Instituto Nacional de Matemática Pura e Aplicada

Calibragem de Superfície de Volatilidade Local

Autor: **Vinícius Hessel**

Orientador: **Vinícius Viana Luiz Albani**

Rio de Janeiro
Março de 2014

Para Francisco e Eliana, meus pais e maiores incentivadores.

Agradecimentos

Agradeço a todos aqueles que me ajudaram e contribuíram para o desenvolvimento deste trabalho. Primeiramente à Vinícius Albani, professor orientador, cuja orientação permitiu a conclusão deste trabalho. Ao Instituto de Matemática Pura e Aplicada, na figura do professor e coordenador Jorge Passamani Zubelli, por toda atenção e esforço dispensado ao meu aprendizado. E aos amigos e familiares por prover o incentivo necessário a superação dos inúmeros desafios ao longo dos meus estudos.

Abstract

This work deals with the inverse problem of local volatility calibration from Dupire Equation. The solution is obtained by adjusting, in the least squares sense, the prices of European call options with observations of prices in the market. To cope with the instability of the problem, Tikhonov regularization was used and the regularization parameter was chosen based on the Morozov discrepancy principle. The minimization is carried out using a quasi-Newton method of large-scale optimization and the gradient of the objective function was obtained by adjoint state method. Simulation and real data results are presented.

Key words: local volatility; adjoint state; dupire equation.

Sumário

Índice **vii**

Introdução **1**

1 Equação de Dupire **3**

2 Calibragem da Volatilidade Local **6**
2.1 Problema dos Mínimos Quadrados 6
2.2 Regularização 7
2.2.1 Regularização de Tikhonov 7
2.2.2 Princípio da Discrepância de Morozov 8
2.3 Método do Estado Adjunto 9
2.3.1 Motivação 9
2.3.2 Derivação 12
2.3.3 Gradiente de J 14

3 Método dos Elementos Finitos **15**
3.1 Formulação Fraca 15
3.1.1 Alguns Espaços de Funções 16
3.1.2 Formulação Fraca da Equação de Dupire 16
3.2 Problema Parabólico de Valor de Contorno 17
3.2.1 Discretização de Tempo 18
3.2.2 Discretização de Tempo e Espaço 18
3.2.3 Forma Matricial 19
3.2.4 Escolha da Base 19
3.3 Discretização da Equação de Dupire 20
3.4 Discretização do Estado Adjunto 22

4 Problema de Minimização **24**
4.1 Métodos Line Search 24
4.1.1 Comprimento do Passo 24
4.1.2 Condições de Wolfe 25
4.1.3 Algoritmo de Escolha do Comprimento do Passo 25
4.2 Quasi-Newton LBFGS 28
4.2.1 Método BFGS 28
4.2.2 Método L-BFGS 29

5 Resultados Numéricos **31**
5.1 Simulação 31
5.1.1 Geração de Dados 31
5.1.2 Volatilidade Local 33
5.2 Dados Reais 35
5.2.1 Preparação dos Dados 35
5.2.2 Volatilidade Local 38

6 Conclusão **43**

A Códigos **44**
A.1 Opções 44
A.2 Otimização 47
A.3 Conversão de Dados 53

Referências Bibliográficas **57**

Introdução

No mercado financeiro, uma opção é um contrato que dá ao seu detentor o direito, mas não a obrigação, de comprar ou vender um determinado ativo financeiro, chamado de ativo subjacente, a um certo preço, o preço de exercício, até uma data limite chamada de vencimento. O vendedor do contrato de opção tem a obrigação de vender, ou comprar, o ativo subjacente pelo valor acordado, caso o detentor da opção deseje exercer seu direito. Uma opção que dá a seu detentor o direito de compra do ativo subjacente é chamada de opção de compra e uma opção que dá a seu detentor o direito a venda do ativo subjacente é chamada de opção de venda. Em relação ao ativo subjacente, o direito de compra ou venda das opções podem recair sobre qualquer ativo financeiro: ações, contratos futuros, índices futuros, etc. Existem muitos estilos de opções mas os mais comuns, abordados neste trabalho, são os americanos e europeus. As opções europeias permitem ao seu detentor o exercício do direito de compra ou venda do ativo subjacente apenas na data de seu vencimento. Por outro lado, as opções americanas permitem o exercício do direito de compra e venda na data do seu vencimento ou em qualquer momento anterior.

O mais famoso resultado teórico na área de precificação de opções é devido à Fischer Black e Myron Scholes em seu artigo "The pricing of Options and Corporate Liabilities"[4]. Baseado na hipótese de volatilidade constante, o celebrado modelo de Black-Scholes pode ser usado para precificar opções europeias utilizando, como parâmetro, uma volatilidade estimada. O modelo considera uma opção europeia de compra, ou venda, com vencimento T e preço de exercício K, sobre um ativo subjacente S e um mercado teórico com dois ativos negociáveis: um ativo livre de risco com taxa de juros constante r e um ativo arriscado cujo processo de preços é originado por um movimento Browniano padrão W:

$$dS_t \;=\; S_t(r\,dt + \sigma\,dW_t), \quad t > t_0; \quad S_{t_0} = S_0,$$

A equação diferencial parcial, tradicionalmente conhecida como Equação de Black-Scholes tem a forma

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + (r-q)S\frac{\partial C}{\partial S} - rC = 0, \tag{0.0.1}$$

Se o modelo de Black-Scholes descrevesse perfeitamente o mercado, a volatilidade implícita, obtida à partir dos preços das opções europeias negociadas no mercado, deveria ser constante para as opções europeias, de todos os vencimentos e preços de exercício, sobre um mesmo ativo subjacente. Entretanto, no mercado, observa-se que a volatilidade implícada pelo modelo de Black-Scholes varia com o preço de exercício e vencimento das opções sobre um mesmo ativo subjacente.

Ao longo dos anos, foram feitas diversas tentativas de estender a teoria de Black-Scholes para concordar com as características da volatilidade observada no mercado. Uma classe de modelos, introduz uma fonte de risco não negociável como saltos e volatilidade estocástica [13].

Outra classe considera a volatilidade como função determinística do preço do ativo subjacente e do tempo, normalmente chamada de volatilidade local. Para esta última classe de problemas, o modelo de Black-Scholes continua sendo um modelo de um parâmetro e mantém a completude do mercado, isto é, a possibilidade de realizar *hedge* de opções com o ativo arriscado.

O problema de obter a função volatilidade local à partir do ajuste de uma modelo paramétrico, cujo parâmetro é a própria volatilidade local, configura um problema inverso. Muitos problemas inversos são mal-postos no sentido de Hadamard e, para tratar este problema, uma técnica comumente utilizada é a chamada Regularização de Tikhonov, usada para estabilizar o problema. Muitas fontes na literatura provam questões à respeito da unicidade e estabilidade do problema regularizado e a bibliografia sobre o problema de calibragem da volatilidade local com regularização de Tikhonov é vasta [11] [2] [12].

Neste trabalho, nós tratamos da calibragem da função de volatilidade local, comumente chamada de superfície de volatilidade local, atráves do ajuste do modelo com dados de preço de opções europeias de compra observados no mercado. A calibragem será feita através da minimização de mínimos quadrados utilizando uma regularização de Tikhonov. O gradiente da função de mínimos quadrados foi obtida através do método do estado adjunto. No primeiro capítulo, nós apresentaremos a Equação de Dupire que tem papel central no problema aqui tratado. Em seguida, nós apresentamos o problema de minimização associado à calibragem da volatilidade local e traremos dos aspectos de regularização e obtenção do gradiente da função objetivo. No capítulo três, nós apresentamos o método dos elementos finitos, utilizado para discretização e resolução dos diversos problemas envolvendo equações diferenciais parciais. Um método quasi-Newton de otimização de larga escala será apresentado no capítulo quatro. Por fim, nós apresentamos os resultados numéricos obtidos com simulações e dados reais do mercado de opções sobre o contrato futuro de *commodities*.

Capítulo 1

Equação de Dupire

A equação de Black-Scholes é a mais utilizada para a precificação de opções europeias mas, como dito anteriormente, ela falha quando assume que a volatilidade do ativo subjacente é constante. A experiência prática nos diz que a volatilidade implicada pelo modelo de Black-Scholes é fortemente dependente do tempo e do preço de execício, efeito conhecido como *smiles*. As tentativas de extensão do modelo, inserindo ativos de risco não negociáveis como saltos, volatilidade estocástica ou custo de negociação terminam por eliminar a completude do mercado. Tendo isso em mente, Dupire [8] procura encontrar uma extensão do modelo que, ao mesmo tempo, concorde com a dependência da volatilidade com o tempo e preço do exercício e mantenha a completude do mercado. Ele, então, imagina um processo de difusão de preço do ativo subjacente que apresente uma função de volatilidade dependente do tempo e preço do ativo

$$dS = S(r(t)dt + \sigma(S,t))dW.$$

onde W é um processo de Wiener e a volatilidade instantânea σ é uma função determinística do tempo t e do preço do ativo subjacente S. De fato, Dupire mostrou que a função de volatilidade local pode ser unicamente determinada à partir do preço das opções europeias de todos os vencimentos e preços de exercício [8] . Na prática, esse cenário não existe. Nós só temos acesso a opções europeias com poucos valores de preço de exercício e vencimento. Dessa forma, o problema de encontrar a volatilidade local pode ser considerado um problema de aproximação feita à partir de um conjunto finito de observações.

Nesse capítulo, nós deduziremos a equação de Dupire em sua forma mais geral [17], dada por:

$$\frac{\partial C}{\partial T} - \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} + (r_T - q_T)K\frac{\partial C}{\partial K} + q_T C = 0 \tag{1.0.1}$$

Denotemos $P(t,T)$ o fator de desconto no tempo t:

$$P(t,T) = \exp\left(\int_t^T r_t \, dt\right)$$

O preço $C = C(S_t, K)$ de uma Call europeia de *strike* K, no tempo t, com preço do ativo objeto S_t é dado por:

$$\begin{aligned} C &= P(t,T)E^Q[(S_T - K)^+] \\ &= P(t,T)E^Q[(S_T - K)\mathbb{1}_{S_T > K}] \\ &= P(t,T)\int_K^\infty (S_T - K)f(S,T)\, dS \end{aligned} \tag{1.0.2}$$

onde $f(S_t, t)$ denota a função densidade de probabilidade do preço S_t, do ativo objeto, no tempo t e Q refere-se à medida de probabilidade neutra ao risco. A função f satisfaz a equação de Fokker-Planck:

$$\frac{\partial f}{\partial t} = -\frac{\partial}{\partial S}[\mu S f(S,t)] + \frac{1}{2}\frac{\partial^2}{\partial S^2}[\sigma^2 S^2 f(S,t)]. \tag{1.0.3}$$

Derivando (1.0.2) com respeito ao *strike* :

$$\begin{aligned}\frac{\partial C}{\partial K} &= P(t,T)\int_K^\infty \frac{\partial(S_T - K)}{\partial K} f(S,T)\, dS \\ &= -P(t,T)\int_K^\infty f(S,T)\, dS\end{aligned} \tag{1.0.4}$$

A seguda derivada com respeito ao *strike*:

$$\begin{aligned}\frac{\partial^2 C}{\partial K^2} &= -P(t,T)[f(S_T - K)]_{S+K}^{S=\infty} \\ &= P(t,T)f(K,T).\end{aligned} \tag{1.0.5}$$

assumindo que $\lim_{S\to\infty} f(S,T) = 0$.
Tomando, agora, a derivada com respeito ao vencimento:

$$\frac{\partial C}{\partial T} = \frac{\partial C}{\partial T}P(t,T) \times \int_K^\infty (S_T - K) f(S,T)\, dS + P(t,T) \times \int_K^\infty (S_T - K)\frac{\partial f(S,T)}{\partial T}\, dS. \tag{1.0.6}$$

mas $\frac{\partial P}{\partial T} = -r_T P(t,T)$, logo podemos reescreve (1.0.6) como :

$$\frac{\partial C}{\partial T} = -r_T C + P(t,T) \times \int_K^\infty (S_T - K)\frac{\partial f(S,T)}{\partial T}\, dS. \tag{1.0.7}$$

Utilizando a equação de Fokker Planck (1.0.3) para $\frac{\partial f}{\partial t}$, com $t = T$, em (1.0.7):

$$\begin{aligned}\frac{\partial C}{\partial T} + r_T C = P(t,T)\int_K^\infty (S_T - K)\times \\ \left\{-\frac{\partial[\mu_T S f(S,T)]}{\partial S} + \frac{1}{2}\frac{\partial^2[\sigma^2 S^2 f(S,T)]}{\partial S^2}\right\}\, dS.\end{aligned} \tag{1.0.8}$$

Na equação (1.0.8), devemos avaliar duas integrais:

$$\begin{aligned}I_1 &= \mu_T \int_K^\infty (S_T - K)\frac{\partial[S f(S,T)]}{\partial S}\, dS, \\ I_2 &= \int_K^\infty (S_T - K)\frac{\partial^2[\sigma^2 S^2 f(S,T)]}{\partial S^2}\, dS.\end{aligned} \tag{1.0.9}$$

Para esse cálculo, utilizaremos duas identidades. A primeira segue da equação (1.0.5) :

$$f(K,T) = \frac{1}{P(t,T)}\frac{\partial^2 C}{\partial K^2} \tag{1.0.10}$$

A segunda identidade, decorre das equações (1.0.2) e (1.0.4) :

$$\begin{aligned}\frac{C}{P(t,T)} &= \int_K^\infty (S_T - K) f(S,T)\, dS \\ &= \int_K^\infty S_T f(S,T)\, dS - K\int_K^\infty f(S,T)\, dS.\end{aligned} \tag{1.0.11}$$

Mas

$$\int_K^\infty f(S,T)\,dS = -\frac{1}{P(t,T)}\frac{\partial C}{\partial K}$$

Substituindo na equação (1.0.11), obtemos a segunda identidade:

$$\int_K^\infty f(S,T)\,dS = \frac{C}{P(t,T)} - \frac{K}{P(t,T)}\frac{\partial C}{\partial K} \tag{1.0.12}$$

Podemos, agora, calcular as integrais I_1 e I_2. Integrando I_1 por partes com $u = S_T - K,\ u' = 1,\ v' = \frac{\partial[SF(S,T)]}{\partial S},\ v = Sf(S,T)$:

$$\begin{aligned} I_1 &= [\mu_T(S_T - K)S_T f(S,T)]_{S=K}^{S=\infty} - \mu_T \int_K^\infty Sf(S,T)\,dS \\ &= [0-0] - \mu_T \int_K^\infty Sf(S,T)\,dS \end{aligned} \tag{1.0.13}$$

onde nós assumimos que $\lim_{S\to\infty}(S-K)Sf(S,T) = 0$. Da segunda identidade (1.0.12), temos:

$$I_1 = \frac{-\mu_T C}{P(t,T)} + \frac{\mu_T K}{P(t,T)}\frac{\partial C}{\partial K} \tag{1.0.14}$$

Integrando I_2 por partes, com $u = S_T - K, u' = 1, v' = \frac{\partial^2[\sigma^2 S^2 f(S,T)]}{\partial S^2}, v = \frac{\partial[\sigma^2 S^2 f(S,T)]}{\partial S}$:

$$\begin{aligned} I_2 &= \left[(S_T - K)\frac{\partial\{\sigma^2 S^2 f(S,T)\}}{\partial S}\right]_{S=K}^{S=\infty} - \int_K^\infty \frac{\partial[\sigma^2 S^2 f(S,T)]}{\partial S}\,dS \\ &= [0-0] - [\sigma^2 S^2 f(S,T)]_{S=K}^{S+\infty} \\ &= \sigma^2 K^2 f(K,T) \end{aligned} \tag{1.0.15}$$

onde $\sigma^2 = \sigma(K,T^2)$. Assumimos que $\lim_{S\to\infty}\frac{\partial\{\sigma^2 S^2 f(S,T)\}}{\partial S} = 0$. Agora, podemos avaliar a equação (1.0.8) :

$$\frac{\partial C}{\partial T} + r_T C = P(t,T)\left[-I_1 + \frac{1}{2}I_2\right]$$

Substituindo I_1 e I_2, temos:

$$\frac{\partial C}{\partial T} + r_T C = \mu_T C - \mu_T K \partial_K C + \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2}$$

Substituindo o *drift* pela taxa de juros descontada do dividendos, $\mu_T = r_T - q_T$, obtemos a equação de Dupire (1.0.1) :

$$\frac{\partial C}{\partial T} = \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} - (r_T - q_T)K\frac{\partial C}{\partial K} - q_T C$$

Capítulo 2

Calibragem da Volatilidade Local

Neste capítulo, nós apresentaremos o problema de mínimos quadrados para calibragem da volatilidade local através do ajuste de preços de opções europeias de compra com valores encontrados no mercado. A utilização da Equação de Dupire elimina grande custo computacional na avaliação da função erro. Em razão da instabilidade natural dessa classe de problemas mal-postos, utilizaremos um termo de regularização na função objetivo de mínimos quadrados.

Por fim, nós apresentaremos o Método do Estado Adjunto, largamente utilizado para análise de sensibilidade em diversos campos da ciência e engenharia. Neste trabalho, nós utilizaremos o método para obter, de forma eficiente, o gradiente da função de mínimos quadrados eliminando grande parte do trabalho computacional que seria utilizado no cálculo do gradiente através de métodos de diferenciação automática.

2.1 Problema dos Mínimos Quadrados

Denote η o quadrado da volatilidade local. Consideraremos a equação de Dupire no retângulo $Q = [0, \tilde{K}] \times [0, \tilde{\tau}]$, tomando $\tilde{K}$ e $\tilde{\tau}$ suficientemente grandes. Chamando S_0 o preço atual do ativo objeto, o preço $C(K, \tau) := C(S_0, 0, K, \tau)$ é solução do problema de condição de contorno:

$$\begin{aligned} \partial_t C - \frac{\eta K^2}{2}\partial^2_{KK} C + (r-q)K\partial_K C + qC = 0, \quad (\tau, K) \in K \\ C(K,0) = (S_0 - K)^+, \quad K \in (0, \tilde{K}), \\ C(0,\tau) = S_0 . \exp(-qt), \quad t \in (0, \tilde{\tau}], \\ C(\tilde{K}, \tau) = 0, \quad \tau \in (0, \tilde{\tau}). \end{aligned} \tag{2.1.1}$$

A taxa de juros e dividendos são assumidos como constantes. O problema de calibragem consiste na obtenção de η das observações de:

- O preço atual S_0 do ativo objeto;
- Os preços $(c_i)_{i \in I}$ de opções do tipo Call europeia simples com diferentes vencimentos e preços de exercício em $(\tau_i, K_i)_{i \in I}$

Pode-se escolher $\tilde{\tau}$ e $\tilde{K}$ à partir das observações de preços como $\tilde{\tau} > max_{i \in I}\ \tau_i$ e $\tilde{K} >> max((K_i)_{i \in I},\ S_0)$. Consideramos o problema de mínimos quadrados: encontrar $\eta \in \mathcal{H}_h$ minimizando:

$$J(\eta) + J_R(\eta), \qquad J(\eta) = \sum_{i \in I} |C(K_i, \tau_i) - c_i| \tag{2.1.2}$$

onde $\mathcal{H}_h$ é subconjunto adequado de um espaço de funções possivelmente infinito-dimensional, J_R é um funcional de Tikhonov adequado e C é solução de (2.1.1).

2.2 Regularização

Como citado anteriormente, o problema que consiste em encontrar a função volatilidade local à partir da minmização de (2.1.2) é mal-posto no sentido de Hadamard. Em particular, não há garantia de dependência contínua da função volatilidade com as observações de preço do mercado e pequenas perturbações nos dados de preço podem resultar em grandes mudanças na função objetivo. Além disso, o número finito e discreto das observações de preço podem gerar infinitas soluções para o problema. Para tornar o problema bem-posto, uma regularização deve ser imposta ao funcional J. A forma mais comum de regularização pode, por exemplo, assumir a forma:

$$F(\eta) = J(\eta) + \lambda \|\eta - \eta_0\|^2 \tag{2.2.1}$$

onde λ é o parâmetro de regularização de Tikhonov e (2.2.1) representa a regularização de Tikhonov de ordem zero. Em outra maneira de regularizar o problema, temos a forma:

$$F(\eta) = J(\eta) + \lambda \|\nabla \eta\|^2 \tag{2.2.2}$$

que caracteriza a regularização de Tikhonov de ordem um.
Na sessão seguinte, será utilizado um exemplo simples e genérico para ilustrar as ideias e conceitos envolvidos com a Regularização de Tikhonov e, por fim, será apresentado um método de escolha do parâmetro de regularização baseado no Princípio da Discrepância de Morozov.

