UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Sistema para testes de stress em uma carteira de opções de moedas

Moreno Siqueira e Mello

Dissertação de Mestrado do Programa de Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria (MECAI)

ICMC USP SÃO CARLOS

Moreno Siqueira e Mello

Sistema para testes de stress em uma carteira de opções de moedas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria. *VERSÃO REVISADA*

Área de Concentração: Matemática, Estatística e Computação

Orientador: Prof. Dr. Adenilso da Silva Simão

USP – São Carlos
Novembro de 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

M527s

Mello, Moreno Siqueira e
Sistema para testes de stress em uma carteira de opções de moedas / Moreno Siqueira e Mello; orientador Adenilso da Silva Simão. -- São Carlos, 2017.
75 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2017.

1. Mercado de câmbio. 2. Opções de moedas. 3. Testes de stress. I. Simão, Adenilso da Silva, orient. II. Título.

Moreno Siqueira e Mello

System for stress test in an FX option portfolio

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master – Professional Masters in Mathematics, Statistics and Computing Applied to Industry. *FINAL VERSION*

Concentration Area: Mathematics, Statistics and Computing

Advisor: Prof. Dr. Adenilso da Silva Simão

USP – São Carlos
November 2017

Este trabalho é dedicado à minha amada esposa Paula Festino,
pelo seu companheirismo, por me despertar o desejo de sempre evoluir
e pelo incentivo durante os anos deste curso.

AGRADECIMENTOS

Agradeço aos meus pais, pela base que me proporcionaram e que me permitiu chegar até aqui.

Agradeço ao colega Carlos Ferrari por todas a caronas de ida e volta para as aulas e pelo conhecimento transmitido durante essas caronas.

Aos colegas Alice Singer e Maurício Pereira pela paciência e por todos os ensinamentos no meu primeiro contato com o mercado financeiro.

Agradeço ao meu orientador Prof. Adenilso da Silva Simão pela cobrança e por não me deixar desistir.

Ao Prof. Dorival Leão, pela dedicação a esta turma e pela vontade em fazer este curso prosperar.

RESUMO

MELLO, M. S. **Sistema para testes de stress em uma carteira de opções de moedas**. 2017. 75 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Na gestão de recursos financeiros, visualizar e gerenciar em tempo real os riscos inerentes a uma carteira de investimentos é uma tarefa crucial para que o objetivo de gerar lucro possa ser atingido, ou que pelo menos as perdas possam ser minimizadas. Uma das formas de realizar esse gerenciamento é submeter essas carteiras a simulações onde são definidos cenários contendo variações de fatores que possam influenciar os ativos nelas contidos. Dependendo da classe dos ativos financeiros analisados, essas simulações requerem uma ferramenta mais sofisticada, capaz de lidar com modelos complexos de precificação.

O objetivo deste trabalho consiste em resolver uma demanda real de uma gestora de recursos onde este autor atua: o desenvolvimento de uma ferramenta capaz de realizar testes de *stress* em uma carteira de investimentos contendo mais especificamente opções de moedas. Foi desenvolvido um sistema no formato de *add-in* de Excel em que os gestores podem definir cenários com as variações desejadas e, em conjunto com dados de mercado em tempo real, avaliar o impacto dessas variações em seu *portfolio*. O desenvolvimento foi realizado em etapas, e a versão atual da ferramenta trouxe ganhos no tempo de execução das simulações na ordem de dez vezes, quando comparado à versão anterior.

Nesta dissertação serão mostrados detalhes da implementação do sistema, bem como o embasamento teórico utilizado no seu desenvolvimento. Será apresentada uma breve descrição sobre o mercado de câmbio e seus instrumentos, incluindo opções de moedas. Também será descrito um modelo para precificação e mensuração de risco desses instrumentos.

Palavras-chave: Mercado de câmbio, Opções de moedas, Teste de *stress*.

ABSTRACT

MELLO, M. S. **System for stress test in an FX option portfolio**. 2017. 75 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

In the financial resources management, visualizing and handling risks inherent in an investment portfolio in real time are key tasks to ensure that the objective of profit is accomplished, or at least that the losses are mitigated. One way to perform this kind of management is to submit the portfolio to scenario simulations, in which factors that might affect the assets held in the portfolio are stressed. Depending on the class of these assets, there is the need of a more sophisticated tool, capable of handling complex pricing models.

The main purpose of this work is to solve a real demand for an investment management company for which this author works: the development of a tool capable to perform stress tests in an investment portfolio containing more specifically Foreign eXchange options. An Excel add-in has been developed and managers can use it to define scenarios with the desired bumps and, along with real time market data, analyze the impact of these bumps in the portfolio. The development has been made in phases and the tool's current version has brought a reasonable improvement to the execution time of the simulations.

In this thesis we will discuss system's implementation details, as well as the theoretical basis used in its development. An overview of the FX market and its instruments will be presented, including FX options. Also, there will be a description of a model for pricing and risk measurement of these instruments.

Keywords: FX Market, FX Options, Stress test.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE CÓDIGOS-FONTE

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

De acordo com a mais recente pesquisa trienal realizada pelo Banco Internacional de Compensações, o mercado internacional de câmbio movimenta diariamente cerca de US$ 5,1 trilhões (BIS, 2016), o que o torna o mercado financeiro com maior liquidez. Como observado por Pereira (2011), "este volume de negociação tem apresentado crescimento consistente e significativamente superior tanto ao PIB dos países envolvidos quanto ao fluxo de comércio internacional, indicando que parte representativa das operações de câmbio estaria ligada à especulação com moedas".

De fato, muitos fundos de investimentos fazem operações especulativas no mercado de câmbio com o único objetivo de rentabilizar as aplicações de seus clientes. Esses investimentos podem ser feitos, por exemplo, montando uma carteira com posições compradas e/ou vendidas em opções de moedas. Uma opção, conforme será mostrado e detalhado adiante, é um contrato que confere ao seu detentor o direito de compra ou venda de um determinado ativo numa data futura por um preço pré-estabelecido, gerando lucro ou prejuízo dependendo do valor do ativo na data de vencimento.

Um grande desafio na definição de estratégias de investimentos no mercado financeiro é a análise do comportamento de uma determinada carteira de ativos, dada a variação dos fatores de risco a que esses estão expostos. Variações hipotéticas ou baseadas em eventos históricos desses fatores, agrupadas em cenários, podem ser utilizadas por gestores para simular o comportamento da carteira em casos de *stress* de mercado. Aumentando o número de simulações, resultados mais precisos e confiáveis podem ser atingidos. Assim, pode-se aumentar os ganhos, ou diminuir as perdas, em períodos de crise do mercado.

No caso de opções, esse desafio é ainda maior, pois a determinação de seu valor de mercado depende de diversos fatores, visto que a natureza desse tipo de contrato traz uma incerteza sobre o seu exercício ou não na data de vencimento. Até mesmo os modelos mais simples de precificação, como os que serviram de base para este trabalho, possuem algoritmos

de precificação complexos que exigem um grande processamento computacional.

Este trabalho visa a apresentar um sistema desenvolvido para o cálculo de resultado e exposição de risco de um carteira de opções de moedas, que pode ser utilizado como um *add-in* de Excel. Dessa forma, os gestores podem utilizar dados de mercado fornecidos por provedores, como Bloomberg Professional[1] e Thomson Reuters Eikon[2], para monitorar sua carteira em tempo real. Nele é possível determinar diversos cenários de mercado, conjunto de variações ou choques aos parâmetros de entrada para o modelo escolhido, bem como avaliar o desempenho da carteira ao considerar-se tais cenários.

Tal sistema foi desenvolvido por este autor dentro do ambiente de uma gestora de recursos, não só baseado nas teorias de matemática financeira já exploradas por diversos outros autores mas também recebendo visões reais dos próprios gestores. Estes tinham como ferramenta de monitoramento das carteiras apenas planilhas que não eram capazes de realizar todas as simulações necessárias. Assim, além de contribuir com informações para o desenvolvimento, tornaram-se usuários responsáveis pela validação do sistema, utilizando-o diariamente nas simulações de *stress* de mercado. O desenvolvimento foi realizado em etapas, sendo que na primeira versão apenas fórmulas de precificação foram disponibilizadas, e a lógica de simulação era realizada por meio de macros que alteravam os parâmetros de entrada dessas funções. Já na segunda versão da ferramenta, que é a atual, toda a lógica é realizada internamente no sistema, trazendo ganhos consideráveis no tempo de execução das simulações.

Os próximos capítulos apresentam o embasamento teórico e os principais pontos do desenvolvimento. Uma breve descrição sobre o mercado de câmbio e seus diferentes instrumentos é apresentada no Capítulo 2. Em seguida, no Capítulo 3, há um detalhamento sobre os modelos de Black-Scholes e Black 1976, além de suas derivadas, que serão utilizadas no sistema para os cálculos de resultado e exposição de risco. O Capítulo 4 mostra as especificidades da construção e interpolação das superfícies de volatilidade, que serão utilizadas para a extração de dados de entrada do modelo de precificação. O Capítulo 5, por sua vez, traz detalhes sobre a implementação do sistema e sobre os ganhos obtidos com sua utilização. Por fim, o Capítulo 6 traz algumas considerações e propostas para a realização de trabalhos futuros com intuito de evoluir o sistema.

[1] https://www.bloomberg.com/professional/

[2] http://financial.thomsonreuters.com/en/products/tools-applications/trading-investment-tools/eikon-trading-software.html

CAPÍTULO

2

MERCADO DE CÂMBIO

No mercado financeiro, mais especificamente na gestão de investimentos, o objetivo dos gestores é captar investimentos de seus clientes e aplicá-los em ativos financeiros com a finalidade de rentabilizar esses investimentos e dar um retorno positivo para os investidores. Para atingir esse objetivo, existem diversas classes de ativos ou nichos em que se pode aplicar, sendo um deles o mercado de câmbio. Neste capítulo será realizada uma breve descrição sobre esse mercado e sobre os instrumentos disponíveis para realização de operações pelos gestores.

Uma operação de câmbio nada mais é que a troca de um valor em determinada moeda pelo valor correspondente em uma moeda estrangeira. Essa troca pode ser feita em espécie rapidamente em uma casa de câmbio, por exemplo, por um turista viajando para um país que utiliza outra moeda.

Agora, supondo que uma pessoa deseje fazer um investimento de grande valor em moeda estrangeira, ela pode entrar em contato com alguma contraparte (um banco ou corretora) e negociar uma troca de moedas. Com a taxa de conversão acordada, o cliente depositará o valor para a contraparte e receberá o valor correspondente em uma outra conta bancária. Nesse caso, dizemos que a operação será executada no mercado de câmbio *spot* e sua liquidação ocorrerá geralmente em dois dias úteis (CLARK, 2011).

Além da troca imediata de moedas, existem outros instrumentos que permitem às partes negociar uma troca futura, tais como *forwards* e opções. Essas operações serão descritas nas próximas seções e são geralmente realizadas por empresas exportadoras e importadoras que desejam proteger-se da variação cambial ou por especuladores que almejam monetizar essa variação.

Como observado em todos os casos citados, a operação é realizada diretamente entre duas partes sem a necessidade de um intermediador ou regulador, caracterizando esse mercado como de balcão.

2.1 Par de moedas e Ordem de precedência

As moedas envolvidas em uma operação de câmbio são chamadas de "par de moedas" e, nesse par, há uma moeda chamada de base e outra chamada de termo. A cotação de um par de moedas representa a quantidade de moeda termo necessária para a compra de uma quantidade da moeda base. Essa cotação ou taxa de conversão é referenciada no mercado pelo código ISO 4217[1] da moeda base acrescido do código correspondente à moeda termo, no formato XXXYYY.

Não há uma regra que define qual moeda é a base e qual é o termo em um par de moedas mas sim uma convenção de mercado. Essa convenção pode ser generalizada pela hierarquia apresentada por Clark (2011):

EUR > GBP > AUD > NZD > USD > CAD > CHF > JPY > ...

Dessa forma, podemos concluir que o par de moedas formado por reais brasileiros e dólares americanos é representado no formato USDBRL.

No dia 02 de junho de 2017, por exemplo, a cotação de USDBRL 3,2484 indica que nessa data eram necessárias 3,2484 quantidades de reais brasileiros para a compra de uma unidade de dólar americano.

2.2 *Forwards*

De acordo com Hull (2006), um *forward* é um acordo de compra ou venda de um ativo em um certo tempo no futuro (T) por um preço pré-determinado (K).

