Modelos de precificação de Opções Americanas a partir de plataformas paralelas

Lucas Vioto dos Santos Ribeiro

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

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Lucas Vioto dos Santos Ribeiro

Modelos de precificação de Opções Americanas a partir de plataformas paralelas

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. Francisco Louzada Neto

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)

V799m VIOTO DOS SANTOS RIBEIRO, LUCAS
Modelos de precificação de Opções Americanas a
partir de plataformas paralelas / LUCAS VIOTO DOS
SANTOS RIBEIRO; orientador Francisco Louzada Neto. -
- São Carlos, 2017.
78 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. Opções Americanas. 2. Modelos de Precificação.
3. Plataforma Paralela. 4. TPL. 5. CUDA. I. Louzada
Neto, Francisco, orient. II. Título.

Lucas Vioto dos Santos Ribeiro

Pricing models of American Options from parallel platforms

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. Francisco Louzada Neto

**USP – São Carlos
November 2017**

Dedico este trabalho às pessoas mais importantes da minha vida. Os grandes mestres, meus pais Roseli e José Antônio, e o amor da minha vida, minha Esposa Ana Paula Duarte.

Agradecimentos

Gostaria primeiramente de agradecer a Deus por toda saúde e disposição me concedida na construção desse trabalho.

Em segundo lugar agradeço meus pais, por sempre me oferecerem bons exemplos de vida aos quais foram fundamentais para a conclusão desse trabalho.

Agradeço o meu orientador professor Dr. Francisco Louzada Neto e seu pós-doutorado José Augusto Fiorucci pelo grande apoio nas discussões para a confecção deste trabalho.

Agradeço os amigos do trabalho que tiveram a paciência necessária em compreender a minha dedicação também ao mestrado.

Agradeço meus familiares pelas orações.

Agradeço, em especial, a minha esposa Ana Paula Duarte diante todo o apoio e incentivo com muitas frases de motivação e de muito café, ao longo desta grande jornada.

*“Nunca desista do seu objetivo, a
dedicação é a chave do sucesso...”*

Lucas Vioto.

RESUMO

RIBEIRO, L. V. S. **Modelos de precificação de Opções Americanas a partir de plataformas paralelas**. 2017. 78p. Dissertação (Mestrado Profissional em Matemática, Estatística e Computação Aplicadas a Indústria) – Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Carlos - SP, 2017

O objetivo desta dissertação é fornecer primeiramente o arcabouço necessário para o entendimento do derivativo opções, muito utilizado nos mercados financeiros mundiais, e posteriormente executar precificações de opções americanas a partir dos modelos dos mínimos quadrados de Monte Carlo (LSM), o modelo de árvore binomial com extrapolação de Richardson e a aproximação analítica de *Bjerk Sund* e *Stensland* (B&S), aplicando duas plataformas de processamento paralelo computacional, a *TPL (Task Parallel Library)* nativa no .NET framework 4.5 e a plataforma *CUDA (Compute Unified Device Architecture)*, demonstrando o comparativo dos resultados obtidos a cada modelo diante de cada plataforma.

Palavras-chave: Opções Americanas, Modelos de precificação, Plataforma paralela, TPL, CUDA.

ABSTRACT

RIBEIRO, L. V. S. **Pricing models of American Options from parallel platforms**. 2017. 78p. Dissertação (Mestrado Profissional em Matemática, Estatística e Computação Aplicadas a Indústria) – Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Carlos - SP, 2017

The objective of this dissertation is to provide first the necessary framework for the understanding of the derivative options, widely used in the world financial markets, and later to execute the American option pricing from Monte Carlo least squares models (LSM), the binomial tree model with Richardson extrapolation and the Bjerk Sund and Stensland analytic approach (BJS) by applying two parallel computational processing platforms, the native TPL (Task Parallel Library) in the .NET framework 4.5 and the CUDA platform (*Compute Unified Device Architecture*), demonstrating the comparison of the obtained results to each model before each platform.

Keywords: American options, Pricing Models, Parallel platforms, TPL, CUDA.

Lista de Figuras

Figura 1 - <i>Payoff</i> de uma compra de opção <i>call</i>	28
Figura 2 - Processo de Wiener generalizado $a=0.3$ e $b=1.5$	33
Figura 3 - Árvore Binomial de 3 períodos.....	39
Figura 4 - Arquitetura TPL.	47
Figura 5 - Padrões de Paralelismo TPL.	48
Figura 6 - Aplicação CPU e GPU.....	50
Figura 7 - SMX Kepler.	51
Figura 8 - Comparativo Arquiteturas GPU.....	52
Figura 9 - Arquitetura CUDA - Dados CPU para GPU.....	53
Figura 10 - Arquitetura CUDA - Execução código GPU.....	53
Figura 11 - Arquitetura CUDA - Resultado GPU para CPU.	54
Figura 12 - Arquitetura CUDA - <i>Threads</i> por bloco.	55
Figura 13 - Desempenho dos geradores de números aleatórios em relação às funções de distribuição Log-Normal, Normal e Uniforme.....	57
Figura 14 - Descrição da CPU.	61
Figura 15 - Descrição da GPU.	61
Figura 16 - Tempo de Processamento Modelo B&S 2002.	63
Figura 17 - Comparação plataformas modelo Árvore Binomial Tempo de Processamento contra Períodos.	65
Figura 18 - Árvore Binomial - uso CPU no processo sequencial.....	66
Figura 19 - Árvore Binomial - uso CPU no processo paralelo TPL.	67
Figura 20 - Comparação plataformas modelo LSM tempo de processamento contra trajetórias e número de passos.	69
Figura 21 - Modelo LSM - uso CPU processo sequencial.	69
Figura 22 - Modelo LSM – uso CPU processo paralelo TPL.	70
Figura 23 - Comparação taxa de convergência modelo Árvore Binomial contra Árvore Binomial com Extrapolação de Richardson com linha de tendência.	72

Lista de Tabelas

Tabela 1 - Ativo Objeto x <i>Strikes</i>	58
Tabela 2 - Dados Financeiros.....	59
Tabela 3 - Modelo B&S 2002 – EQM.....	62
Tabela 4 - Modelo B&S 2002 - Tempo de Processamento e uso da CPU.....	63
Tabela 5 - Modelo Árvore Binomial – EQM.....	64
Tabela 6 - Modelo Árvore Binomial com 15000 períodos.	64
Tabela 7 - Modelo Árvore Binomial com 600 períodos.	65
Tabela 8 - Modelo LSM – EQM.....	68
Tabela 9 - Modelo LSM – Tempo de processamento e uso de CPU.	68
Tabela 10 - Árvore Binomial contra Árvore Binomial com Extrapolação de Richardson - Prêmio de Mercado da opção.	71

Lista de Siglas

CUDA *Compute Unified Device Architecture*

TPL *Task Parallel Library*

CPU *Central Processing Unit*

GPU *Graphics Processing Unit*

LSM *Monte Carlo Least Square Model*

B&S *Bjersund & Stensland*

EWMA *Exponentially Weighted Moving Average*

EDP *Equação diferencial parcial*

BBSR *Binomial Black & Scholes Richardson*

DI *Depósito Interbancário*

API *Application Programming Interface*

Sumário

1. INTRODUÇÃO	25
2. CONCEITOS FUNDAMENTAIS.....	27
2.1. Opções	27
2.1.1. Opções Europeias	27
2.1.2. Opções Americanas	29
2.2. Processo Estocástico	29
2.2.1. Processo de Markov	30
2.2.2. Processo de Wiener	30
2.2.3. Processo de Wiener Generalizado.....	31
2.2.4. Processo de Itô	33
2.2.5. Lema de Itô	34
3. MODELOS DE PRECIFICAÇÃO DE OPÇÕES.....	35
3.1. Modelo de Black & Scholes	35
3.2. Método dos Mínimos Quadrados de Monte Carlo (LSM).....	36
3.3. Modelo de Árvore Binomial com Extrapolação de Richardson	38
3.4. Aproximação Analítica de Bjerksund & Stensland (B&S)	42
4. PLATAFORMAS PARALELAS.....	46
4.1. CPU - Biblioteca de tarefas paralelas (TPL)	46
4.2. GPU - CUDA	49
5. METODOLOGIA	58
5.1. Portfólio	58
5.2. Dados de Mercado	58
5.3. Processos de Cálculo	60
5.4. Hardware	60
6. RESULTADOS	62
6.1. Simulação da Aproximação Analítica Bjerksund & Stensland 2002.....	62
6.2. Simulação da Árvore Binomial Padrão.....	63

6.3.	Simulação do Método dos Mínimos Quadrados de Monte Carlo.....	67
6.4.	Convergência do Modelo Árvore Binomial com Extrapolação de Richardson	70
7.	CONCLUSÃO.....	73
	REFERÊNCIA BIBLIOGRÁFICA.....	75
	GLOSSÁRIO	77

1. Introdução

O intuito mais relevante da teoria de finanças quantitativas é o de determinar o preço justo de um ativo financeiro. Os modelos de precificação de ativos buscam justamente descrever o comportamento dos preços ao longo do tempo. Uma vez que os ativos possuem liquidez, é factível a verificação se o preço calculado a partir do modelo está aderente com o preço real.

Do ponto de vista de opções, o princípio supracitado também pode ser observado. As opções são geralmente classificadas de acordo com sua possibilidade de exercício antecipado. Enquanto as opções europeias só podem ser exercidas na sua data de expiração, as americanas podem ser exercidas a qualquer momento até seu vencimento. Assim, a precificação de opções europeias, por exigir apenas o conhecimento do preço final do ativo, apresenta-se como uma tarefa bastante simples se comparada com opções mais complexas do estilo americano.

A característica de exercício antecipado tem sido a maior dificuldade no desenvolvimento de soluções analíticas, que ainda não são conhecidas, para opções de estilo americano. Assim, ao longo dos anos, a maior parte das pesquisas realizadas nessa área foram no desenvolvimento de modelos baseado em Simulações de Monte Carlo, Métodos Numéricos e aproximações analíticas para a resolução deste problema, que também é conhecido como problema da parada ótima.