2.2.1 Regularização de Tikhonov

Considere um problema linear inverso:

$$GM = D \tag{2.2.3}$$

onde G é o operador que representa o modelo, M são os parâmetros de entrada e D são os dados observados. O problema direto tem por objetivo calcular D dado M. O problema inverso visa obter M dado D. Por simplicidade, tomemos G como uma matriz $n \times m$ e M e D vetores de tamanho n e m, respectivamente. Uma abordagem para encontrar a melhor solução aproximada de (2.2.3), é encontrar M que minimize o erro entre os dados observados e os dados calculados no problema direto. Uma estratégia tradicional é minimizar a norma em L^2 do resíduo:

$$\|GM - D\|_{L^2}^2$$

Este é um problema geral de mínimos quadrados e um método de resolução, de particular interesse em problemas mal-postos, é através da decomposição em valores singulares (*Singular Value Decomposition* - SVD). Com SVD, G é fatorado com a forma:

$$G = U \Sigma V^*$$

onde U é uma matriz unitária $m \times m$, V é uma matriz unitária $n \times n$ e S é uma matriz diagonal $m \times n$ com os valores singulares em sua diagonal arranjados em ordem decresente. É importante

notar que podem existir valores singulares nulos.
Usando a pseudo-inversa de G, a solução para o problema (2.2.3) pode ser expressa por:

$$M = V\Sigma_p^{-1}U^*D = \sum_{i=1}^{p} \frac{U_{.,i}^* D}{s_i} V_{.,i} \tag{2.2.4}$$

onde $s_i,\ i = 1, 2, ..., min(m, n)$ denotam os valores singulares em Σ. Pode-se perceber que a solução do problema inverso se torna instável quando um ou mais valores singulares estão próximos de zero.
Dentre as possíveis soluções para o problema (2.2.3), a Regularização de Tikhonov seleciona aquela mais adequada ao domínio do problema. Uma maneira da regularização tem a forma:

$$\min \|GM - D\|_{L^2}^2 + \alpha^2 \|M\|_{L^2}^2 \tag{2.2.5}$$

onde α é o parâmetro de regularização.
A solução de (2.2.5) pode ser obtido através do problema de mínimos quadrados aumentado para (2.2.3) da seguinte maneira:

$$\min \left\| \begin{bmatrix} G \\ \alpha I \end{bmatrix} - \begin{bmatrix} D \\ 0 \end{bmatrix} \right\|$$

Utilizando a pseudo-inversa de $\begin{bmatrix} G \\ \alpha I \end{bmatrix}$, a solução para o problema (2.2.5) é dada por:

$$M_\alpha = \sum_{i=1}^{k} \frac{s_i^2}{s_i^2 + \alpha_i^2} \frac{U_{i,i}^* D}{s_i} V_{.,i}$$

Considere

$$f_i = \frac{s_i^2}{s_i^2 + \alpha_i^2}$$

Para $s_i \gg \alpha_i$, $f_i \approx 1$, e para $s_i \ll \alpha_i$, $f_i \approx 0$. Para valores singulares entre esses extremos, f_i produz uma contribuição monotonicamente decrescente, em s_i, do vetor $V_{.,i}$ correspondente.
Para cada valor do parâmtero de regularização α, existe os valores da solução de minimização para $\|GM - D\|_{L^2}^2$ e $\|M\|_{L^2}^2$. A curva do logaritmo de $\|M\|_{L^2}^2$ versus $\|GM - D\|_{L^2}^2$ geralmente toma a forma de uma curva em L. Uma possível estratégia para a escolha do parâmetro de regularização é tomar o valor de α que gera a solução próxima ao vértice da curva em L. Este é o chamado Critério da Curva em L.

2.2.2 Princípio da Discrepância de Morozov

Em problemas inversos mal-postos, a escolha adequada do parâmetro de regularização tem grande influência sobre o resultado final da minimização. A bibliografia que trata desse assunto é vasta e não temos intenção de fazer uma revisão detalhada desses métodos. Neste trabalho faremos uma abordagem com um viés bastante prático, seguindo a abordagem de [18], de um método de escolha do parâmetro de regularização baseado no Princípio da Discrepância de Morozov.
Suponha um problema de minimização com regularização de Tikhonov

$$F_\delta(x) = \|Ax - y\|^2 + \delta\|x\|^2, \tag{2.2.6}$$

e suponha que um minimizador x_δ exista. O parâmetro $\delta > 0$ é o parâmetro de regularização.

Assumindo que nós tenhamos um estimador $\epsilon > 0$ da norma do erro no vetor de dados de (2.2.6) então, qualquer vetor x tal que

$$\|Ax - y\| \leq \epsilon, \tag{2.2.7}$$

deveria ser considerada uma solução aproximada. A inequação (2.2.7) nos fornece uma boa indicação para a parada do algoritmo de minimização. De fato, uma redução da função objetivo a níveis inferiores ao ruído não nos fornece uma solução mais próxima da solução ideal. Se nós definirmos a função discrepância por

$$f : \mathbb{R}_+ \to \mathbb{R}_+, f(\delta) = \|Ax_\delta - y\|,$$

o Princípio da Discrepância de Morozov afirma que o parâmetro de regularização δ deveria ser escolhido satisfazendo

$$f(\delta) = \|Ax_\delta - y\| = \epsilon, \tag{2.2.8}$$

se possível, isto é, a solução regularizada não deve ajustar a função aos dados com mais precisão do que o nível de ruído, como indicado de (2.2.7).

A Figura 5.3 ilustra graficamente o Princípio da Discrepância de Morozov. A escolha do parâmetro de regularização é feita através do valor de δ que gera um valor de discrepância igual ao estimador do ruído. Vale notar que a função discrepância é estritamente crescente com o valor de δ. Por outro lado, o valor da regularização é estritamente decrescente com o valor de δ como ilustrado na Figura 2.2.

2.3 Método do Estado Adjunto

A análise de sensibilidade é um campo de larga aplicação em ciências e engenharia, incluindo otimização, estimação de parâmetros, simplificação de modelos, controle ótimo, análise de incerteza e modelagem experimental. Em análise de sensibilidade, estuda-se o comportamento de funções em relação à um conjunto de parâmetros de interesse. Nos métodos de otimização baseados no gradiente, o custo computacional para a obtenção do gradiente ganha importância à medida que o número de variáveis envolvidas na otimização aumenta, sendo que diversas ferramentas de otimização utilizam técnicas de diferenciação automática, como diferenças finitas, para esse cálculo. Neste trabalho, utilizaremos uma técnica conhecida como Método do Estado Adjunto para a obtenção do gradiente da função custo J. Nas próximas sessões, apresentaremos a motivação e aspectos teóricos relevantes ao método do estado adjunto, mas uma introdução ao assunto pode ser encontrada em [9]. Por fim, apresentaremos a derivação do estado adjunto aplicado ao problema inverso objeto de estudo deste trabalho.

2.3.1 Motivação

Considere $x \in \mathbb{R}^{n_x}$ e $p \in \mathbb{R}^{n_p}$. Seja uma função $g(x,p) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$ e uma função $f(x,p) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ tal que $f(x,p) = 0$ e cuja derivada parcial $\frac{\partial f}{\partial_x}$ é não singular em toda parte. A equação $f(x,p) = 0$ é, normalmente, resolvida por software que implementa o problema direto. Dado valores para os parâmetros p, o programa calcula os valores x. Por outro lado, $g(x,p)$ é tomado como uma medida de mérito sendo, por exemplo, uma medida de erro de ajuste da função, uma medida de suavidade de x e p ou qualquer função que mensure

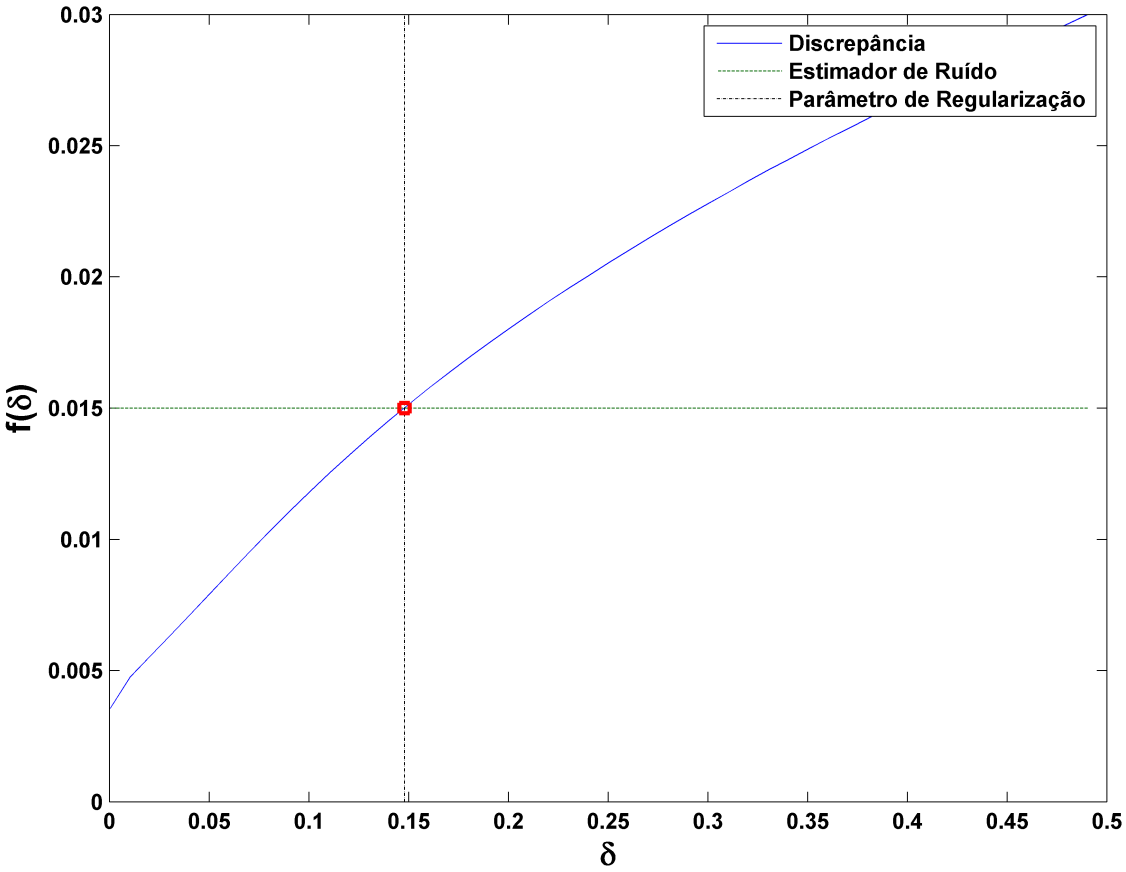

Figura 2.1: Princípio de Morozov: O parâmetro de regularização corresponde à intersecção da função discrepância com o estimador de erro

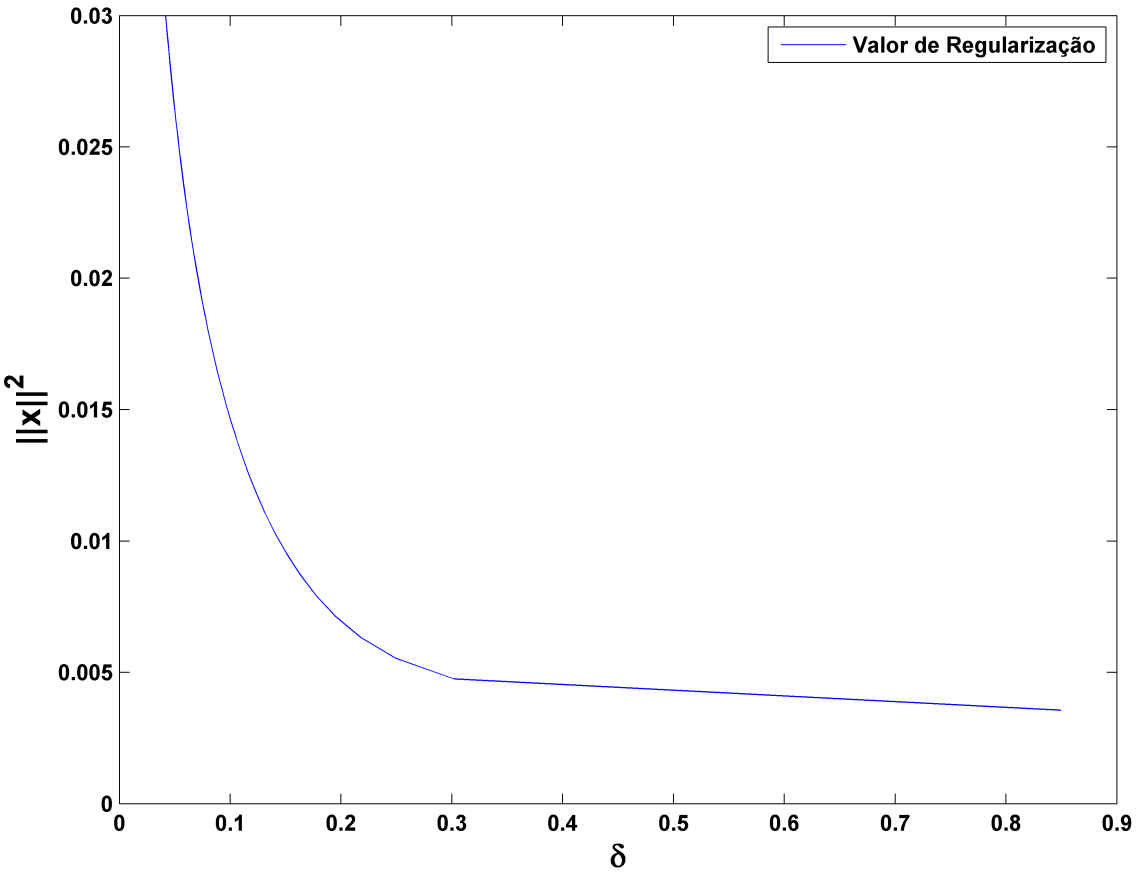

Figura 2.2: Valor da Regularização de Tikhonov

o grau em que p atinge certo objetivo. A minimização de f pertence à classe de problemas inversos.
Nesse contexto, é desejável obter um método eficiente para o cálculo do gradiente $\nabla_p g$. Uma estratégia seria a resolução de uma diferença finita para cada uma das n_p dimensões de p. Cada diferença finita envolve, ao menos, uma solução de $f(x,p) = 0$. É fácil perceber que esse método de cálculo do gradiente se torna ineficiente para grandes dimensões de p.

2.3.2 Derivação

Nesta seção, abordaremos o método do estado adjunto aplicado à problemas que envolvam equações diferenciais parciais com dependência de tempo. No método do estado adjunto, o gradiente é calculado indiretamente através da solução do problema adjunto. Considere $x \in \mathbb{R}^{n_x}$, $p \in \mathbb{R}^{n_p}$, um sistema de equações diferenciais parciais (EDP):

$$F(t, u, u_t, u_x, u_{xx}, p) = 0, \tag{2.3.1}$$

e uma função objetivo $G(x, u, p)$ que depende de u e p. Pela regra da cadeia, o gradiente $\nabla_p G$ é dado por:

$$\nabla_p G = \frac{dG}{dp} = \frac{\partial G}{\partial u}\frac{\partial u}{\partial p} + \frac{\partial G}{\partial P}$$

Tomando (2.3.1) como um sistema não linear sobre u e p, $H(u,p) = F(t, u, u_x, u_{xx}, p)$, tem-se a seguinte relação:

$$\frac{\partial H}{\partial u}\frac{\partial u}{\partial p} + \frac{\partial H}{\partial p} = 0$$

Assumindo que $\frac{\partial H}{\partial u}$ é inversível, o gradiente $\nabla_p G$ é dado por:

$$\frac{dG}{dp} = -\frac{\partial G}{\partial u}\left(\frac{\partial H}{\partial u}\right)^{-1}\frac{\partial H}{\partial p} + \frac{\partial G}{\partial p} \tag{2.3.2}$$

Existem dois métodos para o cálculo do gradiente à partir de (2.3.2): Método direto e método adjunto.
No método direto calcula-se, primeiramente, $\frac{\partial u}{\partial p} = \left(\frac{\partial H}{\partial u}\right)^{-1}\frac{\partial H}{\partial p}$ que é a hessiana da solução u da EDP em relação aos parâmetros p. Por outro lado, o método adjunto primeiramente calcula $\frac{\partial G}{\partial u}\left(\frac{\partial H}{\partial u}\right)^{-1}$ que é solução do problema adjunto. Embora as soluções sejam as mesmas para ambos os métodos, o custo computacional envolvido é diferente. O método direto é adequado para problemas que envolvam poucos parâmetros de otimização ou com grande número de funções objetivo enquanto o método adjunto é mais eficiente em problemas envolvendo um grande número de parâmetros e poucas funções objetivo.
Existem duas abordagens para o método do estado adjunto: No primeiro, chamado abordagem discreta, primeiro discretiza-se o problema original e, então, diferencia-se o sistema discreto com respeito aos parâmetros. No segundo caso, chamado abordagem contínua, primeiramente diferencia-se o sistema original em relação aos parâmetros e, só então, discretiza-se as PDEs adjuntas. A abordagem discreta é a de mais simples implementação e pode ser resolvida através de técnicas de diferenciação automática enquanto a abordagem contínua leva à derivação e solução de um novo problema, o problema adjunto. Neste trabalho utilizou-se a abordagem contínua que gera resultados mais precisos além de não necessitar de restrições sobre o domínio discretizado.