Um contrato de *forward* no mercado de câmbio é firmado entre duas partes, por meio do qual uma assume o compromisso de compra de uma certa quantidade de moeda A numa determinada data futura, pagando por isso uma quantidade pré-estabelecida de moeda B. Por outro lado, a outra parte assume o compromisso de entrega daquela quantidade de moeda A, recebendo em troca a quantidade de moeda B. Dizemos que o comprador assumiu uma posição *long* e o vendedor uma posição *short*.

Vale notar que a variação cambial pode gerar lucro para a parte compradora e, consequentemente, perda para a parte vendedora ou vice-versa. Se, no momento da entrega, a taxa de conversão S_T estiver maior que K, a parte em posição *long* registrará um lucro, pois será equivalente a realizar uma compra abaixo do valor de mercado. Entretanto, se $S_T < K$, ela registrará um prejuízo. Logo, o resultado esperado é dado por:

$$S_T - K. \tag{2.1}$$

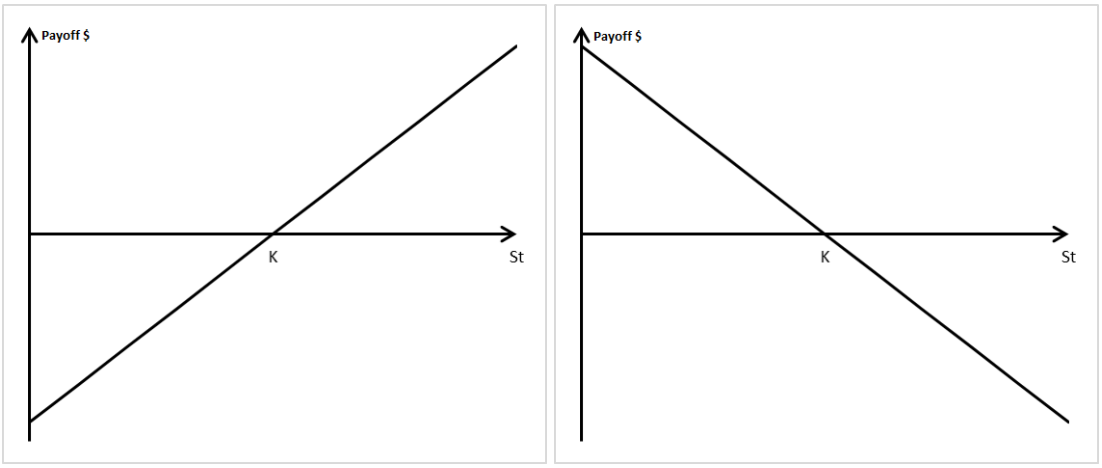
[1] ISO 4217: www.iso.org

E para a parte em posição *short*:

$$K - S_T. \tag{2.2}$$

Tais resultados podem ser visualizados através da Figura 1.

Figura 1 – Resultado esperado de uma operação *forward*



(a) *long* (b) *short*

Fonte: Hull (2006).

Exemplo 1. Pelas taxas apresentadas na Tabela 1, se duas partes celebrarem em 02 de junho de 2017 um contrato *forward* de um ano no valor de US$ 1.000.000,00, na data de liquidação, a parte compradora deverá pagar R$ 3.473.200,00 à parte vendedora, e esta deverá entregar a quantidade de dólares americanos acordada. Supondo que em 04 de junho de 2018 a taxa *spot* S_T de USDBRL seja 3,5000, a parte em posição *long* registraria ganho, pois estaria comprando dólares por um valor abaixo do valor de mercado. Por outro lado, se a taxa de conversão USDBRL estiver em 3,4000, o ganho será da parte em posição *short*.

Outra observação importante levantada por Hull (2006) é que podemos relacionar a taxa *spot* S_0 e a taxa *forward* F_0 utilizando o conceito da paridade de taxa de juros:

$$F_0 = S_0 e^{(r_d - r_f)\tau} \tag{2.3}$$

onde r_d é a taxa de juros livre de risco da moeda doméstica; r_f é a taxa de juros livre de risco da moeda estrangeira e τ é o prazo até a data de maturidade T.

A Tabela 1 traz exemplos da relação *spot* e *forward* para diferentes prazos.

Tabela 1 – Taxa de conversão *spot* e *forward* de USDBRL em 02 de junho de 2017

	USDBRL
spot	3,2484
forward de um mês	3,2727
forward de seis meses	3,3148
forward de um ano	3,4732

Fonte – Citi Velocity

2.3 Opções

Diferentemente de um *forward*, uma opção dá ao investidor o direito, e não a obrigação, de comprar ou vender um ativo em uma data futura, chamada de data de liquidação ou maturidade (T), por um preço pré-determinado. Esse preço é denominado preço de exercício ou *strike* (K). Para iniciar um contrato desse tipo, a parte compradora deve fazer o pagamento para a parte vendedora de um valor conhecido como *prêmio*. Quando o contrato negociado dá um direito de compra, essa opção é chamada *call* (c) e, quando dá um direito de venda, é chamada *put* (p). O ativo em questão é denominado ativo objeto.

No mercado de câmbio especificamente, o prêmio da opção é expressado em valor percentual, como exemplificado na Tabela 2. Assim como em outros mercados, as opções de moedas podem ser classificadas, de acordo com seu estilo de liquidação, como Europeias ou Americanas. Numa opção Europeia, o comprador só pode escolher na data de liquidação se exerce ou não seu direito de compra ou venda do ativo objeto do contrato. Já no caso de uma Americana, ele pode fazer essa escolha a qualquer momento entre as datas de negociação e de maturidade.

Tabela 2 – Preço de opções de USDBRL de três meses em 02 de junho de 2017. O valor do *spot* nesta data era 3,2484 e do *forward* de mesmo vencimento da opção, 3,3148

Strike	*Call*	*Put*
3,0450	8,49%	0,32%
3,3148	2,96%	2,96%
3,5950	0,91%	9,31%

Fonte – Citi Velocity

Como o comprador tem o poder de escolha, ele somente exercerá a opção caso o resultado final gere lucro. Assim, em contraste ao *forward*, esse tipo de operação tem uma perda limitada ao prêmio pago. Logo, o resultado financeiro esperado por um detentor de uma *call* é:

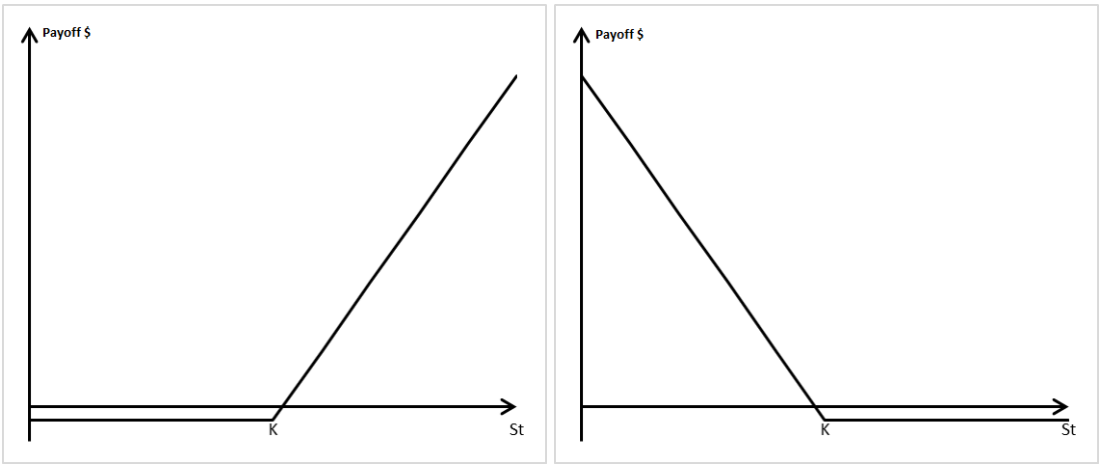
$$max(0, S_T - K) \tag{2.4}$$

e de uma *put*:

$$max(0, K - S_T), \tag{2.5}$$

conforme Figura 2.

Figura 2 – Resultado esperado de uma operação com opção



(a) *call* (b) *put*

Fonte: Hull (2006).

2.4 Regras de liquidação

Conforme mencionado nas seções anteriores, em uma operação de câmbio, seja ela *spot*, *forward* ou uma opção, é importante que algumas datas sejam observadas.

Numa operação *spot*, as moedas são trocadas geralmente dois dias úteis após a negociação. Essa data é conhecida como data *spot* e pode ser referenciada como D+2. Essa regra vale para a maioria dos pares de moeda, mas há exceções. Alguns pares podem ser operados com liquidação D+1 ou até mesmo D+0. (CLARK, 2011) lista algumas dessas exceções, conforme Tabela 3.

Tabela 3 – Exceções à regra de liquidação D+2

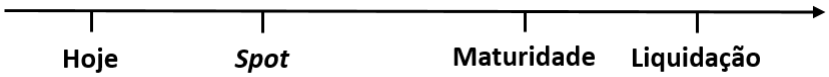
	Moeda Termo				
Moeda Base	USD	EUR	CAD	TRY	RUB
USD			D+1	D+1	D+1
EUR	D+2		D+2	D+1	D+1
CAD				D+1	D+1
TRY					D+1
RUB					

Fonte: Clark (2011, p. 6).

No caso de uma operação *forward*, há a data em que o acordo é estabelecido entre as partes e a data em que as moedas serão de fato trocadas, a data de liquidação. Já para opções, as datas importantes são a data de negociação, a data de pagamento do prêmio, a maturidade, quando a parte decide se vai exercer ou não seu direito de compra ou venda e a data da troca em si das moedas. Vale ressaltar que a quantidade de dias entre a data de negociação e a data de pagamento, e entre a data de maturidade e a data de liquidação seguem a regra D+X estabelecida anteriormente.

A Figura 3 traz uma linha do tempo contendo essas datas citadas.

Figura 3 – Datas de importância no mercado de câmbio



Fonte: Clark (2011, p. 2).

2.5 Considerações finais

Neste capítulo, além de introduzir alguns conceitos e terminologias importantes do mercado de câmbio, foram apresentados os principais instrumentos utilizados para a troca de moedas. Ao passo que numa operação *forward* as partes acordam a troca em uma data futura sem o pagamento de nenhum valor adiantado, em uma operação com opções uma parte compra de outra o direito e não a obrigação de realizar essa troca e, para isso, desembolsa um prêmio. Como há incerteza sobre o preço do ativo objeto na data de maturidade (S_T), a determinação desse prêmio não é uma tarefa trivial.

O próximo capítulo será dedicado a descrever um modelo de grande utilização no mercado para encontrar o valor justo para este prêmio, bem como para mensurar os riscos de uma opção.

CAPÍTULO

3

PRECIFICAÇÃO E RISCO DE OPÇÕES

Diversos ativos do mercado financeiro apresentam variação de preço por causa de diversos fatores. No caso de pares de moedas, sua cotação pode variar dependendo da política monetária dos bancos centrais, fluxo de investidores estrangeiros, etc. Desse modo, ao entrar em contrato de opção, a parte compradora deve pagar um prêmio que reflita a probabilidade desse contrato ser ou não exercido. Se existe uma baixa chance de exercício, o prêmio também deve ser baixo. Por outro lado, deve ser realizado um desembolso maior para o prêmio caso haja uma grande probabilidade de exercício.

Logo, o cálculo do prêmio deve ser capaz de "prever"a variação de preço do ativo objeto, levando em consideração os fatores específicos de cada classe. Outro ponto importante é que o prêmio da opção continua variando após início do contrato, trazendo riscos para as partes envolvidas.

Em 1973, foi criado por Fisher Black e Myron Scholes um modelo para precificação e mensuração de risco de opções de ações, que se tornou amplamente utilizado e ficou conhecido como modelo de Black-Scholes. Posteriormente esse modelo foi estendido para as demais classes de ativos. A seguir, será apresentada sua evolução até que sua utilização seja possível para o cálculo de opções de moedas.

3.1 Black-Scholes

Black e Scholes (1973) propuseram um modelo para precificar opções europeias em que o ativo subjacente fosse uma ação e, para tal, assumiram algumas premissas:

1. a taxa de juros livre de riscos é constante e conhecida;
2. o ativo objeto, no caso uma ação, não paga dividendos;
3. não há custos de transação;

4. é permitida a venda descoberta;
5. não há oportunidades de arbitragem;
6. as operações podem ser realizadas continuamente sem impactar o mercado.

O modelo proposto também assume que as variações do preço do ativo objeto em um curto espaço de tempo sigam uma distribuição normal. Conforme a descrição feita por Hull (2006), definimos μ como o retorno esperado e σ como a volatilidade do ativo objeto. Em um espaço de tempo Δt o retorno esperado é dado por $\mu\Delta t$ e o desvio padrão desse retorno por $\sigma\sqrt{\Delta t}$.