A divisão deste trabalho foi feita da seguinte maneira: no segundo capítulo teremos a introdução sobre conceitos fundamentais que teremos que enunciar para darmos os subsídios necessários à abordagem do tema apontado. Tais subsídios se concentram na concepção do derivativo opções, exemplificando os estilos europeias e americanas, processo estocástico de *Markov*, de *Wiener* (também chamado de movimento browniano), *Wiener* Generalizado, processo e lema de Itô. No terceiro capítulo falaremos sobre os modelos de precificação de opções americanas que iremos utilizar neste trabalho, como: o método dos mínimos quadrados de Monte Carlo (LSM), o modelo de árvore binomial com extrapolação de Richardson e a aproximação analítica de *Bjerk Sund* e *Stensland* (B&S). No quarto capítulo apresentaremos as plataformas de processamento paralelo *TPL* (*Task Parallel Library*) e *CUDA* (*Compute Unified Device Architecture*) presentes no desenvolvimento da solução. No quinto capítulo, referente à metodologia, evidenciaremos a origem e os dados de mercado, o portfólio de opções de venda de VALE5, a descrição da implementação dos modelos de cálculo de opções americanas a partir da IDE Visual Studio, framework .NET e o Kit de ferramentas com

compilador *C/C++ CUDA Toolkit*. Além disso, traremos os detalhes da configuração dos *hardwares* aplicados CPU e GPU. No sexto capítulo iremos apresentar os resultados das simulações diante as variáveis utilizadas na metodologia descrita, priorizando a medida de erro quadrático médio entre o preço de tela e o preço da simulação, tempo de processamento e uso da CPU e GPU. E, por fim, teceremos comentários sobre os resultados alcançados, as dificuldades encontradas e os novos possíveis desafios a seguir sobre o tema abordado.

Portanto, o trabalho proposto tem como objetivo realizar o estudo da precificação de opções americanas através de plataformas paralelas, a fim de desenvolver comparações entre os modelos supracitados em termos de tempo de processamento, robustez, consistência e complexidade computacional para avaliarmos o uso apropriado de cada modelo dependendo da circunstância do momento.

Esclareço que os textos em algumas figuras foram mantidos em inglês para manter a autenticidade da informação, em parte devido ao uso das tecnologias que foram consideradas na dissertação.

2. Conceitos Fundamentais

Neste capítulo iremos descrever o significado do derivativo opções no mercado financeiro brasileiro, cujo derivativo será amplamente utilizado neste trabalho. Conceitos básicos sobre alguns estilos de opções, como as europeias e americanas, também serão elucidados, com suas principais propriedades. Além disso, abordaremos os processos estocásticos de *Markov*, *Wiener*, *Wiener* Generalizado, processo e lema de Itô, que utilizaremos posteriormente nos modelos de precificação.

2.1. Opções

Opções podem ser definidas como contratos que dão o direito, mas não a obrigação, ao seu comprador (ou detentor) de realizar uma transação pré-determinada com o vendedor (ou lançador). Os dois principais tipos de opção são as de compra (*call*) e as de venda (*put*). Uma opção de compra dá o direito ao comprador de, numa data futura, comprar um ativo (ou derivativo) do vendedor da opção a um preço pré-estabelecido. Já uma opção de venda dá direito ao seu comprador de, numa data futura, vender um ativo (ou derivativo) ao vendedor da opção a um preço pré-estabelecido.

As opções são negociadas em plataformas de bolsas organizadas, como a BM&FBovespa. Na BM&FBovespa são negociadas opções no mercado à vista, como as opções de ações, de índice da Bovespa, de IDI (índice de taxa média de DI), sobre Dólar Americano, de ouro à vista, no mercado futuro como opções sobre futuros de mercadorias (açúcar, álcool, algodão, boi gordo, soja, café). Além disso, existem as opções de séries padronizadas e as opções flexíveis que permitem que as partes envolvidas na negociação definam data, preço de exercício e qualidade da mercadoria diferente das séries padronizadas.

2.1.1. Opções Europeias

As opções europeias referem-se, por convenção de mercado, ao fato que o exercício do contrato, isto é, o direito do comprador da opção de exercer a compra (ou venda), ser válido

somente no dia de vencimento da operação. O preço de referência para o ativo subjacente é determinado na data *fixing* da operação, podendo ser essa igual ou inferior à data de vencimento. O *payoff* da opção de compra e de venda sobre o ativo de preço S é dado pelas equações, *call*: $\max(S - X, 0)$ (2.1) e *put*: $\max(X - S, 0)$ (2.2), onde S é o valor do ativo objeto na data de vencimento da opção e X é o *strike* ou preço de exercício da opção.

Figura 1 - Payoff de uma compra de opção *call*.

Fonte: produzido pelo próprio Autor.

A Figura 1 mostra o *payoff* de uma compra de opção *call* de ação. Podemos ver o prejuízo no início do gráfico dado o valor gasto na compra da opção e, com a valorização do ativo objeto em relação ao *strike*, o *payoff* se torna positivo, como vimos na equação (2.1). O mesmo acontece com a opção de venda.

2.1.2. Opções Americanas

As opções americanas diferem das opções europeias por poderem ser exercidas a qualquer momento ao longo de sua vida e não somente na data de vencimento. Grande parte do problema de precificação das opções americanas está relacionado à determinação de um momento de exercício ótimo antecipado. Por este motivo as tornam mais complexas que as europeias.

Atualmente, não é conhecida uma solução analítica para as opções desse estilo. Em algumas situações mais simples, como uma *call* americana cujo ativo subjacente não paga dividendos, esta poderia ser avaliada como sendo uma de estilo europeia por não ser vantajoso exercê-la antes do tempo de expiração (vencimento)

Uma das mais importantes propriedades das opções americanas é que seu valor $V(S_i, t_i)$, onde $S_i = S(t)$ representa o valor do ativo subjacente em t , deve ser maior ou igual ao seu valor intrínseco $\Pi(S_i, t_i)$ em qualquer momento até a data de expiração, $V(S_i, t_i) \geq \Pi(S_i, t_i)$. O valor intrínseco de uma opção se trata do lucro imediato da opção considerando apenas o *spot* e o *strike*. No caso da *call*: $\Pi(S_i, t_i) = \max(S_i - X, 0)$, e da *put* $\Pi(S_i, t_i) = \max(X - S_i, 0)$. No entanto, se essa propriedade não for atendida, surge uma oportunidade de arbitragem onde compraríamos uma opção americana e a venderíamos imediatamente de modo a lucrar $\Pi - V$ sem risco.

Antes de entrarmos nos modelos de precificação de opções americanas, é fundamental relembrarmos alguns conceitos básicos de um processo estocástico, como o processo de Markov, de Wiener e o lema de Itô. Estes serão abordados no tópico a seguir.

2.2. Processo Estocástico

Uma variável que tem o seu valor alterado ao longo do tempo de forma incerta podemos afirmar que a variável segue um processo estocástico. Processo estocástico pode ser classificado basicamente em dois tipos: um processo com verificação contínua no tempo da variável (ativo), ou verificação discretizada em determinados instantes de tempo. Os ativos

financeiros seguem um processo contínuo estocástico, no entanto, a análise nos mercados financeiros é feita em tempos espaçados (discretos).

2.2.1. Processo de Markov

O processo de Markov é um exemplo de processo estocástico onde o valor presente da variável aleatória é suficiente para descobrirmos o seu valor futuro. Todo o histórico é irrelevante para análise futura. Podemos demonstrar essa propriedade a partir da probabilidade condicional:

$$\begin{aligned} P\{X(t_i) = x_i | X(t_{i-1}) = x_{i-1}, X(t_{i-2}) = x_{i-2}, \dots, X(t_1) = x_1, X(t_0) = x_0\} \\ = P\{X(t_i) = x_i | X(t_{i-1}) = x_{i-1}\} \text{ , onde, } t_0 \leq t_1 \leq \dots \leq t_{i-1} \leq t_i \end{aligned} \quad (2.3)$$

A expressão (2.3) diz respeito à probabilidade condicional das ocorrências futuras da variável aleatória serem independentes do passado, levando em consideração apenas o estado presente.

Com isso podemos descrever este Processo Estocástico como *memoryless process* (processo sem memória), uma vez que o passado pode ser desprezado das amostras. A probabilidade de transição de n-passos pode ser definida como:

$$p_n(i, j) = P\{X_{n+m} = j | X_m = i\}.$$

2.2.2. Processo de Wiener

Processo de Wiener é uma extensão do processo de Markov com características de sua distribuição ter média 0 e variância 1 para a variável aleatória. Este processo foi muito utilizado primeiramente na física para descrever o movimento de uma partícula sujeita a um grande número de pequenos choques moleculares e este também conhecido como Movimento Browniano.

De acordo com HULL (2016, p. 325):

Uma variável z segue um processo de Wiener se possui as duas seguintes propriedades a seguir:

Propriedade 1: A mudança Δz durante um período curto Δt é:

$$\Delta z = \varepsilon \sqrt{\Delta t} \quad (2.4)$$

onde ε tem uma distribuição normal $\phi(0, 1)$.

Propriedade 2: Os valores de Δz para quaisquer dois intervalos curtos de tempo diferentes, Δt , são independentes. Logo, segundo a primeira propriedade, Δz em si tem uma distribuição normal com média de $\Delta z = 0$, desvio-padrão de $\Delta z = \sqrt{\Delta t}$, variância de $\Delta z = \Delta t$.

A segunda propriedade implica que z segue um processo de *Markov*.

Considere a mudança no valor de z durante um período relativamente longo, T . Esta pode ser denotada por $z(T) - z(0)$. Ela pode ser considerada como sendo igual à soma das mudanças em z em N intervalos de pequena duração Δt , onde:

$$N = \frac{T}{\Delta t}.$$

Assim:

$$z(T) - z(0) = \sum_{i=1}^N \varepsilon_i \sqrt{\Delta t} \quad (2.5)$$

onde os ε_i ($i = 1, 2, \dots, N$) são distribuídos $\phi(0, 1)$. Da segunda propriedade dos processos de *Wiener*, sabemos que os ε_i são independentes uns dos outros. Logo da equação (2.5), $z(T) - z(0)$ é normalmente distribuído com: média de $[z(T) - z(0)] = 0$, desvio padrão de $[z(T) - z(0)] = \sqrt{T}$, variância de $[z(T) - z(0)] = N\Delta t = T$.