Vamos novamente considerar um sistema de EDP sobre o domínio Ω e assumir que a PDE envolve apenas derivadas até a segunda ordem:

$$F(t, u_{,t}, D_x(u), D_x^2(u)) = 0$$

Definindo a função objetivo como uma integral sobre o domínio de tempo e espaço, temos:

$$G(x, u, p) = \int_0^T \int_\Omega g(t, x, u, p)\, dx\, dt \tag{2.3.3}$$

O gradiente de G com respeito a p é dado por:

$$\nabla_p G = \frac{dG}{dp} = \int_0^T \int_\Omega (g_p + g_u u_p)$$

onde utilizamos a seguinte notação para derivada parcial:

$$\frac{\partial f}{\partial x} = f_x$$

Para derivar o problema adjunto, nós introduzimos a variável adjunta ν. Como $F(x, u, p) = 0$ em todo domínio Ω, temos:

$$G(x, u, p) = G(x, u, p) - \int_0^T \int_\Omega \nu F\, dx\, dt$$

e o gradiente $\nabla_p G$ é dado por:

$$\frac{dG}{dp} = \int_0^T \int_\Omega (g_u u_p + g_p) - \int_0^T \int_\Omega \nu \frac{dF}{dp}\, dx\, dt \tag{2.3.4}$$

Usando integração por partes, temos:

$$\int_0^T \int_\Omega \nu \frac{dF}{dp} = \int_0^T \int_\Omega \nu F_p + (\nu F_u - (\nu F_{u_t})_t - (\nu F_{u_x})_x + (\nu F_{u_{xx}})_{xx}) u_p\, dx\, dt + \int_\Omega \nu F_{u_t} u_p|_0^T dx$$
$$\int_0^T \int_{\partial\Omega} (\nu F_{u_x} u_p + \nu F_{u_{xx}} (u_p)_x - (\nu F_{u_{xx}})_x u_p)\, dl$$

Pode-se derivar condições de contorno para ν tais que u_p e $(u_p)_x$ desapareçam ou sejam especificados via condições de Dirichlet. Vamos denotar o último termo por B e definir ν satisfazendo:

$$\nu F_u - (\nu F_{u_t})_t - (\upsilon F_{u_x})_x + (\nu F_{u_x x})_{xx} = g_u \tag{2.3.5}$$

A equação (2.3.2) pode ser reescrita como:

$$\frac{dG}{dp} = \int_0^T \int_\Omega (g_p - \nu F_p) - \int_\Omega (\nu F_{u_t}) u_p|_0^T\, dx - B$$

Com condições iniciais adequadas em $t = T$, (2.3.5) pode ser resolvida no sentido contrário ao tempo. Se a PDE original tem índice não maior que um, no tempo, pode-se escolher ν em $t = T$ tal que $\nu F_{u_t}|_T = 0$. Dessa forma, podemos reescrever (2.3.2):

$$\frac{dG}{dp} = \int_0^T \int_\Omega (g_p - \nu F_p) - \int_\Omega (\nu F_{u_t})|_{t=0} u_p(0)\, dx - B \tag{2.3.6}$$

Para sistemas de EDPs comuns, onde F_{u_t} é uma matriz identidade, a condição final para o problema adjunto (2.3.5) é $\nu = 0$ [15]. Dessa forma, o problema adjunto é dado por:

$$\begin{aligned} \nu F_u - (\nu F_{u_t})_t - (\upsilon F_{u_x})_x + (\nu F_{u_x x})_{xx} &= g_u \\ \nu(T) &= 0 \end{aligned} \tag{2.3.7}$$

2.3.3 Gradiente de J

A partir das condições de fronteira de (2.1.1), nós obtemos:

$$\frac{\partial C}{\partial \eta}(0,\tau) = 0,$$
$$\frac{\partial C}{\partial \eta}(\tilde{K},\tau) = 0,$$

Denotando por P a variável adjunta, temos:

$$\int_0^T \int_{\partial Q} (PF_{C_K} C_\eta + PF_{C_{KK}}(C_\eta)_K - (PF_{C_{KK}})_K C_\eta)\, dl = 0,$$
$$\int_Q (PF_{C_\tau})|_{\tau=0} C_\eta(0)\, dx = 0. \tag{2.3.8}$$

Introduzindo (2.3.8) em (2.3.6), nós obtemos a expressão do gradiente de J:

$$\frac{dJ}{d\eta} = \int_0^T \int_Q (\frac{\partial J}{\partial \eta} - P\frac{\partial F}{\partial \eta})$$

Considerando a Equação de Dupire em (2.1.1) e a expressão de J em (2.1.2), é fácil verificar que:

$$\frac{\partial J}{\partial \eta} = 0$$
$$\frac{\partial F}{\partial \eta} = -\frac{K^2}{2}\frac{\partial^2 C}{\partial K^2}.$$

Dessa forma, podemos obter a expressão final para o gradiente de J:

$$\nabla_\eta J = \frac{dJ}{d\eta} = -\int_0^T \int_Q \frac{K^2}{2} P \frac{\partial^2 C}{\partial K^2}. \tag{2.3.9}$$

Para obter o problema adjunto, cuja solução fornece P, considere a Equação de Dupire (2.1.1) e o problema (2.3.7). Tomando as derivadas sobre as soluções na Equação de Dupire, nós obtemos:

$$\frac{\partial P}{\partial \tau} + \frac{\partial^2}{\partial K^2}\left(\frac{\eta K^2}{2} P\right) - \frac{\partial}{\partial K}(rPK) = -2\sum_{i\in I}(C(K_i,\tau_i) - c_i)\delta_{K_i,\tau_i},$$
$$P(K,\tilde{\tau}) = 0,$$
$$P(\tilde{K},\tau) = 0. \tag{2.3.10}$$

onde $\delta_{K,\tau}$ denota a função delta de Dirac no tempo e preço de exercício dos preços das opções do mercado.

Capítulo 3

Método dos Elementos Finitos

O Método dos Elementos Finitos (MEF) foi inventado por engenheiros por volta de 1950 para resolução de equações diferenciais parciais (EDP) provenientes de problemas mecânicos. Generalizações para outros campos da física e engenharia foram feitas por matemáticos através dos conceitos de formulação variacional e formas fracas das equações diferenciais parciais.
Em finanças, as EDPs provenientes de problemas de precificação de opções podem ter suas soluções aproximadas por quatro classes de métodos numéricos:

- Método das Diferenças Finitas é o mais simples mas apresenta problemas quando a solução requer um grid flexível, não regular, sendo difícil controlar o erro numérico.

- Método dos Volumes Finitos são mais adequados para EDPs hiperbólicas. Em finanças, o método pode ser útil para precificação de opções Asiáticas, onde a PDE se torna hiperbólica próximo ao vencimento.

- Métodos Espectrais são métodos de Galerkin com séries de Fourier de alto grau polinomial. Eles são utilizados para resolução de PDEs com coeficientes constantes que raramente aparecem em finanças.

- Método dos Elementos Finitos parecem, à primeira vista, desnecesssariamente complexos para finanças onde uma grande classe de problemas é unidimensional no espaço. Ainda assim, o método é bastante flexível para grids não regulares e as dificuldades de implementação são, apenas, aparente.

3.1 Formulação Fraca

Formulações fracas constituem uma ferramenta importante em análise de equações matemáticas e permitem a transferência de conceitos de álgebra linear para outros campos como equações diferenciais parciais. Na formulação fraca, uma EDP tem soluções fracas apenas com respeito a certas funções testes.

3.1.1 Alguns Espaços de Funções

Nós denotamos por $L^2(\mathbb{R}_+)$ o espaço de Hilbert das funções quadrado-integrávels sobre $\mathbb{R}_+$, dotado com a norma $\|\nu\|_{L^2(\mathbb{R}_+)} = \left(\int_{\mathbb{R}_+} \nu(x)^2\, dx\right)^{\frac{1}{2}}$ e produto interno $(\nu, w)_{L^2(\mathbb{R}_+)} = \int_{\mathbb{R}_+} \nu(x)w(x)\, dx$. Chamando de $\mathcal{D}(\mathbb{R}_+)$ o espaço das funções suaves com suporte compacto em $\mathbb{R}_+$, nós introduzimos o espaço

$$W = \left\{ w \text{ contínua em } [0, +\infty) : \; w(x) = \int_0^x \phi(s)\, ds, \text{ com } \phi \in L^2(\mathbb{R}_+) \right\}.$$

3.1.2 Formulação Fraca da Equação de Dupire

Consideremos o problema de precificação de opções Européias de compra através da equação de Dupire. É conveniente substituir a variável de tempo t pela variável de tempo para o vencimento $T - t$ transformando, assim, o problema de valor final em problema de valor inicial. Para $S > 0$ e $t \in (0, T]$.

$$\frac{\partial C}{\partial T} - \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} + (r_T - q_T)K\frac{\partial C}{\partial K} + q_T C = 0, \tag{3.1.1}$$

com condição de Dirichlet

$$C(K, 0) = C_0(K), \quad S \in \mathbb{R}_+, \tag{3.1.2}$$

onde C_0 é a função de *payoff*.
Vamos multiplicar (3.1.1) por uma função suave ϕ, de valor real, sobre $\mathbb{R}_+$ e integrar em K sobre $\mathbb{R}_+$. Assumindo que integrações por partes são permitidas, temos:

$$\begin{aligned}
0 =& \frac{d}{dT}\int_{\mathbb{R}_+} C(K,T)\phi(K)\, dK \\
&+ \int_{\mathbb{R}_+} \frac{K^2\sigma^2(K,T)}{2}\frac{\partial C(K,T)}{\partial K}\frac{\partial \phi(K)}{\partial K}\, dK \\
&+ \int_{\mathbb{R}_+} \left(-r_T + \sigma^2(K,T) + K\sigma(K,T)\frac{\partial \sigma(K,T)}{\partial K}\right) K\frac{\partial C(K,T)}{\partial K}\frac{\partial \phi(K)}{\partial K}\, dK \\
&+ r_T \int_{\mathbb{R}_+} C(K,T)\phi(K)\, dK.
\end{aligned}$$

Levando à forma bilinear a_t :

$$\begin{aligned}
a_t(v, w) =& \int_{\mathbb{R}_+} \frac{K^2\sigma^2(K,T)}{2}\frac{\partial v}{\partial K}\frac{\partial w}{\partial K}\, dK \\
&+ \int_{\mathbb{R}_+} \left(-r_T + \sigma^2(K,T) + K\sigma(K,T)\frac{\partial \sigma(K,T)}{\partial K}\right) K\frac{\partial v}{\partial K}\frac{\partial w}{\partial K}\, dK \\
&+ q_T \int_{\mathbb{R}_+} vw\, dK,
\end{aligned} \tag{3.1.3}$$

Temos a **Formulação Fraca de** (3.1.1)**,** (3.1.2) **:** Encontrar $C \in \mathbb{C}^0([0,T], L^2(\mathbb{R}_+)) \cap L^2(0,T;V)$, tal que $\frac{\partial C}{\partial T} \in L^2(0,T;V')$, satisfazendo

$$C(K, 0) = C_0(K) \quad em\, \mathbb{R}_+ \text{ e para todo } T \in (0, T), \tag{3.1.4}$$

$$\forall v \in V, \quad \left(\frac{\partial C}{\partial T}, v\right) + a_t(C, v) = 0. \tag{3.1.5}$$

3.2 Problema Parabólico de Valor de Contorno

Começaremos apresentando o Método dos Elementos Finitos para um problema parabólico genérico. O *framework* é o mesmo para qualquer dimensão do espaço d: para um formulação fraca tomada sobre um espaço de funções infinito-dimensional V, por exemplo

$$V = H^1(\Omega) = \{w \in L^2(\Omega) \ : \ \nabla w \in L^2(\Omega)^d\},$$

consiste na escolha de um subspaço finito-dimensional V_h de V, por exemplo, o espaço das funções afins por partes sobre uma triangularização de Ω, e a resolução do problema com funções teste em V_h no lugar das funções em V. A construção de V_h é feita como se segue:

- O domínio é particionado em células não sobrepostas como, por exemplo, intervalos em espaços unidimensional, triângulos ou quadriláteros em espaços bidimensionais e prismas ou tetraedros em espaços tridimensionais. O conjunto de elementos da partição é chamado triangularização;

- Escolhe-se o grau máximo k da aproximação polinomial nos elementos;

- V_h é construído à partir das funções de V cuja restrição nos elementos são polinomiais de grau inferior a k.

Considere o seguinte:

- Seja Ω um domínio polinomial de $\mathbb{R}^2$ aberto e limitado;

- Seja Γ a fronteira de Ω, assumindo que $\Omega = \Omega_d \cup \Omega_n$, onde a medida unidimensional de $\Omega_d \cap \Omega_n$ É 0;

- Para $x \in \Gamma$, nós denotamos por n o vetor unitário normal a Γ em x, apontando para o exterior;

- Consideramos as seguintes funções suaves:

$$\kappa : \Omega \to (R)^{2\times 2}, \quad \alpha : \Omega \to \mathbb{R}^2, \quad \beta : \Omega \to \mathbb{R}, \quad b : \Gamma_n \to \mathbb{R}.$$

Nós desejamos encontrar $u(x,t)$, resolvendo numéricamente o problema parabólico de condição de contorno

$$\begin{aligned} \partial_t u - \nabla(\kappa\nabla u) - \nabla(\alpha u) + \beta u = \phi, &\qquad em\ \Omega \times (0,T), \\ u(x,0) = u_0(x) &\qquad em\ \Omega, \\ u = g &\qquad sobre\ \Gamma_d \times (0,T), \\ -bu - (\kappa\nabla u)n = f &\qquad sobre\ \Gamma_n \times (0,T). \end{aligned} \tag{3.2.1}$$

Introduzindo a forma bilinear sobre W:

$$a(w,v) = \int_\Omega ((\kappa\nabla w)\nabla v - \nabla(\alpha w)v + \beta wv) + \int_{\Gamma_n} bwv, \tag{3.2.2}$$

obtém-se a formulação variacional de (3.2.1) como: Encontrar $u : u - u_g \in L^2((0,T) : V)$, $u \in \mathbb{C}^0([0,T], L^2(\Omega))$, e $\partial_t u \in L^2((0,T) : V')$, com $u(x,0) = u_0(x)$ e, para todo $t \in (0,T)$,

$$\forall v \in V, \quad (\partial_t u, v) + a(u,v) = \int_\Omega \phi(t)v + \int_{\Gamma_n} f(t)v. \tag{3.2.3}$$

Pode-se demonstrar que, se existe u_g satisfazendo as condições acimas, então o problema variacional tem solução única que satisfaz (3.2.1) no sentido de distribuições [1].

3.2.1 Discretização de Tempo

Considere uma partição do intervalo $[0,T]$ em subintervalos $[t_{m-1}, t_m]$, $1 \leq m \leq M$, tal que $0 = t_0 < t_1 < ... < t_m = T$.
Denotando por Δt_m a diferença $t_m - t_{m-1}$ and por Δt o máximo Δt_m. Assuma que $u_0 \in W$.
A discretização de (3.2.3) por meio do Esquema de Cranck-Nicolson resolve $u_m \in W$, $m = 0, ..., M$, tal que $u^0 = u_0$ e para todo $m = 1, ..., M$, $u^m - u_g(t_m) \in V$, e para todo $v \in V$,

$$\left(\frac{u^m - u^{m-1}}{\Delta t_m}, v\right) + \frac{1}{2}(a(u^m, v) + a(u^{m-1}, v)) = \int_\Omega \phi^{m-1/2} v + \int_{\Gamma_n} f^{m-1/2} v, \tag{3.2.4}$$

3.2.2 Discretização de Tempo e Espaço

A discretização do espaço é obtida através da substituição de W por um subspaço de dimensão finita $W_h \subset W$. Pode-se, por exemplo, tomar W_h como o espaço das funções polinomiais contínuas por partes sobre uma triangularização de Ω. Para um número real positivo h, seja uma partição $\mathcal{T}_h$ de Ω em triângulos não sobrepostos, tais que:

- $\tilde{\Omega} = \cup_{K \in \mathcal{T}_h} K$;
- Para todo $K \neq K'$, dois triângulos de $\mathcal{T}_h$, a medida de $K \cap K'$ é vazio, ou igual a um vértice em comum de K e K', ou igual a uma aresta em comum de K e K';
- Para todo $K \in \mathcal{T}_h$, a medida de $K \cap \Gamma_d (K \cap \Gamma_n)$ é zero ou uma aresta de K;
- $max_{K \in \mathcal{T}_h} diametro(K) = h$.

Neste trabalho, utilizaremos $k = 1$, considerando W_h um subspaço de funções afins por partes. Dessa forma, W_h é um subespaço finito-dimensional de W e V_h é um subespaço finito-dimensional de V.
Assuma que para todo $m = 1, ..., M$, existe a função $u^m_{g,h} \in W_h$, tal que o traço de $u^m_{g,h}$ sobre Γ_d é $g(t_m)$. Supondo que $u_0 \in W_h$, a discretização total, de tempo e espaço, da formulação variacional (3.2.3) consiste na obtenção de $u^m_h \in W_h$, $m = 1, ..., M$, tal que $u^m_h - u^m_{g,h} \in V_h$, e, com $u^0 = u_0$,

$$\left(\frac{u^m_h - u^{m-1}_h}{\Delta t_m}, v_h\right) + \frac{1}{2}(a(u^m_h, v_h) + a(u^{m-1}_h, v_h)) = \int_\Omega \phi^{m-1/2} v_h + \int_{\Gamma_n} f^{m-1/2} v_h, \tag{3.2.5}$$

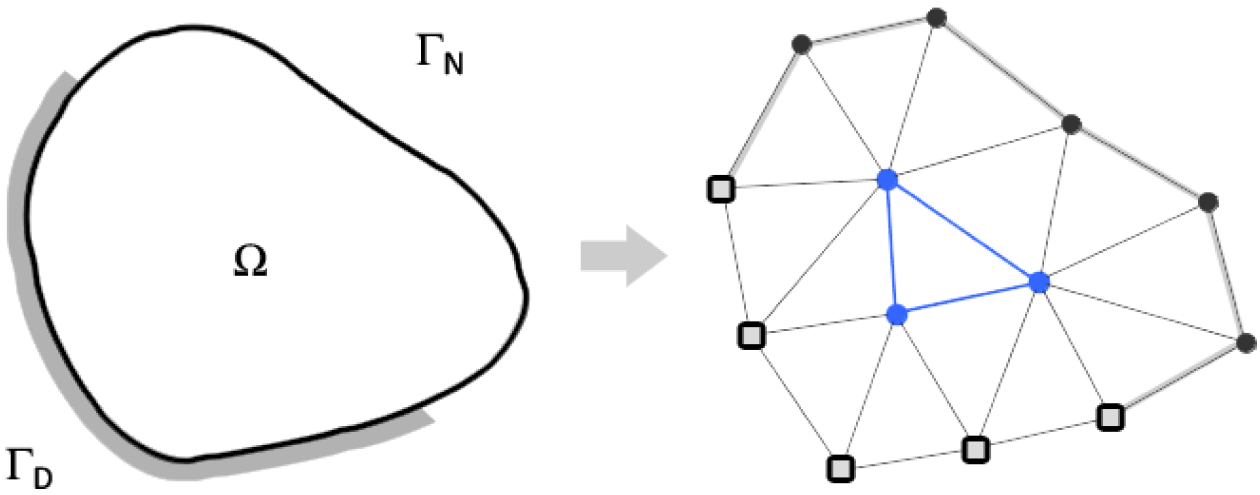

Figura 3.1: Triangularização sobre o domínio Ω

3.2.3 Forma Matricial

Considere $(w_i)_{i=1,\dots,N}$ uma base para o subspaço finito-dimensional V_h. Assim, u_h^m, para qualquer $m = 1, \dots, M$, pode ser escrito como

$$u_h^m(x) = u_{g,h}^m + \sum_{j=1}^{N} u_j^m w_j^m, \tag{3.2.6}$$

Aplicando (3.2.6) em (3.2.5), tomando $v_h = w_i$, obtém-se um sistema linear para $\mathbf{u}_m = (u_j^m)_{j=1,\dots,N}^T$:

$$\mathbf{M}(\mathbf{u}^m - \mathbf{u}^{m-1}) + \frac{\Delta t_m}{2}\mathbf{A}(\mathbf{u}^m + \mathbf{u}^{m-1}) = \mathbf{f}, \tag{3.2.7}$$

onde $\mathbf{M}$ e $\mathbf{A}$ são matrizes em $\mathbb{R}^{N\times N}$ com:

$$\begin{aligned} \mathbf{M}_{i,j} &= \int_\Omega w_i w_j, \\ \mathbf{A}_{i,j} &= a(w_i, w_j), \end{aligned} \tag{3.2.8}$$

e

$$\mathbf{f}_i = \Delta t_m \left(\int_\Omega \phi^{m-1/2} w_i + \int_{\Gamma_n} f^{m-1/2} w_i - \frac{1}{2} a(u_{g,h}^{m-1} + u_{g,h}^m, w_i) \right) - \int_\Omega (u_{g,h}^{m-1} - u_{g,h}^m) w_i$$

Pode-se provar que, se Δt é suficientemente pequeno, $\mathbf{M} + \frac{\Delta t_m}{2}\mathbf{A}$ é inversível, logo (3.2.7) tem solução. [1]

3.2.4 Escolha da Base

Com a escolha de $k = 1$, nós trabalhamos sobre elementos finitos lineares. EM cada triângulo $K \in \mathcal{T}_h$, sejam q_i, $i = 1, 2, 3$, os vértices de K. Definimos, para $x \in \mathbb{R}^2$, as coordenadas baricêntricas de x como solução de

$$\sum_{i=1,2,3} \lambda_i^K(x) q^i = x, \qquad \sum_{i=1,2,3} \lambda_i^K(x) = 1.$$

As coordenadas baricênctricas λ_i^K são funções afins de x. Dessa forma, para qualquer função $v_h \in W_h$, sobre cada triângulo $K \in \mathbb{T}_h$, temos:

$$v_h(x) = v_h(\sum_{i=1,2,3} \lambda_i^K(x) q_i^K) = \sum_{i=1,2,3} v_h(q_i^K)\lambda_i^K(x)$$

Dessa forma, as funções em W_h são unicamente determinadas por seus valores em cada nó de $\mathcal{T}_h$ e as funções de V_h são unicamente definidas pelos valores em cada nó de $\mathcal{T}_h$ não contido na fronteira Γ_d.
Sejam $(q^i) = 1, ..., N$ os nós de $\mathcal{T}_h$ não localizados em Γ_d, e sejam w_i funções únicas em V_h tal que $w_i(q^j) = \delta_{i,j}$ para todo $j = 1, ..., N$. Em cada triângulo K, que contém o vértice q^i, w_i coincide com a coordenada baricêntrica correspondente a K. Dessa forma:

$$v_h = \sum_{i=1}^{N} v_h(q^i) w^i, \tag{3.2.9}$$

mostrando que $(w^i)_{i=1,...,N}$ é uma base de V_h. Essa base, como definida anteriormente, é chamada de base nodal de V_h. Como o suporte das funções bases w^i e w^j se intersectam apenas quando q^i e q^j pertencem ao mesmo triângulo, as matrizes $\mathbf{M}$ e $\mathbf{A}$, construídas com a base nodal, é muito esparsa. Isso reduz a complexidade dos cálculos computacionais na resolução de (3.2.7).