Conforme levantado por Hoek (2012), o preço de uma ação é uma variável cuja variação no tempo é imprevisível e, portanto, pode ser representada por um processo estocástico. Além disso, é esperado que as variações de preço no passado e no futuro sejam independentes tornando esse um processo sem memória e satisfazendo a propriedade Markov. Dadas essas propriedades, o preço do ativo objeto no tempo pode ser modelado por um movimento Browniano.

Definição 1. Um processo $\{B_t : 0 \leqslant t < \infty\}$ contínuo no tempo é dito movimento Browniano se satisfaz as seguintes condições:

1. $B_0 = 0$;
2. Se $t_1 < t_2 \leqslant t_3 < t_4$ então os incrementos $B_{t_2} - B_{t_1}$ e $B_{t_4} - B_{t_3}$ são independentes ;
3. $B_t - B_s \sim N(0, t-s)$ para $0 \leqslant t \leqslant s$;
4. $B = \{B_t : 0 \leqslant t < \infty\}$ tem trajetórias contínuas.

O movimento Browniano descrito acima não possui *drift*, e, portanto, o seu valor esperado a qualquer momento no futuro é igual ao seu valor atual. Entretanto, foi dito anteriormente que o retorno esperado do ativo objeto é dado por $\mu\Delta t$. Assim, o modelo Black-Scholes assume uma equação diferencial estocástica (EDE) para modelar o preço da ação em questão, representada por um movimento Browniano geométrico, dada na seguinte foma:

$$dS_t = \mu dS_t dt + \sigma S_t dBt.$$

Aplicando o lema de Itô, pode-se facilmente verificar que:

$$dln(S_t) = \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dBt.$$

Vale notar que $ln(S_t)$ possui *drift* e volatilidade constantes e, portanto, segue uma distribuição normal. Como consequência, tem-se que $ln(S_T)$ segue uma distribuição lognormal:

$$ln(S_T) \sim N\left(ln(S_0) + \left(\mu - \frac{1}{2}\sigma^2\right)T, \sigma^2 T\right)$$

$$S_T \sim N\left(ln(S_0) + \left(\mu - \frac{1}{2}\sigma^2\right)T, \sigma^2 T\right)$$

$$\mathbf{E}[S_T] = S_0 e^{\mu T}$$

Com base nesse modelo capaz de descrever o comportamento do preço de uma ação, Black e Scholes (1973) demonstram através do lema de Itô, que o preço de uma *call* europeia de valor $V(S,t)$ pode ser descrito pela seguinte equação diferencial parcial:

$$dV = \left(\frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right)dt + \sigma S \frac{\partial V}{\partial S} dB.$$

Baseado no princípio de não arbitragem é possível construir uma carteira livre de risco, composta pela *call* descrita acima e por uma certa quantidade do ativo objeto, e finalmente chegar à equação diferencial parcial de Black e Scholes (1973):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.1}$$

Resolvendo a Equação 3.1 considerando o resultado esperado da opção do tipo *call* europeia que vem sido utilizada nesta seção, pode-se chegar a uma solução analítica para o preço:

$$C(S,t) = S\phi(d_1) - Ke^{-r\tau}\phi(d_2) \tag{3.2}$$

onde $\phi(.)$ é a função de distribuição acumulada de uma distribuição normal padrão e τ é o prazo até a data de maturidade da opção $(T-t)$. Os termos d_1 e d_2 são dados por:

$$d_1 = \frac{ln(S/K) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}},$$

e

$$d_2 = d_1 - \sigma\sqrt{\tau}.$$

3.1.1 Paridade call-put

Para que não haja condições de arbitragem, Stoll (1969) afirma que o preço de *call* implica um certo preço justo para uma *put* de mesmo *strike* e data de maturidade. Hull (2006) demonstra tal afirmação com a utilização de duas carteiras:

- Carteira A: uma *call* europeia mais uma quantidade em dinheiro igual a $Ke^{-r\tau}$
- Carteira B: uma *put* com mesmo vencimento mais uma quantidade do ativo objeto

Como as duas carteiras possuem o mesmo valor esperado $max(S_T, K)$, e as opções são do tipo europeias, os seus valores presentes devem ser iguais. Assim, a paridade *call-put* pode ser definida por:

$$c + Ke^{-r\tau} = p + S_0. \tag{3.3}$$

Dessa forma, conclui-se que uma *put* também europeia tem seu preço dado por:

$$P(S,t) = Ke^{-r\tau}\phi(-d_2) - S\phi(-d_1). \tag{3.4}$$

3.2 Black 1976

Com algumas modificações, o modelo de Black-Scholes pode ser estendido para as demais classes de ativos financeiros. Na implementação do sistema proposto neste trabalho, as opções de moedas serão precificadas utilizando as fórmulas derivadas do modelo de Black (1976), inicialmente proposto para a precificação de opções de futuros. O porquê de tal utilização será explicado a seguir.

Começando pelo caso de ações que pagam dividendos, Hull (2006, p. 313) mostra-nos que o valor inicial da ação S_0 pode ser reduzido pela taxa de dividendo q resultando em $S_0e^{-q\tau}$. Fazendo essa substituição nas equações (3.2) e (3.4), obtem-se:

$$c = S_0e^{-q\tau}\phi(d_1) - Ke^{-r\tau}\phi(d_2) \tag{3.5}$$

$$p = Ke^{-r\tau}\phi(-d_2) - S_0e^{-q\tau}\phi(-d_1) \tag{3.6}$$

Como:

$$ln\left(\frac{S_0e^{-q\tau}}{K}\right) = ln\frac{S_0}{K} - q\tau$$

os parâmetros d_1 e d_2 são reescritos na forma:

$$d_1 = \frac{ln(S_0/K) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_2 = d_1 - \sigma\sqrt{\tau}$$

Ao mudar para o mercado de opções de moedas, pode ser utilizada uma analogia novamente de Hull (2006, p. 115), que compara um par de moedas a um ativo pagando uma taxa de juros ou dividendos conhecida, nesse caso $q = r_f$. Com isso, partindo da Equação 2.3 e substituindo em (3.5) e (3.6), tem-se que o preço de uma opção de moedas é dado por

$$c = e^{-rT}[F_0\phi(d_1) - K\phi(d_2)] \tag{3.7}$$

$$p = e^{-rT}[K\phi(-d_2) - F_0\phi(-d_1)] \tag{3.8}$$

onde:

$$d_1 = \frac{ln(F_0/K) + \sigma^2 T/2}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

As fórmulas (3.7) e (3.8) são idênticas às fórmulas do modelo proposto por Black (1976) para precificação de opções de futuros. De fato, quando a taxa livre de risco é constante ou varia em função do tempo, o preço de um futuro é igual ao preço de *forward* com vencimento na mesma data. Assim, fica explicada a utilização deste modelo durante o desenvolvimento do sistema.

3.3 Gregas

Observando as equações que dão o preço de opções *call* e *put*, é fácil verificar que essas são funções que dependem de pelo menos cinco parâmetros: valor do ativo objeto, volatilidade, taxa de juros, *strike* e tempo. Logo, o valor de uma opção apresenta sensibilidade em decorrência da alteração destes parâmetros. A esse conjunto de sensibilidades é dado o nome de Gregas e algumas delas serão detalhadas a seguir.

Vale recapitular que V denota o preço de uma opção e que as derivações apresentadas nas próxima seções originam do modelo de Black (1976). Portanto, nas fórmulas que seguem, a variação do valor do ativo será representada pelo *forward* F, uma vez que eles possuem relção direta.

3.3.1 Delta

O *delta* (Δ) é a derivada do preço da opção em relação ao preço do ativo objeto.

$$\Delta = \frac{\partial V}{\partial F}$$

Ele mede o quanto a variação do preço do ativo objeto impacta o preço da opção. No nosso caso:

$$\Delta_c = e^{-r\tau}\phi(d_1) \tag{3.9}$$

e

$$\Delta_p = e^{-r\tau}(\phi(d_1) - 1). \tag{3.10}$$

Na construção de uma carteira de investimentos com opções, o Δ indica a quantidade necessária de unidades do ativo objeto para torná-la livre de risco. Entretanto, quando o preço do ativo muda, o Δ da opção também muda e é necessário recalibrar a carteira. Esse conceito é chamado de *delta-hedging*.

Pode-se notar que o Δ é sempre positivo para uma *call*, o que faz sentido, pois um aumento no preço do ativo objeto implica o aumento da probabilidade de exercício da opção. Por outro lado, ele é negativo para uma *put*, pois a direção da variação do preço desta é inversa à direção da variação do preço do ativo.

3.3.2 Gamma

A segunda derivada do preço da opção em relação ao preço do ativo objeto é chamada de *gamma* (Γ).

$$\Gamma = \frac{\partial^2 V}{\partial F^2} = \frac{\partial \Delta}{\partial F}$$

Essa medida de sensibilidade indica o quanto o Δ vai variar em relação ao movimento de preço do ativo. Um Γ alto indica que mudanças no preço do ativo são rapidamente refletidas no *delta*, e, assim, o *delta-hedge* deve ser realizado com maior frequência. Seguindo o mesmo raciocínio, o rebalanceamento da carteira pode ser feito com menor frequência quando o Γ é baixo.

Para o modelo utilizado neste trabalho, o *gamma* pode ser calculado por:

$$\Gamma_c = \Gamma_p = e^{-r\tau}\frac{\varphi(d_1)}{F\sigma\sqrt{\tau}}. \tag{3.11}$$

onde $\varphi(.)$ é a função densidade de probabilidade da distribuição normal padrão.

3.3.3 Vega

Dá-se o nome de *vega* (ν) à derivada do preço da opção em relação à volatilidade.

$$\nu = \frac{\partial V}{\partial \sigma}$$

Ele indica o quanto o preço da opção varia no caso de aumento ou diminuição da volatilidade implícita. Quanto maior o ν, maior a sensibilidade da opção a pequenas mudanças de volatilidade.

A equação do ν tanto para opções do tipo *call* quanto para do tipo *put* no modelo de Black é dada por:

$$\nu_c = \nu_p = Fe^{-r\tau}\varphi(d_1)\sqrt{\tau}. \tag{3.12}$$

3.3.4 Theta

O *theta* (θ) é a derivada do preço da opção em relação ao tempo.

$$\theta = \frac{\partial V}{\partial t}$$

Também conhecido como decaimento pelo tempo, ele indica o quanto o valor da opção é alterado devido apenas a passagem do tempo e é geralmente negativo.

O valor do θ para o modelo base deste trabalho pode ser obtido pelas seguinte equações:

$$\theta_c = -\frac{Fe^{-r\tau}\varphi(d_1)\sigma}{2\sqrt{\tau}} + rFe^{-r\tau}\phi(d_1) - rKe^{-r\tau}\phi(d_2) \tag{3.13}$$

e

$$\theta_p = -\frac{Fe^{-r\tau}\varphi(d_1)\sigma}{2\sqrt{\tau}} - rFe^{-r\tau}\phi(-d_1) + rKe^{-r\tau}\phi(-d_2). \tag{3.14}$$

3.3.5 Rho

Nomeia-se *rho* (ρ) a derivada do preço da opção em relação à variação da taxa de juros.

$$\rho = \frac{\partial V}{\partial r}$$

No modelo de Black 1976, seu valor é dado por:

$$\rho_c = -\tau c \tag{3.15}$$

e

$$\rho_p = -\tau p. \tag{3.16}$$

Apesar de ter sido apresentado nesse texto, o sistema objeto desse trabalho não traz a implementação do cálculo do *rho*.

3.4 Considerações finais

A publicação do modelo de Black-Scholes representou um grande marco no mundo de opções, possibilitando encontrar o prêmio de maneira analítica e simplificada. Apesar de ser limitado a opções de ações e de possuir premissas que não refletem o comportamento do mercado real, sua dinâmica serviu como ponto de partida para outros modelos que conseguem contornar essas premissas e que também o adaptaram para outras classes de ativos.

A volatilidade, por exemplo, não é constante como assume o modelo. Assim, faz-se necessária a utilização de algum mecanismo capaz de traduzir esse comportamento para aplicação no modelo. Tal mecanismo será detalhado no próximo capítulo.