2.2.3. Processo de Wiener Generalizado

De acordo com HULL (2016, p. 328):

A mudança média por unidade de tempo para um processo estocástico é conhecida como a taxa de deriva e a variância por unidade de tempo é conhecida como taxa de variância. O processo de *Wiener* básico, dz , que vimos no tópico anterior tem taxa de deriva zero e taxa de variância 1 (um). A taxa de deriva zero significa que o valor esperado de z em qualquer momento futuro é igual a seu valor atual. A taxa de variância 1 (um) significa que a variância da mudança em z em um intervalo de tempo T é igual a T . Um processo de *Wiener* generalizado para uma variável x pode ser definido em termos de dz como

$$dx = a dt + b dz \quad (2.6)$$

onde a e b são constantes.

Para entender a equação (2.6) é útil considerar os dois componentes no lado direito separadamente. O termo adt implica que x tem uma taxa de deriva esperada de a por unidade de tempo. Sem o termo bdz a equação é $dx = adt$ que implica que $dx / dt = a$. Integrando com relação ao tempo, obtemos: $x = x_0 + at$, onde x_0 é o valor de x no tempo 0. Em um período de tempo de duração T , a variável x aumenta na quantidade aT . O termo bdz no lado direito da equação (2.6) pode ser considerado como adicionando ruído ou variabilidade ao caminho seguido por x . O nível desse ruído ou variabilidade é b vezes um processo de *Wiener*. Um processo de *Wiener* tem uma taxa de variância por unidade de tempo 1. Logo, b vezes um processo de *Wiener* tem uma taxa de variância por unidade de tempo de b^2 . Em um pequeno intervalo Δt , a mudança Δx no valor de x é dada pelas equações (2.4) e (2.6) como:

$$\Delta x = a\Delta t + b\varepsilon\sqrt{\Delta t},$$

onde, assim como antes, ε tem uma distribuição normal padrão $\phi(0, 1)$. Assim, Δx tem uma distribuição normal com: média de $\Delta x = a\Delta t$, desvio padrão de $\Delta x = b\sqrt{\Delta t}$ e variância de $\Delta x = b^2\Delta t$.

Argumentos semelhantes àqueles dados para um processo de *Wiener* mostram que a mudança de valor de x em qualquer intervalo de tempo T é normalmente distribuída com: média da mudança em $x = aT$, desvio padrão da mudança em $x = b\sqrt{T}$ e variância da mudança em $x = b^2T$.

Figura 2 - Processo de Wiener generalizado $a=0.3$ e $b=1.5$.

Fonte: HULL (2016).

Em suma, o processo de *Wiener* generalizado dado na equação (2.6) tem uma taxa de deriva esperada (ou seja, deriva média por unidade de tempo) de a e uma taxa de variância (ou seja, variância por unidade de tempo) de b^2 . A Figura 2 ilustra a situação.

2.2.4. Processo de Itô

De acordo com HULL (2016, p. 329):

O processo de Itô é um processo de *Wiener* generalizado no qual os parâmetros a e b são funções do valor da variável subjacente x e do tempo t . Assim, um processo de Itô pode ser escrito como:

$$dx = a(x, t) dt + b(x, t) dz.$$

A taxa de deriva e a taxa de variância esperadas de um processo de Itô podem mudar com o tempo. Em um pequeno intervalo entre t e $t + \Delta t$, a variável muda de x para $x + \Delta x$, onde $\Delta x = a(x, t) \Delta t + b(x, t) \varepsilon \sqrt{\Delta t}$.

Essa equação envolve uma pequena aproximação. Ela pressupõe que a taxa de deriva e de variância de x permanecem constantes, iguais a seus valores no tempo t , durante o intervalo de tempo entre t e $t + \Delta t$.

Observe que o processo na equação é de *Markov* porque a mudança em x no tempo t depende apenas do valor de x no tempo t , não de seu histórico. Um processo não *Markoviano* poderia ser definido fazendo com que a e b na equação dependessem dos valores de x antes do tempo t .

2.2.5. Lema de Itô

De acordo com HULL (2016, p. 335):

O preço de uma opção sobre ações é uma função do preço da ação subjacente e do tempo. Em termos mais gerais, podemos dizer que o preço de qualquer derivativo é uma função das variáveis estocásticas subjacentes ao derivativo e do tempo.

Suponha que o valor de uma variável x segue o processo de Itô:

$$dx = a(x, t)dt + b(x, t)dz \quad (2.7)$$

onde dz é um processo *Wiener* e a e b são funções de x e t . A variável x tem uma taxa de deriva de a e a taxa de variância de b^2 . O lema de Itô mostra que, para uma função qualquer G de x e t , segue o processo:

$$dG = \left(\frac{\partial G}{\partial x} a + \frac{\partial G}{\partial t} + \frac{1}{2} \frac{\partial^2 G}{\partial x^2} b^2 \right) dt + \frac{\partial G}{\partial x} b dz,$$

onde dz é o mesmo processo de *Wiener* que na equação (2.7). Assim, G também

segue um processo de Itô, com taxa de deriva de: $\frac{\partial G}{\partial x} a + \frac{\partial G}{\partial t} + \frac{1}{2} \frac{\partial^2 G}{\partial x^2} b^2$,

e a taxa de variância de: $\left(\frac{\partial G}{\partial x} \right)^2 b^2$.

Uma prova mais rigorosa do lema de Itô pode ser encontrado, em: ITO (1944). Este lema vai ser utilizado posteriormente na precificação do modelo de *Black-Scholes*.

3. Modelos de Precificação de Opções

Neste capítulo iremos apresentar o modelo de *Black & Scholes* e os modelos de precificação de opções americanas que serão abordados neste trabalho.

3.1. Modelo de *Black & Scholes*

O modelo de *Black & Scholes* permite calcularmos o preço de uma opção *Plain Vanilla* (instrumento financeiro de opção mais elementar) *call* ou *put* europeia através de uma solução analítica. O ativo objeto subjacente da opção segue um movimento browniano geométrico que pode ser representado pela equação diferencial estocástica abaixo, de BLACK & SCHOLES (1973):

$$dS(t) = \mu S dt + \sigma S dW(t) \quad (3.1)$$

sendo:

$S(t)$: Valor do Ativo Subjacente em t ,

μ : *Drift* (tendência) Constante na medida de probabilidade real P

σ : Volatilidade constante do ativo subjacente S

$dW(t)$: Processo estocástico em relação ao ativo S (Movimento Browniano Geométrico)

Pelo lema de Itô citado no capítulo anterior, e a partir de uma carteira dinâmica livre de risco, ao qual podemos traduzir o *drift* (μ) do modelo teoricamente utilizado para a taxa livre de risco r , o preço da opção C (*call* ou *put*), relacionado ao comportamento do valor de S , pode ser representado pela equação diferencial parcial:

$$\frac{1}{2} \sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS \frac{\partial C}{\partial S} + \frac{\partial C}{\partial t} - rC = 0 \quad (3.2)$$

A resolução da EDP acima trará a função prêmio para uma opção de *call* europeia com *strike* K como sendo:

$$C(S, t) = SN(d_1) - Ke^{-r(T-t)}N(d_2) \quad (3.3)$$

Para uma opção *put* europeia será:

$$P(S, t) = Xe^{-r(T-t)} N(-d_2) - SN(-d_1) \quad (3.4)$$

onde temos:

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{(T-t)}} ,$$

$$d_2 = d_1 - \sigma\sqrt{(T-t)} .$$

A função $N(x)$ é uma função de distribuição de probabilidade cumulativa de uma variável normalmente padronizada. $N(d_1)$, ou delta da opção, representa a quantidade de papéis necessários para termos em um portfólio replicante protegido em relação à exposição de seu derivativo, ou seja, um portfólio sem risco de mercado.

3.2. Método dos Mínimos Quadrados de Monte Carlo (LSM)

LSM é o método dos mínimos quadrados de Monte Carlo, que consiste na avaliação de opções americanas através de simulações e de regressões simples.

As principais características do método são:

- Simulação do ativo é um processo *Forward*
- Avaliação da parada ótima é um processo *Backward*
- A expectativa condicionada do valor de continuação pode ser aproximada por método dos mínimos quadrados.
- Pode ser utilizado na avaliação de opções que dependem do caminho, de ativos que seguem um processo estocástico com *jumps* e de opções que dependem de múltiplos fatores de risco.

A ideia que está por trás do método é a da programação dinâmica: a cada instante anterior à data de vencimento de uma opção americana, o proprietário desta opção compara o *payoff* do exercício antecipado com o seu valor de continuação, para assim tomar uma decisão ótima.

A estratégia de exercício ótimo de uma opção americana é determinada fundamentalmente pela expectativa condicionada do seu valor de continuação.

O objetivo do algoritmo LSM é fornecer uma aproximação para a regra de exercício ótimo que maximiza o valor de uma opção americana. Seja $C(w, s; t, T)$ os fluxos de caixa gerados pela opção, condicionada à restrição da opção não ser exercida antes do instante t , onde w corresponde a uma simulação de trajetória de preços. O dono da opção seguirá uma estratégia de exercício ótimo para cada instante s , onde $t < s < T$.

Na data de vencimento T da opção, o investidor exercerá a opção se ela estiver no lucro. Em qualquer instante t_n anterior à data T , porém, o investidor deve escolher entre exercer imediatamente a opção ou esperar até o instante t_{n+1} para novamente tomar uma decisão. Em cada trajetória de fluxos de caixa, a opção será maximizada se for exercida quando o valor de exercício antecipado for superior ou igual ao valor de continuação. Em t_n , o valor de exercício antecipado é conhecido pelo investidor, mas o valor de continuação não é. A teoria de avaliação de ativos sem arbitragem, porém, indica que o valor de continuação da opção é dado pelo valor esperado dos fluxos de caixa futuros descontados até t_n com base numa medida Q de probabilidade de risco neutra. Assim, o valor de continuação em t_n , definido como $F(w, T_n)$, será dado pela seguinte fórmula de LONGSTAFF & SCHWARTZ (2001, p. 9):

$$F(w, t_n) = E_Q \left[\sum_{j=n+1}^N \exp \left(- \int_{t_n}^{t_j} r(w, s) ds \right) C(w, t_j; t_n, T) \mid I(t_n) \right] \quad (3.5)$$

onde, $r(w, t)$ é a taxa de desconto livre de risco, $I(t_n)$ corresponde ao conjunto de informações disponíveis em t_n para a tomada de decisão, e n é o número de intervalos de tempo entre o instante inicial e T ($n = T / \Delta t$ e $t_n = T$).

