é dado por

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2}{\partial x^2}, \quad 0 < x < 1 \\ u(x,0) = \sin(\pi x) \qquad \text{(C.I.)} \\ u(0,t) = 0, \quad u(1,t) = exp(-\pi^2 t)\sin(\pi) \qquad \text{(C.C.)} \end{cases} \tag{6.2}$$

que é um problema bem posto e portanto a princípio admite uma única solução.

Aqui, estamos utilizando a técnica descrita na seção 3.3, para mostrar como se comporta a ordem do método para diferentes escolhas de Δt. Se escolhemos $\Delta t \approx h$ a ordem de convergência tende a um, ao passo que $\Delta t \approx h^2$, resulta em convergência tendendo a dois. Por outro lado, dado que a segunda escolha de Δt implica em elevar ao quadrado também o número de passos necessários para se alcançar o instante de tempo máximo fixado, o tempo de execução aumenta quadraticamente. A tabela 6.1 mostra a ordem de convergência do Método de Euler implícito para $\Delta t = h$ e $\Delta t = h^2$.

As colunas *Razão conv.*, com as razões de convergência do erro entre dois tamanhos de malha consecutivos, mostram os valores 2^p, onde p é a ordem de convergência.

Na figura 6.1 mostramos o gráfico do logarítimo na base 2 da norma infinito do erro cometido na aproximação da solução. Variamos o valor de Δt em relação a h, para dar uma idéia de como podemos escolher Δt para maximizar o tamanho do passo no tempo e obter o menor erro possível. Para as escolhas $\Delta t = 0.5h$ e $\Delta t = h$, múltiplos de h, observa-se que o logarítimo do erro cai com ordem menor que quando tomamos $\Delta t = 2h^2$ e $\Delta t = h^2$. No primeiro caso observa-se a dominância da ordem de convergência do método de Euler ímplicito no tempo, enquanto no segundo, dada a dependência de Δt em relação a h^2, observa-se a predominância da ordem de convergência do método de diferenças centradas no espaço.

No gráfico mostramos o efeito de se escolher dois valores arbitrários para Δt, $\Delta t = 0.01h$ e $\Delta t = 0.001h$. Nesses dois casos, o comportamento das curvas passa a variar conforme o tamanho da malha. Quando $\Delta t = 0.01h$, o erro até a malha de 128 nós fica abaixo do erro para Δt múltiplo de h^2. Esse mesmo efeito ocorre até a malha 1024 para $\Delta t = 0.001h$. Essas mudanças denotam a

alteração de ordem do método a partir de determinado tamanho de malha, e podem auxiliar na escolha de Δt em relação a h para a precisão que se deseja alcançar para a aproximação da solução.

Malha	n	Norma inf.	Razão conv. ($\Delta t = h$)	Tempo	Norma inf.	Razão conv. ($\Delta t = h^2$)	Tempo
0	8	3.4133e-02	-	0.000	3.5385e-03	-	0.000
1	16	1.4430e-02	2.365382	0.000	8.2019e-04	4.314206	0.000
2	32	6.4017e-03	2.254126	0.000	2.0092e-04	4.082128	0.000
3	64	2.9759e-03	2.151157	0.000	4.9971e-05	4.020753	0.000
4	128	1.4290e-03	2.082513	0.000	1.2477e-05	4.005202	0.050
5	256	6.9943e-04	2.043104	0.000	3.1181e-06	4.001301	0.380
6	512	3.4590e-04	2.022028	0.000	7.7947e-07	4.000326	2.980
7	1024	1.7199e-04	2.011134	0.020	1.9486e-07	4.000084	24.640
8	2048	8.5757e-05	2.005598	0.090	4.8715e-08	4.000057	197.310
9	4096	4.2819e-05	2.002807	0.370			

Tabela 6.1: Tabela de convergência para Fatorização de Crout.