CAPÍTULO

4

SUPERFÍCIE DE VOLATILIDADE

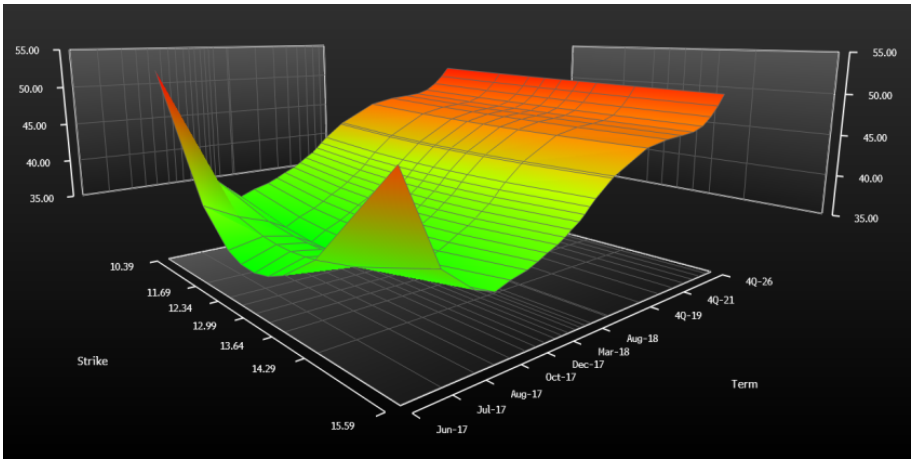
Uma das premissas do modelo de Black-Scholes, citada no Capítulo 3, é a volatilidade constante. Entretanto, no mundo real a volatilidade varia com o prazo e com o *strike* ou *delta* da opção. No mercado é possível encontrar valores de volatilidade para opções com prazos e *strikes* ou *deltas* mais líquidos. Essas volatilidades são chamadas de volatilidade implícita pois, quando aplicadas na fórmula de Black-Scholes, refletem os preços das opções observados no mercado.

O conjunto de volatilidades para *strikes* ou *deltas* de um mesmo prazo possui uma assimetria e é comumente chamado de *skew*. Já para a variação da volatilidade de acordo com o prazo damos o nome de estrutura de volatilidade no tempo. Por fim, uma superfície ou *smile* de volatilidade é formada pelo conjunto de *skews* e estruturas de volatilidade no tempo. A partir dessa superfície, pode-se obter os valores de volatilidade implícita para as opções com menor liquidez e até mesmo precificar derivativos mais complexos (CATTARUZZI, 2009).

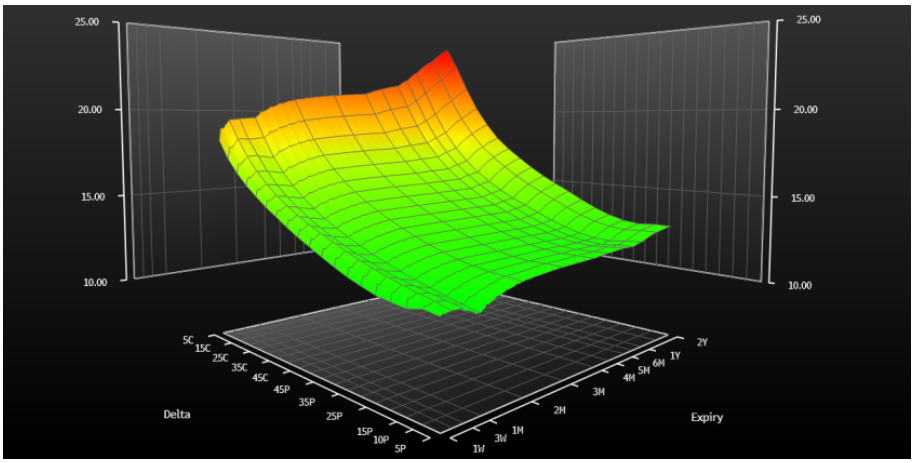
Em mercados como o de ações ou índices, a volatilidade de um determinado prazo e *strike* permanece inalterada à medida que o valor do ativo objeto se move e, por isso, diz-se que a superfície possui uma característica *sticky-strike*. Por outro lado, o mercado de câmbio possui uma característica *sticky-delta* pois, à medida que o valor do *spot* se altera, a volatilidade implícita de opções com mesmo delta permanece inalterada (DERMAN, 1999). A Figura 4 traz exemplo de uma superfície *sticky-strike*, construída a partir de volatilidades de opções sobre a ação PETR4, e de uma superfície *sticky-delta*, elaborada com base em volatilidades do par de moedas USDBRL.

As seções seguintes serão destinadas a descrever um método para que, a partir de informações observáveis no mercado de câmbio, seja possível construir uma superfície de volatilidade *sticky-delta*. Também será apresentada uma forma de interpolar essa superfície a fim de se encontrar volatilidades que servirão como parâmetro de entrada para o modelo de Black-Sholes.

Figura 4 – Exemplos de superfícies de volatilidade de opções de PETR4 e USDBRL em 02 de junho de 2017



(a) *sticky-strike*



(b) *sticky-delta*

Fonte – Bloomberg

4.1 Construção da superfície a partir das volatilidades de mercado

A fim de construir uma superfície de volatilidade, que no caso de opções de moedas será *sticky-delta*, deve-se utilizar as informações disponíveis no mercado. Entretanto, ao consultar as volatilidades, estas são apresentadas não no formato por *delta* mas sim como *at-the-money* (ATM), *risk-reversal* (RR) e *market-strangle* (STR).

Como descrito por França (2010, p 37), uma opção *at-the-money* é aquela cujo *strike* é igual ao *forward* de mesmo vencimento e, então, a volatilidade ATM refere-se à volatilidade implícita dessa opção. Numa aproximação simples, pode-se afirmar que o *delta* dessa opção é

igual a 50% e que este é o ponto de partida para construção da superfície. Logo:

$$\sigma_{50\Delta} = \sigma_{ATM}. \tag{4.1}$$

Os termos *risk-reversal* e *market-strangle* referem-se a estratégias formadas por mais de uma opção. Tomando como partida uma *call* e uma *put* de mesmo *delta*, por exemplo 25%, o *risk-reversal* mede a assimetria ou diferença entre as volatilidades destas duas opções, ou seja:

$$\sigma_{RR25\Delta} = \sigma_{25\Delta c} - \sigma_{25\Delta p}. \tag{4.2}$$

Já o *strangle* representa a diferença entra a média das volatilidades dessas duas opções e a volatilidade de uma terceira opção do tipo ATM:

$$\sigma_{STR25\Delta} = \frac{\sigma_{25\Delta c} + \sigma_{25\Delta p}}{2} - \sigma_{ATM}. \tag{4.3}$$

Fazendo alguns rearranjos e substituindo (4.2) em (4.3), podemos facilmente chegar às seguintes fórmulas para cálculo da volatilidade por *delta*:

$$\sigma_{25\Delta c} = \sigma_{ATM} + \sigma_{STR25\Delta} + 0.5 \times \sigma_{RR25\Delta} \tag{4.4}$$

e

$$\sigma_{25\Delta p} = \sigma_{ATM} + \sigma_{STR25\Delta} - 0.5 \times \sigma_{RR25\Delta}. \tag{4.5}$$

Neste trabalho, serão utilizadas superfícies compostas pelos seguintes *deltas*: 10% *call*, 10% *put*, 25% *call*, 25% *put* e 50% *call/put*. Para tal, faz-se necessário coletar do mercado as volatilidades ATM, RR25, STR25, RR10 e STR10, como exemplificado na Tabela 4.

A Tabela 5 utiliza as equações (4.1), (4.4) e (4.5) para traduzir as volatilidades da Tabela 4 em uma superfície por *deltas*.

4.2 Interpolação da superfície

Muitas vezes o gestor de uma carteira de opções realiza uma operação para um prazo e *delta* com observação não disponível no mercado. Então, a fim de precificar e calcular as gregas dessas opções, é necessário interpolar a superfície para encontrar a volatilidade implícita específica para a opção em questão.

Dado um prazo t e *delta* não observáveis, deve-se primeiro encontrar o *skew* de cada um dos dois prazos t_i e t_{i+1} existentes na superfície que delimitam o prazo desejado, ou seja, $t_i \leq t \leq t_{i+1}$. Em cada um desses *skews* é necessário realizar uma interpolação qualquer que

Tabela 4 – Volatilidades ATM, RR e STR para opções de USDBRL em 02 de junho de 2017

Prazo	ATM	RR10	RR25	STR10	STR25
1D	11,50%	4,54%	2,45%	1,21%	0,29%
1W	14,69%	4,41%	2,84%	1,05%	0,27%
2W	14,89%	5,14%	2,76%	1,11%	0,34%
3W	14,41%	5,20%	2,83%	1,27%	0,42%
1M	14,23%	5,45%	2,88%	1,29%	0,43%
2M	14,64%	5,58%	3,03%	1,54%	0,49%
3M	14,62%	5,65%	3,10%	1,70%	0,53%
6M	14,55%	6,06%	3,23%	2,05%	0,63%
9M	14,60%	6,36%	3,38%	2,13%	0,65%
1Y	14,62%	6,69%	3,52%	2,34%	0,69%
18M	15,14%	7,15%	3,76%	2,39%	0,73%
2Y	15,42%	7,74%	3,98%	2,57%	0,74%
3Y	16,59%	9,08%	4,66%	3,24%	0,81%
5Y	18,20%	11,73%	5,89%	3,87%	0,94%

Fonte – Bloomberg

Tabela 5 – Superfície por delta calculada a partir da Tabela 4 e das equações (4.1), (4.4) e (4.5)

Prazo	10C%	25C%	50C ou 50P%	25P%	10P%
1D	14,97%	13,01%	11,50%	10,56%	10,44%
1W	17,94%	16,38%	14,69%	13,54%	13,53%
2W	18,57%	16,61%	14,89%	13,85%	13,43%
3W	18,28%	16,24%	14,41%	13,42%	13,08%
1M	18,25%	16,10%	14,23%	13,22%	12,80%
2M	18,98%	16,65%	14,64%	13,62%	13,39%
3M	19,14%	16,70%	14,62%	13,60%	13,49%
6M	19,62%	16,79%	14,55%	13,56%	13,57%
9M	19,91%	16,93%	14,60%	13,55%	13,55%
1Y	20,30%	17,07%	14,62%	13,54%	13,61%
18M	21,10%	17,75%	15,14%	13,98%	13,95%
2Y	21,86%	18,15%	15,42%	14,16%	14,12%
3Y	24,36%	19,73%	16,59%	15,07%	15,29%
5Y	27,93%	22,08%	18,20%	16,20%	16,20%

Fonte: Elaborada pelo autor.

garanta as condições de não arbitragem. No sistema objeto deste trabalho, essas condições foram relaxadas e foram utilizados *splines* cúbicos naturais, implementados conforme Apêndice A. Como cada *skew* tende a ser uma curva suave, esse tipo de interpolação se ajusta melhor do que outra do tipo linear.

De posse das duas volatilidades interpoladas nos *skews* de t_i e t_{i+1}, para encontrar um valor de volatilidade implícita final, Clark (2011, p. 63-67) sugere uma interpolação *flat-forward* na volatilidade, que seria similar a uma interpolação linear na variância total entre os prazos,

dada por:

$$\sigma_{imp}^{flat\,fwd}(t) = \begin{cases} \sigma_1 & \text{se } t < t_1 \\ \sqrt{[\sigma^2 t_i + \sigma_{i,i+1}^2 (t - t_i)]/t} & \text{se } t_i \leq t < t_{i+1} \text{ para } i < N \\ \sigma_N & \text{se } t \geq t_N \end{cases} \tag{4.6}$$

onde:

$$\sigma_{i,i+1}^2 = \frac{\sigma_{i+1}^2 t_{i+1} - \sigma_i^2 t_i}{t_{i+1} - t_i}.$$

4.3 Estimando a volatilidade implícita a partir do *forward*

Até agora foi falado em obter a volatilidade implícita de uma opção a partir do seu prazo e *delta*. Entretanto, como foi visto no Capítulo 3, o *delta* é uma função que depende da volatilidade entre outros parâmetros. Faz-se necessário então a utilização de um método iterativo, descrito pelo Algoritmo 1, que, a partir dos dados da opção e um chute inicial para o *delta*, vá refinando a busca até encontrar a volatilidade correta.

Algoritmo 1 – Algoritmo de busca da volatilidade implícita a partir do *forward*

```
1:  função VolImplicita(F, K, r, T, Superficie)
2:      Δ ← 0.5
3:      σ_imp ← Superficie.Volatilidade(T, Δ)
4:      erro ← 1
5:      precisao ← 1E − 5
6:      i ← 0
7:      enquanto erro > precisao e i < 100 faça
8:          Δ ← Delta(F, K, r, T, σ_imp)
9:          σ_anterior ← σ_imp
10:         σ_imp ← Superficie.Volatilidade(T, Δ)
11:         erro ← Modulo(σ_anterior − σ_imp)
12:         i ← i + 1
13:     fim enquanto
14:     retorna σ_imp
15: fim função
```

Exemplo 2. Suponha que em 02 de junho de 2017, um gestor tenha comprado uma opção de USDBRL do tipo *call* com vencimento T em 40 dias e *strike* $K = 3,4000$. É dado que no momento da compra $F = 3,2867$.