3.3 Discretização da Equação de Dupire

Nós estamos interessados em discretizar a Equação de Dupire para uma opção Européia de compra considerando a ausência de pagamento de dividendos, $q_T = 0$. Vamos denotar $\eta(K,T) = \sigma^2(K,T)$, tomar $\tilde{K}$ e $\tilde{\tau}$ suficientemente grandes e considerar a Equação de Dupire sobre o retângulo $[0, \tilde{K}] \times [0, \tilde{\tau}]$. Sendo S_0 o preço do ativo subjacente, temos

$$\frac{\partial C}{\partial T} - \frac{1}{2}\eta K^2 \frac{\partial^2 C}{\partial K^2} + r_T K \frac{\partial C}{\partial K} = 0, \tag{3.3.1}$$

com condição de Dirichlet

$$\begin{aligned} C(K,0) &= (S_0 - K)^+, \quad S \in \mathbb{R}_+, \\ C(\tilde{K},T) &= 0. \end{aligned} \tag{3.3.2}$$

Nós introduzimos uma partição do intervalo $[0, \tilde{\tau}]$ em subintervalos $[t_{n-1}, t_n]$, $1 \leq n \leq N$, com $\Delta t_i = t_i - t_{i-1}$. Também consideramos uma partição do intervalo $[0, \tilde{K}]$ em subintervalos $w_i = [k_{i-1}, k_i]$, $1 \leq i \leq N_h + 1$, tal que $0 = k_0 < k_1 < ... < k_{N_h} < k_{N_h+1} = \tilde{K}$. Tomamos h_i como o tamanho do intervalo w_i e definimos a triangularização $\mathcal{T}_h$ de $[0, \tilde{K}]$ como o conjunto $\{w_1, ..., w_{N_h+1}\}$. Devemos assumir que o preço do ativo S_0 coincide com algum nó de $\mathcal{T}_h$. O subspaço discreto V_h é definido por

$$V_h = \{v_h \in \mathcal{C}^0([0, \tilde{K}]) : v_h(\tilde{K}) = 0, \forall k \in \mathcal{T}_h, v_{h|k} \text{ é afim}\}.$$

O problema discreto, resultante da aplicação do Esquema de Euler no tempo, é dado por: Encontrar $(C^m)_{0 \leq m \leq N}$, $C^m \in V_h$ satisfazendo

$$C^0(K) = (S_0 - K)^+$$

e para todo m, $1 \leq m \leq N$,

$$\forall v_h \in V_h, \quad \left(C_h^m - C_h^{m-1}, v_h\right) + \Delta t_m a_m(C_h^m, v_h) = 0. \tag{3.3.3}$$

onde a forma bilinear $a_m = a_{t_m}$ é dada por (3.1.3).
Seja $(w^i)_{i=0,\ldots,N_h}$ a base nodal de V_h e $\mathbf{M}$ e $\mathbf{A}^m$, em $\mathbb{R}^{N_h+1\times N_h+1}$, as matrizes referentes à forma matricial de (3.3.3) definidas por $\mathbf{M}_{i,j} = (w^i, w^j)$, $\mathbf{A}^m_{i,j} = a_{t_m}(w^i, w^j)$, para $0 \leq i, j \leq N_h$. Tomando $C^m = (C^m(k_0), \ldots, C^m(k_{N_h}))^T$ e $C^0 = ((S_0 - k_0)^+, \ldots, (S_0 - k_{N_h})^+)^T$, temos que (3.3.3) é equivalente à

$$\mathbf{M}(C^m - C^{m-1}) + \Delta t_m \mathbf{A}^m C^m = 0.$$

reorganizando os termos, obtemos

$$(\mathbf{M} + \Delta t_m \mathbf{A}^m) C^m = \mathbf{M} C^{m-1}. \tag{3.3.4}$$

A escolha da base nodal para V_h torna as matrizes $\mathbf{M}$ e $\mathbf{A}$ tridiagonais pois quando $|i-j| > 1$, a intersecção dos suportes de w^i e w^j tem medida nula. Dessa forma:

$$\begin{aligned}
w^i(K) &= \frac{K - K_{i-1}}{h_i}, & \partial_K w^i(K) &= \frac{1}{h_i} & \forall K \in (K_{i-1}, K_i) \\
w^i(K) &= \frac{K_{i-1} - K}{h_{i+1}}, & \partial_K w^i(K) &= -\frac{1}{h_{i+1}} & \forall K \in (K_i, K_{i+1})
\end{aligned} \tag{3.3.5}$$

E, dessa forma:

$$\begin{aligned}
\int_0^{\tilde{K}} w^{i-1} w^i &= \frac{h_i}{6}, & \int_0^{\tilde{K}} S w^i \partial_K w^{i-1} &= -\frac{K_{i-1}}{6} - \frac{K_i}{3}, \\
\int_0^{\tilde{K}} (w^i)^2 &= \frac{h_i + h_{i+1}}{3}, & \int_0^{\tilde{K}} S w^i \partial_K w^i &= -\frac{h_i + h_{i+1}}{6}, \\
\int_0^{\tilde{K}} (w^0)^2 &= \frac{h_i}{3}, & \int_0^{\tilde{K}} S w^0 \partial_K w^0 &= -\frac{h_i}{6}, \\
\int_0^{\tilde{K}} w^{i+1} w^i &= \frac{h_{i+1}}{6}, & \int_0^{\tilde{K}} S w^i \partial_K w^{i+1} &= \frac{K_{i+1}}{6} + \frac{K_i}{3},
\end{aligned} \tag{3.3.6}$$

Com isso, obtemos a seguinte forma para as matrizes M e A^m

$$\begin{aligned}
A^m_{0,0} &= -\frac{r}{6} h_1, \\
A^m_{i,i} &= \frac{k_i^2 \eta(k_i, t_m)}{2} \left(\frac{1}{h_i} + \frac{1}{h_{i+1}} \right) - \frac{r}{6}(h_i + h_{i+1}), \quad 1 \leq i \leq N, \\
A^m_{i,i-1} &= -\frac{k_i^2 \eta(k_i, t_m)}{2h_i} - \frac{k_i r}{2} + \frac{h_i r}{6}, \quad 1 \leq i \leq N, \\
A^m_{i,i+1} &= -\frac{k_i^2 \eta(k_i, t_m)}{2h_{i+1}} + \frac{k_i r}{2} + \frac{h_{i+1} r}{6}, \quad 0 \leq i \leq N-1.
\end{aligned}$$

$$
\begin{aligned}
M_{0,0} &= -\frac{h_i}{6}, \\
M_{i,i} &= \frac{h_i + h_{i+1}}{3}, \quad 1 \leq i \leq N, \\
M_{i,i-1} &= \frac{h_i}{6}, \quad 1 \leq i \leq N, \\
M_{i,i+1} &= \frac{h_{i+1}}{6}, \quad 0 \leq i \leq N-1.
\end{aligned}
$$

3.4 Discretização do Estado Adjunto

Considerando, agora, o problema (2.3.10), o estado adjunto discreto $(\mathbf{P}^m)_{0\leq m\leq N}$ é solução para $\mathbf{P}^N = 0$, e

$$M(P^m - P^{m+1}) + \Delta t_m A^{T,m} P^m = 2\sum_{i\in I} G^m,$$

onde

$$G^m = \left(0, \sum_{i\in I} \delta_{k_1=K_i}(C_1^m - c_i), ..., \sum_{i\in I} \delta_{k_j=K_i}(C_j^m - c_i), ...\right)^T,$$

onde $A^{T,m}$ denota a transposta da matriz A^m.
Considere uma variação $\delta\eta$ e $\delta C = C(\eta + \delta\eta) - C(\eta)$. Da Equação de Dupire, temos

$$
\begin{aligned}
\frac{\partial \delta C}{\partial T} - \frac{\eta K^2}{2}\frac{\partial^2 \delta C}{\partial K^2} + rK\frac{\partial \delta C}{\partial K} &= \frac{\delta\eta K^2}{2}\frac{\partial^2 C}{\partial K^2}, \\
\delta C(K, 0) &= 0, \\
\delta C(0, \tau) &= 0, \\
\delta C(\tilde{k}, \tau) &= 0.
\end{aligned} \tag{3.4.1}
$$

Assim, a variação de $\mathbf{C}^m$ satisfaz $\delta C^0 = 0$ e

$$M(\delta C^m - \delta C^{m-1}) + \Delta t_m A^m \delta C^m = -\Delta t_m \delta A_m C^m,$$

onde a variação da matriz A^m, causada por uma variação $\delta\eta$, é dada por

$$
\begin{aligned}
\delta A_{0,0}^m &= 0, \\
\delta A_{i,i}^m &= \frac{k_i^2 \eta(k_i, t_m)}{2}\left(\frac{1}{h_i} + \frac{1}{h_{i+1}}\right), \quad 1 \leq i \leq N, \\
\delta A_{i,i-1}^m &= -\frac{k_i^2 \eta(k_i, t_m)}{2h_i}, \quad 1 \leq i \leq N, \\
\delta A_{i,i+1}^m &= -\frac{k_i^2 \eta(k_i, t_m)}{2h_{i+1}}, \quad 0 \leq i \leq N-1.
\end{aligned}
$$

Dessa forma, podemos reescrever (2.3.9) como

$$\nabla_\eta J = \sum_{i=0}^{N_h}\sum_{j=0}^{N} \Delta t_m P^{T,m} \delta A_m C^m \tag{3.4.2}$$

Observe que, dada a condição final do problema adjunto, $P^n = 0$, nós devemos estender a dimensão do problema adjunto no tempo para a utilização de todos os valores dos preços de opções de compra observados no mercado.

Capítulo 4

Problema de Minimização

Neste capítulo nós trataremos dos métodos necessários à minimização do problema de mínimos quadrados da calibragem de volatilidade local. Inicialmente, apresentaremos algumas condições de convergência e uma estratégia de escolha do comprimento do passo das iterações baseado nessas condições. Por fim, apresentaremos um método quasi-Newton, que não requer o cálculo direto da Hessiana da função objetivo, e uma adaptação do conhecido método BFGS, o LBFGS, que limita a memória computacional utilizada.

4.1 Métodos Line Search

Considere um método de otimização que, dado uma direção de busca p_k, decide quão distante procurar ao longo dessa direção. Podemos definir um valor escalar positivo α_k, chamado de comprimento do passo, e escrever a equações das iterações por:

$$x_{k+1} = x_k + \alpha_k p_k. \tag{4.1.1}$$

Em geral, a direção de busca toma a forma:

$$p_k = -B_k^{-1} \nabla f_k,$$

onde B_k é uma matriz simétrica não singular. Enquanto no método de Newton B_k toma a forma da Hessiana $\nabla^2 f(x_k)$, nos métodos de quasi-Newton B_k é uma aproximação para a Hessiana que é atualizada a cada iteração do algoritmo.
Métodos de Line Search buscam valores adequados de α_k que satisfaçam certas condições necessárias à convergência do algoritmo de minimização.

4.1.1 Comprimento do Passo

A escolha do comprimento do passo α_k em cada iteração (4.1.1) de um método de otimização é de importância central para a eficiência do algoritmo. Nós procuramos por um comprimento de passo que forneça uma relevante redução da função objetivo f, mas não podemos podemos investir muito tempo e custo computacional na busca por esse valor. Considere uma função escalar $\phi_k : \mathbb{R} \to \mathbb{R}$ definida por:

$$\phi_k(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0. \tag{4.1.2}$$

Um minimizador global poderia ser escolhido de tal forma que

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}^+} \phi(\alpha),$$

mas em geral, a escolha desse minimizador exige muito tempo computacional envolvendo muitas avaliações da função objetivo f e, possivelmente, de seu gradiente ∇f. Métodos mais práticos realizam uma busca inexata procurando equilibrar baixo custo com a obtenção de valores que tragam reduções adequadas à f.
Algoritmos de de busca do comprimento de passo agem em duas etapas. Na primeira, busca-se um intervalo de comprimento de passo que satisfaça condições de convergência enquanto na segunda etapa escolhe-se um passo adequado, dentro do intervalo obtido, que forneça uma redução relevante de f.

4.1.2 Condições de Wolfe

A convergência de um método de otimização está, também, associada ao comprimento do passo em cada iteração. Uma primeira condição simples, que surge intuitivamente, seria a imposição de que o comprimento do passo α_k gere uma redução de f, isto é, $f(x_k + \alpha_k p_k) < f(x_k)$. Não é difícil notar, porém, que essa única condição não garante a convergência do método uma vez reduções insuficientes em f levam a progressos computacionalmente insignificantes nas iterações. Dessa forma, nós devemos garantir uma condição de redução suficientemente grande para f, em cada iteração.
Uma condição popularmente utilizada em métodos line search, define valor para o comprimento do passo α_k baseado em um valor mínimo de redução para a função objetivo f. Essa condição, conhecida como Condição de Armijo, segue a seguinte desigualdade:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{4.1.3}$$

definida para uma constante $c_1 \in (0, 1)$. Na pratica, c_1 é escolhido para ser pequeno, por exemplo $c_1 = 10^{-4}$[20].
Observe que (4.1.3) é satisfeita para valores arbitrariamente pequenos de α_k. Dessa forma, apenas a Condição de Armijo não garante a convergência do método de minimização. Para definir um limite inferior de valores inaceitavelmente pequenos de α_k, nós introduzimos a chamada Condição de Curvatura, que segue a seguinte desigualdade:

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \tag{4.1.4}$$

definida para uma constante $c_2 \in (c_1, 1)$. Note que $\phi'(\alpha) = \nabla f(x_k + \alpha p_k)^T p_k$. Dessa forma, se a inclinação $\phi'(\alpha)$ é significantemente negativa, nós podemos reduzir f através da direção de descida p_k. Por outro lado, se a inclinação é pouco negativa ou mesmo positiva, nenhum avanço relevante pode ser feito em relação e minimização de f e o método line search deve ser finalizado. Tipicamente, $c_2 = 0.9$, quando a direção de descida p_k é obtida por um método quasi-Newton ou Newton[20].
Juntas, as Condições de Curvatura e Armijo são conhecidas como Condições de Wolfe:

$$\begin{aligned} f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \\ \nabla f(x_k + \alpha_k p_k)^T p_k &\geq c_2 \nabla f_k^T p_k, \end{aligned} \tag{4.1.5}$$

com $0 < c_1 < c_2 < 1$.

4.1.3 Algoritmo de Escolha do Comprimento do Passo

A partir deste ponto, nós passaremos a considerar o problema de minimizar uma função escalar

$$\phi(\alpha) = f(x_k + \alpha p_k),$$

e encontrar um comprimento de passo α_k que satisfaça as Condições de Wolfe. Para confinar os valores de α_k em um intervalo positivo, nós consideramos que p_k é uma direção de descida, isto é, $\phi'(0) < 0$.
Para funções não lineares, geralmente utiliza-se um procedimento iterativo para a escolha do comprimento do passo da iteração. Esse procedimento merece atenção especial pois tem importância central na eficiência dos métodos de otimização.
Os procedimentos de escolha de passo podem utilizar a apenas informação da função objetivo ou, adicionalmente, de seu gradiente. Procedimentos que utilizam apenas a função objetivo são pouco eficientes pois precisam iterar até que a busca por um minimizador seja reduzida a um pequeno intervalo. Por outro lado, com o uso da informação do gradiente da função objetivo, pode-se afirmar se um valor candidato à passo da iteração satisfaz as Condições de Wolfe. Em geral, a primeira escolha de α_k satisfaz as condições e o procedimento de procura pelo comprimento do passo não precisa ser acionado. Para métodos de Newton e quasi-Newton, a valor $a_0 = 1$ sempre deve ser usado como valor inicial para o passo. Essa condição garante que o valor sempre será usado quando as Condições de Wolfe forem satisfeitas levando os métodos a apresentarem rápidas taxas de convergência. Todo procedimento de busca do comprimento do passo requer uma estimativa inicial a_0 e, então, gera uma sequência $\{a_i\}$ de valores que termina com as Condições de Wolfe satisfeitas ou determina que tal comprimento não existe. Tipicamente, os procedimentos dessa classe realizam duas etapas: uma etapa de agrupamento, que encontra um intervalo $[\tilde{a}, \tilde{b}]$ contendo valores aceitáveis de comprimento do passo, e uma fase de seleção que determina um valor final dentro do intervalo. A etapa de agrupamento começa com uma valor inicial α_1 que vai sendo incrementado, a cada iteração, até um valor, ou intervalo de valores, aceitável. Se um valor aceitável é encontrado, o algoritmo termina e retorna o comprimento de passo satisfazendo as condições de Wolfe. Se o algoritmo encontra um intervalo, a etapa de seleção se inicia sobre esse intervalo.
No Algoritmo 1, nós apresentamos a etapa de agrupamento do método Line Search discutido.

Algoritmo 1: Line Search: Etapa de Agrupamento

```
α_0 ← 0, choose α_max > 0 and α_1 ∈ (0, α_max);
i ← 1;
repeat
    Evaluate φ(α_i);
    if φ(α_i) > φ(0) + c_i α_i φ'(0) or (φ(α_i) ≥ φ(α_{i-1}) and i ≥ 1) then
        α ← selecao(α_{i-1}, α_i) and stop;
    end
    Evaluate φ'(α_i);
    if φ'(α_i) ≥ c_2 φ'(0) then
        α ← α_i and stop;
    end
    if φ'(α_i) ≥ 0 then
        α ← selecao(α_i, α_{i-1}) and stop;
    end
    Choose α_{i+1} ∈ (α_i, α_max);
    i ← i + 1;
until;
```

A última etapa do algoritmo realiza uma extrapolação para encontrar o próximo valor candidato α_{i+1}. É importante notar que os valores devem crescer suficientemente rápido para

alcançar o limite superior α_{max} em um número finito de iterações.
A etapa de seleção é apresentada no Algoritmo 2. Os parâmetros de entrada seguem a forma **selecao**$(\alpha_{lo}, \alpha_{hi})$, onde

- O intervalo definido por α_{lo} e α_{hi} contem comprimentos de passo que satisfazem as Condições de Wolfe;
- α_{lo} é, entre todos os valores do intervalo, aquele que gera o menor valor da função objetivo;
- α_{hi} é escolhido tal que $\phi'(\alpha_{lo})(\alpha_{hi} - \alpha_{lo}) < 0$.