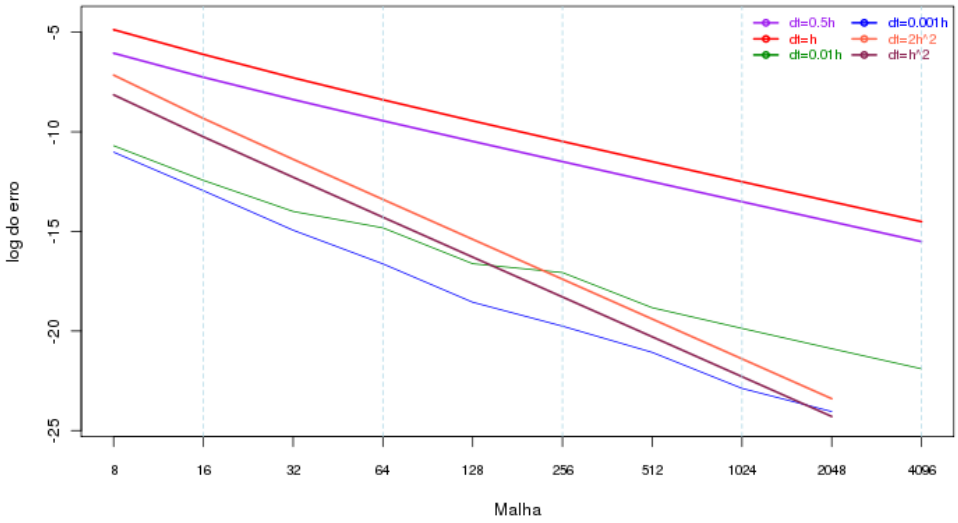


Figura 6.1: Efeito no erro cometido ao aproximar a solução para algumas escolhas de Δt em relação a h.

As tabelas 6.2, 6.3, fornecem os resultados para Jacobi e Gauss-Seidel. Observando a coluna *tempo* das tabelas, vemos que o custo computacional de usar os dois métodos indiretos é muito mais alto do que usar um método direto como a Fatorização de Crout.

O tempo de $0.24s$ obtido para uma malha de 4096 pontos, que resulta em uma norma do erro da ordem de $10e-5$, mostra-se apropriado para obter uma solução próxima o suficiente da solução manufaturada. Assim, poderia ser empregado para obter os resultados da equação de Black-Sholes original.

Embora em termos práticos a resolução de um sistema tridiagonal até a ordem que estamos trabalhando seja resolvido de forma eficiente usando-se um método direto como a Fatorização de Crout, podemos verificar bons ganhos de escalabilidade quando introduzimos OpenMP na resolução dos sistemas lineares pelo Método de Jacobi.

Para essas simulações, vamos escolher na dimensão temporal apenas o primeiro instante de tempo $t = \Delta t$. Uma medida comumente usada para verificação de escalabilidade, é o *Speedup*, dada por

Malha	n	Norma 2	Razão conv.	Norma inf.	Razão conv.	Tempo
0	8	8.0895e-03	-	1.1440e-02	-	0.000
1	16	4.3823e-03	1.845962	6.1975e-03	1.845962	0.000
2	32	2.2623e-03	1.937109	3.1993e-03	1.937109	0.020
3	64	1.1439e-03	1.977764	1.6177e-03	1.977764	0.200
4	128	5.7410e-04	1.992442	8.1190e-04	1.992442	1.560
5	256	2.8743e-04	1.997325	4.0649e-04	1.997325	12.660
6	512	1.4379e-04	1.998970	2.0335e-04	1.998970	100.160
7	1024	7.1911e-05	1.999568	1.0170e-04	1.999568	792.340

Tabela 6.2: Tabela de convergência para Método de Jacobi.

Malha	n	Norma 2	Razão conv.	Norma inf.	Razão conv.	Tempo
0	8	8.0895e-03	-	1.1440e-02	-	0.000
1	16	4.3823e-03	1.845962	6.1975e-03	1.845962	0.000
2	32	2.2623e-03	1.937109	3.1993e-03	1.937109	0.010
3	64	1.1439e-03	1.977764	1.6177e-03	1.977764	0.110
4	128	5.7410e-04	1.992442	8.1190e-04	1.992442	0.940
5	256	2.8743e-04	1.997325	4.0649e-04	1.997325	7.580
6	512	1.4379e-04	1.998970	2.0335e-04	1.998970	60.670
7	1024	7.1911e-05	1.999567	1.0170e-04	1.999567	482.160

Tabela 6.3: Tabela de convergência para Método de Gauss-Seidel.

$$S = \frac{T_{old}}{T_{new}},$$

onde T_{old} é o tempo gasto no método sem aprimoramento - que em nosso caso são os métodos de Gauss-Seidel e o método de Jacobi em paralelo - e T_{new} representa o método aprimorado.

Na figura 6.2 vemos os resultados de escalabilidade para a Equação Parabólica 1D. Esses resultados constituem a média aritimética de três execuções do programa.

Um *Speedup* maior que 1, indica ganho de escalabilidade. Pelo gráfico vemos que a depender da malha adotada, obtemos ganhos de escalabilidade diferentes. Para malhas mais grossas, como as malhas de 256 e 512 nós, obtemos *speedups* maiores com o uso de poucos processadores. Já para as outras malhas, mais finas, os ganhos maiores se dão com o aumento do número de processadores. Para essas malhas, o pico é alcançado com o uso de 8 processadores. Com 32 processadores o ganho de escalabilidade já é bem menor.

Esses resultados ocorrem devido ao *trade-off* entre o aumento do número de processadores na execução em paralelo e o custo envolvido no gerenciamento da própria execução em paralelo. Desse modo, em situações práticas esse tipo de gráfico auxilia na escolha adequada do número de processadores a serem utilizados para a paralelização de problemas de modelos particulares.