Com os dados fornecidos, é possível utilizar o Algoritmo 1 para obter a volatilidade implícita dessa opção. Para o chute inicial de *delta* igual a 50% e aplicando sobre a Tabela 5 o

mecanismo de interpolação indicado anteriormente, chega-se uma volatilidade $\sigma_{imp} = 14,4154\%$. A partir daí, a volatilidade implícita final $\sigma_{imp} = 15,2359\%$ pode ser encontrada por meio do processo iterativo ilustrado na Tabela 6.

Tabela 6 – Demonstração do processo iterativo para o Exemplo 2

i	Δ	σ_{imp}	$\sigma_{anterior}$	*erro*
	50,0000%	14,4154%		
0	35,7254%	15,2878%	14,4154%	0,008723
1	36,4415%	15,2329%	15,2878%	0,000548
2	36,3989%	15,2361%	15,2329%	0,000032
3	36,4014%	15,2359%	15,2361%	0,000001

Fonte: Elaborada pelo autor.

4.4 Considerações finais

Neste capítulo foi demonstrado como a superfície de volatilidade permite tomar como base as poucas volatilidades implícitas encontradas no mercado para se calcular a volatilidade de qualquer prazo e *delta* desejado.

De posse dessa ferramenta, do modelo de Black 1976 e das noções de mercado apresentadas até aqui, é possível precificar e mensurar os riscos de uma opção de moedas. Esses conhecimentos servem de base para o sistema proposto neste trabalho, que será detalhado no próximo capítulo.

CAPÍTULO

5

SIRI: SISTEMA DE INFORMAÇÕES DE RISCO INTEGRADAS

Os gestores de carteiras de investimentos estão a todo instante observando as variáveis de mercado e avaliando o momento certo para aumentar ou diminuir sua posição em um ou mais ativos a fim de rentabilizar o investimento de seus clientes. Geralmente esses gestores utilizam para o controle da carteira simples planilhas do Microsoft Excel ligadas a sistemas que fornecem em tempo real as variações de preço dos ativos, tais como Bloomberg Professional, Thomson Reuter Eikon, AE Broadcast, etc. Nessas planilhas, eles programam fórmulas que calculam o resultado de cada posição, dados os valores de entrada, e também podem fazer simulações de como a carteira se comportaria em algum cenário de *stress* pré-determinado.

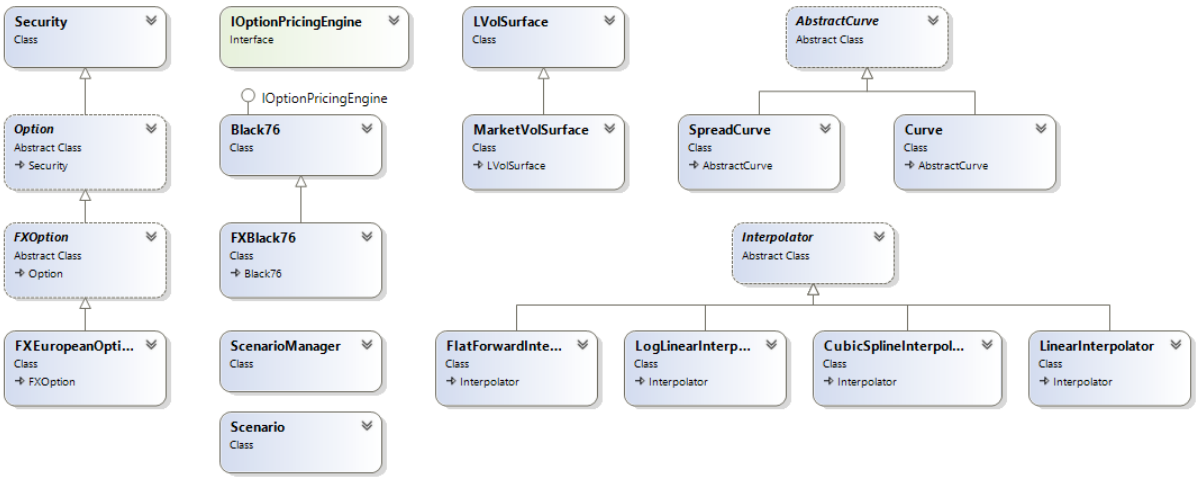
Para ativos como ações, alguns tipos de *bonds* ou até mesmo derivativos mais simples que dependem de poucas variáveis, essas planilhas apresentam um desempenho razoável e são suficientes para o gerenciamento da carteira. Entretanto, como observado nos capítulos anteriores, o mundo de opções exige uma série de cálculos que, dependendo da complexidade do cenário ou cenários que se deseja testar, podem onerar a velocidade de processamento da planilha e não entregar os resultados a tempo para que se possa tomar uma decisão sobre a composição da carteira.

Grandes bancos ou até mesmo gestoras com um grande volume de recursos sob gestão recorrem a sistemas sofisticados para o gerenciamento de seus *portfolios*. Esses sistemas exigem grandes investimentos em licenças, infraestrutura, treinamento e manutenção, tornando-se inviáveis para empresas de menor porte. Assim, surgiu a ideia do desenvolvimento de um sistema no formato de *add-in* que pode ser acoplado às planilhas já utilizada pelos gestores, mas que leva o cálculo mais intensivo para um ambiente de mais alto nível e melhor desempenho e que será detalhado nas próximas seções. Este é o SIRI: Sistema de Informações de Risco Integradas.

5.1 Arquitetura

Por ter sido criado pela Microsoft e, com isso, oferecer algumas facilidades para integração com o Excel, o C# foi a linguagem escolhida para o desenvolvimento do sistema. Ademais, sua natureza orientada a objetos permite a reutilização de código além da herança e polimorfismo das classes, que agilizam a programação e facilitam expansões futuras. Na Figura 5, pode ser observado o diagrama de classes do sistema.

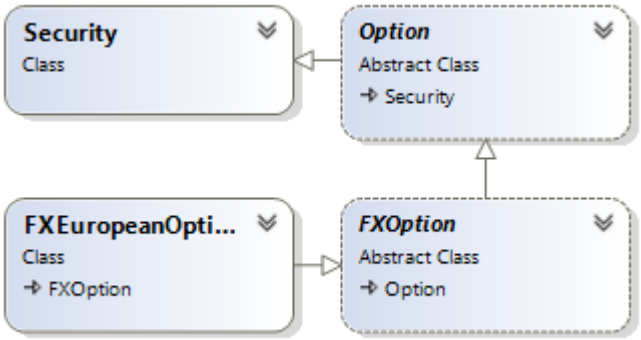
Figura 5 – Diagrama de classes



Fonte: Elaborada pelo autor.

A Figura 6 traz a hierarquia da classe `FXEuropeanOption`, que implementa as especificidades de uma opção de moedas do tipo Europeia. Nota-se que ela herda de `FXOption`, que por sua vez herda de `Option`, detalhada no Apêndice B, e que, finalmente, herda de `Security`, que é a classe base para todos os ativos financeiros. Dessa forma, quando for necessário incluir um novo tipo de ativo, ou de opção ou até mesmo de opção de moedas, vários métodos poderão ser reaproveitados.

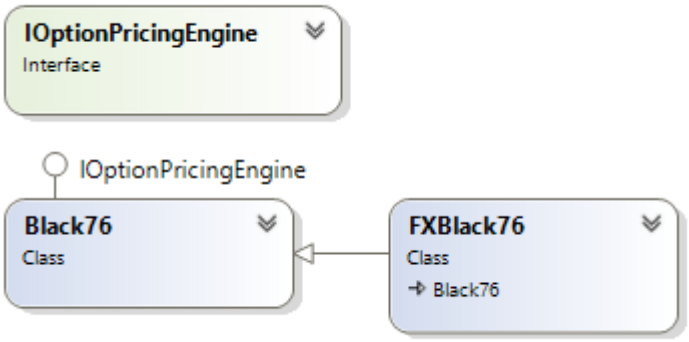
Figura 6 – Hierarquia das classes que representam ativos financeiros



Fonte: Elaborada pelo autor.

A classe `Option` tem como atributos os parâmetros da opção, como tipo (*call* ou *put*), *strike*, data de vencimento, superfície de volatilidade, curva de juros e ativo objeto. Além disso, essa classe possui um outro atributo do tipo `IOptionPricingEngine`, que é uma interface que define os métodos responsáveis pela precificação e cálculo de risco, como prêmio e gregas. O modelo de Black (1976) apresentado no Capítulo 3, por exemplo, foi codificado na classe `Black76`, que implementa essa interface e que contém os cálculos descritos nas equações (3.7) a (3.14). Por sua vez, esta é estendida pela classe `FXBlack76`, como pode ser visto na Figura 7, que é instanciada pela `FXEuropeanOption`.

Figura 7 – Hierarquia dos modelos de precificação



Fonte: Elaborada pelo autor.

5.2 Implementação da superfície

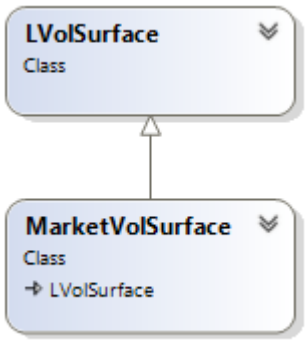
Como observado no Capítulo 4, a superfície de volatilidade pode ser apresentada no formato *sticky-strike* ou *sticky-delta*, dependendo do mercado. Entretanto, essa diferença diz respeito apenas à forma de interpretação e utilização dos valores. Matematicamente, isso quer dizer que ambos os formatos podem ser representados da mesma maneira, mas que no primeiro caso o eixo X representa *strikes*; e, no segundo, *deltas*. No eixo Y, encontram-se os prazos; e, no eixo Z, as volatilidades.

Para cada *skew*, conjunto de volatilidades para um mesmo prazo, é necessário um interpolador que ligue os seus pontos. Logo, para uma superfície com n prazos, são necessários n interpoladores. Finalmente, acrescentando a Equação 4.6 chega-se a um mecanismo completo para utilização de uma superfície de volatilidade, conforme implementado na classe `LVolSurface`, detalhada no Apêndice C.

Avançando na representação da superfície, faz-se necessária uma forma de receber as volatilidades no formato observado no mercado de câmbio. Para tal finalidade foi criada a classe `MarketVolSurface`, que estende `LVolSurface`, conforme ilustrado na Figura 8, e cuja implementação pode ser encontrada no Apêndice D. Essa classe recebe como entrada as

volatilidades ATM, RR e STR e as converte para a classe pai utilizando as equações (4.1), (4.4) e (4.5).

Figura 8 – Hierarquia da superfície de volatilidade



Fonte: Elaborada pelo autor.

5.3 Integração com o Excel

Como um dos objetivos do sistema proposto era a integração com o Microsoft Excel, seria necessária ou a implementação de interfaces disponibilizadas pela própria Microsoft ou a utilização de alguma biblioteca que facilitasse essa integração. Após algumas pesquisas foi decidida a utilização do Excel-DNA (DRIMMELEN, 2006–2017), uma biblioteca de *software* de código aberto distribuída sob licença MIT.

O Excel-DNA permite a criação de comandos que podem ser disponibilizados em uma aba de botões personalizados e de funções que podem ser chamadas dentro de uma célula da planilha. Para tal, a biblioteca disponibiliza um *add-in* no formato `.xll` que deve ser importado no Excel e um arquivo de configuração com extensão `.dna`. É nesse arquivo de configuração que se deve referenciar a biblioteca que traz as customizações que se deseja expor. A partir daí, o *add-in* busca na biblioteca todos os métodos estáticos que possuem a anotação `ExcelFunction` e os disponibiliza no formato de funções do Excel, conforme Figura 9.