Cada iteração do algoritmo gera um valor α_j entre α_{lo} e α_{hi} e, então, substitui um dos extremos do intervalo por esse novo valor. Se esse novo valor satisfaz as Condições de Wolfe, o algoritmo se encerra retornando α_j. A primeira parte do algoritmo, que realiza uma interpolação para

Algoritmo 2: Line Search: selecao

```
repeat
    Interpolate to find a trial step lenght α_j between α_lo and α_hi;
    Evaluate φ(α_j);
    if φ(α_j) > φ(0) + c_i α_j φ'(0) or α(α_j) ≥ φ(α_lo) then
        α_hi ← α_j;
    end
    else
        Evaluate φ'(α_j);
        if φ'(α_j) ≥ c_2 φ'(0) then
            α ← α_j and stop;
        end
        if φ(α_j)(α_hi − α_lo) ≥ 0 then
            α_hi ← α_lo;
        end
        α_lo ← α_j;
    end
until;
```

encontrar α_j, deve garantir que o novo valor não esteja muito próximo dos valores extremos do intervalo. Na prática, o algoritmo pode fazer uso de propriedades de interpolação polinomial para encontrar candidatos adequados do próximo comprimento do passo.
Nós podemos utilizar as informações dos valores da função $\phi(\alpha)$ e da sua derivada para criar um procedimento de interpolação na busca de α_j que satisfaça a Condição de Armijo. O procedimento gera uma sequência decrescente de valores α_i tais que α_i não é muito menor do que seu predecessor α_{i-1}. Nós podemos podemos utilizar os valores de $\phi(\alpha_{lo})$, $\phi'(\alpha_{lo})$ e $\phi(\alpha_{hi})$ para construir uma aproximação quadrática

$$\phi_q(\alpha) = \frac{(\phi(\alpha_{hi}) - \phi(\alpha_{lo}) - (\alpha_{hi} - \alpha_{lo})\phi'(\alpha_{lo}))}{(\alpha_{hi} - \alpha_{lo})^2}(\alpha - \alpha_{lo})^2 + \phi'(\alpha_{lo})(\alpha - \alpha_{lo}) + \phi(\alpha_{lo}), \quad (4.1.6)$$

O valor candidato do novo comprimento α_j será definido como o minimizador de (4.1.6)

$$\alpha_j = \frac{(\alpha_{hi} - \alpha_{lo})^2\phi'(\alpha_{lo})}{2[\phi(\alpha_{hi}) - \phi(\alpha_{lo}) - (\alpha_{hi} - \alpha_{lo})\phi'(\alpha_{lo})]} + \alpha_{lo}$$

4.2 Quasi-Newton LBFGS

Métodos quasi-Newton, assim como o método do gradiente simples, necessitam apenas do cálculo da função objetivo e seu gradiente em cada iteração. Através de mudanças dos valores do gradiente, essa classe de métodos constroem aproximações para a Hessiana da função objetivo, produzindo convergência superlinear. A vantagem sobre o método do gradiente simples é enorme e, como as segundas derivadas não são utilizas explicitamente, métodos quasi-Newton são, geralmente, mais eficientes que o método de Newton.

4.2.1 Método BFGS

O algoritmo quasi-Newton mais popular é o Método BFGS, nomeado por seus criadores Broyden, Fletcher, Goldfarb e Shanno. Considere um modelo quadrático da função objetivo na iteração k

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p.$$

B_k é uma matriz simétrica $n \times n$ positiva definida que será atualizada a cada iteração. Observe que $m_k(0) = f_k$ e $\nabla m_k(0) = \nabla f_k$. O minimizador desse modelo quadrático pode ser facilmente obtido algebricamente e tem a forma:

$$p_k = -B_k^{-1} \nabla f_k,$$

Utilizando o minimizador como direção de busca, a equação das iterações pode ser escrita como

$$x_k + 1 = x_k + \alpha_k p_k,$$

onde α_k é o comprimento de passo satisfazendo as Condições de Wolfe (4.1.5). No lugar de recalcular B_k a cada iteração, pode-se atualizar a matriz, de uma maneira simples, através das medidas de curvatura durante os últimos passos. Nós podemos impor que o gradiente de m_{k+1} coincida com os valores do gradiente de f nas iterações x_k e x_{k+1}. Como $\nabla m_{k+1}(0) = \nabla f_{k+1}$, a segunda condição é satisfeita trivialmente. A primeira condição pode ser escrita como

$$\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k = \nabla f_k.$$

Dessa forma, temos

$$B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k. \tag{4.2.1}$$

Podemos simplificar a notação definindo os vetores

$$s_k = x_{k+1} - x_k = \alpha_k p_k \qquad y_k = \nabla f_{k+1} - \nabla f_k,$$

Assim, (4.2.1) se torna

$$B_{k+1} s_k = y_k. \tag{4.2.2}$$

Dadas as mudanças de valores s_k e y_k, (4.2.2) define um mapeamento, através uma matriz positivo definida B_{k+1}, de s_k em y_k. Isso é possível apenas se s_k e y_k satisfizerem a condição de curvatura

$$s_k^T y_k > 0. \tag{4.2.3}$$

Quando f é fortemente convexa, (4.2.3) será satisfeita para quaisquer dois pontos x_k e x_{k+1}. Para funções não convexas, a escolha do comprimento do passo satisfazendo as Condições de Wolfe garante que $\nabla f_{k+1}^T s_k \geq c_2 \nabla f_k^T s_k$ e, por isso

$$y_k^T s_k \geq (c_2 - 1) \alpha_k \nabla f_k^T p_k > 0.$$

Quando as Condições de Wolfe são satisfeitas, a equação (4.2.2) sempre tem solução B_{k+1}. De fato, (4.2.2) admite infinitas soluções. Para determinar B_{k+1} unicamente, nós devemos impor condições adicionais tais que entre todas as matrizes simétricas positivo definidas, B_{k+1} esteja, em algum sentido, próxima ao valor de B_k. Dessa forma, devemos resolver o problema

$$\begin{aligned} &\min_B \|B - B_k\| \\ &\text{sujeito a } B = B^T, \quad Bs_k = y_k, \end{aligned} \tag{4.2.4}$$

onde s_k e y_k satisfazem (4.2.3) e B_k é simétrica e positivo definida. Diferentes normas matriciais podem ser usadas e dão origem à diferentes métodos quasi-Newton. Para derivar o método BFGS, nós podemos impor as mesmas condições à matriz inversa da aproximação da Hessiana, H_k, sendo $H_k = B_k^{-1}$.
O problema (4.2.4) toma a forma análoga

$$\begin{aligned} &\min_H \|H - H_k\| \\ &\text{sujeito a } H = H^T, \quad Hy_k = s_k, \end{aligned} \tag{4.2.5}$$

Considerando a norma de Frobenius ponderada

$$\|A\|_W \equiv \|W^{\frac{1}{2}} A W^{\frac{1}{2}}\|_F,$$

onde $\|C\|_F = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^2$ e W é qualquer matriz satisfazendo $Ws_k = y_k$, a solução única H_{k+1} para o problema de minimização (4.2.5) é dada por

$$\text{(BFGS)} \quad H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \tag{4.2.6}$$

com $\rho_k = \frac{1}{y_k^T s_k}$.

4.2.2 Método L-BFGS

Cada passo do método BFGS tem a forma

$$x_{k+1} = x_k - \alpha_k H_k \nabla f_k,$$

onde H_k é atualizado segundo (4.2.6).
Como a matriz inversa da aproximação da Hessiana H_k será, geralmente, densa, o custo computacional de armazenamento e manipulação dessa matriz cresce a medida que o número de parâmetros de otimização aumenta. Para lidar com essa dificuldade, nós podemos armazenar implicitamente uma versão modificada de H_k através de um certo número de pares de vetores $\{s_i, y_i\}$. O produto $H_k \nabla f_k$ pode ser obtido através de produtos internos e somas envolvendo ∇f_k e os pares $\{s_i, y_i\}$ armazenados. Após cada cálculo de H_k, o novo par $\{s_k, y_k\}$ é armazenado e o mais antigo par do conjunto $\{s_i, y_i\}$ é descartado.
Considere que o algoritmo armazena m pares de vetores $\{s_i, y_i\}$, $i = k-m, ..., k-1$. Nós devemos tomar uma aproximação inicial para a Hessiana H_k^0, que pode variar a cada iteração e, utilizando (4.2.6), verificamos que no método L-BFGS a aproximação H_k pode ser obtida pela fórmula

$$\begin{aligned} H_k =& (V_{k-1}^T ... V_{k-m}^T) H_k^0 (V_{k-m} ... V_{k-1}) \\ &+ \rho_{k-m} (V_{k-1}^T ... V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} ... V_{k-1}) \\ &+ \rho_{k-m+1} (V_{k-1}^T ... V_{k-m+2}^T) s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} ... V_{k-1}) \\ &+ ... \\ &+ \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{4.2.7}$$

onde $V_k = I - \rho_k y_k s_k^T$. Da expressão (4.2.7), podemos derivar um procedimento para cálculo do produto $H_k \nabla f_k$. Esse procedimento é dado pelo Algoritmo 3.
A cada iteração, nos deparamos com o problema de escolha da aproximação inicial H_k^0. Um

Algoritmo 3: L-BFGS:Dupla Recursão

$q \leftarrow \nabla f_k$;
for $i = k-1, k-1, ..., k-m$ **do**
  $\alpha_i \leftarrow \rho_i s_i^T q$;
  $q \leftarrow q - \alpha_i y_i$;
end
$r \leftarrow H_k^0 q$;
for $i = k-m, k-m+1, ..., k-1$ **do**
  $\beta \leftarrow \rho_i y_i^T r$;
  $r \leftarrow r + s_i(\alpha_i - \beta)$;
end
stop with result $H_k \nabla f_k = r$.

método para escolha que se mostra efetivo na prática é considerar $H_k^0 = \gamma_k I$, onde

$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}.$$

O algoritmo final quasi-Newton L-BFGS é apresentado no Algoritmo 4. Durante as primeiras

Algoritmo 4: L-BFGS

Choose starting point x_0, integer $m > 0$;
$k \leftarrow 0$;
repeat
  Choose H_k^0;
  Compute $p_k \leftarrow -H_k \nabla f_k$ from (3);
  Compute $x_{k+1} \leftarrow x_k + \alpha_k p_k$, where α_k is chosen to satisfy the Wolfe conditions;
  if $k > m$ **then**
    Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage;
  end
  Compute and save $s_k \leftarrow x_{k+1} - x_k,\ y_k = \nabla f_{k+1} - \nabla f_k$;
  $k \leftarrow k+1$;
until *convergence*;

$m-1$ iterações, os métodos BFGS e L-BFGS são equivalentes se a escolha para a aproximação inicial for a mesma em ambos os métodos.

Capítulo 5

Resultados Numéricos

Nesse capítulo, nós apresentaremos os detalhes de implementação e resultados numéricos obtidos com a solução do problema de calibragem da volatilidade local. Foram utilizados dados sintéticos, gerados por simulação, e dados reais de opções americanas de compra sobre futuro de *commodities*. Começaremos por apresentar os resultados referentes aos dados simulados onde, também, cabe uma pequena discussão em relação à metodologia de geração de dados e ruídos. Por fim, apresentaremos os resultados obtidos com a utilização de dados reais do mercado de opções.

5.1 Simulação

A etapa de resolução do problema de calibragem para dados simulados é importante para a validação de implementação e observação de características do problema. Nesta etapa, os dados referentes à negociação, no mercado, das opções europeias de compra são gerados à partir de uma superfície de volatilidade local arbitrariamente definida. O problema de calibragem, nesse caso, consiste na obtenção de uma superfície de volatilidade local suficientemente próxima àquela utilizada na geração de dados.

5.1.1 Geração de Dados

Não é incomum que, em testes de modelos teóricos exatos ou aproximados, sejam utilizados dados sintéticos. Seja por falta de dados reais ou pela conveniência na criação dos próprios dados, a simulação de dados sintéticos merece atenção e a falta de cuidado nesta etapa pode, eventualmente, produzir resultados que não concordam com a realidade. A primeira preocupação que deve ser tomada na criação de dados sintéticos para a resolução de problemas inversos diz respeito ao Crime Inverso. Essa expressão foi, pela primeira vez, utilizada por Colton e Kress em [5] e consiste na inversão trivial de problemas finito-dimensional. Dessa forma, os dados sintéticas deveriam ser gerados através de um problema direto que não mantivesse conexão com o problema inverso testado. Para evitar o crime inverso trivial, aquele no qual o modelo do problema direto usado para a geração de dados é utilizado, exatamente com a mesma forma, no problema inverso, nós utilizamos um *grid* mais fino para a simulação de dados do que o utilizado para a resolução do problema inverso[18]. Na Figura 5.1, nós apresentamos os *grids* utilizados no presente trabalho.

A superfície de volatilidade local utilizada para a simulação dos dados é apresentada na Figura 5.2. Nós adicionamos um componente temporal de decaimento de forma que a volatilidade local tenda a decrescer com a aproximação do vencimento. Além disso, a superfície apresenta

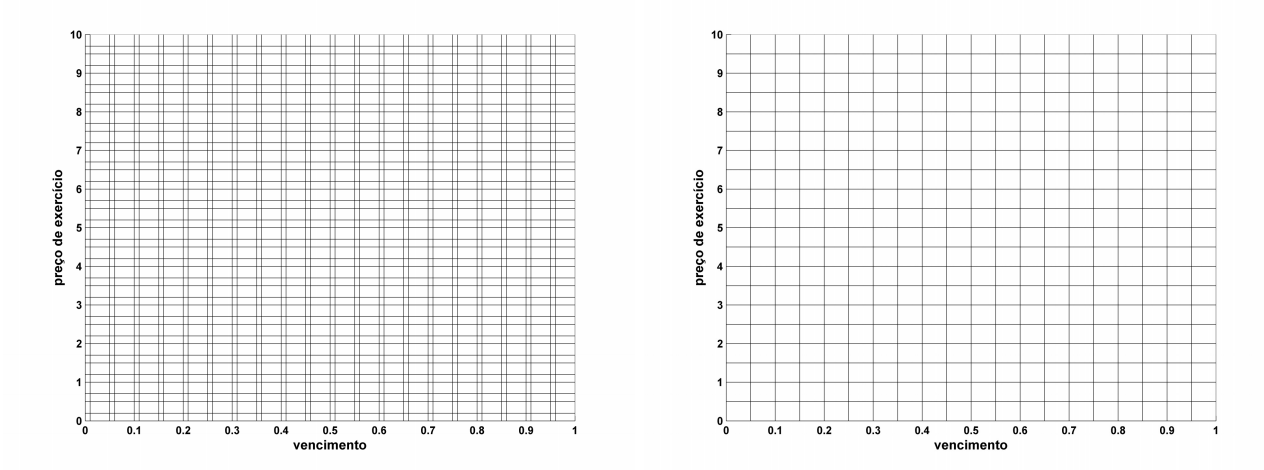

Figura 5.1: *Grids* utilizados para a simulação de dados e solução do problema inverso

convexidade ao longo do eixo do preço de exercício, nos preços de exercício próximos ao valor do ativo subjacente, exibindo uma característica semelhante ao *smile* encontrado nas curvas de volatilidade implícita.

O parâmetro de regularização utilizado, foi escolhido através do Critério da Discrepância

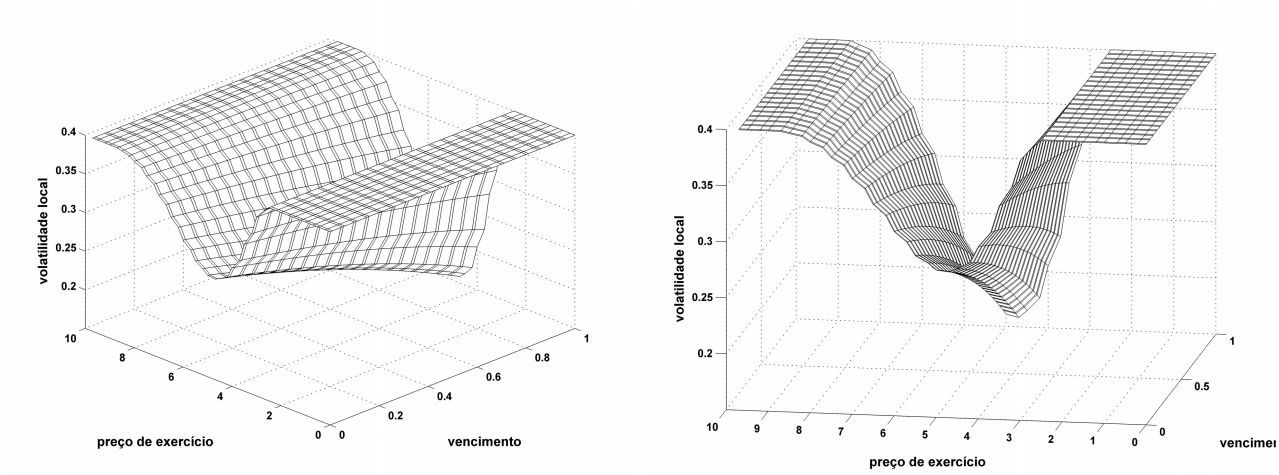

Figura 5.2: Superfície de Volatilidade Local utilizada para simulação de dados

de Morozov. O método de escolha consiste em resolver várias instâncias do problema inverso decrementando o parâmetro de regularização, à partir de um valor escolhido como máximo inicial, em cada instância. Pode-se utilizar um método numérico, como o método da bisseção, para a obtenção do parâmetro de regularização ótimo δ_{opt} que gera uma discrepância igual ao estimador de ruído do problema. Nós, no entanto, utilizamos uma metodologia diferente para a escolha. Resolveu-se diversas instâncias problema inverso com pequenos decrementos do parâmetro de regularização e o valor final foi escolhido como o último parâmetro que gera uma discrepância superior ao estimador de ruído do problema. Vale notar que as instâncias do problema inverso resolvidas na escolha do parâmetro de regularização deve ter o mesmo *grid* do problema inverso original. Isso se deve ao fato do problema não ser independente de escala, gerando valores diferentes da função objetivo otimizada para diferentes *grids* e, conseqüentemente, diferentes parâmetros de regularização.

Para dados sintéticos, o ruído do problema pode ser obtido com exatidão. O ruído referente

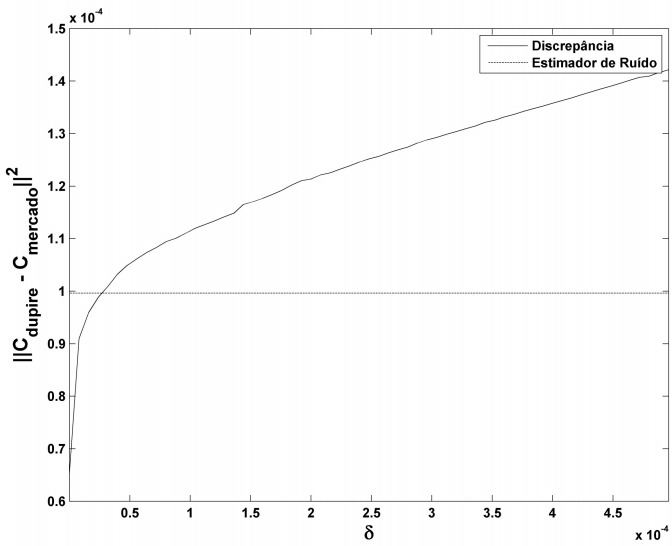

Figura 5.3: Curva de discrepância do problema simulado

à discretização, gerado pela a utilização de *grids* distintos na geração de dados e resolução do problema, é obtido pela diferença dos dados sintéticos com dados gerados pelo problema direto e mesmo *grid* da calibragem da volatilidade local. Adicionalmente, foi inserido ruído branco gaussiano com distribuição normal de média zero e desvio padrão 10^{-6}, gerando a superfície de ruído apresentada na Figura 5.4.

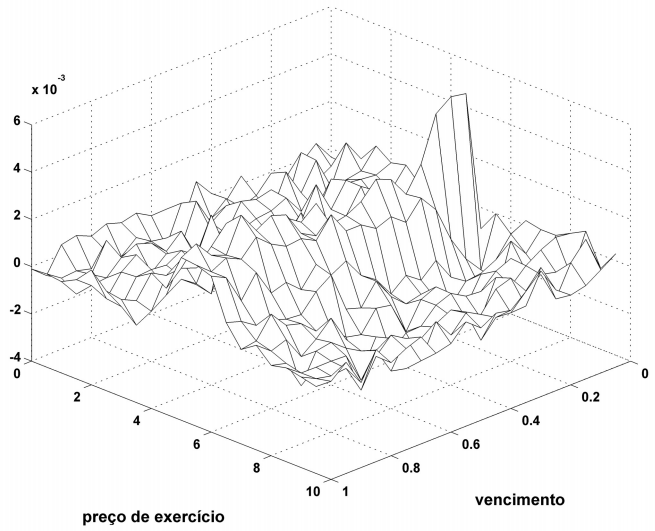

Figura 5.4: Ruído inserido nos dados do problema

5.1.2 Volatilidade Local

Afim de apresentar os efeitos que os diferentes parâmetros de regularização causam na solução do problema inverso, nós apresentamos, na Figura 5.5, uma solução sub-regularizada, com parâmetro de regularização inferior a δ_{opt}, uma solução super-regularizada, com parâmetro

de regularização superior a δ_{opt}, e uma solução regularizada com parâmetro de regularização adequado. O funcional de Tikhonov utilizado foi a norma quadrática da Hessiana da variância, $\|\nabla\eta(K,T)\|^2$, isto é, regularização de primeira ordem. O cálculo da Hessiana foi realizado através de diferenças finitas. Os parâmetros utilizados na simulação são apresentados na Tabela 5.1.