Para calcular a expectativa condicionada ao valor de continuação, o LSM utiliza uma aproximação pelo método dos mínimos quadrados. Em t_{n-1} , assumimos que $F(w, T_{n-1})$ é desconhecido, mas pode ser representado como uma combinação linear de um grupo de funções básicas $L_j(X)$, onde X é uma variável de estado. Assim, o valor de continuação pode ser escrito da seguinte forma (LONGSTAFF & SCHWARTZ, 2001, p. 10):

$$F(w, t_{n-1}) = \sum_{j=0}^{\infty} a_j L_j(X) \quad (3.6)$$

onde os coeficientes a_j são constantes. As funções básicas podem ser diversas como, funções exponenciais, potências, polinômios de Jacobi, entre outros. Assim, a aproximação é dada através da regressão dos valores descontados $C(w, s; t_{n-1}, T)$ nas funções básicas para as trajetórias onde a opção está *in-the-money* no instante t_{n-1} . Com a função expectativa condicional estimada no instante t_{n-1} , pode-se determinar se o exercício antecipado neste instante é ótimo através da comparação com o resultado da função para cada simulação. Do mesmo modo o procedimento é repetido para todos os instantes até o inicial.

O valor da opção será a média dos fluxos de caixa provenientes do exercício ótimo em cada simulação, descontados até o instante inicial. A equação (3.7) abaixo mostra em detalhes o valor da opção referente a BAIDYA & ARAUJO (2004, p. 3):

$$V_{Opcao} = \frac{1}{K} \sum_{w=1}^K FC(w, t_w^*) \exp(-t_w^* r) \quad (3.7)$$

onde,

r = taxa de desconto livre de risco,

K = número de trajetórias de preços do ativo subjacente,

t_w^* = data de exercício ótima da opção na simulação w ,

$FC(w, t_w^*)$ = fluxo de caixa gerado pelo exercício da opção no instante t_w^* na simulação w .

3.3. Modelo de Árvore Binomial com Extrapolação de Richardson

Cox, Ross & Rubinstein (1979) desenvolveram um método que converge para a solução encontrada por Black & Scholes (1973) descrita em (3.1). Essa convergência pode ser obtida como um limite contínuo de uma trajetória aleatória em tempo discreto. Para isso, o tempo deve ser dividido em períodos discretos de intervalos Δt e então utilizar o modelo de programação dinâmica para encontrar o preço da opção. No modelo binomial os preços do ativo subjacente S move-se para cima ou para baixo a partir de incrementos u e d . A probabilidade de mover-se para cima é dada por uma medida de risco neutra p e para baixo por $q = (1 - p)$. Abaixo temos um exemplo de uma Árvore Binomial com três períodos.

Figura 3 - Árvore Binomial de 3 períodos.

Fonte: BAIDYA & CASTRO (2001, p. 3).

Para encontrarmos os parâmetros da árvore u , d e p precisaremos retornar à equação (3.1) assumindo um mercado completo com uma taxa de juros livre de risco r em uma medida *Martingale* Equivalente, ou seja, uma medida ao qual acontecimentos passados não ajudam a prever resultados futuros. Além disso, levar em consideração os dividendos pagos pelo ativo subjacente. Portanto, reescrevendo a equação, teremos:

$$dS(t) = (r - \delta)Sdt + \sigma SdW(t) \quad (3.8)$$

Dividindo a equação (3.8) por S , substituindo a variável S por x e utilizando a relação $x = \ln(S)$ e fazendo a mudança de variável:

$$\frac{dx}{dS} = \frac{1}{S} \Rightarrow dx = \frac{dS}{S} \quad (3.9)$$

teremos:

$$dx = (r - \delta)dt + \sigma dW(t) \quad (3.10)$$

Substituindo $v = r - \delta$, e tomando variações instantâneas discretas de x e t a equação (3.10) pode ser escrita da seguinte forma:

$$\Delta x = v\Delta t + \sigma\Delta W \quad (3.11)$$

onde $\Delta t = T / n$, sendo T o prazo até o vencimento e n o número de intervalos de tempo. O valor esperado e a variância de Δx , de acordo com o movimento browniano será:

$$E(\Delta x) = v\Delta t \quad (3.12)$$

$$Var(\Delta x) = \sigma^2 \Delta t \quad (3.13)$$

lembrando que $E(\Delta W) = 0$ e $Var(\Delta W) = \Delta t$.

Como houve troca de variáveis de S para x , a árvore binomial deve também ser alterada para exibir variações em x . Portanto, a variável x move-se para cima ou para baixo segundo incrementos: $u_x = \ln(u)$, $d_x = \ln(d)$. Então o valor esperado e a variância a partir da distribuição de valores de toda árvore em x será:

$$E(\Delta x) = p \ln(u) + q \ln(d) \quad (3.14)$$

$$Var(\Delta x) = pq(\ln(u) - \ln(d))^2 \quad (3.15)$$

Como a distribuição de probabilidade de cada período da árvore é dado por uma distribuição binomial, então quanto maior o número de intervalos de tempo, mais essa distribuição se aproxima de uma distribuição normal. Portanto, devemos estabelecer os parâmetros para que no limite, as médias e as variâncias entre as duas distribuições sejam iguais. Para isso, igualamos as equações (3.12) e (3.14), (3.13) e (3.15), ou seja,

$$v\Delta t = p \ln(u) + q \ln(d),$$

$$\sigma^2 \Delta t = pq(\ln(u) - \ln(d))^2,$$

$$p + q = 1.$$

Este sistema possui três equações e quatro incógnitas. De modo a obter uma solução para o sistema descrito acima foi acrescentado uma nova equação: $\ln(u) = -\ln(d)$, com isso a solução foi encontrada assumindo que potências de Δt superiores a unidade deve tender a zero à medida que $n \rightarrow \infty$. Portanto, os parâmetros da árvore serão: $u = e^{\sigma\sqrt{\Delta t}}$, $d = e^{-\sigma\sqrt{\Delta t}}$,

$$p = \frac{1}{2} + \frac{1}{2} \frac{v}{\sigma} \sqrt{\Delta t}.$$

Ao assumir $\ln(u) = -\ln(d)$, ou $u = 1/d$ do sistema: a árvore binomial torna-se recombinante e com isso o número de nós diminui drasticamente em cada período, à medida que n aumenta.

Uma árvore binomial onde o período entre o início da opção e seu vencimento foi dividido em n intervalos de tempo, possui $n + 1$ nós no n -ésimo período. Um algoritmo para calcular o valor de uma opção usando os modelos de árvores binomiais não necessita armazenar mais do que $n + 1$ nós. Assim, o preço da ação é calculado recursivamente e o valor em cada nó é dado por $Su^j d^{n-j}$, como podemos ver na Figura 3.

A partir do processo de programação dinâmica *backward* é inicializada estabelecendo o preço da *call* igual a $C_T(S_T) = \max(S_T - K, 0)$, em cada um dos nós terminais T . No instante, $T - \Delta t$ correspondente ao nó Su^2 no nosso exemplo, o valor da opção é dado por BAIDYA & CASTRO (2001, p.7):

$$C_{T-\Delta t}(Su^2) = \max\{Su^2 - K, e^{-r\Delta t}(pC_T(Su^3) + qC_T(Su^2))\},$$

ou seja, o valor de uma opção americana é o máximo entre o valor de exercício imediato e o valor presente de manter a opção viva (valor de continuação). O valor da opção nos demais nós são determinados recursivamente de maneira similar.

3.3.1. Convergência com BBSR

Um dos problemas do modelo de árvore binomial diz respeito à convergência oscilatória lenta. De modo a resolver este problema, o presente trabalho irá utilizar a extrapolação de Richardson presente no estudo de BROADIE & DETEMPLE (1996, p.35) que consiste em substituir o valor de continuação no último período antes do vencimento pela fórmula de Black & Scholes. Este tratamento também é conhecido como Binomial Black & Scholes (BBS). Em seguida irá ser adicionado uma extrapolação de dois pontos diretamente sobre o número de períodos de tempo do modelo BBS. Com isso, o preço da opção americana com n períodos de tempo irá ter os seguintes passos:

- Calcular o preço de uma opção usando o modelo BBS com n períodos de tempo (C_n);

- Calcular o preço de uma opção usando o modelo BBS com $m = n / 2$ períodos de tempo (C_m);
- Usar a fórmula de extrapolação $2C_n - C_m$.

3.4. Aproximação Analítica de *Bjerk Sund & Stensland* (B&S)

A aproximação de *Bjerk Sund & Stensland* (1993) é baseada em uma estratégia de exercício correspondente a um limite de fronteira, ou seja, partir de um preço gatilho (barreira I). Dada pela estratégia viável, mas não ótima, a *call* americana se resume a:

- Uma opção *Knockout UP* com barreira I e *strike* K ,
- Rebate que é recebido dado se bateu a barreira antes do vencimento.

De acordo com BRANKA, ZDRAVKA, TEA (2014, p.4):

Logo, esta aproximação é dada por:

$$c = \alpha S^\beta - \alpha \phi(S, T, \beta, I, I) + \phi(S, T, 1, I, I) - \phi(S, T, 1, K, I) - K \phi(S, T, 0, I, I) + K \phi(S, T, 0, K, I),$$

onde,

$$\alpha = (I - K)I^\beta,$$

$$\beta = \left(\frac{1}{2} - \frac{b}{\sigma^2} \right) + \sqrt{\left(\frac{b}{\sigma^2} - \frac{1}{2} \right)^2 + 2 \frac{r}{\sigma^2}},$$

A função $\phi(S, T, \gamma, H, I)$ é dada por:

$$\phi(S, T, \gamma, H, I) = e^{\lambda S^\gamma} \left[N(d) - \left(\frac{I}{S} \right)^k N \left(d - \frac{2 \ln \left(\frac{I}{S} \right)}{\sigma \sqrt{T}} \right) \right],$$

$$\lambda = \left[-r + \gamma b + \frac{1}{2} \gamma (\gamma - 1) \sigma^2 \right] T,$$

$$d = \frac{\ln \left(\frac{S}{H} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}},$$

$$k = \frac{2b}{\sigma^2} + (2\gamma - 1).$$

A barreira I é definida por:

$$I = B_0 + (B_\infty - B_0)(1 - e^{h(T)}),$$

$$h(T) = -(bT + 2\sigma\sqrt{T}) \left(\frac{B_0}{B_\infty - B_0} \right), B_0 = \frac{\beta}{\beta - 1} K,$$

$$B_\infty = \max \left\{ K, \frac{r}{r - b} K \right\}.$$

Se $S > I$ então, é ideal exercer a opção imediatamente. O valor de uma opção americana *put* é dada pela transformação *put-call* no qual teremos:

$$p(S, K, T, r, b, \sigma) = c(S, K, T, r - b, -b, \sigma).$$

A paridade *call-put* e o entendimento da aproximação B&S (2002) também é demonstrada em: BJERKSUND & STENSLAND (2002, p.6)

A aproximação de *Bjerk Sund & Stensland (2002)* se diferencia da anterior ao dividir o tempo até o vencimento em duas partes, no qual cada parte terá uma barreira, ou seja, teremos uma barreira I_1 que representa a barreira de 0 até t , e I_2 que representa a barreira de t até T (vencimento).