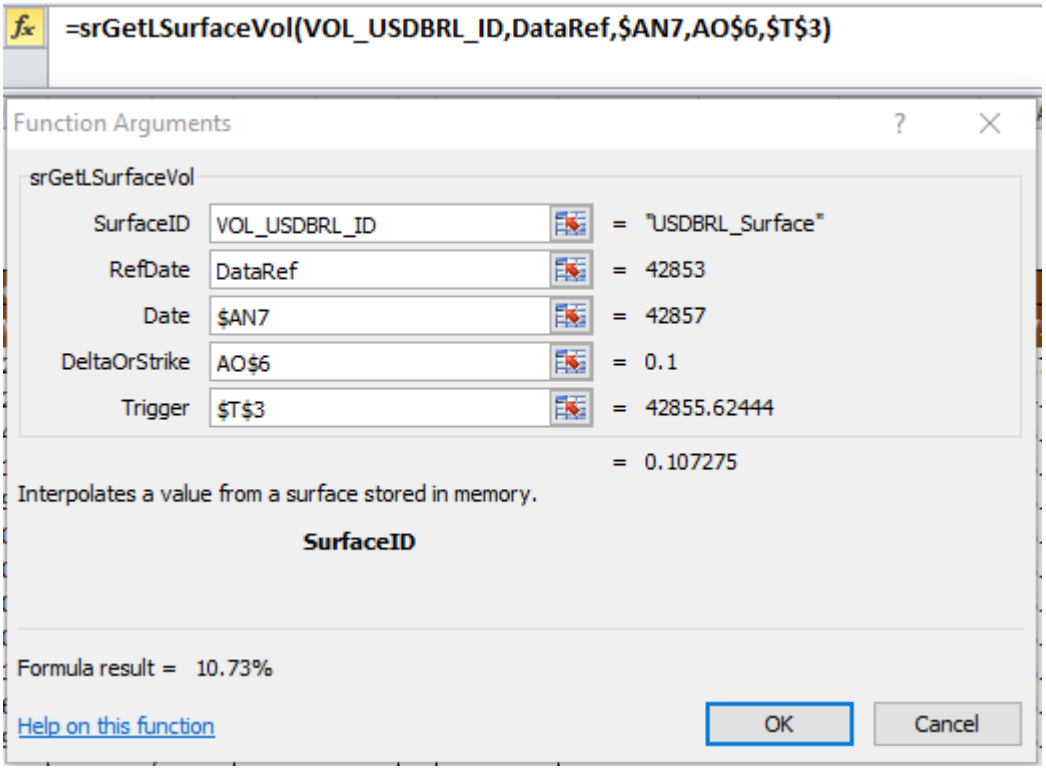
É no arquivo `.dna` também que são definidas as customizações da interface gráfica do Excel, podendo inserir botões que chamarão funções desenvolvidas no C#. Um exemplo de configuração pode ser visto no Código-fonte 1.

Código-fonte 1 – Exemplo de configuração do arquivo `.dna`

```
1: <DnaLibrary Name="SIRI Math" RuntimeVersion="v4.0"
     ShadowCopyFiles="true">
2:   <ExternalLibrary Path="SIRI-Math.Excel.dll" LoadFromBytes="
     true" />
3:   <Reference Path="SIRI-Math.dll" LoadFromBytes="true" />
```

```
4:   <Reference Path="SIRI-Cal.dll" LoadFromBytes="true" />
5:
6:   <CustomUI>
7:     <customUI xmlns='http://schemas.microsoft.com/office
     /2009/07/customui' loadImage='LoadImage'>
8:       <ribbon>
9:         <tabs>
10:          <tab id='tabSiriVol' label='SIRI Vol'>
11:            <group id='grpSimulation' label='Simulation'>
12:              <button id='btnRunSimulation' label='Run
     Simulation' size='normal' onAction='RunTagMacro' tag='
     RunOptionPortfolioSimulation'/>
13:            </group >
14:          </tab>
15:        </tabs>
16:      </ribbon>
17:    </customUI>
18:  </CustomUI>
19: </DnaLibrary>
```

Figura 9 – Exemplo de função exposta ao Excel



Fonte: Elaborada pelo autor.

5.4 *Triggers*

Ao longo do desenvolvimento foram disponibilizadas funções seguindo a mesma lógica de utilização de objetos dentro do código: primeiro um novo objeto deve ser instanciado para que depois seus métodos possam ser invocados. Por exemplo, quando o usuário deseja criar uma superfície de volatilidade ele deve chamar a função `srCreateMarketVolSurface` para que essa seja inicializada e guardada em memória. Depois, deve aplicar as volatilidades pela função `srSetMarketSurfaceVols`. Então, poderá utilizar a função `srGetLSurfaceVol` para interpolar alguma volatilidade.

Analogamente, ativos financeiros devem ser instanciados antes de serem precificados. A função `srFXEuropeanOption`, por exemplo, cria uma opção de moedas do tipo europeia a partir dos parâmetros de entrada. Logo após, a função `srFXOptionEvaluate` pode ser chamada para se obter dados como prêmio e gregas.

Quando o Excel recebe um comando para recalcular a planilha como um todo, ele cria uma árvore de dependências e executa as funções de células independentes de forma paralela; e, as células com dependência, de modo ordenado. Não há como ele respeitar dependências internas ao código do *add-in*. Assim, para que as funções do SIRI sejam executadas de maneira organizada, foi adicionado um parâmetro *trigger* em cada uma delas para que o usuário seja responsável por apontar as células de que o cálculo atual seja dependente, tirando proveito dessa árvore de dependências.

5.5 Simulação de cenários

Uma das formas de se testar os resultados de uma carteira sob condições de *stress* de mercado é a definição de cenários compostos por variações pré-definidas dos fatores de risco que influenciam essa carteira. Um gestor pode querer, por exemplo, saber qual seria seu resultado e sua exposição de risco caso o valor do *spot* suba 3% e a volatilidade 1%.

O mecanismo de simulação proposto no sistema objeto deste trabalho permite que o gestor defina uma lista de variações para o *spot*, volatilidades *at-the-money* e *risk-reversal*, e também para a passagem do tempo. Cada combinação possível dentre todas essas variações definidas é considerada um cenário. Supondo uma simulação com quinze variações do *spot*, cinco da volatilidade ATM, três do tempo e duas da volatilidade RR, teremos quatrocentos e cinquenta cenários diferentes.

Para representar esse comportamento no sistema, foi criada a classe `Scenario`, descrita no Apêndice E, que funciona como um *cache* em memória e que armazena os dados de mercado: valor dos *spots*, superfícies de volatilidade e curvas de juros. Durante a simulação, o cenário base é clonado, e são aplicadas as variações definidas. A carteira é então precificada com esse novo cenário aplicado. Como a passagem do tempo não é um valor de mercado, seu efeito é simulado

recriando cada opção com um prazo inferior ao original.

De forma a facilitar a visualização, o resultado da simulação é apresentado em uma planilha no formato de cubo onde a primeira e segunda dimensões representam as variações do *spot* e volatilidade ATM, respectivamente. A terceira dimensão é composta por todos os conjuntos formados pela combinação das variações do tempo e da volatilidade RR, e cada um desses conjuntos é apresentado em uma planilha separada.

Para cada cenário definido o sistema calcula os seguintes valores da carteira, conforme Figura 10:

- Resultado (perda ou ganho);
- Novo valor de mercado;
- Gregas (*delta, gamma, vega e theta*);
- Sensibilidade à variação de 1% das volatilidades RR e STR.

5.6 Resultados

Para validar o sistema, foi definida pelo gestor uma carteira teórica de opções de moedas, e, sobre ela, foi calculado analiticamente o valor de mercado e todas as gregas. Para tal, foi preciso interpolar manualmente as curvas de juros e a superfície de volatilidade. Os resultados obtidos foram comparados com informações de mercado extraídas do Citi Velocity e, após validados, armazenados para servir de base para comparações futuras.

Após a criação das primeiras funções de precificação e interpolação, ainda sem a facilidade do *cache* e mecanismos de simulação internos ao C#, foi gerada uma macro no Excel que, a partir de cada cenário definido na planilha, apenas altera as células que servem de parâmetros de entrada para o *add-in* e que recalcula todas as fórmulas. Ou seja, a primeira versão não manual do sistema apenas automatizava a mudança de parâmetros. Para os quatrocentos e cinquenta cenários citados na seção anterior, por exemplo, a simulação era executada em vinte e sete minutos em média.

A etapa final do desenvolvimento consistiu em desenvolver a classe `Scenario`, citada anteriormente, e transportar a lógica de troca de parâmetros, baseada nos cenários, para o C#. Com essa mudança, é necessário executar as funções do Excel apenas uma vez, pois a lógica foi internalizada, e, com isso, o tempo médio de execução para os mesmos quatrocentos e cinquenta cenários foi reduzido para dois minutos e quarenta segundos.

Apesar de a primeira versão ter representado uma grande evolução pois eliminou a necessidade de cálculos manuais, o seu alto tempo de execução ainda permitia ao gestor executar

Figura 10 – Exemplo de cubo resultante de uma simulação

	Spot	3.0880	3.1530	3.1855	3.2180	3.2505	3.2830	3.3155	3.3481	3.4131
	FX Bumps	-5.00%	-3.00%	-2.00%	-1.00%	0.00%	1.00%	2.00%	3.00%	5.00%
VOL 2.0%	PnL USD	469,996	270,609	187,696	130,224	100,880	98,213	121,011	169,493	344,791
VOL 2.0%	Mkt Value	542,031	342,644	259,731	202,259	172,915	170,249	193,047	241,528	416,827
VOL 2.0%	Spot Sml Dlta (USD)	(9,477,081)	(8,788,219)	(6,883,200)	(4,297,253)	(1,600,520)	1,016,472	3,635,674	6,344,627	11,844,936
VOL 2.0%	Gamma (USD/1%)	(506,070)	1,308,313	2,204,696	2,616,257	2,656,863	2,644,652	2,717,624	2,820,419	2,798,797
VOL 2.0%	Smile Vega	294	14,464	20,226	23,781	25,380	25,738	25,567	25,217	23,316
VOL 2.0%	Theta	(1,264)	(7,755)	(10,264)	(11,832)	(12,555)	(12,679)	(12,488)	(12,156)	(10,885)
VOL 2.0%	Strgl	526,551	265,478	187,392	156,038	150,443	157,516	175,707	210,271	364,176
VOL 2.0%	RR	506,407	312,396	218,324	142,817	101,866	97,869	123,225	172,612	339,701
VOL 1.0%	PnL USD	476,920	238,699	141,300	77,982	47,944	46,617	70,108	117,805	293,380
VOL 1.0%	Mkt Value	548,955	310,734	213,335	150,018	119,979	118,652	142,143	189,841	365,415
VOL 1.0%	Spot Sml Dlta (USD)	(10,985,441)	(10,476,647)	(7,878,529)	(4,619,781)	(1,568,274)	1,118,390	3,631,382	6,265,732	12,112,693
VOL 1.0%	Gamma (USD/1%)	(1,025,860)	1,735,941	2,877,332	3,125,169	2,869,086	2,626,783	2,626,799	2,831,657	3,095,759
VOL 1.0%	Smile Vega	(1,597)	15,483	21,388	23,844	23,924	23,187	22,754	22,878	22,255
VOL 1.0%	Theta	(501)	(7,106)	(9,251)	(10,214)	(10,328)	(10,087)	(9,856)	(9,733)	(9,016)
VOL 1.0%	Strgl	532,616	216,089	139,970	115,906	112,781	118,552	132,546	160,676	308,204
VOL 1.0%	RR	514,334	280,955	162,793	76,253	42,424	48,089	77,575	126,206	289,426
VOL 0.0%	PnL USD	490,012	201,914	90,640	25,916	(0)	2,991	27,547	72,458	243,708
VOL 0.0%	Mkt Value	562,048	273,949	162,675	97,951	72,035	75,026	99,582	144,493	315,744
VOL 0.0%	Spot Sml Dlta (USD)	(12,781,744)	(12,418,453)	(8,623,079)	(4,483,431)	(1,146,252)	1,389,513	3,544,525	5,940,448	12,297,814
VOL 0.0%	Gamma (USD/1%)	(1,882,938)	2,551,399	3,900,660	3,702,344	2,936,472	2,370,801	2,324,791	2,763,682	3,566,275
VOL 0.0%	Smile Vega	(3,644)	17,894	23,088	22,938	20,587	18,916	19,005	20,404	21,654
VOL 0.0%	Theta	15	(6,743)	(8,206)	(8,218)	(7,631)	(7,236)	(7,275)	(7,571)	(7,422)
VOL 0.0%	Strgl	547,196	161,284	95,347	79,980	78,507	82,832	92,934	114,291	253,395
VOL 0.0%	RR	529,328	243,027	91,624	9,298	(4,721)	10,941	41,156	85,314	239,615
VOL -1.0%	PnL USD	510,996	155,277	35,111	(22,165)	(37,731)	(29,269)	(5,577)	33,280	194,652
VOL -1.0%	Mkt Value	583,031	227,312	107,146	49,870	34,304	42,766	66,458	105,316	266,687
VOL -1.0%	Spot Sml Dlta (USD)	(14,972,789)	(14,544,074)	(8,683,043)	(3,598,880)	(355,195)	1,622,304	3,193,216	5,257,921	12,460,396
VOL -1.0%	Gamma (USD/1%)	(3,333,423)	4,135,693	5,382,083	4,121,679	2,610,592	1,795,001	1,859,907	2,685,376	4,303,614
VOL -1.0%	Smile Vega	(6,073)	22,702	24,708	19,383	14,492	13,346	15,231	18,434	21,431
VOL -1.0%	Theta	285	(6,661)	(6,887)	(5,601)	(4,586)	(4,536)	(5,135)	(5,856)	(5,993)
VOL -1.0%	Strgl	573,852	103,039	56,398	48,245	47,437	50,234	57,143	72,165	200,361
VOL -1.0%	RR	553,350	185,575	3,005	(39,679)	(33,533)	(14,942)	11,502	48,426	189,452
VOL -2.0%	PnL USD	543,257	91,402	(21,989)	(57,325)	(59,947)	(49,843)	(30,819)	(562)	145,552
VOL -2.0%	Mkt Value	615,292	163,437	50,046	14,711	12,088	22,193	41,216	71,473	217,587
VOL -2.0%	Spot Sml Dlta (USD)	(17,699,265)	(16,457,919)	(7,252,470)	(1,868,096)	374,100	1,470,912	2,516,901	4,262,608	12,717,144
VOL -2.0%	Gamma (USD/1%)	(5,791,175)	7,168,378	7,169,097	3,768,182	1,669,504	1,084,656	1,429,914	2,636,831	5,390,159
VOL -2.0%	Smile Vega	(9,495)	30,931	22,998	11,024	6,922	8,503	12,324	16,824	21,231
VOL -2.0%	Theta	343	(6,604)	(4,714)	(2,482)	(1,956)	(2,588)	(3,579)	(4,460)	(4,581)
VOL -2.0%	Strgl	617,660	47,401	24,305	20,402	19,288	20,637	25,391	35,595	149,880
VOL -2.0%	RR	589,369	64,810	(59,913)	(59,502)	(49,631)	(34,640)	(13,702)	15,395	138,353