O erro da solução otimizada em relação à superfície de volatilidade esperada é apresentado na

Parâmetro	$\tilde{K}$	$\tilde{T}$	S_0	r	δ
Valor	10	1	5	0.1	4×10^{-5}

Tabela 5.1: Parâmetros de simulação

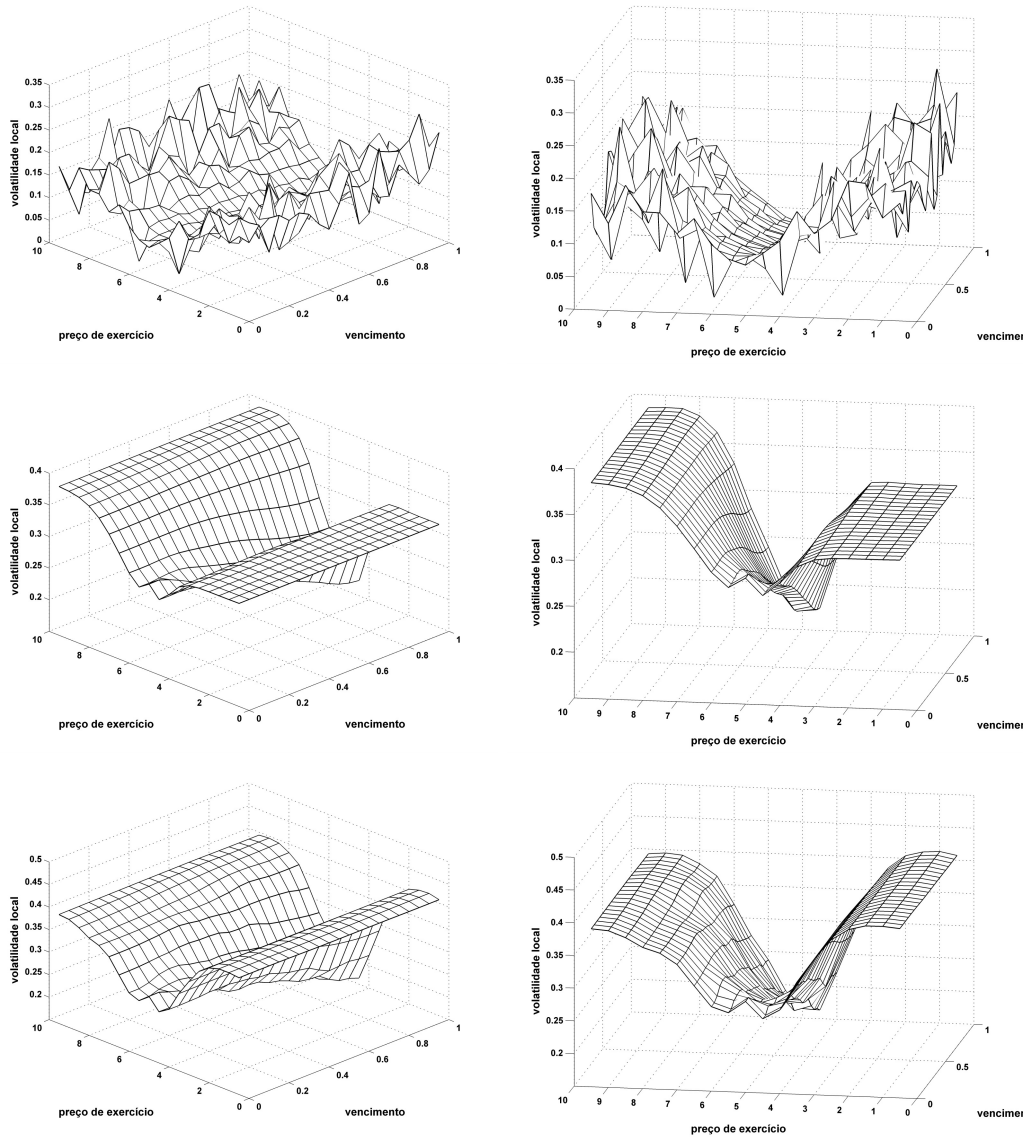

Figura 5.5: Solução do problema inverso sub-regularizada, super-regularizada e regularizada

Figura 5.6

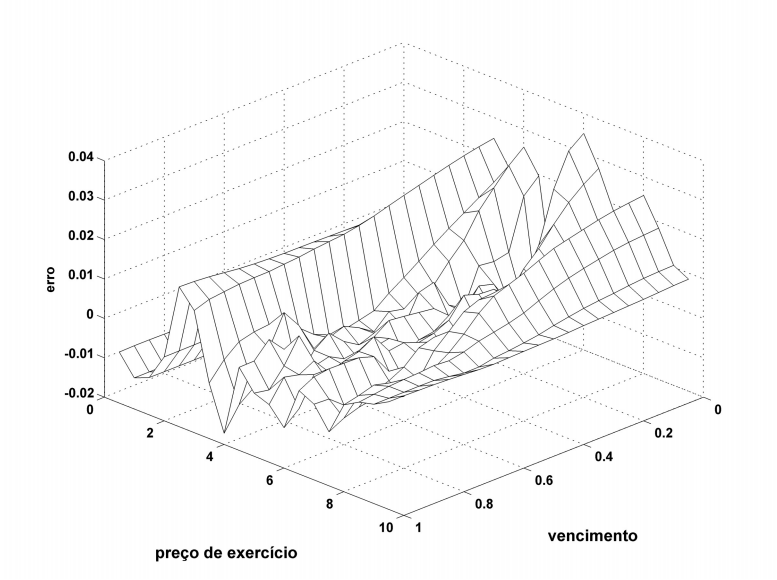

Figura 5.6: Erro da solução do problema inverso simulado

5.2 Dados Reais

Os dados reais utilizados correspondem aos negócios de opções americanas de compra, sobre o contrato futuro de óleo cru *West Texas Intermediate* (WTI), entre 05/09/2013 e 16/11/2013. Nesta seção nós trataremos do pré-processamento dos dados, para adequação ao problema de calibragem aqui tratado, e por fim, apresentaremos os resultados do problema de calibragem da superfície local.

5.2.1 Preparação dos Dados

As opções sobre *commodities* negociadas no mercado são, comumente, do tipo americana. Para adequar os dados reais de negociação ao nosso problema, devemos obter uma aproximação para o preço das opções europeias de compra à partir da versão americana. Em seu artigo de 1987, Barone, Adesi e Whaley propõem uma aproximação para o preço das opções americanas sobre *commodities* à partir das opções europeias. Nós utilizaremos essa aproximação para a obtenção dos preços de opções europeias.

Vamos considerar um modelo geral de precificação de opções sobre *commodities*. As hipóteses assumidas são consistentes com as introduzidas por Black e Scholes [4]. A taxa livre de risco r e o custo de carrego das *commodities* b são considerados constantes. Para opções dobre ações que não pagam dividendos, o custo de carrego é igual à taxa livre de risco e o preço para opções de compra são iguais para o tipos americanos e europeus. Esse não é o caso geral para *commodities*.

Sob a hipótese de não-arbitragem, a relação entre o contrato futuro e a *commodity* correspondente é dada por

$$F = Se^{bT}, \tag{5.2.1}$$

onde F e S são os preços do futuro e *spot*, respectivamente, e T é o tempo para o vencimento do contrato futuro.

Uma segunda hipótese, comumente adotada, é a de que o preço da *commodity* segue uma equação diferencial estocástica

$$dS = S(\alpha dt + \sigma dW), \tag{5.2.2}$$

onde α é valor esperado para a mudança instantânea do preço, σ é o desvio padrão instantâneo e W é um processo de Wiener. De (5.2.1) e (5.2.2), nós obtemos a seguinte equação diferencial estocástica para o movimento de preço do contrato futuro de *commodity*

$$dF = F(\alpha - b)dt + \sigma dW. \tag{5.2.3}$$

Assumindo a possibilidade de um *hedge* livre de risco entre a opção e a *commodity*, a equação diferencial parcial que governa o preço V da opção ao longo do tempo é

$$\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} + bS\frac{\partial V}{\partial S} - rV = 0. \tag{5.2.4}$$

Quando o custo de carrego da *commodity* é igual à taxa livre de risco, (5.2.4) se reduz à Equação de Black e Scholes. Se, por outro lado, nós tomamos uma opção sobre o contrato futuro da *commodity*, então, $b = 0$ e nós obtemos a Equação de Black.
Para as opções europeias de compra e venda, a equação (5.2.4) tem solução analítica bastante conhecida. Neste trabalho discutiremos a aproximação quadrática de Barone, Adesi e Whaley quando as condições de contorno das opções americanas de compra são aplicados à (5.2.4). A ideia principal dessa abordagem vem da aplicação da equação (5.2.4) ao prêmio de exercício antecipado da opção americana, uma vez que essa equação se aplica à precificação de opções americanas e europeias. Definindo o prêmio do exercício antecipado por

$$\varepsilon(S,T) = C(S,T) - c(S,T),$$

onde $C(S,T)$ é preço da opção americana e $c(S,T)$ é o preço da opção europeia. A equação diferencial parcial do exercício antecipado é

$$\frac{\partial \varepsilon}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 \varepsilon}{\partial S^2} + bS\frac{\partial \varepsilon}{\partial S} - r\varepsilon = 0. \tag{5.2.5}$$

Para simplificar a notação, nós devemos considerar a mudança de variável de tempo $T = t^* - t$ que evolui do vencimento da opção para o presente. Além disso, denotando $M = \frac{2r}{\sigma^2}$, $N = \frac{2b}{\sigma^2}$ e dividindo a equação (5.2.5) por $\frac{2}{\sigma^2}$, nós obtemos

$$\frac{M}{r}\frac{\partial \varepsilon}{\partial T} - S^2\frac{\partial^2 \varepsilon}{\partial S^2} - NS\frac{\partial \varepsilon}{\partial S} + M\varepsilon = 0.$$

O prêmio de exercício antecipado é, então, definido por $\varepsilon(S,R) = R(T)f(S,K)$. Dessa forma

$$S^2\frac{\partial^2 f}{\partial S^2} + NS\frac{\partial f}{\partial S} - Mf\left[1 + \left(\frac{\frac{dS}{dT}}{rS}\right)\left(1 + \frac{R\frac{\partial f}{\partial R}}{f}\right)\right] = 0. \tag{5.2.6}$$

Tomando $R(T) = 1 - e^{-rt}$ na equação (5.2.6), nós obtemos

$$S^2\frac{\partial^2 f}{\partial S^2} + NS\frac{\partial f}{\partial S} - \frac{M}{R}f - (1-R)M\frac{\partial f}{\partial R} = 0. \tag{5.2.7}$$

Nesse ponto, será introduzida a primeira aproximação do método. Para opções sobre *commodities* com tempo para o vencimento muito curto, ou muito longo, o último termo da equação (5.2.7) se aproxima de zero. Vamos considerar esse termo com valor nulo. Assim temos

$$S^2\frac{\partial^2 f}{\partial S^2} + Ns\frac{\partial f}{\partial S} - \frac{M}{R}f = 0. \tag{5.2.8}$$

A equação (5.2.8) é uma equação diferencial ordinária de segunda ordem com duas soluções linearmente independentes. A solução geral tem a forma

$$f(S) = a_1 S^{q_1} + a_2 S^{q_2},$$

onde $q_1 = \left[-(N-1) - \frac{\sqrt{(N-1)^2 + \frac{4M}{R}}}{2}\right]$ e $q_2 = \left[-(N-1) + \frac{\sqrt{(N-1)^2 + \frac{4M}{R}}}{2}\right]$.
Como $q_1 < 0$, se tivermos $a_1 \neq 0$, a função f tende a ∞ quando o preço S, da *commodity*, se aproxima de zero. Isso não é aceitável uma vez que o prêmio de exercício antecipado da opção de compra americana se torna desprezível quando o preço da *commdity* cai a zero. Dessa forma, devemos impor $a_1 = 0$, e o valor aproximado da opção americana de compra será dado por

$$C(S,T) = c(S,T) + Ra_2 S^{q_2}. \tag{5.2.9}$$

Na equação (5.2.9), o valor de $C(S,T)$ aumenta por dois fatores, quando S tem seu valor aumentado: $c(S,T)$ e $Ra_2 S^{q_2}$, assumindo $a_2 > 0$. Observe, no entanto, que a função à direita de (5.2.9) deveria tocar, mas não intersectar, a condição de fronteira, $S - K$, imposta pelo exercício antecipado da opção americana de compra. Abaixo do preço crítico S^*, tangente à condição de fronteira, o valor da opção americana de compra segue (5.2.9). Acima de S^*, o seu preço é igual ao valor de exercício $S - K$.
Para encontra o valor S^*, nós tomamos a condição limite

$$S^* - K = c(S^*, T) + Ra_2 S^{q_2}, \tag{5.2.10}$$

e a inclinação do valor de exercício da opção de compra é igualado à inclinação de $C(S^*, T)$, isso é

$$1 = e^{(b-r)T} N[d_1(S^*)] + Ra_2 S^{q_2}, \tag{5.2.11}$$

onde $\frac{\partial c(S^*,T)}{\partial S^*} = e^{(b-r)T} N[d_1(S^*)]$ e $d_1 = \frac{[\log\left(\frac{S^*}{K}\right) + \frac{(b+0.5*\sigma^2)}{T}]}{\sigma\sqrt{T}}$ Da equação (5.2.11), nós encontramos

$$a_2 = \frac{[1 - e^{(b-r)T} N[d_1(S^*)]}{Rq_2 S^{*(q_2-1)}} \tag{5.2.12}$$

Substituindo (5.2.12) em (5.2.10) e realizando as simplificações adequadas, nós obtemos

$$S^* - K = c(S^*, T) + \frac{[1 - e^{(b-r)T} N[d_1(S^*)]]S^*}{q_2} \tag{5.2.13}$$

O valor S^* na equação (5.2.13) deve ser determinado iterativamente. Com S^*, nós obtemos o valor de aproximação de Barone, Adesi e Whaley para opções americanas de compra

$$\begin{aligned} C(S,T) &= c(S,T) + A_2 \left(\frac{S}{S^*}\right)^{q_2}, \text{ quando } S < S^*, \\ C(S,T) &= S - K, \text{ quando } S \geq S^*, \end{aligned} \tag{5.2.14}$$

onde $A_2 = \left(\frac{S^*}{q_2}\right)[1 - e^{(b-r)T} N[d_1(S^*)]]$. Note que, com a equação (5.2.14), depois de obtido o valor S^*, nós podemos encontrar o valor aproximado das opções europeias de compra à partir de dados reais de opções americanas de compra.
Com a conversão de dados realizada, nós devemos lidar com o problema que surge dos diferentes valores do ativo subjacente em cada dia de negociação. Observe que a Equação de

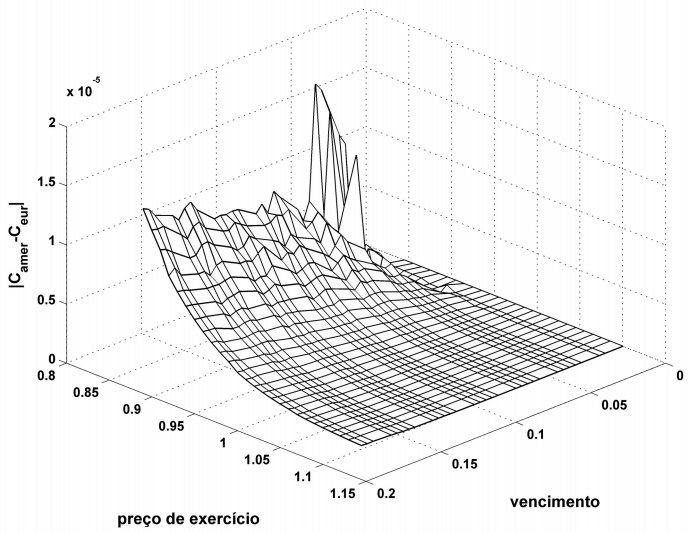

Figura 5.7: Diferença de preços entre opções americanas e europeias

Dupire (1.0.1) é linear em relação ao preço do ativo subjacente S_0 e, por isso, nós podemos fazer uma mudança de variável $C_s(K,T) = \frac{C(K,T)}{S_0}$. Considerando-se uma opção européia de compra de preço C, vencimento T, preço de exercício K sobre um contrato futuro de valor F, a mudança de variável proposta gera uma opção sintética de preço $C_s = \frac{C}{F}$, vencimento T sobre um contrato futuro de valor 1. Realizamos essa mudança de variável em cada dia disponível de dados, dividindo o preço das opções europeias de compra pelo valor do contrato futuro no dia correspondente, obtendo, dessa forma, um novo conjunto de preços de opções europeias de compra sobre um contrato futuro de preço $S_0 = 1$.
Observe que a divisão feita pelo valor dos contratos futuros, que é diferente em cada dia de negociação, gera um conjunto de opções que difere nos preços de exercício em cada dia. Nesse ponto nós realizamos um realinhamento dos dados de forma a reduzir o erro entre os novos preços de exercício correspondentes. Vale ressaltar que uma interpolação de dados poderia ser realizada, reduzindo o erro do descasamento de preços de exercícios mas, a metodologia adotada neste trabalho, além de simples, apresentou resultados satisfatórios. Na Figura 5.7, nós apresentamos o resultado final da conversão de preços das opções americanas para preços de opções europeias e na Figura 5.8, o resultado do realinhamento da opção sintética. Em ambas as Figuras nós percebemos a presença de ruído, característica esperada dos dados reais. A conversão de preços de opções americanas para opções europeias, no entanto, gera, eventualmente, ruído decorrente da má convergência do método iterativo de cálculo do preço crítico de exercício antecipado, S^*, das opções americanas.