Nesse caso, a aproximação do preço da opção americana *call* é:

$$\begin{aligned} c = & \alpha_2 S^\beta - \alpha_2 \phi(S, t_1, \beta, I_2, I_2) + \phi(S, t_1, 1, I_2, I_2) - \phi(S, t_1, 1, I_1, I_2) - \\ & K \phi(S, t_1, 0, I_2, I_2) + K \phi(S, t_1, 0, I_1, I_2) + \alpha_1 \phi(S, t_1, \beta, I_1, I_2) - \\ & \alpha_1 \psi(S, T, \beta, I_1, I_2, I_1, t_1) + \psi(S, T, 1, I_1, I_2, I_1, t_1) - \\ & \psi(S, T, 1, K, I_2, I_1, t_1) - K \psi(S, T, 0, I_1, I_2, I_1, t_1) + \\ & \psi(S, T, 0, K, I_2, I_1, t_1), \end{aligned}$$

onde,

$$\alpha_1 = (I_1 - K)I_1^{-\beta}, a_2 = (I_2 - K)I_2^{-\beta},$$

$$\beta = \left(\frac{1}{2} - \frac{b}{\sigma^2} \right) + \sqrt{\left(\frac{b}{\sigma^2} - \frac{1}{2} \right)^2 + 2 \frac{r}{\sigma^2}}.$$

A função $\phi(S, T, \gamma, H, I)$ é dada por:

$$\phi(S, T, \gamma, H, I) = e^{\lambda} S^{\gamma} \left[N(-d) - \left(\frac{I}{S} \right)^k N(-d_2) \right],$$

$$\lambda = \left[-r + \gamma b + \frac{1}{2} \gamma(\gamma - 1) \sigma^2 \right],$$

$$d = \frac{\ln\left(\frac{S}{H}\right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}},$$

$$d_2 = \frac{\ln\left(\frac{I^2}{SH}\right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}},$$

$$k = \frac{2b}{\sigma^2} + (2\gamma - 1).$$

As barreiras I_1 e I_2 são definidas por:

$$I_1 = B_0 + (B_{\infty} - B_0)(1 - e^{h_1}),$$

$$I_2 = B_0 + (B_{\infty} - B_0)(1 - e^{h_2}),$$

$$h_1 = -(bt_1 + 2\sigma\sqrt{T}) \left(\frac{K^2}{(B_{\infty} - B_0)B_0} \right),$$

$$h_2 = -(bT + 2\sigma\sqrt{T}) \left(\frac{K^2}{(B_{\infty} - B_0)B_0} \right),$$

$$t_1 = \frac{1}{2}(\sqrt{5} - 1)T,$$

$$B_0 = \frac{\beta}{\beta - 1}K,$$

$$B_\infty = \max \left\{ K, \frac{r}{r - b} K \right\}.$$

E a função $\psi(S, T, \gamma, H, I_2, I_1, t_1, r, b, \sigma)$ é dada por:

$$\psi(S, T, \gamma, H, I_2, I_1, t_1, r, b, \sigma) = e^{\lambda T S^\gamma} \left[M \left(-e_1, -f_1, \sqrt{\frac{t_1}{T}} \right) - \left(\frac{I_2}{S} \right)^K M \left(-e_2, -f_2, \sqrt{\frac{t_1}{T}} \right) - \left(\frac{I_1}{S} \right)^K M \left(-e_3, -f_3, \sqrt{\frac{t_1}{T}} \right) + \left(\frac{I_1}{I_2} \right)^K M \left(-e_4, -f_4, \sqrt{\frac{t_1}{T}} \right) \right],$$

onde $M(\cdot, \cdot, \cdot)$ é a distribuição normal bivariada acumulada e

$$\begin{aligned} e_1 &= \frac{\ln \left(\frac{S}{I_1} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] t_1}{\sigma \sqrt{t_1}}, e_2 = \frac{\ln \left(\frac{I_2^2}{S I_1} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] t_1}{\sigma \sqrt{t_1}}, \\ e_3 &= \frac{\ln \left(\frac{S}{I_1} \right) - \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] t_1}{\sigma \sqrt{t_1}}, e_4 = \frac{\ln \left(\frac{I_2^2}{S I_1} \right) - \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] t_1}{\sigma \sqrt{t_1}}, \\ f_1 &= \frac{\ln \left(\frac{S}{H} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}}, f_2 = \frac{\ln \left(\frac{I_2^2}{S H} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}}, \\ f_3 &= \frac{\ln \left(\frac{I_1^2}{S H} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}}, f_4 = \frac{\ln \left(\frac{I_1^2}{H I_2^2} \right) + \left[b + \left(\gamma - \frac{1}{2} \right) \sigma^2 \right] T}{\sigma \sqrt{T}}. \end{aligned}$$

Neste trabalho foi implantado a aproximação de *Bjerk Sund & Stensland (2002)* com duas barreiras.

4. Plataformas paralelas

Neste capítulo iremos apresentar as tecnologias de computação paralela oriundas da CPU como a *TPL (Task Parallel Library)* fornecida pela empresa Microsoft e oriundas da GPU como a plataforma *CUDA (Compute Unified Device Architecture)* da empresa NVIDIA utilizadas na implementação dos métodos de precificação de opções americanas abordadas neste trabalho e consequentemente os processos realizados com as quais.

4.1. CPU - Biblioteca de tarefas paralelas (*TPL*)

De acordo com MICROSOFT (2017), a Biblioteca Paralela de Tarefas (*TPL*) é um conjunto de tipos e APIs públicos acessíveis pela IDE Visual Studio em *System.Threading* e *namespaces* do *System.Threading.Tasks* referentes ao .NET Framework. O objetivo da *TPL* é tornar os desenvolvedores mais produtivos ao simplificar o processo de adicionar paralelismo e concorrência em aplicativos. A *TPL* dimensiona o grau de simultaneidade dinamicamente para usar todos os processadores disponíveis de forma mais eficiente. Além disso, a *TPL* lida com o particionamento do trabalho, a programação de threads em *ThreadPool*, o suporte ao cancelamento, o gerenciamento de estados e outros detalhes de baixo nível. Ao usar a *TPL*, você pode maximizar o desempenho do seu código enquanto se concentra no trabalho para o qual seu programa foi criado para realizar. Começando no .NET Framework 4, a *TPL* é a maneira preferencial para escrever código multissegmentado e paralelo. No entanto, nem todos os códigos são apropriados para paralelização; por exemplo, se um *loop* executa somente uma pequena quantidade de trabalho em cada iteração, ou se ele não é executado para muitas iterações, a sobrecarga da paralelização pode fazer com que o código seja executado mais lentamente. Além disso, a paralelização, assim como qualquer código *multithread*, acrescenta complexidade à execução do seu programa. Embora a *TPL* simplifique cenários *multithread*, recomendamos que você tenha uma compreensão básica dos conceitos de *threads*, por exemplo, bloqueios, *deadlocks* e condições de corrida para que possa usar a *TPL* efetivamente.

Figura 4 - Arquitetura TPL.

Fonte: MICROSOFT (2017).

A Figura 4 descreve o funcionamento da arquitetura *TPL* desde a preparação do código até sua execução.

Os padrões de computação paralela disponibilizados pela biblioteca *TPL* de acordo com CAMPBELL, JOHNSON, MILLER, TOUB (2010) “se concentram basicamente em paralelismo de dependência de controle, e de controle e dados”. Em relação ao de controle ao qual abordam casos em que as operações assíncronas são ordenadas apenas por restrição ao fluxo de controle, temos:

- *Parallel Loops*: Utilizado quando se deseja executar o mesmo cálculo em cada membro de uma estrutura de coleção, por exemplo uma lista, ou para uma variedade de índices e onde não há dependências entre os membros da coleção.
- *Parallel Tasks*: Utilizado quando se possui diversas operações assíncronas distintas para execução.

Em relação ao padrão de fluxo de controle e dependência de dados, temos:

- *Parallel Aggregation*: São apropriados quando se existem dependências de dados na iteração com a estrutura de coleção.

- *Futures*: Refere-se quando algumas operações produzem saídas que são necessárias como entradas de outras. A ordem das operações é restringida por um gráfico de dependência de dados. Algumas operações são realizadas em paralelo e outras em série, dependendo de quando as entradas se tornam acessíveis.
- *Dynamic Task Parallelism*: Em alguns casos, as operações são adicionadas dinamicamente à lista de tarefas a serem executadas à medida que a computação prossegue. Este padrão aplica-se a vários domínios, incluindo algoritmos de gráficos e classificação.
- *Pipelines*: Utilizado quando deseja-se alimentar sucessivas saídas de um componente para a fila de entrada de outro componente, de modo paralelo.

Figura 5 - Padrões de Paralelismo TPL.

Fonte: CAMPBELL, JOHNSON, MILLER, TOUB (2010).

A Figura 5 apresenta os padrões de paralelismo supracitados separados em dependência do fluxo de controle e de controle e dados.