6.2 Equação do Calor 2D

Dada a ênfase deste capítulo nos resultados de escalabilidade obtidos na implementação dos algoritimos numéricos para a Equação do calor em duas dimensões espaciais, não faremos aqui a recuperação dos valores obtidos até o apreçamento da opção em si.

Vamos estabelecer apenas uma função suficientemente diferenciável para efeitos de comparação com a solução numérica e da validação dos algoritimos.

Não estamos comparando a resolução numérica da equação do calor padrão com a resolução numérica. Dessa forma a comparação é simplemente entre os três métodos de resolução - dois

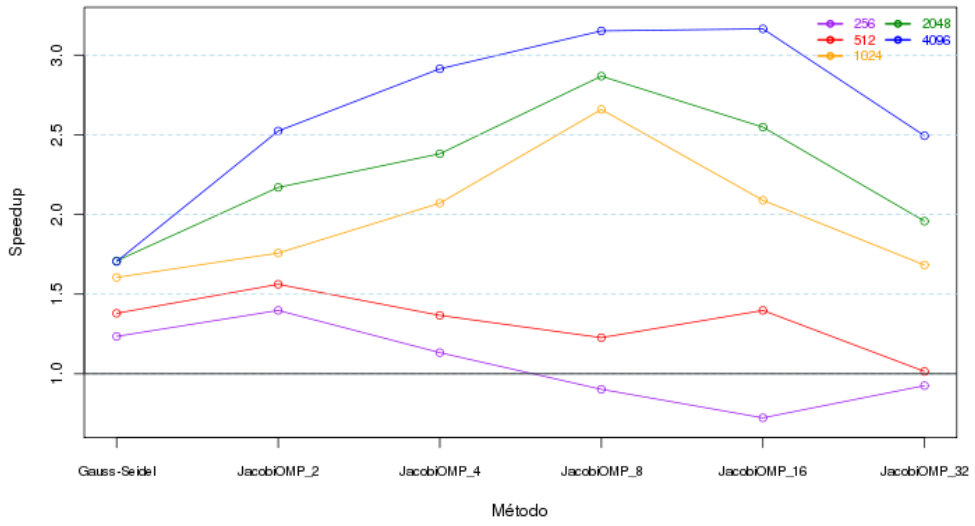


Figura 6.2: Escalabilidade de Gauss-Seidel e da paralelização de Jacobi na equação parabólica unidimensional

sequenciais e um em paralelo.

Mas mesmo assim como o procedimento para recuperar a solução seria idêntico nos três casos, os resultados são válidos para fins da aplicação a casos reais com dados provenientes dos mercados de ativos.

Como no caso unidimensional aplicamos os algoritimos para uma EDP com solução analítica conhecida e dada por

$$u(x, y, t) = exp(-t)\sin(xcos(\pi) + ysin(\pi)) \tag{6.3}$$

no domínio $[-1, 1] \times [-1, 1] \times [0, 2]$. Dessa forma o problema que estamos resolvendo é dado por

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}, \quad -1 < x < 1, \, -1 < y < 1 \\ u(x, y, 0) = \sin(xcos(\pi) + ysin(\pi)) \qquad \text{(C.I.)} \\ u(-1, y, t) = exp(-t)\sin(-\cos(\pi) + y\sin(\pi)), \\ u(1, y, t) = exp(-t)\sin(cos(\pi) + y\sin(\pi)), \\ u(x, -1, t) = exp(-t)\sin(xcos(\pi) - \sin(\pi)), \\ u(x, 1, t) = exp(-t)\sin(xcos(\pi) + \sin(\pi)) \qquad \text{(C.C.)} \end{cases} \tag{6.4}$$

A Figura 6.3 mostra o *speedup* para os métodos de Jacobi e Gauss-Seidel, e de Jacobi com paralelização via OpenMP. Como no caso unidimensional, estamos considerando apenas o primeiro instante de tempo para a comparação, e anotando a média de tempo gasto em três execuções do programa.

O programa foi executado na máquina CFD do Laboratório de Matemática Aplicada. Está máquina contém 8 processadores com 8 núcleos cada e portanto podemos ter ao todo 64 *threads* rodando em paralelo, uma *thread* para cada núcleo.

Como podemos ver na Figura 6.3 os ganhos de escalabilidade variam bastante com o tamanho da malha de discretização do problema. Para as malhas mais finas, 128×128 e 256×256, o maior ganho da paralelização se dá ao usar 16 procesadores, enquanto para as outras malhas, mais grossas, o uso de 32 processadores resulta em maior eficiência.

Esse comportamento destoante do caso unidimensional quando estamos usando 32 processadores também é relacionado ao gerenciamento interno da execução em paralelo.