5.2.2 Volatilidade Local

Diferentemente dos dados sintéticos, a utilização de dados reais não permite o cálculo preciso do ruído presente no problema. Como estimador do ruído, nós utilizamos 50% do valor extrínseco das opções, isto é, da diferença $C - (S_0 - K)$. O valor extrínseco dos dados utilizados é apresentado na Figura 5.9.
A escolha do parâmetro de Tikhonov foi, também, baseado no Príncipio de Morozov e a metodologia de escolha, com decrementos do parâmetro nas soluções das intâncias do problema inverso, foi repetida para os dados reais. A curva de discrepância, resultante da escolha do

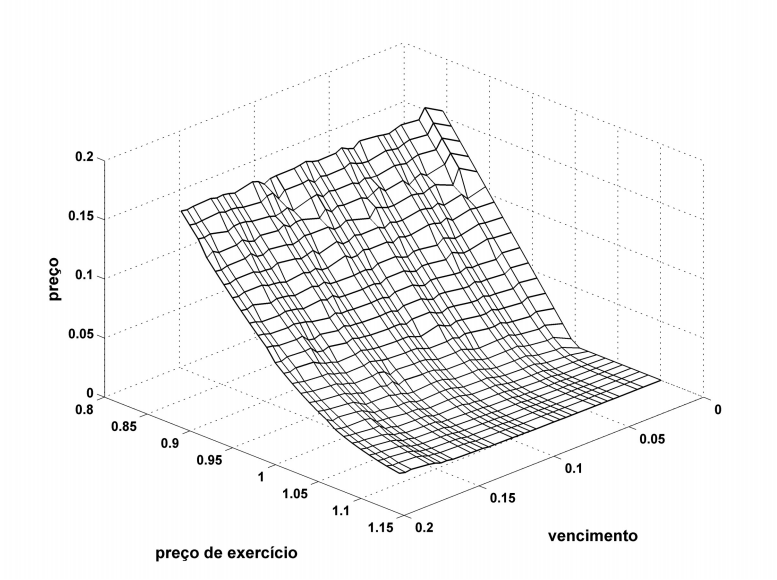

Figura 5.8: Preços convertido de opções europeias de compra

parâmetro de regularização, é apresentada na Figura 5.10. O resultado final do problema de calibragem da volatilidade local, aplicado à dados reais, é apresentado na Figura 5.11. Para

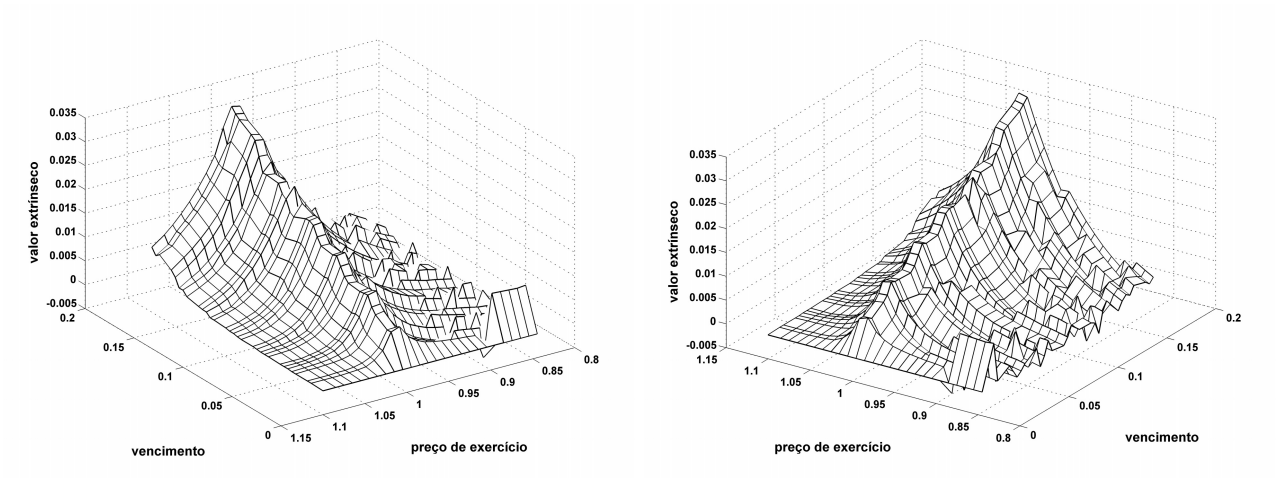

Figura 5.9: Valor extrínseco das opções europeias de compra

efeito de comparação, nós calculamos a volatilidade implícita de Black à partir dos dados gerados pela solução do problema direto aplicado à superfície de volatilidade local obtida. O resultado é apresentado na Figura 5.12.

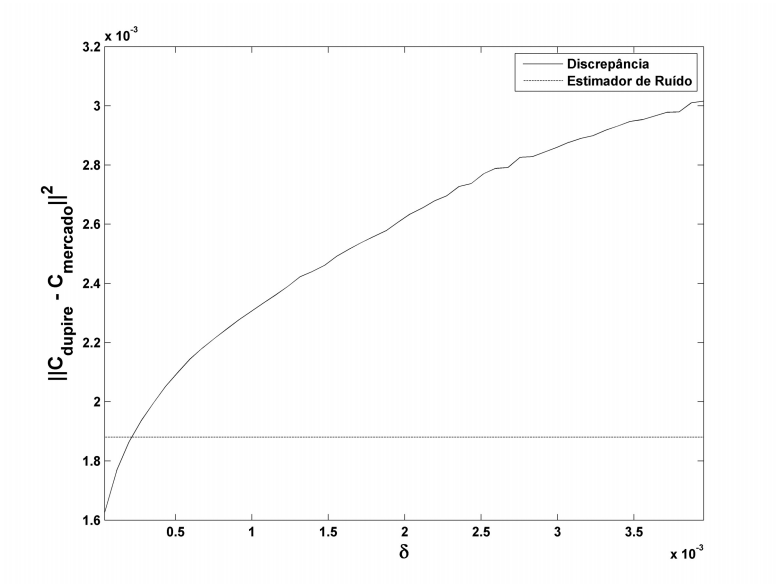

Figura 5.10: Curva de discrepância para dados reais

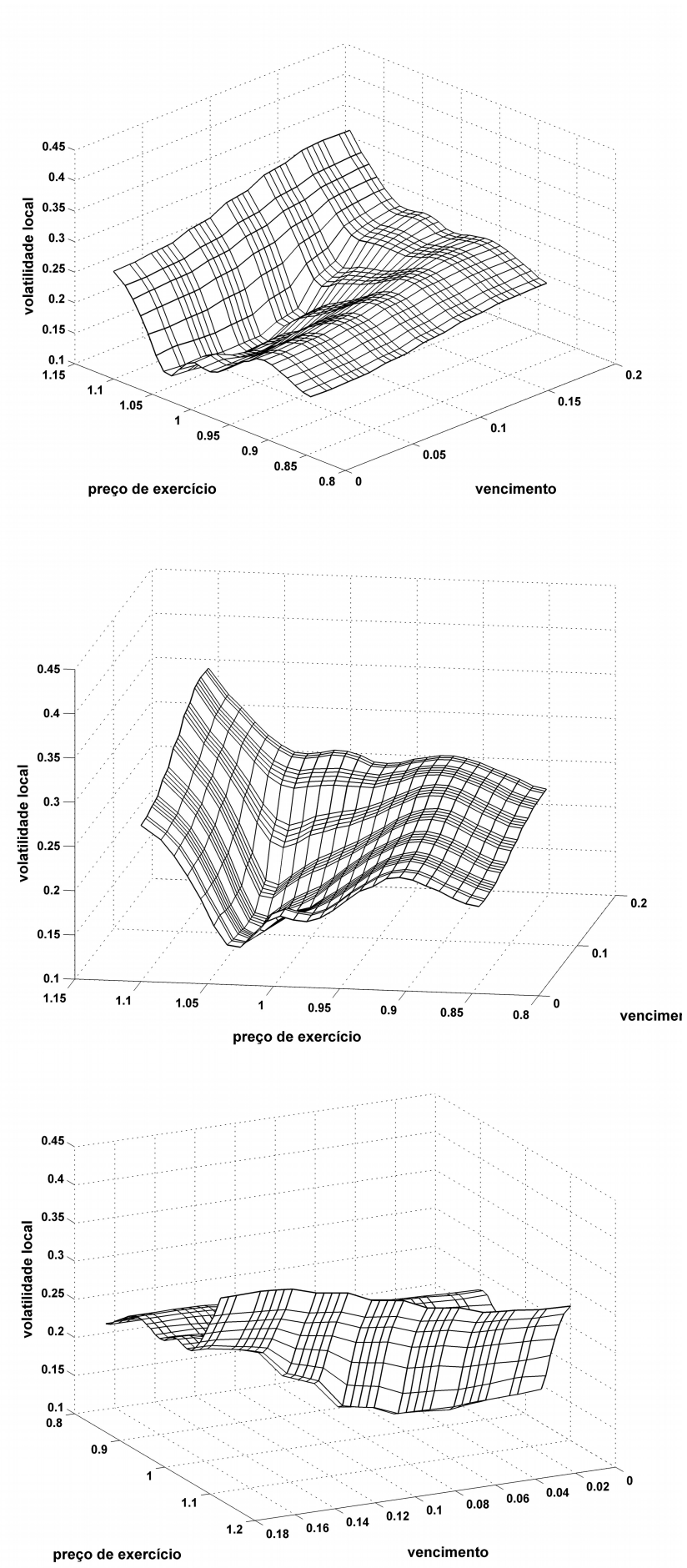

Figura 5.11: Volatilidade Local solução do problema de calibragem de volatilidade local

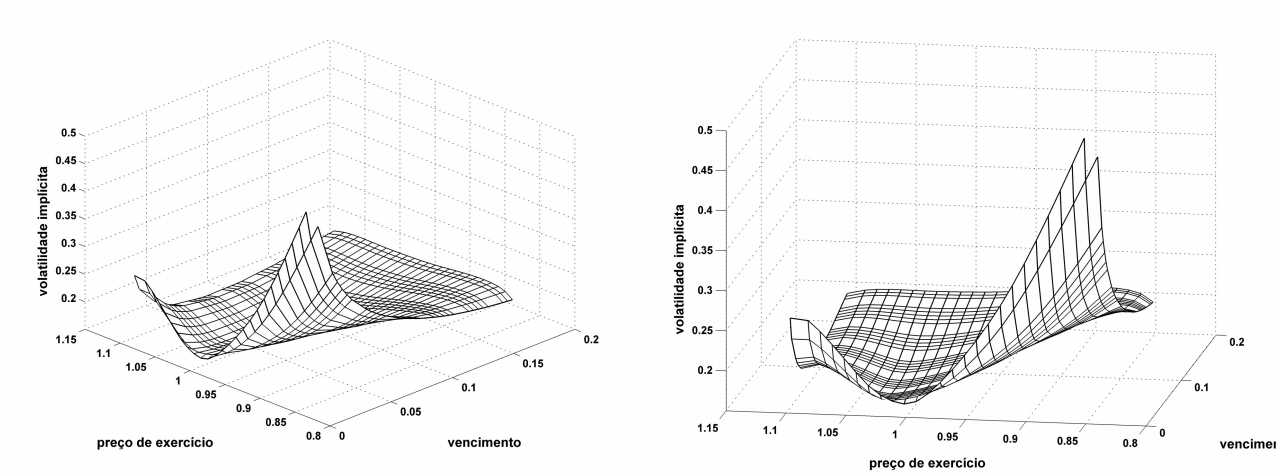

Figura 5.12: Volatilidade implícita de Black

Capítulo 6

Conclusão

Ao longo deste trabalho, nós especificamos e resolvemos o problema que consiste na calibragem da superfície de volatilidade local à partir de dados de preço de opções europeias de compra negociadas no mercado. O uso do Método dos Elemento s Finitos, de utilização pouco comum nas áreas de finanças [2], mostrou que complexidade matemática e de implementação do método é apenas aparente. Com a vantagem de permitir a utilização de um domínio não regular, mantendo os padrões de erro do método em níveis baixos, sobre outros métodos como o Método das Diferenças Finitas, ele se torna particularmente atrativo para a solução de problemas onde a disponibilidade de dados é pequena e fragmentada como em alguns mercados de opções com pouca liquidez.

A utlização do Método do Estado Adjunto também mostrou grande vantagem em relação a outros métodos de derivação automática como, por exemplo, Diferenças Finitas. Enquanto o cálculo do gradiente utilizando o primeiro método tem complexidade computacional próxima a resolução do problema direto, o cálculo através de diferenças finitas exige, ao menos, uma resolução do problema direto para cada variável envolvida no problema de otimização. O aumento do número de observações, de mercado, de preço de opções inviabiliza a utilização de muitos métodos de diferenciação automática. Nós pudemos perceber que, na dimensão dos exemplos numéricos que utilizamos neste trabalho, a utilização do Método do Estado Adjunto obtém a solução em poucos segundos, enquanto o Método das Diferenças Finitas leva uma fração de horas para alcançar a mesma solução.

A literatura utiliza, principalmente, dois métodos para a escolha do parâmetro de regularização: Curvas-L [11] e Princípio da Discrepância de Morozov. A escolha do parâmetro de regularização de Tikhonov através do Princípio da Discrepância de Morozov, utilizada neste trabalho, apresentou bons resultados quando o estimador de ruído era conhecido com exatidão, como o caso de dados simulados, e quando a estimação de ruído foi feita através de uma aproximação pelo valor extrínseco das opções utilizadas.

Por fim, todas as rotinas e funções implementadas em Matlab® foram validadas e apresentaram resultados que concordam com a literatura. Uma função com particular utilidade em problemas que utilizam dados reais, é aquela que implementa a conversão de preços de opções americanas sobre contratos futuros de *commodities* para preços de opções europeias sobre os mesmos futuros. Todo o código produzido e utilizado neste trabalho é apresentado nas sessões do Apêndice.

Apêndice A

Códigos

A.1 Opções

```
function [U] = Dupire_FEM(K_nodes, T_nodes, rate, eta, S_0)
%
% Equacao de Dupire - Metodo dos Elementos Finitos
%
% Entradas:
%   K_nodes: Vetor coluna de precos de exercicio
%   T_nodes: Vetor coluna de vencimentos
%   rate: Taxa livre de risco anualizada (escalar ou dependente do tempo)
%   eta: Quadrado da volatilidade local
%   S_0: Preco do ativo subjacente
%
% Saidas:
%   U: Precos das opcoes europeias de compra

    %Discrete Problem
    % M(C(m)-C(m-1))+(delta_t/2)*A(m)*C(m) = 0

    K = size(K_nodes,1);
    T = size(T_nodes,1);

    if(length(rate)==1)
        rate = ones(size(T_nodes))*rate;
    elseif (length(rate)~=length(T_nodes))
        display('rate e T_nodes devem ter a mesma dimensao');
    end

    if(size(eta) == [1 1])
        eta=ones(K-1,T)*eta;
    elseif (size(eta) ~= [K-1 T] )
        display('sigma e [K_nodes-1 S_nodes] devem ter a mesma dimensao');
    end

    delta_t = diff(T_nodes);

    M = Build_M(K_nodes);

    C = zeros(K-1,T);
    %Condicoes Iniciais (Payoff)
```

```
    C(:,1) = max((S_0-K_nodes(1:end-1)),zeros(K-1,1));

    for t=1:T-1
       B1 = M + delta_t(t)*Build_A(K_nodes, T_nodes, rate, eta,t+1);
       B2 = M;

       C(:,t+1)=( B1 )\( B2*C(:,t) );
    end
    U=C;
end
```

```
function P = AdjointState(K_nodes, T_nodes, C_nodes, C_obs, rate, eta)
%
% Problema Adjunto da Equacao de Dupire - Metodo dos Elementos Finitos
%
% Entradas:
%   K_nodes: Vetor coluna de precos de exercicio
%   T_nodes: Vetor coluna de vencimentos
%   C_nodes: Vetor de precos calculados de opcoes europeias de compra
%   C_obs: Vetor de precos observados de opcoes europeias de compra
%   rate: Taxa livre de risco anualizada (escalar ou dependente do tempo)
%   eta: Quadrado da volatilidade local
%   S_0: Preco do ativo subjacente
%
% Saidas:
%   P: Solucao do problema adjunto

    K = size(K_nodes,1);
    T = size(T_nodes,1);

    delta_t = diff(T_nodes);
    %Devemos estender a dimensao de tempo para utilizar todos os precos
    %observados
    delta_t = [delta_t; delta_t(end)];

    M = Build_M(K_nodes);

    P = zeros(K-1,T+1);


    G = C_nodes-C_obs;
    %Backward Problem
    for m=T:-1:1
        B = M + delta_t(m)*transpose(Build_A(K_nodes,T_nodes,rate,eta,m));
        P(:,m) = B\( M*P(:,m+1) + 2*G(:,m));
    end

    P(:,end)=[];

end
```

```
function g = BuildGrad(K_nodes, T_nodes, C_nodes, C_obs, rate, eta)
%
% Gradiente da Funcao de Minimos Quadrados
%
```

```
% Entradas:
%   K_nodes: Vetor coluna de precos de exercicio
%   T_nodes: Vetor coluna de vencimentos
%   C_nodes: Vetor de precos calculados de opcoes europeias de compra
%   C_obs: Vetor de precos observados de opcoes europeias de compra
%   rate: Taxa livre de risco anualizada (escalar ou dependente do tempo)
%   eta: Quadrado da volatilidade local
%   S_0: Preco do ativo subjacente
%
% Saidas:
%   g: gradiente da funcao de minimos quadrados em relacao ao quadrado
%      da volatilidade local

    K = size(K_nodes,1);
    T = size(T_nodes,1);

    g = zeros(K-1,T);

    if(length(rate)==1)
        rate = ones(size(T_nodes))*rate;
    elseif (length(rate)~=length(T_nodes))
        display('rate e T_nodes devem ter a mesma dimensao');
    end

    if(size(eta)==[1 1])
        eta=ones(K-1,T)*eta;
    elseif (size(eta) ~= [K-1 T] )
        display('eta e [K_nodes-1 S_nodes] devem ter a mesma dimensao');
    end

    delta_t = diff(T_nodes);

    P = AdjointState(K_nodes,T_nodes,C_nodes,C_obs,rate,eta);

    delta_eta=1;
    d_eta(:,:)=ones(size(eta));

    %Devemos estender a dimensao de tempo para concordar com a extensao
    %realizada no problema adjunto
    delta_t = [delta_t; delta_t(end)];

    for i=1:T
       g(:,i) = -P(:,i).*((1/delta_eta)*delta_t(i)* ...
           Build_Delta_A(K_nodes,d_eta,i)*C_nodes(:,i));
    end

    %Condicoes inicial e de contorno nao variam com a volatilidade local
    g(:,1)=0;
    g(1,:)=0;
end
```

```
function A = Build_A(K_nodes, T_nodes, rate, eta, m)
%
% Matriz Rigidez
%
% Entradas:
%   K_nodes: Vetor coluna de precos de exercicio
```

```
%   T_nodes: Vetor coluna de vencimentos
%   rate: Taxa livre de risco anualizada (escalar ou dependente do tempo)
%   eta: Quadrado da volatilidade local
%   m: Coluna do vencimento
%
% Saidas:
%   A: Matriz Rigidez

    h = diff(K_nodes);

    a_diag = ( (K_nodes(2:end-1).^2).*eta(2:end,m) )/2;
    a_diag = a_diag.*(1./h(1:end-1) + 1./h(2:end) );
    a_diag = a_diag - (rate(m)/6)*(h(1:end-1) + h(2:end));
    a_diag = [-rate(m)*h(1)/6; a_diag];

    a_diag1 = -( (K_nodes(2:end-1).^2).*eta(2:end,m) )./(2*h(1:end-1));
    a_diag1 = a_diag1 - ( rate(m)*K_nodes(2:end-1) )/2 + (rate(m)*h(1:end-1))/6;

    a_diag2 = -( (K_nodes(1:end-2).^2).*eta(1:end-1,m) )./(2*h(1:end-1));
    a_diag2 = a_diag2 + ( rate(m)*K_nodes(1:end-2) )/2 + (rate(m)*h(1:end-1))/6;

    A = diag(a_diag) + diag(a_diag1,-1) + diag(a_diag2, 1);
end
```

```
function M = Build_M(K_nodes)
%
% Matriz Massa
%
% Entradas:
%   K_nodes: Vetor coluna de precos de exercicio
%
% Saidas:
%   M: Matriz Massa

    h = diff(K_nodes);

    m_diag = (h(1:end-1)+h(2:end))/3;
    m_diag = [h(1)/3; m_diag];

    m_diag1 = h(1:end-1)/6;
    m_diag2 = h(1:end-1)/6;

    M = diag(m_diag) + diag(m_diag1,-1) + diag(m_diag2,1);
end
```