4.2. GPU - CUDA

A GPU (*Graphics Processing Unit*) trata-se da unidade gráfica de processamento de tarefas amplamente presente nos computadores modernos. À medida que a demanda por computação paralela de alto desempenho cresce em áreas de engenharia, finanças e medicina, por exemplo, mais se existe a procura de soluções robustas que atendam tais requisitos de processamento paralelo. Esta tecnologia está disseminada nos mais variados setores, como: identificação de placas ocultas em artérias, análise do fluxo de tráfego aéreo, visualização de moléculas, modelos computacionais, modelagem climática, dinâmica computacional de fluidos, entre outros.

As primeiras GPUs foram projetadas como aceleradores gráficos, suportando somente *pipelines* de função fixa específicos. A partir do final dos anos 90, o *hardware* tornou-se cada vez mais programável, culminando na primeira GPU NVIDIA em 1999. Com o passar dos anos foi se evoluindo o *hardware* de forma considerável através das arquiteturas NVIDIA FERMI (2010), NVIDIA KEPLER (2012), NVIDIA MAXWELL (2014) até os dias atuais com NVIDIA PASCAL (2016).

4.2.1. GPU NVIDIA

Uma maneira de demonstrar a grande diferença entre uma GPU e uma CPU é comparar o modo que elas processam as tarefas. Uma CPU tem alguns núcleos otimizados para o processamento serial sequencial, enquanto uma GPU tem uma arquitetura paralela robusta que consiste em milhares de núcleos menores e mais eficientes criados para lidar com múltiplas tarefas simultaneamente. Consequentemente, a GPU é construída para aplicações com demandas diferentes da CPU, ou seja, para cálculos paralelos grandes com mais ênfase no *throughput* que na latência. Por essa razão, sua arquitetura tem progredido em uma direção diferente da CPU. As GPU da NVIDIA são divididas em vários SMs (*Streaming Multiprocessors*), que por sua vez, executam grupos de *threads*, chamados de *warps*. Cada SM possui vários núcleos, chamados de CUDA *cores*.

Figura 6 - Aplicação CPU e GPU.

Fonte: ANGERER (2015).

A Figura 6 simboliza uma aplicação em execução mostrando a quantidade de núcleos de processamento distintos entre a CPU e a GPU.

No tópico a seguir iremos detalhar a arquitetura KEPLER presente na placa de vídeo utilizada neste trabalho.

4.2.2. Arquitetura Kepler

De acordo com NVIDIA KEPLER (2012), a arquitetura Kepler introduziu uma nova tecnologia, chamada *Hyper-Q*, que permite diferentes programas de CPU iniciar *kernels* (funções) específicos e diferentes no mesmo dispositivo. Na prática, isso significa que diferentes programas ou sistemas podem usar simultaneamente a mesma GPU. Antes disso, sempre que um *kernel* fosse iniciado, a GPU não poderia receber trabalhos de outros segmentos de CPU ou de programas, sendo necessário um controle de sincronização explícito. Além disso, um novo recurso importante que foi incluído pela geração Kepler é a possibilidade de lançar *kernels* dentro de outros *kernels*. Nas arquiteturas passadas, somente a CPU conseguia iniciar um *kernel*, mas as melhorias da arquitetura Kepler SM possibilitaram chamar um *kernel* diretamente do dispositivo. Esse recurso é chamado de paralelismo dinâmico e oferece um gerenciamento de *kernel* mais complexo com até chamadas de *kernel* recursivas. Para habilitar

esse recurso, a Arquitetura Kepler introduziu um novo controle de *grid* de unidade de gerenciamento (GMU), que é capaz de pausar, ou suspender atividades até a GPU estar pronta para executar. Normalmente, cada nova geração de GPU tem novos modelos SM. O modelo *Kepler* SM é chamado SMX e está demonstrado na Figura 7. O SMX é composto por 192 núcleos e 32 SPU (*Special Function Unit*), responsável pelas operações transcendentais rápidas e aproximadas, como em GPUs de geração anterior. Outra melhoria da arquitetura Kepler diz respeito ao gerenciamento de energia mais eficiente, permitindo quase 3 vezes mais desempenho por watt. Uma versão completa do Kepler é composta por 15 unidades SMX, com seis de 64 bits controladores de memória.

Figura 7 - SMX Kepler.

Fonte: NVIDIA KEPLER (2012).

Figura 8 - Comparativo Arquiteturas GPU.

	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K	16K	16K	16K
	48K	48K	32K	32K
			48K	48K
Max X Grid Dimension	$2^{16}-1$	$2^{16}-1$	$2^{32}-1$	$2^{32}-1$
Hyper-Q	No	No	No	Yes
Dynamic Parallelism	No	No	No	Yes

Compute Capability of Fermi and Kepler GPUs

Fonte: NVIDIA KEPLER (2012).

A Figura 8 exibe a diferença entre os componentes da arquitetura Kepler com as anteriores.

4.2.3. Arquitetura CUDA

CUDA, sigla para *Compute Unified Device Architecture*, é uma plataforma de computação paralela que possui um modelo de programação inventado pela empresa NVIDIA. Com a plataforma é possível ter ganhos expressivos de performance nas execuções de tarefas pois utiliza-se a potência do *hardware* da unidade gráfica de processamento (GPU) conforme citado anteriormente.

Em 2006 Ian Buck, grande responsável pela criação da plataforma, se juntou à empresa NVIDIA e começou a desenvolver uma solução para executar a linguagem de programação C diretamente na GPU. A partir dessa evolução nascia a importante plataforma de computação paralela **CUDA** que utilizaremos neste trabalho.

A arquitetura CUDA é baseada em alguns conceitos, dentre eles, podemos citar:

- Computação Heterógena: O código CUDA é composto por funções executadas no *Host* (CPU) e no dispositivo (GPU). O fluxo das informações entre os *hardwares* CPU e GPU pode ser visto nas Figuras 9, 10, e 11.

Figura 9 - Arquitetura CUDA - Dados CPU para GPU.

Fonte: ANGERER (2015).

Figura 10 - Arquitetura CUDA - Execução código GPU.

Fonte: ANGERER (2015).

Figura 11 - Arquitetura CUDA - Resultado GPU para CPU.

Fonte: ANGERER (2015).

- Blocos: Os blocos referem-se a cada invocação de execução das funções na GPU de forma paralela, ou seja, o número de blocos em uma aplicação é uma das variáveis determinantes no paralelismo na GPU. Um conjunto de blocos define o *grid size*.
- Threads: Cada bloco pode conter múltiplas ações a serem executadas em paralelo. Essas ações podem ser definidas como *threads*. Um conjunto de *threads* define um *block size*.
- Indexação: A configuração da combinação entre o número de blocos e *threads* em uma aplicação trata-se da indexação. Cada indexação pode ter diversos blocos com uma *thread* cada, ou, um bloco com múltiplas *threads*. Com uma melhor indexação a aplicação se torna mais eficiente.

Figura 12 - Arquitetura CUDA - Threads por bloco.

Fonte: NVIDIA (2017).

A Figura 12 acima exibe uma configuração de 6 blocos por *grid* e com 15 *threads* cada bloco.

- Memória Compartilhada: É possível definir dados que podem ser compartilhados por *threads* de dentro de cada bloco maximizando o processamento das aplicações.

- Operações assíncronas: Funções executadas na GPU são invocadas de forma assíncrona a CPU até sua conclusão.

4.2.4. Biblioteca cuRAND

O Kit de ferramentas CUDA (*CUDA Toolkit*) fornece um ambiente de desenvolvimento para a criação de aplicativos acelerados por GPU de alto desempenho. Nele inclui diversas bibliotecas de otimização de processamento, ferramentas de depuração e um compilador na linguagem C/C++. As bibliotecas fornecem funções dos mais variados ramos de aplicações, como: de álgebra linear, processamento de imagem e vídeo, *deep learning*, geração de números aleatórios, entre outros. Especificamente neste trabalho foi utilizada a biblioteca de geração de números aleatórios chamada (*cuRAND*).

De acordo com NVIDIA cuRAND (2017), a biblioteca cuRAND fornece facilidades na geração simples e eficiente de números pseudoaleatórios e quasi-aleatórios de alta qualidade tanto na CPU quanto na GPU. Uma sequência de números pseudorrandômicos satisfaz a maioria das propriedades estatísticas de uma sequência verdadeiramente aleatória, mas é gerada por um algoritmo determinista. Uma sequência quasirandômica de pontos n-dimensionais é gerada por um algoritmo determinista projetado para preencher uniformemente o espaço n-dimensional.

Os números aleatórios são produzidos por geradores. Um gerador em cuRAND encapsula todo o estado interno necessário para produzir uma sequência de números pseudoaleatórios ou quasi-aleatórios. A sequência normal de operações para produção destes números é a seguinte:

1. Criar um novo gerador. Existem cinco tipos diferentes de geradores, sendo três geradores de números pseudoaleatórios: XORWOW, MRG32K3A, MTGP32 e dois geradores de números quasi-aleatórios: SOBOL e SCRAMBLED SOBOL, estes possuindo algumas categorias em 32 e 64 bits.
2. Definição da semente para inicialização do número aleatório.
3. Possibilidade de definição de um *offset* para saltar a sequência dos números aleatórios gerados.

4. O parâmetro da ordem é usado para escolher como os resultados são ordenados na memória global.

Figura 13 - Desempenho dos geradores de números aleatórios em relação às funções de distribuição Log-Normal, Normal e Uniforme.

cuRAND: HIGH PERFORMANCE RNGS

Fonte: NVIDIA cuRAND (2017).

A Figura 13 apresenta a performance em relação a giga amostras por segundo da geração de números pseudoaleatórios e quasi-aleatórios a partir das funções de distribuição Log-Normal, Normal e Uniforme.

5. Metodologia

Neste capítulo iremos descrever os insumos de dados de mercado, o portfólio, os processos de cálculos, o *hardware* utilizado.

5.1. Portfólio

Neste presente trabalho foram utilizadas 48 opções americanas de venda de ações VALE5 referentes à data base 17/07/2017. Foram calculadas opções com diferentes vencimentos e diferentes *strikes*. Os *strikes* foram produzidos para fins de simulação e comparação dos modelos apresentados. Na Tabela 1 temos os *strikes*:

Tabela 1 - Ativo Objeto x Strikes.

Ativo Objeto	Strikes
VALE 5	24
VALE 5	25
VALE 5	26
VALE 5	27
VALE 5	28
VALE 5	29

Fonte: produzida pelo próprio Autor.