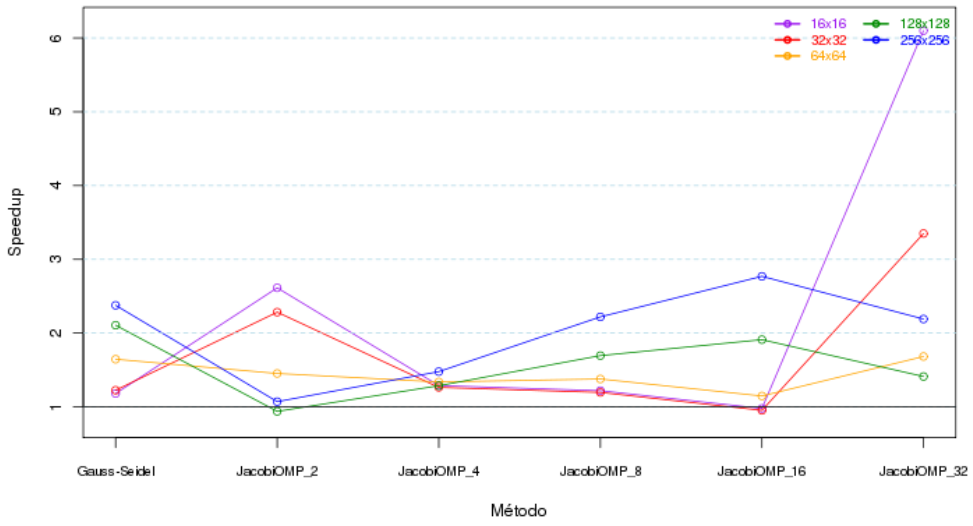


Figura 6.3: Escalabilidade de Gauss-Seidel e da paralelização de Jacobi na Equação parabólica bidimensional

Finalmente, nota-se também que sempre há ganho de escalabilidade para alguma malha em relação ao algoritimo de Gauss-Seidel nos dois casos, unidimensional e bidimensional.

Capítulo 7

Conclusões

A resolução numérica da Equação do Calor em uma dimensão se mostrou bastante eficiente ao se adotar um Método implícito para a discretização do tempo e diferenças centradas para discretização do espaço. A Fatorização de Crout consegue gerar uma aproximação muito boa - norma do erro da ordem de 10^-4 - para malhas bastante finas.

Assim os resultados obtidos no caso unidimensional favorecem a resolução numérica da equação do calor com a finalidade de se recuperar posteriormente os preços de opções européias dados pela Equação de Black-Scholes original.

Por outro lado, a implementação em paralelo do Método de Jacobi obtêm bons ganhos de escalabilidade em relação tanto ao próprio Método de Jacobi, quanto ao Método de Gauss-Seidel.

No caso da Equação Parabólica em duas dimensões, a paralelização do Algoritimo de Jacobi em ambiente de memória compartilhada se mostrou eficiente na comparação com a versão sequencial, e também com o Método de Gauss-Seidel.

Dessa forma, a paralelização tanto em uma quanto em duas dimensões mostrou bons ganhos de eficiência. Sendo que no caso bidimensional, no qual métodos iterativos são uma boa opção para a resolução dos sistemas lineares, essa poderia ser usada aproximar a solução da equação parabólica e, posteriormente, encontrar a aproximação do preço da opção utilizando a equação de Black-Scholes bidimensional.

Apêndice A

Programas

1. Aqui vão os programas com implementações dos 3 métodos em 1D e 2D.

q

Referências Bibliográficas

[Bul92] J. Stoer R. Bulirsch. *Introduction to numerical analysis*. Springer, 1992. 10

[eCS98] Les Clewlow e Chris Strickland. *Implementing derivatives models*. Wiley, 1998. 6

[eDF10] Richard Burden e Douglas Faires. *Numerical Analysis*. Brooks Cole, 2010. ix, 8, 13, 15

[eHBK66] Eugene Issacson e Herbert B. Keller. *Analysis of Numerical Methods*. Courier Dover Publications, 1966. 16

[eKWM67] R. D. Richtmyer e K. W. Morton. *Difference Methods for Initial-value Problems*. Wiley-Interscience, 1967. 10

[eMS73] Fischer Black e Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81:637–654, 1973. 3, 4

[eYK13] Joonglee Jo e Yongsik Kim. Comparison of numerical schemes on multi-dimensional black-scholes equations. *Bulletin of the Korean Mathematical Society*, 50:2035–2051, 2013. 6

[LeV05] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics (SIAM, 2005. 9, 18

[Ort72] James M. Ortega. *Numerical Analysis: A Second Course*. Society for Industrial and Applied Mathematics, 1972. 18

[You71] David M. Young. *Iterative solution of large linear systems*. Academic Press, 1971. 18