A.2 Otimização

```
function [x, f, exitFlag] = minFunction(funObj, x0, options)
% Otimizador quasi-Newton irrestrito com metodo LBFGS e algoritmo Line Search
%(Pseudo-Codigo: Capitulo 7 - Numerical Optimization. Nocedal, J. and Wright, S.)
% Algoritmo 7.5 - L-BFGS
%
% Entradas:
```

```
%   funObj: Handler da funcao objetivo e gradiente [f g] = funObj
%   x0: ponto inicial
%
% Saidas:
%   x: valor de minimo
%   f: Valor da funcao objetivo em x
%   exitFlag: Condicao de parada

global K_nodes T_nodes K T LineSize;

%Dimensao do problema
n = size(x0,1)*size(x0,2);

% Valores padrao
maxIter = 500; %Maximo numero de iteracao
optTol = 1e-5; %Tolerance para a condicao de primeira ordem
progressTol = 1e-9; %Tolerancia para progresso minimo
x = x0;
m = 10;
c1=1e-4;
c2=0.9;
wolfeLineSearchMaxIter = 100;
verbose = true;


%Verifica otimalidade no ponto inicial
[f g] = funObj(x);
maxG = max(abs(g));
%Condicao necessaria de primeira ordem para Otimalidade
if maxG <= optTol
    exitFlag = 1;
    return
end

% Valores Iniciais
iteration = 0;

f_obj = [f];
subplot(2,2,4);
plot(f_obj);
y_index=1;

a_lenght = [1];

while iteration <= maxIter
    %Usa o metodo do pradiente simples para a primeira iteracao ( p = -I*grad(f) )
    if iteration == 0
        p = -g;
        S = zeros(n,m);
        Y = zeros(n,m);
    else
        [S Y] = UpdateLBFGSVectors(g-g_old,a*p,S,Y,iteration);
        p = LBFGS(g,S,Y, iteration);
    end

    a= WolfeLineSearch(x,p,c1,c2,wolfeLineSearchMaxIter,funObj);
    if verbose
        fprintf('%03d\tObjective Function: %f\n',iteration,f) ;
```

```
    end
    x = x +a*p;

    g_old = g;

    f_old = f;

    [f g] = funObj(x);

    gtd = g'*p;

    maxG = max(abs(g));
    % Condicao necessaria de primeira ordem para otimalidade
    if maxG <= optTol
        exitFlag = 1;
        return
    end

    % Condicao de parada por progresso minimo
    if max(abs(a*p)) <= progressTol
        exitFlag = 2;
        return
    end

    if abs(f-f_old) < progressTol
        exitFlag=3;
        return;
    end

    iteration = iteration + 1;
end

exitFlag = 4;

end
```

```
function [a f g] = WolfeLineSearch(x,p,c1,c2,maxIter,funObj, options)
%
% Algoritmo Line Search Para as Condicoes de Wolfe
% (Pseudo-Codigo: Capitulo 3 - Numerical Optimization. Nocedal, J. and Wright, S.)
% (Algoritmos 3.5 and 3.6)
%
% Entradas:
%   x: Ponto inicial
%   p: Direcao de descida
%   c1: Parametro de suficiencia de descida
%   c2: Parametro de curvatura
%   maxIter: Numero maximo de iteracoes
%   funObj: Funcao objetivo
%
% Saidas:
%   a: Comprimento do passo
%   f: Valor da funcao objetivo em x+a*p
%   g: Valor do gradiente em x+a*p

if nargin == 7
    numericTol = options.NumericTolerance;
```

```
    verbose = options.Verbose;
else
    numericTol = 1e-9;
    verbose = true;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FASE DE AGRUPAMENTO %%%%%%%%%%%%%%%%%%%%%%%%

[f g] = funObj(x);

% Comprimento inicial do passo
% Para metodos de Newton e quasi-Newton, o comprimento a0=1 sempre deveria
% ser usado como valor inicial (Capitulo 3, pg 59)
a = 1;
a_min = 0;
% a_max e um valor fornecido por usuario para o maior comprimento do passo
%(utilizamos 10*a0)
a_max = 10;

a_old = 0;
f_old = f;
d_phi_old = g'*p;
% E importante que o comprimento do passo nas iteracoes aumente rapidamente
% para alcancar a_max em um numero finito de iteracoes
step_size = (a_max-a_min)/10;

[f_new, g_new] = funObj(x + a*p);

%phi(a) = f(x + a*p) => phi'(a) = transpose( grad(f) )*p
d_phi = g'*p;
d_phi_new = g_new'*p;

iteration = 0;
finished = 0;

while iteration < maxIter
    if f_new > f + c1*a*d_phi || (iteration > 1 && f_new >= f_old)
        bracket = [a_old a];
        fBracket = [f_old f_new];
        gBracket = [d_phi_old d_phi];
        break;
    elseif abs(d_phi_new) <= -c2*d_phi
        bracket = a;
        fBracket = f_new;
        gBracket = d_phi_new;
        finished = 1;
        break;
    elseif d_phi_new >= 0
        bracket = [a a_old];
        fBracket = [f_new f_old];
        gBracket = [d_phi_new d_phi_old];
        break;
    end

    a_old = a;
    f_old = f_new;

    a = a + step_size;
```

```
    [f_new, g_new] = funObj(x + a*p);
    d_phi_new = g_new'*p;

    iteration = iteration + 1;

end
%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FASE DE AGRUPAMENTO %%%%%%%%%%%%%%%%%%%%
if(iteration >= maxIter)
    if verbose
        fprintf('WolfeLineSearch Alerta: Numero Maximo de Iteracoes Excedido\n');
    end
    return;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FASE DE SELECAO %%%%%%%%%%%%%%%%%%%%%%%%%%%
if finished==0
    a_lo = bracket(1);
    a_hi = bracket(2);
    f_lo = fBracket(1); f_hi = fBracket(2);
    d_phi_lo = gBracket(1);

    while iteration < maxIter

        if abs(a_hi-a_lo)<numericTol
            if verbose
                fprintf('WolfeLineSearch Alerta: Tolerancia Numerica Alcancada\n');
            end
            break
        end

        %%%%%%% Aproximacao Quadratica %%%%%%%%%%%%%
        %a = -(a_hi-a_lo)^2*d_phi_lo / ( 2*( f_hi - f_lo - ...
        %(a_hi-a_lo)*d_phi_lo ) ) + a_lo;
        a = (a_lo+a_hi)/2;

        [f_new g_new] = funObj(x + a*p);
        d_phi_new = g_new'*p;

        % Armijo Condition Test
        if f_new > f + c1*a*d_phi ||  f_new >= f_lo
            a_hi = a;
            f_hi = f_new;
        else
            %Wolfe Condition Test
            if abs(d_phi_new) <= -c2*d_phi
                %a* found!
                break;
            elseif d_phi_new*(a_hi-a_lo) >= 0
                a_hi = a_lo;
                f_hi = f_lo;
            end

            a_lo = a;
            f_lo = f_new;
            d_phi_lo = d_phi_new;
        end

```

```
        iteration = iteration + 1;
    end

    if(iteration >= maxIter)
        if verbose
            fprintf('WolfeLineSearch Alerta: Numero Maximo de Iteracoes Excedido\n');
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FASE DE SELECAO %%%%%%%%%%%%%%%%%%%%%%%

end
```

```
function [p] = LBFGS(g, s, y, iteration)
%
% Algoritmo de Atualizacao LBFGS para Metodo quasi-Newton
%(Pseudo-Codigo: Capitulo 7 - Numerical Optimization. Nocedal, J. and Wright, S.)
% Algoritmo 7.4 - L-BFGS Two-Loop Recursion
%
% Entradas:
%   g: Gradiente da funcao objetivo
%   s: Ultimo vetor de direcao de busca( x(k+1)-x(k) )
%   y: Ultimo comprimento de passo ( grad(f(k+1))-grad(f(k)) )
%
% Output:
%   p: Nova direcao de busca ( H*grad(f) )
%
%n e a variavel dimensao e m e o numero de vetores armazenados no L-BFGS
[n, m] = size(s);
m = min(m,iteration);
idt = eye(n);

for i = 1:m
    rho(i,1) = 1/( y(:,i)'*s(:,i) );
end

q = zeros(n,1);
r = zeros(n,1);
a =zeros(n,1);
beta = 0;

q = -g;

for i = m:-1:1
    a(i) = rho(i)*s(:,i)'*q;
    q = q-a(i)*y(:,i);
end

%(Capitulo 7, pg 178)
%Um metodo para escolha de H0 que se mostra efetivo na pratica e definir
% H0 = gamma*I, onde
gamma = ( s(:,m)'*y(:,m) )/( y(:,m)'*y(:,m) ) ;
H0 = gamma*idt;

r = H0*q;

for i = 1:m
```

```
    beta(i) = rho(i)*y(:,i)'*r;
    r = r + s(:,i)*(a(i)-beta(i));
end

p=r;

end
```

```
function [S Y] = UpdateLBFGSVectors(y,s,S,Y,k)
% Algoritmo de Atualizacao de vetores historicos de LBFGS
%
% Entradas:
%   y: Ultimo comprimento de passo ( grad(f(k+1))-grad(f(k)) )
%   s: Ultimo vetor de direcao de busca( x(k+1)-x(k) )
%   S: Historico dos ultimos m vetores s
%   Y: Historico dos ultimos m vetores y
%   k: Iteracao atual
%
% Output:
%   S: Novo conjunto de vetor s
%   Y: Novo conjunto de vetor y
%
    ys = y'*s;
    m = size(S,2);
    if ys > 1e-10
        if (k>m)
           S(:,1)=[];
           Y(:,1)=[];
           S = [S s];
           Y = [Y y];
        else
           S(:,k) = s;
           Y(:,k) = y;
        end
    else
       fprintf('Ignorando Atualizacao\n');
    end

end
```

A.3 Conversão de Dados

```
function [europeanOptionPrice, optionVola, exitFlag] = ...
  convertAmericanToEuropean(isCall, optionPrice, S, X, r, T, sigma_user, p_user)
%-------------------------------------------------------------------------
% Converte um conjunto de precos de opcoes europeias em opcoes americanas
% usando a formula de Barone-Adesi Whaley para obter a vol implicita e
% dai calcular o preco da opcao europeia com a formula de Black
%
% A Funcao retorna o preco da opcao europeia e a volatilidade implicita
%
% isCall        = Call = 1, Put = 0
% optionPrice   = Preco de mercado da opcao americana
```

```
% S              = Preco do ativo subjacente (S=F opcao sobre futuro)
% X              = Preco de exercicio da opcao
% r              = Taxa livre de risco
% T              = Tempo para o vencimento
%
% Valores de Retorno
% optionPrice    = Preco da opcao europeia
% optionVola     = Volatilidade implicita
%-------------------------------------------------------------------------

    % Valor inicial para o fsolve
    if (isCall)
        if nargin < 8
            S_Star_0 = S+10;
        else
            S_Star_0 = S+p_user;
        end
        if nargin <7
            Sigma_0 = blkimpv(S, X, r, T, optionPrice, [], [], {'call'});
        else
            Sigma_0 = sigma_user;
        end
    else
        S_Star_0 = S-10;
        Sigma_0 = blkimpv(S, X, r, T, optionPrice, [], [], {'put'});
    end

    initialGuess = [S_Star_0; Sigma_0^2];

    options = optimset('Jacobian','On');

    [x fval exitFlag] = fsolve(@impliedVolatility_BAW_fsolve, initialGuess,...
        options, isCall, optionPrice, S, X, r, 0.0, T);

    optionVola = sqrt(x(2));
    S_Star = x(1);

    b=0;
    d1_star = (log(S_Star/X) + (b+0.5*optionVola^2)*T) / (optionVola * sqrt(T));
    M = (2*r)/(optionVola^2);
    N = (2*b)/(optionVola^2);
    K = 1 - exp(-r*T);
    q2 = 0.5 * ( -(N-1) + sqrt( (N-1)^2 + (4*M)/K ) );
    N_d1_Star = normcdf(d1_star,0,1);
    A2 = (S_Star/q2) * ( 1 - exp((b-r)*T) * N_d1_Star );

    if(S>=S_Star)
       europeanOptionPrice = S-X;
    else
       europeanOptionPrice = optionPrice - A2*(S/S_Star)^q2;
    end

end
```

```
function [F,J] = impliedVolatility_BAW(x, isCall, marketPrice, S, X, r, b, T)

    try
```

```
        % Valores iniciais
        S_Star = x(1);
        Sigma = sqrt(x(2));


        if (isCall)

            % 1.) Variaveis necessarias para calcular f e g
            M = (2*r)/(Sigma^2);
            N = (2*b)/(Sigma^2);
            K = 1 - exp(-r*T);
            q2 = 0.5 * ( -(N-1) + sqrt( (N-1)^2 + (4*M)/K ) );

            d1_star = (log(S_Star/X) + (b+0.5*Sigma^2)*T) / (Sigma * sqrt(T));
            d2_star = d1_star - Sigma * sqrt(T);
            N_d1_Star = normcdf(d1_star,0,1);
            n_d1_Star = normpdf(d1_star,0,1);
            n_d2_Star = normpdf(d2_star,0,1);

            d1 = (log(S/X) + (b+0.5*Sigma^2)*T) / (Sigma * sqrt(T));
            d2 = d1 - Sigma * sqrt(T);
            n_d2 = normpdf(d2,0,1);

            A2 = (S_Star/q2) * ( 1 - exp((b-r)*T) * N_d1_Star );

            c_S_Star = EurpeanOptionOnCommodity(1, S_Star, X, r, b, T, Sigma);
            c = EurpeanOptionOnCommodity(1, S, X, r, b, T, Sigma);


            % 2.) Calcula f(x) = 0 e g(x) = 0
            f = c_S_Star + S_Star/q2 * ( 1 - exp((b-r)*T) * N_d1_Star ) - ...
                (S_Star - X);
            g = c + A2 * ((S/S_Star)^q2) - marketPrice;

            F = [f g];


            % 3.) Calcula a matriz Jacobiana de f e g

            % Variaveis de Substituicao
            y1 = 1 - exp((b-r)*T) * N_d1_Star;
            y2 = n_d1_Star / (S_Star * Sigma * sqrt(T));
            y3 = exp((b-r)*T);
            y4 = b/(Sigma^4) - ( ( (N-1)*b + 2*r/K ) / ...
                ( Sigma^4 * sqrt((N-1)^2 + 4*M/K) ) );
            y5 = X * sqrt(T) * exp(-r*T) * n_d2 / (2*Sigma);
            y6 = -0.5 * ( log(S_Star/X) + b*T ) / ( (Sigma^3 * sqrt(T)) / ...
                (4*Sigma) );
            y7 = X * sqrt(T) * exp(-r*T) * n_d2_Star / (2*Sigma);

            % Derivadas parciais da matriz Jacobiana
            dg_dS_Star = ((S/S_Star)^q2) * ( (1/q2)*y1 - ...
                (S_Star/q2)*y2*y3 - q2*A2*(1/S_Star) );
            df_dS_Star = y3 * N_d1_Star + (y1/q2) - (S_Star/q2)*(y2*y3) - 1;
            dg_dSigmaSqr = y5 + ((S/S_Star)^q2) * ( A2*log(S/S_Star)*y4 - ...
                (S_Star/q2)*y3*y6*n_d1_Star - y1*y4*(S_Star/(q2^2)) );
            df_dSigmaSqr = y7 - y1*y4*(S_Star/(q2^2)) - ...
```

```
                (S_Star/q2)*y3*y6*n_d1_Star;

            J = [df_dS_Star df_dSigmaSqr ; dg_dS_Star dg_dSigmaSqr ];

        else

            % 1.) Variaveis necessarias para calcular f e g
            M = (2*r)/(Sigma^2);
            N = (2*b)/(Sigma^2);
            K = 1 - exp(-r*T);
            q1 = ( -(N-1) - sqrt( (N-1)^2 + (4*M)/K ) ) / 2;

            d1_star = (log(S_Star/X) + (b+0.5*Sigma^2)*T) / (Sigma * sqrt(T));
            d2_star = d1_star - Sigma * sqrt(T);
            N_d1_Star = normcdf(-d1_star,0,1);
            n_d1_Star = normpdf(-d1_star,0,1);
            n_d2_Star = normpdf(-d2_star,0,1);

            d1 = (log(S/X) + (b+0.5*Sigma^2)*T) / (Sigma * sqrt(T));
            d2 = d1 - Sigma * sqrt(T);
            n_d2 = normpdf(-d2,0,1);

            A1 = -(S_Star/q1) * ( 1 - exp((b-r)*T) * N_d1_Star );

            p_S_Star = EurpeanOptionOnCommodity(0, S_Star, X, r, b, T, Sigma);
            p = EurpeanOptionOnCommodity(0, S, X, r, b, T, Sigma);


            % 2.) Calcula f(x) = 0 e g(x) = 0
            f = p_S_Star - S_Star/q1 * ( 1 - exp((b-r)*T) * N_d1_Star ) - ...
                (X - S_Star);
            g = p + A1*((S/S_Star)^q1) - marketPrice;

            F = [f ; g];


             % 3.) Calcula a matriz Jacobiana de f e g

            % Variaveis de Substituicao
            z1 = 1 - exp((b-r)*T) * N_d1_Star;
            z2 = -n_d1_Star * exp((b-r)*T) / (S_Star * Sigma * sqrt(T));
            z3 = ( 0.5*(log(S_Star/X) + b*T)/(Sigma^3*sqrt(T)) ) - ...
                (sqrt(T)/(4*Sigma));
            z4 = ( b + ( ((N-1)*b + 2*r/K) / sqrt((N-1)^2 + 4*M/K) ) ) / ...
                (Sigma^4);
            z5 = (S_Star/q1) * ( exp((b-r)*T) * n_d1_Star * z3 ) + ...
                (S_Star/(q1^2))*z1*z4;

            % Derivadas parciais da matriz Jacobiana
            dg_dS_Star = ((S/S_Star)^q1) * ( (S_Star/q1)*z2 - (1/q1)*z1 - ...
                q1*A1*(1/S_Star) );
            df_dS_Star = z1 * (1 - 1/q1) + (S_Star/q1)*z2;
            dg_dSigmaSqr = ( n_d2*X*sqrt(T)*exp(-r*T) / (2*Sigma) ) + ...
                ((S/S_Star)^q1) * ( A1*log(S/S_Star)*z4+z5 );
            df_dSigmaSqr = ( n_d2_Star*X*sqrt(T)*exp(-r*T) / (2*Sigma) ) + ...
                (S_Star/q1)*exp((b-r)*T)*n_d1_Star*z3 + (S_Star/(q1^2))*z1*z4;

            J = [df_dS_Star df_dSigmaSqr; dg_dS_Star dg_dSigmaSqr];
```

```

        end
    catch
        F = [NaN NaN];
        J = [NaN NaN ; NaN NaN];
    end

end
```

Referências Bibliográficas

[1] Yves Achdou and Olivier Pironneau, *Computational Methods for Option Pricing*, Frontiers in Applied Mathematics, SIAM, 2005.

[2] ———, *Finite element methods for option pricing*, Université Pierre et Marie Curie (2007).

[3] GIOVANNI BARONE-ADESI and Robert E Whaley, *Efficient analytic approximation of american option values*, The Journal of Finance **42** (1987), no. 2, 301–320.

[4] Fischer Black and Myron Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy **81** (1973), no. 3, 637–654.

[5] David L Colton and Rainer Kress, *Inverse acoustic and electromagnetic scattering theory*, 1992.

[6] S Crépey, *Calibration of the local volatility in a trinomial tree using tikhonov regularization*, Inverse Problems **19** (2003), no. 1, 91.

[7] Austen C Duffy, *An introduction to gradient computation by the discrete adjoint method*, Tech. report, Technical report, Florida State University, 2009.

[8] Bruno Dupire, *Pricing with a smile*, Risk Magazine **7** (1994), 18–20.

[9] Ronald M Errico, *What is an adjoint model?*, Bulletin of the American Meteorological Society **78** (1997), no. 11, 2577–2591.

[10] Jim Gatheral, *The Volatility Surface: A Practitioner's Guide*, Wiley Finance, John Wiley & Sons, 2006.

[11] Jian Geng, *Calibration of a Local Volatility Surface using Tikhnov Regularization*, Ph.D. thesis, Florida State University.

[12] Martin Hanke and Elisabeth Rösler, *Computation of local volatilities from regularized dupire equations*, International Journal of Theoretical and Applied Finance **8** (2005), no. 02, 207–221.

[13] Steven L Heston, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, Review of financial studies **6** (1993), no. 2, 327–343.

[14] George W Kutner, *Determining the implied volatility for american options using the qam*, Financial Review **33** (1998), no. 1, 119–130.

[15] Shengtai Li and Linda Petzold, *Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement*, Journal of Computational Physics **198** (2004), no. 1, 310–325.

[16] Ricardo Rebonato, *Volatility and Correlation: The Perfect Hedger and the Fox*, second ed., John Wiley & Sons, 2004.

[17] Fabrice Douglas Rouah, *Derivation of local volatility.*

[18] Erkki Somersalo and Jari Kapio, *Statistical and Computational Inverse Problems*, Applied Mathematical Sciences, vol. 160, Springer, 2004.

[19] Armand Wirgin, *The inverse crime*, arXiv preprint math-ph/0401050 (2004).

[20] SJ Wright and J Nocedal, *Numerical optimization*, vol. 2, Springer New York, 1999.