5.2. Dados de Mercado

Os dados de mercado empregados neste trabalho foram retirados da plataforma da Bloomberg a partir do comando “VALE BZ EQUITY OVME”. A Bloomberg é uma plataforma de dados, notícias e análises em tempo real para tomar decisões de negócios para o setor financeiro.

Dentre os dados de mercado de BLOOMBERG (2017), estão: vencimento das opções de ações VALE5, taxa pré (livre de risco) no período, *spot* de Vencimento do *Underling* (VALE5), volatilidade implícita da opção, prêmio de mercado de Tela, cujos dados estão exibidos na Tabela 2 a seguir.

Tabela 2 - Dados Financeiros.

Vencimento	Taxa pré	Spots	Volatilidade Implícita	Premio
21/08/2017	0,09598	27,92	0,45797	0,24
21/08/2017	0,09598	27,92	0,40446	0,3
21/08/2017	0,09598	27,92	0,35868	0,41
21/08/2017	0,09598	27,92	0,32818	0,63
21/08/2017	0,09598	27,92	0,31439	1,02
21/08/2017	0,09598	27,92	0,31135	1,58
18/09/2017	0,09271	27,92	0,44866	0,49
18/09/2017	0,09271	27,92	0,40329	0,58
18/09/2017	0,09271	27,92	0,36292	0,72
18/09/2017	0,09271	27,92	0,3317	0,94
18/09/2017	0,09271	27,92	0,3121	1,3
18/09/2017	0,09271	27,92	0,30292	1,81
16/10/2017	0,09052	27,92	0,42868	0,66
16/10/2017	0,09052	27,92	0,39444	0,78
16/10/2017	0,09052	27,92	0,36645	0,97
16/10/2017	0,09052	27,92	0,34575	1,25
16/10/2017	0,09052	27,92	0,33286	1,63
16/10/2017	0,09052	27,92	0,32611	2,14
21/11/2017	0,08852	27,92	0,43115	0,92
21/11/2017	0,08852	27,92	0,40333	1,08
21/11/2017	0,08852	27,92	0,38005	1,3
21/11/2017	0,08852	27,92	0,36157	1,59
21/11/2017	0,08852	27,92	0,3488	1,98
21/11/2017	0,08852	27,92	0,3408	2,46
18/12/2017	0,08719	27,92	0,43292	1,11
18/12/2017	0,08719	27,92	0,4068	1,28
18/12/2017	0,08719	27,92	0,38515	1,51
18/12/2017	0,08719	27,92	0,36759	1,81
18/12/2017	0,08719	27,92	0,35483	2,19
18/12/2017	0,08719	27,92	0,34617	2,66
15/01/2017	0,08614	27,92	0,43594	1,29
15/01/2017	0,08614	27,92	0,41247	1,48
15/01/2017	0,08614	27,92	0,39339	1,73
15/01/2017	0,08614	27,92	0,37856	2,05
15/01/2017	0,08614	27,92	0,36757	2,44
15/01/2017	0,08614	27,92	0,35981	2,91
19/02/2017	0,08529	27,92	0,43701	1,49
19/02/2017	0,08529	27,92	0,4173	1,71
19/02/2017	0,08529	27,92	0,40119	1,98
19/02/2017	0,08529	27,92	0,38848	2,31
19/02/2017	0,08529	27,92	0,37881	2,72
19/02/2017	0,08529	27,92	0,37172	3,19
19/03/2017	0,08487	27,92	0,43835	1,64
19/03/2017	0,08487	27,92	0,42067	1,87
19/03/2017	0,08487	27,92	0,40613	2,16
19/03/2017	0,08487	27,92	0,39454	2,5
19/03/2017	0,08487	27,92	0,38556	2,91
19/03/2017	0,08487	27,92	0,37882	3,38

Fonte: produzida pelo próprio Autor.

5.3. Processos de Cálculo

Para a avaliação e simulação dos modelos de precificação de opções americanas foram desenvolvidas duas aplicações:

1. Na linguagem C# através da IDE Visual Studio 2013 .NETFramework 4.5 foram gerados processos de cálculo sequencial e paralelo (*TPL*) dos modelos de cálculo de:
 - a. Aproximação analítica de *Bjerkstrand & Stensland* 2002,
 - b. Árvore binomial padrão,
 - c. Binomial *Black & Scholes* (BBS),
 - d. Binomial *Black & Scholes* com convergência de Richardson (BBSR),
 - e. Método dos Mínimos Quadrados de Monte Carlo.
2. Na linguagem C através da IDE Visual Studio 2013 e a instalação do CUDA Toolkit 8.0 que oferece o compilador de programação foram gerados processos de cálculo paralelo (CUDA) dos seguintes modelos:
 - a. Aproximação analítica de *Bjerkstrand & Stensland* 2002,
 - b. Árvore binomial padrão,
 - c. Método dos Mínimos Quadrados de Monte Carlo.

5.4. Hardware

A descrição dos *Hardwares* da CPU e da GPU utilizados neste trabalho foram retirados da ferramenta *NVIDIA Parallel Nsight*, vista em: NVIDIA NSIGHT (2017), que permite realizar a depuração e a otimização de códigos construídos em CUDA na GPU e também realiza o rastreamento dos dispositivos disponíveis. As Figuras 14 e 15 fornecem esses dados, respectivamente.

Figura 14 - Descrição da CPU.

Attribute	Value
^ CPU	
Name	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
Architecture	x64
Frequency	2.594 MHz
Number of Cores	4
Page Size	4.096
Total Physical Memory	8.097,00 MB
Available Physical Memory	3.390,00 MB
^ Misc	
Hybrid Graphics Enabled	False
^ Operating System	
Version Name	Windows 10 Home Single Language
Version Number	10.0.15063
^ Tools	
Nsight Version	5.2.0.16321
Nsight Edition	Standard
Visual Studio Version	12.0

Fonte: Ferramenta *NVIDIA NSIGHT* (2017).

Figura 15 - Descrição da GPU.

Attribute	GeForce GT 740M
Driver Version	384.76
Driver Model	WDDM
CUDA Device Index	0
GPU Family	GK208
Compute Capability	3.5
Number of SMs	2
Frame Buffer Physical Size (MB)	2048
Frame Buffer Bandwidth (GB/s)	14.4
Frame Buffer Bus Width (bits)	64
Frame Buffer Location	Dedicated
Graphics Clock (Mhz)	516
Memory Clock (Mhz)	900
Processor Clock (Mhz)	1032
RAM Type	DDR3
Attached Monitors	0

Fonte: Ferramenta *NVIDIA NSIGHT* (2017).

6. Resultados

Neste capítulo iremos apresentar as comparações realizadas entre as simulações de precificação de opções a partir dos modelos abordados neste trabalho com as finalidades de consistência do modelo em relação aos prêmios de mercado de tela coletados da BLOOMBERG; e também o uso de máquina através do monitor de recursos do Windows e as variações de tempo de processamento de plataforma de processamento sequencial, paralela *TPL* e paralela *CUDA*. Por fim, iremos avaliar o estudo da convergência para os modelos de Árvore Binomial, Binomial de Black & Scholes e Binomial de Black & Scholes com extrapolação de Richardson. Foi utilizada a medida de erro quadrático médio para as comparações gerado a partir da fórmula:

$$EQM = \frac{\sum_{i=1}^n e_i^2}{n}$$

sendo e_i , o erro na amostra i .

6.1. Simulação da Aproximação Analítica Bjerksund & Stensland 2002

Como este modelo diz respeito a uma aproximação analítica, as diferenças entre os tempos de processamento e uso de máquina são praticamente imperceptíveis para um portfólio de 48 opções. Em relação à alteração dos parâmetros de *GridSize* e *BlockSize*, demonstraram para esse cenário irrelevância. Seguem os resultados:

6.1.1. Erro Quadrático Médio

Na Tabela 3 podemos ver a consistência entre as aplicações por gerarem o mesmo erro quadrático com diferentes plataformas.

Tabela 3 - Modelo B&S 2002 – EQM.

Modelo	Plataforma	EQM
B&S 2002	Sequencial	0,0009
B&S 2002	Paralela TPL	0,0009
B&S 2002	Paralela CUDA	0,0009

Fonte: produzida pelo próprio Autor.

6.1.2. Tempo de Processamento e Uso de CPU

Tabela 4 - Modelo B&S 2002 - Tempo de Processamento e uso da CPU.

Modelo	Plataforma	Tempo de Process. (s)	Uso CPU (%)	GRID SIZE	BLOCK SIZE
B&S 2002	Sequencial	0,016	30	-	-
B&S 2002	Paralela TPL	0,007	35	-	-
B&S 2002	Paralela CUDA	0,013	30	1	256
B&S 2002	Paralela CUDA	0,012	30	8	256
B&S 2002	Paralela CUDA	0,012	30	16	256
B&S 2002	Paralela CUDA	0,012	30	32	256
B&S 2002	Paralela CUDA	0,012	30	256	256
B&S 2002	Paralela CUDA	0,012	30	1	512
B&S 2002	Paralela CUDA	0,012	30	8	512
B&S 2002	Paralela CUDA	0,012	30	16	512
B&S 2002	Paralela CUDA	0,012	30	32	512
B&S 2002	Paralela CUDA	0,012	30	256	512

Fonte: produzida pelo próprio Autor.

Figura 16 - Tempo de Processamento Modelo B&S 2002.

Fonte: produzida pelo próprio Autor.

A Tabela 4 e a Figura 16 comprovam o que falamos anteriormente, que para a precificação de opções americanas utilizando o modelo B&S a plataforma paralela não chega a ser um fator determinante na definição do tempo de processamento.

6.2. Simulação da Árvore Binomial Padrão

A simulação da árvore binomial, por sua vez, é mais complexa que o método anterior de aproximação analítica por possuir um estudo de otimizar o número de passos dado o critério de convergência a ser adotado. Iremos mostrar duas configurações de convergências, uma com três casas decimais (600 períodos) que visualizaremos na Tabela 7 o tempo de processamento e uso de máquina e a outra com cinco casas decimais (15000 períodos) que estará disponível

na Tabela 6. Ficou claro que com cinco casas decimais o tempo de processamento se tornou exorbitante, aproximadamente 77000% em relação à convergência de três casas decimais no caso sequencial. Este dado pode ser verificado através da Figura 17.