Fonte: Elaborada pelo autor.

os testes somente ao final do dia, para que ele pudesse obter os valores simulados para a abertura do dia seguinte. Com o ganho de pouco mais de dez vezes no tempo de execução proporcionado pela versão totalmente em C#, o gestor agora é capaz de rodar a simulação diversas vezes no dia e estar melhor preparado nos dias em que o mercado está mais volátil.

CAPÍTULO

6

CONCLUSÃO

Realizar testes em carteiras de investimentos variando as condições de mercado é uma tarefa essencial para que os gestores possam ter uma ideia do que pode ocorrer com seu *portfolio* em diferentes cenários. De posse dos resultados destes testes, eles podem tomar decisões sobre aumentar, diminuir ou até mesmo diversificar sua exposição aos fatores de risco. Vale lembrar que no mercado financeiro são encontradas diversas classes de ativos que são influenciadas por diferentes fatores. Entender quais são os fatores que influenciam a carteira em questão é um passo importante na construção de uma ferramenta capaz de auxiliar nos testes.

Neste trabalho foi apresentada uma introdução ao mercado de câmbio e seus principais instrumentos, como *forwards* e opções de moedas. O mundo de opções foi um pouco mais detalhado, sendo mostrada sua dinâmica e a influência da volatilidade, taxa de juros e passagem do tempo. Foram descritos ainda o modelo de Black-Scholes e uma de suas evoluções, o modelo de Black 1976, que permite calcular o preço de uma opção de moedas, bem como quantificar a sensibilidade desse preço com relações aos fatores mencionados, ainda que esse modelo assuma premissas não condizentes com o mundo real. Adiante, foi indicado um mecanismo utilizado para adaptar o comportamento observado no mercado para a premissa da volatilidade constante, a superfície de volatilidade.

De posse desses conhecimentos de matemática financeira foi possível criar um *add-in* de Excel, o SIRI, capaz de interpolar a superfície e computar os valores do prêmio de uma opção e suas gregas. Do ponto de vista computacional, seu desenvolvimento foi realizado com base nos conceitos de orientação a objetos, o que permitiu criar uma arquitetura que possibilita o reuso e facilita o reaproveitamento de código. A versão atual é capaz de executar simulações sobre carteiras de opções de moedas. Cada cenário testado pode representar variações tanto na volatilidade ATM quanto na RR, além de mudanças na taxa de câmbio e passagem do tempo.

Entretanto, como os resultados obtidos nessas simulações auxiliam o gestor na tomada de decisões, faz-se importante ressaltar suas limitações. Uma delas, mencionada na Seção 4.2, é que

foi utilizada uma interpolação do tipo *spline* cúbica que, dependendo da superfície, pode resultar em volatilidades que gerem condições de arbitragem. Outro ponto importante é que o sistema suporta apenas as mais simples opções de moedas do tipo europeia, descritas na Seção 2.3 e também chamadas "baunilha".

Esses pontos mencionados deixam margem para a elaboração de trabalhos futuros, sempre buscando a evolução do sistema. Um primeiro passo poderia ser a inclusão de mecanismos de cálculos para opções exóticas de primeira ordem, como europeias binárias ou digitais. Tais opções podem ser simuladas por duas opções baunilha em um *call-spread* ou *put-spread*.

Um outro trabalho pode ser a adição de suporte a opções cujo resultado dependa não apenas do valor do ativo objeto na data de maturidade, mas também de todo o caminho percorrido por esse preço, como opções do tipo Americanas ou até mesmo Europeias com barreira. Para esse fim, é necessário o desenvolvimento de um mecanismo capaz de simular trajetórias para o preço do ativo a partir de um bom gerador de números aleatórios.

Do ponto de vista computacional, há também espaço para novos trabalhos, como a utilização de técnicas de programação paralela. Como os cenários a serem testados são independentes entre si, a utilização dessas técnicas permitiria que mais de um cenário fosse executado ao mesmo tempo, trazendo ganhos ainda mais expressivos no tempo de execução das simulações.

Contudo, apesar de todas as melhorias possíveis, a versão atual resultante deste trabalho mostrou que a ideia do desenvolvimento inicial foi acertada, pois já representou um ganho de horas de trabalho diárias pelo gestor, que agora é capaz de obter um cubo com os resultados das simulações em apenas alguns minutos.

REFERÊNCIAS

BIS. **Triennial central bank survey - Foreign exchange turnover in April 2016**. [S.l.], 2016. Disponível em: <http://www.bis.org/publ/rpfx16.htm>. Citado na página 23.

BLACK, F. The pricing of commodity contracts. **Journal of financial economics**, Elsevier, v. 3, n. 1-2, p. 167–179, 1976. Citado nas páginas 34, 35 e 47.

BLACK, F.; SCHOLES, M. The pricing of options and corporate liabilities. **Journal of political economy**, The University of Chicago Press, v. 81, n. 3, p. 637–654, 1973. Citado nas páginas 31 e 33.

BURDEN, R. L.; FAIRES, J. D. **Numerical analysis**. 9th. ed. Cengage Learning, 2010. ISBN 9781133169338. Disponível em: <https://books.google.com.br/books?id=Dbw8AAAAQBAJ>. Citado nas páginas 17 e 57.

CATTARUZZI, R. F. **Análise de componentes principais do skew e da superfícies de volatilidade de dólar/reais**. Dissertação (Mestrado) — Faculdade IBMEC São Paulo, 2009. Citado na página 39.

CLARK, I. J. **Foreign exchange option pricing: a practitioner's guide**. [S.l.]: John Wiley & Sons, 2011. Citado nas páginas 25, 26, 29, 30 e 42.

DERMAN, E. Regimes of volatility. **Risk**, v. 4, n. 4, p. 55–59, 1999. Citado na página 39.

DRIMMELEN, G. V. **Excel-DNA**. 2006–2017. <https://excel-dna.net/>. Citado na página 48.

FRANÇA, D. M. **Derivativos cambiais do mercado brasileiro: precificação e administração de riscos**. Dissertação (Mestrado) — Fundação Getulio Vargas, 2010. Citado na página 40.

HOEK, B. v. d. **Alternative swaption valuation methods**. Dissertação (Mestrado) — Tilburg University, 2012. Citado na página 32.

HULL, J. **Options, futures, and other derivatives**. 6th. ed. Pearson Prentice Hall, 2006. ISBN 978-0-13-197705-1. Disponível em: <http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+563580607&sourceid=fbw_bibsonomy>. Citado nas páginas 26, 27, 29, 32, 33, 34 e 35.

PEREIRA, M. **Os riscos do carry trade sob uma abordagem não-paramétrica**. Dissertação (Mestrado) — Insper Instituto de Ensino e Pesquisa, 2011. Citado na página 23.

STOLL, H. The relationship between put and call option prices. **Journal of finance**, v. 24, n. 5, p. 801–24, 1969. Disponível em: <http://EconPapers.repec.org/RePEc:bla:jfinan:v:24:y:1969:i:5:p:801-24>. Citado na página 33.

APÊNDICE

A

SPLINE CÚBICO

Uma interpolação do tipo *spline* cúbica consiste em um conjunto de polinômios de terceira ordem que unem cada subintervalo de uma função. Cada um desses polinômios deve possuir primeira e segunda derivadas contínuas e deve passar obrigatoriamente pelos pontos que delimitam o subintervalo. Como condição de contorno, a segunda derivada dos pontos extremos da função pode ser igualada a zero e, nesse caso, chamamos de "*spline* cúbico natural".

Código-fonte 2 – Implementação de um *spline* cúbico natural seguindo o algortimo proposto por Burden e Faires (2010).

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5:
6: namespace SIRI.Mathematics.Interpolation
7: {
8:     public class CubicSplineInterpolator : Interpolator
9:     {
10:         private double[] _b;
11:         private double[] _c;
12:         private double[] _d;
13:
14:         public CubicSplineInterpolator(bool Extrapolate = false
      )
15:             : base(Extrapolate)
16:         {
17:             _name = "NaturalCubicSpline";
18:         }
```

```
19:
20:         /// <summary>
21:         /// Algorithm 3.4 of:
22:         ///     Numerical Analysis, Ninth Edition (Richard L.
    Burden)
23:         /// </summary>
24:         public override void update()
25:         {
26:             int n = _size - 1;
27:
28:             _b = new double[n];
29:             _c = new double[n+1];
30:             _d = new double[n];
31:
32:             double[] h = new double[n];
33:             for (int i = 0; i <= n-1; i++)
34:             {
35:                 h[i] = _xAxis[i + 1] - _xAxis[i];
36:             }
37:
38:             double[] alpha = new double[n];
39:             for (int i = 1; i <= n-1; i++)
40:             {
41:                 alpha[i] = (3 / h[i]) * (_yAxis[i + 1] - _yAxis
    [i]) - (3 / h[i-1]) * (_yAxis[i] - _yAxis[i-1]);
42:             }
43:
44:             double[] l = new double[n+1];
45:             double[] mi = new double[n];
46:             double[] z = new double[n+1];
47:
48:             l[0] = 1;
49:             mi[0] = 0;
50:             z[0] = 0;
51:
52:             for (int i = 1; i <= n-1; i++)
53:             {
54:                 l[i] = 2 * (_xAxis[i + 1] - _xAxis[i - 1]) - h[
    i - 1] * mi[i - 1];
55:                 mi[i] = h[i] / l[i];
56:                 z[i] = (alpha[i] - h[i - 1] * z[i - 1]) / l[i];
57:             }
```

```
58:
59:                 l[n] = 1;
60:                 z[n] = 0;
61:                 _c[n] = 0;
62:
63:                 for (int j = n - 1; j >= 0; j--)
64:                 {
65:                     _c[j] = z[j] - (mi[j] * _c[j + 1]);
66:                     _b[j] = (_yAxis[j + 1] - _yAxis[j]) / h[j] - h[
       j] * (_c[j + 1] + 2 * _c[j]) / 3;
67:                     _d[j] = (_c[j + 1] - _c[j]) / (3 * h[j]);
68:                 }
69:             }
70:
71:             protected override double Eval(double X, int i)
72:             {
73:                 double dx = X - _xAxis[i];
74:                 return _yAxis[i] + _b[i] * dx + _c[i] * (dx * dx) +
        _d[i] * (dx * dx * dx);
75:             }
76:         }
77: }
```

APÊNDICE

B

CLASSE OPTION

Código-fonte 3 – Implementação da Opção.

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4:
5: using QLNet;
6: using SIRI.Pricer;
7: using SIRI.Pricer.Engine;
8:
9: namespace SIRI.Securities
10: {
11:     public abstract class Option : Security
12:     {
13:         protected Date _refDate;
14:
15:         protected string _callPut;
16:         protected double _strike;
17:         protected double _vol;
18:
19:     protected string _surfaceId;
20:     protected LVolSurface _surface;
21:
22:     protected string _curveId;
23:     protected AbstractCurve _curve;
24:
25:         protected Security _underlying;
26:
```

```
27:         protected IOptionPricingEngine _engine;
28:
29:         public override Scenario Scenario { set { _scenario =
      value; Update(); } }
30:
31:         #region Properties
32:         public double Vol
33:         {
34:             get { return _vol; }
35:             set
36:             {
37:                 _vol = value;
38:                 _engine.Vol = _vol;
39:             }
40:         }
41:         #endregion
42:
43:         public virtual double Premium()
44:         {
45:             return _engine.Premium();
46:         }
47:
48:         public virtual double Delta()
49:         {
50:             return _engine.Delta();
51:         }
52:
53:         public virtual double Gamma()
54:         {
55:             return _engine.Gamma();
56:         }
57:
58:         public virtual double Vega()
59:         {
60:             return _engine.Vega();
61:         }
62:
63:         public virtual double Theta()
64:         {
65:             return _engine.Theta();
66:         }
67:
```

```
68:         public override double Evaluate(string ValueType)
69:         {
70:             double value;
71:             switch (ValueType.ToLower())
72:             {
73:                 case "premium":
74:                     value = Premium();
75:                     break;
76:                 case "delta":
77:                     value = Delta();
78:                     break;
79:                 case "gamma":
80:                     value = Gamma();
81:                     break;
82:                 case "vega":
83:                     value = Vega();
84:                     break;
85:                 case "theta":
86:                     value = Theta();
87:                     break;
88:                 default:
89:                     throw new ArgumentException("Value type not
       applicable: " + ValueType);
90:             }
91:
92:             return value;
93:         }
94:     }
95: }
```

APÊNDICE

C

CLASSE LVOLSURFACE

Código-fonte 4 – Implementação da superfície de volatilidade

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4:
5: using QLNet;
6: using SIRI.Mathematics.Interpolation;
7:
8: namespace SIRI.Pricer
9: {
10:     public class LVolSurface
11:     {
12:         #region Members
13:         protected string _name;
14:         protected Date _refDate;
15:
16:         protected Date[] _dates;
17:
18:         protected double[] _deltasOrStrikes;
19:         protected double[] _timeToMaturity;
20:         protected double[,] _vols;
21:
22:         protected Calendar2 _calendar;
23:         protected DayCounter _dayCounter;
24:
25:         protected Interpolator[] _interpolators;
26:         #endregion
```

```
27:
28:         #region Properties
29:         public string Name { get { return _name; } }
30:         public Date RefDate { get { return _refDate; } }
31:
32:         public double[,] Vols
33:         {
34:             get { return _vols; }
35:             set
36:             {
37:                 if (value.GetLength(0) != _dates.Length ||
    value.GetLength(1) != _deltasOrStrikes.Length)
38:                 {
39:                     throw new ArgumentException("Vols matrix
    rows must be equals to dates array length and columns equals
     to deltas or strikes array length.");
40:                 }
41:
42:                 _vols = value;
43:                 for (int i = 0; i < _dates.Length; i++)
44:                 {
45:                     _interpolators[i].SetAxis(_deltasOrStrikes,
     _vols.SliceRow<double>(i).ToArray());
46:                 }
47:             }
48:         }
49:         #endregion
50:
51:         public LVolSurface(string Name, int RefDate, double[]
    Dates, double[] DeltasOrStrikes, string DayCounter, string
    Calendar = "USD", string Interpolation = "CubicSpline")
52:         {
53:             _name = Name;
54:             _refDate = RefDate;
55:
56:             _calendar = CalendarFactory.Instance.GetCalendar(
    Calendar);
57:             _dayCounter = QLNet.DayCounter.GetDayCounter(
    DayCounter, _calendar);
58:
59:             _deltasOrStrikes = DeltasOrStrikes.TrimDouble();
60:
```

```
61:             Dates = Dates.TrimDouble();
62:             _dates = new Date[Dates.Length];
63:             _interpolators = new Interpolator[Dates.Length];
64:             _timeToMaturity = new double[Dates.Length];
65:             for (int i = 0; i < Dates.Length; i++)
66:             {
67:                 _dates[i] = new Date((int)Dates[i]);
68:                 _interpolators[i] = Interpolator.
    GetInterpolator(Interpolation);
69:                 _interpolators[i].Extrapolate = true;
70:                 _timeToMaturity[i] = _dayCounter.dayCount(
    _refDate, _dates[i]);
71:             }
72:         }
73:
74:         public double GetVol(int EndDate, double DeltaOrStrike)
75:         {
76:             double timeToMaturity = _dayCounter.dayCount(
    _refDate, EndDate);
77:             int x = _timeToMaturity.Locate(timeToMaturity);
78:
79:             double volT = 0;
80:             if (EndDate == _dates[x])
81:             {
82:                 volT = _interpolators[x].Interpolate(
    DeltaOrStrike);
83:             }
84:             else
85:             {
86:                 double t = _dayCounter.yearFraction(_refDate,
    EndDate);
87:                 double t1 = _dayCounter.yearFraction(_refDate,
    _dates[x]);
88:                 double t2 = _dayCounter.yearFraction(_refDate,
    _dates[x + 1]);
89:
90:                 double volT1 = _interpolators[x].Interpolate(
    DeltaOrStrike);
91:                 double volT2 = _interpolators[x + 1].
    Interpolate(DeltaOrStrike);
92:
93:                 double varianceT1 = volT1 * volT1;
```

```
94:                 double varianceT2 = volT2 * volT2;
95:
96:                 double varianceT = ((1 / (t2 - t1)) * (
      varianceT2 * t2 * (t - t1) + varianceT1 * t1 * (t2 - t))) /
      t;
97:                 volT = Math.Sqrt(varianceT);
98:             }
99:
100:            return volT;
101:        }
102:    }
103: }
```

APÊNDICE

D

CLASSE MARKETVOLSURFACE

Código-fonte 5 – Implementação da superfície de volatilidade pelos parâmetros ATM, RR e STR

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4:
5: using QLNet;
6: using SIRI.Mathematics.Interpolation;
7:
8: namespace SIRI.Pricer
9: {
10:     public class MarketVolSurface : LVolSurface
11:     {
12:         private static readonly double[] DELTAS = { 0.1, 0.25,
      0.50, 0.75, 0.9 };
13:
14:         private readonly double STR_RR_FACTOR = 2.25;
15:
16:         #region Members
17:         private double[] _atm;
18:         private double[] _rr10;
19:         private double[] _rr25;
20:         private double[] _str10;
21:         private double[] _str25;
22:
23:         private double _atm3M;
24:
```

```
25:         private double[] _shifts;
26:
27:         private double _atmBump;
28:         private double _rrBump;
29:         private double _strBump;
30:         #endregion
31:
32:         public MarketVolSurface(string Name, int RefDate,
    double[] Dates, string DayCounter, string Calendar = "USD",
    string Interpolation = "CubicSpline")
33:             : base(Name, RefDate, Dates, DELTAS, DayCounter,
    Calendar, Interpolation)
34:         {
35:             CalculateShifts();
36:         }
37:
38:         public MarketVolSurface(string Name, Date RefDate, Date
    [] Dates, string DayCounter, string Calendar = "USD", string
     Interpolation = "CubicSpline")
39:             : base(Name, RefDate, Dates, DELTAS, DayCounter,
    Calendar, Interpolation)
40:         {
41:             CalculateShifts();
42:         }
43:
44:         private void CalculateShifts()
45:         {
46:             _shifts = new double[_dates.Length];
47:             for (int i = 0; i < _dates.Length; i++)
48:             {
49:                 double days = (double)(_dates[i] - _refDate);
50:                 _shifts[i] = Math.Sqrt(90D / days);
51:             }
52:         }
53:
54:         public void SetVols(double[] ATM, double[] RR10, double
    [] RR25, double[] STR10, double[] STR25, double ATM3M)
55:         {
56:             _atm = ATM;
57:             _rr10 = RR10;
58:             _rr25 = RR25;
59:             _str10 = STR10;
```

```
60:             _str25 = STR25;
61:
62:             _atm3M = ATM3M;
63:
64:             UpdateSurfaceVols();
65:         }
66:
67:         public void SetBumps(double ATMBump, double RRBump,
    double STRBump)
68:         {
69:             _atmBump = ATMBump;
70:             _rrBump = RRBump;
71:             _strBump = STRBump;
72:
73:             if (_rrBump != 0)
74:             {
75:                 _strBump += _rrBump / ((_atm3M + ATMBump)  *
    100) * STR_RR_FACTOR;
76:             }
77:
78:             UpdateSurfaceVols();
79:         }
80:
81:         public void UpdateSurfaceVols()
82:         {
83:             double[,] vols = new double[_atm.Length, 5];
84:             for (int i = 0; i < _atm.Length; i++)
85:             {
86:                 double atm = _atm[i] + _atmBump * _shifts[i];
87:                 double rr25 = _rr25[i] + _rrBump * _shifts[i];
88:                 double rr10 = rr25 * (_rr10[i] / _rr25[i]);
89:                 double str25 = _str25[i] + _strBump * _shifts[i
    ];
90:                 double str10 = str25 * (_str10[i] / _str25[i]);
91:
92:                 vols[i, 0] = atm + str10 + 0.5 * rr10;  // 10%
    delta
93:                 vols[i, 1] = atm + str25 + 0.5 * rr25;  // 25%
    delta
94:                 vols[i, 2] = atm;                       // 50%
    delta
```

```
95:                 vols[i, 3] = atm + str25 - 0.5 * rr25;  // 75%
       delta
96:                 vols[i, 4] = atm + str10 - 0.5 * rr10;  // 90%
       delta
97:             }
98:
99:             base.Vols = vols;
100:        }
101:     }
102: }
```

APÊNDICE

E

CLASSE SCENARIO

Código-fonte 6 – Implementação do Cenário

```
1: using System;
2: using System.Collections.Generic;
3: using System.Linq;
4:
5: using QLNet;
6: using SIRI.Securities;
7: using SIRI.Portfolio;
8:
9: namespace SIRI.Pricer
10: {
11:     public class Scenario
12:     {
13:         #region Members
14:         private string _scenarioID;
15:
16:         private Dictionary<Tuple<string, Date>, AbstractCurve>
      _curves;
17:         private Dictionary<Tuple<string, Date>, LVolSurface>
      _surfaces;
18:         private Dictionary<string, Fixing2> _fixings2;
19:         #endregion
20:
21:         #region Properties
22:         public string ScenarioID { get { return _scenarioID; }
      set { _scenarioID = value; } }
```

```
23:         public Dictionary<Tuple<string, Date>, LVolSurface>
    Surfaces { get { return _surfaces; } set { _surfaces = value
    ; } }
24:         public Dictionary<string, Fixing2> Fixings2 { get {
    return _fixings2; } }
25:         #endregion
26:
27:         public Scenario()
28:         {
29:             Reset();
30:         }
31:
32:         public void Reset()
33:         {
34:             _curves = new Dictionary<Tuple<string, Date>,
    AbstractCurve>();
35:             _surfaces = new Dictionary<Tuple<string, Date>,
    LVolSurface>();
36:             _fixings2 = new Dictionary<string, Fixing2>();
37:         }
38:
39:         #region Curve
40:         public void AddCurve(AbstractCurve Curve)
41:         {
42:             _curves[Curve.ID] = Curve;
43:         }
44:
45:         public AbstractCurve GetCurve(string CurveName, Date
    RefDate)
46:         {
47:             Tuple<string, Date> key = new Tuple<string, Date>(
    CurveName, RefDate);
48:             AbstractCurve curve = null;
49:             _curves.TryGetValue(key, out curve);
50:             return curve;
51:         }
52:         #endregion
53:
54:         #region Surface
55:         public void AddSurface(LVolSurface Surface)
56:         {
57:             _surfaces[Surface.ID] = Surface;
```

```
58:         }
59:
60:         public LVolSurface GetSurface(string SurfaceID, Date
      RefDate)
61:         {
62:             Tuple<string, Date> key = new Tuple<string, Date>(
      SurfaceID, RefDate);
63:             LVolSurface surface = null;
64:             _surfaces.TryGetValue(key, out surface);
65:             return surface;
66:         }
67:         #endregion
68:
69:         #region Fixing
70:         public void AddFixing2(Fixing2 Fixing)
71:         {
72:             _fixings2[Fixing.SecurityID] = Fixing;
73:         }
74:
75:         public Fixing2 GetFixing2(string FixingID)
76:         {
77:             Fixing2 fixing = null;
78:             _fixings2.TryGetValue(FixingID, out fixing);
79:             return fixing;
80:         }
81:         #endregion
82:     }
83: }
```

—


ICMC
USP
SÃO CARLOS