6.2.1. Erro Quadrático Médio

Na Tabela 5 podemos ver a consistência entre as aplicações por gerarem o mesmo erro quadrático com as diferentes plataformas. Foi constatado que, independente da convergência das casas decimais com o preço final do modelo, o valor do erro quadrático médio comparado ao preço de tela se manteve em 0.0003.

Tabela 5 - Modelo Árvore Binomial – EQM.

Modelo	Plataforma	EQM
Árvore Binomial	Sequencial	0,0003
Árvore Binomial	Paralela TPL	0,0003
Árvore Binomial	Paralela CUDA	0,0003

Fonte: produzida pelo próprio Autor.

6.2.2. Tempo de Processamento e Uso de CPU

Tabela 6 - Modelo Árvore Binomial com 15000 períodos.

Modelo	Plataforma	Tempo de Process. (s)	Uso CPU (%)	GRID SIZE	BLOCK SIZE	PERIODOS
Arvore Binomial	Sequencial	771,031	35	-	-	15000
Arvore Binomial	Paralela TPL	305,620	100	-	-	15000
Arvore Binomial	Paralela CUDA	3,361	30	1	256	15000
Arvore Binomial	Paralela CUDA	3,202	30	8	256	15000
Arvore Binomial	Paralela CUDA	3,262	30	16	256	15000
Arvore Binomial	Paralela CUDA	3,314	30	32	256	15000
Arvore Binomial	Paralela CUDA	3,251	30	256	256	15000
Arvore Binomial	Paralela CUDA	3,276	30	1	512	15000
Arvore Binomial	Paralela CUDA	3,332	30	8	512	15000
Arvore Binomial	Paralela CUDA	3,212	30	16	512	15000
Arvore Binomial	Paralela CUDA	3,209	30	32	512	15000
Arvore Binomial	Paralela CUDA	3,543	30	256	512	15000

Fonte: produzida pelo próprio Autor.

Tabela 7 - Modelo Árvore Binomial com 600 períodos.

Modelo	Plataforma	Tempo de Process. (s)	Uso CPU (%)	GRID SIZE	BLOCK SIZE	PERÍODOS
Arvore Binomial	Sequencial	0,992	35	-	-	600
Arvore Binomial	Paralela TPL	0,424	100	-	-	600
Arvore Binomial	Paralela CUDA	1,987	30	1	256	600
Arvore Binomial	Paralela CUDA	1,981	30	8	256	600
Arvore Binomial	Paralela CUDA	1,992	30	16	256	600
Arvore Binomial	Paralela CUDA	2,002	30	32	256	600
Arvore Binomial	Paralela CUDA	1,993	30	256	256	600
Arvore Binomial	Paralela CUDA	1,999	30	1	512	600
Arvore Binomial	Paralela CUDA	1,976	30	8	512	600
Arvore Binomial	Paralela CUDA	1,979	30	16	512	600
Arvore Binomial	Paralela CUDA	2,020	30	32	512	600
Arvore Binomial	Paralela CUDA	2,034	30	256	512	600

Fonte: produzida pelo próprio Autor.

Figura 17 – Comparação do tempo de processamento entre plataformas paralelas e quantidade de períodos para o modelo Árvore Binomial.

Fonte: produzida pelo próprio Autor.

Nota-se que a plataforma CUDA adquire uma maior homogeneidade com a escala de processamento de modo a partir de um tempo de processamento de aproximadamente 2 segundos de média com 600 períodos para aproximadamente 3.4 segundos de média com 15000 períodos.

A média da CPU total no processamento sequencial foi de 35% conforme podemos visualizar na Figura 18.

Figura 18 - Árvore Binomial - uso CPU no processo sequencial.

Fonte: Monitor de recursos Windows.

A média da CPU total no processamento paralelo *TPL* foi de 97% conforme podemos visualizar na Figura 19.

Figura 19 - Árvore Binomial - uso CPU no processo paralelo TPL.

Fonte: Monitor de recursos Windows.

6.3. Simulação do Método dos Mínimos Quadrados de Monte Carlo

O método de simulação dos mínimos quadrados de Monte Carlo nos evidenciou o grande destaque de performance do processamento em GPU dado a biblioteca de geração de números pseudoaleatórios e quasi-aleatórios cuRAND e os múltiplos *cores* do *hardware*. Neste modelo é fundamental definir os números de passos e trajetórias que aproxime para a média do preço para o preço ótimo da opção.

6.3.1. Erro Quadrático Médio

Neste caso houve diferenças da análise do erro quadrático médio por plataforma utilizada devido à simulação do comportamento do ativo objeto durante os passos em cada trajetória. Podemos verificar esta diferença a partir da Tabela 8 abaixo.

Tabela 8 - Modelo LSM – EQM.

Modelo	Plataforma	EQM
LSM	Sequencial	0,0002
LSM	Paralela TPL	0,0002
LSM	Paralela CUDA	0,0001

Fonte: produzida pelo próprio Autor.

6.3.2. Tempo de Processamento e Uso de CPU

Novamente podemos verificar a partir da Tabela 9 e a Figura 20 que a plataforma CUDA para simulações de Monte Carlo são muito mais performáticas em relação às demais.

Tabela 9 - Modelo LSM – Tempo de processamento e uso de CPU.

Modelo	Plataforma	Tempo de Process. (s)	Uso CPU (%)	GRID SIZE	BLOCK SIZE	Trajetórias	Passos
LSM	Sequencial	682,433	25	-	-	100	20
LSM	Sequencial	1856,859	25	-	-	100	40
LSM	Sequencial	2421,544	25	-	-	100	60
LSM	Sequencial	11187,345	25	-	-	1000	20
LSM	Sequencial	20454,432	25	-	-	1000	40
LSM	Paralela TPL	201,702	100	-	-	100	20
LSM	Paralela TPL	298,411	100	-	-	100	40
LSM	Paralela TPL	387,124	100	-	-	100	60
LSM	Paralela TPL	876,645	100	-	-	1000	20
LSM	Paralela TPL	1996,122	100	-	-	1000	40
LSM	Paralela CUDA	2,291	30	1	256	1000	20
LSM	Paralela CUDA	2,367	30	1	256	1000	40
LSM	Paralela CUDA	2,481	30	4	256	1000	20
LSM	Paralela CUDA	2,323	30	8	256	1000	40
LSM	Paralela CUDA	2,413	30	4	256	1000	60

Fonte: produzida pelo próprio Autor.

Figura 20 - Comparação do tempo de processamento entre plataformas paralelas, trajetórias e número de passos para o modelo LSM.

Fonte: produzida pelo próprio Autor.

A média da CPU total no processamento sequencial foi de 37% conforme podemos visualizar na Figura 21.

Figura 21 - Modelo LSM - uso CPU processo sequencial.

Fonte: Monitor de recursos Windows.

A média da CPU total no processamento sequencial foi de 98% conforme podemos visualizar na Figura 22.

Figura 22 - Modelo LSM – uso CPU processo paralelo TPL.

Fonte: Monitor de recursos Windows.

6.4. Convergência do Modelo Árvore Binomial com Extrapolação de Richardson

Neste tópico vamos abordar a melhoria na convergência oscilatória do modelo de árvore binomial com a Extrapolação de Richardson.

Assumindo uma opção de venda VALE5 que não paga dividendos do portfólio gerado neste trabalho com volatilidade implícita: 45.7%, taxa de juros: 9.6%, *spot*: 27.92, *strike*: 24 e prazo para o vencimento: 0.0992, podemos demonstrar o aumento do desempenho da convergência com a Tabela 10.

Tabela 10 - Árvore Binomial contra Árvore Binomial com Extrapolação de Richardson - Prêmio de Mercado da opção.

Modelo	Períodos	Prêmio
Árvore Binomial	10	0,26252
Árvore Binomial	20	0,25326
Árvore Binomial	30	0,24792
Árvore Binomial	40	0,25102
Árvore Binomial	50	0,25130
Árvore Binomial	60	0,24650
Árvore Binomial	70	0,25095
Árvore Binomial	80	0,25067
Árvore Binomial	90	0,24728
Árvore Binomial	100	0,24958
Árvore Binomial	200	0,24999
Árvore Binomial	500	0,24951
Árvore Binomial	1000	0,24949
Árvore Binomial	2000	0,24943
Árvore Binomial	5000	0,24933
Árvore Binomial	10000	0,24937
Árvore Binomial	15000	0,24936
Árvore Binomial com Extr. De Richardson	10	0,24987
Árvore Binomial com Extr. De Richardson	20	0,25004
Árvore Binomial com Extr. De Richardson	30	0,24940
Árvore Binomial com Extr. De Richardson	40	0,24950
Árvore Binomial com Extr. De Richardson	50	0,24936
Árvore Binomial com Extr. De Richardson	60	0,24935
Árvore Binomial com Extr. De Richardson	70	0,24936
Árvore Binomial com Extr. De Richardson	80	0,24936
Árvore Binomial com Extr. De Richardson	90	0,24936
Árvore Binomial com Extr. De Richardson	100	0,24936
Árvore Binomial com Extr. De Richardson	200	0,24936
Árvore Binomial com Extr. De Richardson	500	0,24936
Árvore Binomial com Extr. De Richardson	1000	0,24936
Árvore Binomial com Extr. De Richardson	2000	0,24936
Árvore Binomial com Extr. De Richardson	5000	0,24936
Árvore Binomial com Extr. De Richardson	10000	0,24936
Árvore Binomial com Extr. De Richardson	15000	0,24936

Fonte: produzida pelo próprio Autor.

Fica evidente a diferença de taxa de convergência relacionada aos modelos supracitados quando analisamos a Figura 23.

Figura 23 – Comparação da quantidade de períodos entre o modelo Árvore Binomial e Árvore Binomial com Extrapolação de Richardson para convergência do prêmio.

Fonte: produzida pelo próprio Autor.

7. Conclusão

O trabalho proposto teve como objetivo realizar a comparação dos modelos de precificação de opções americanas a partir de múltiplas plataformas paralelas *TPL* e *CUDA* para se adequar à necessidade em termos de tempo de processamento, robustez e complexidade de cada modelo. Houve também a análise da convergência no modelo de Árvore Binomial com o uso da extrapolação de Richardson.

O modelo de precificação de Opções Americanas que teve o melhor desempenho foi a aproximação analítica de B&S com menos de um segundo para execução de todo o portfólio.

O modelo mais robusto e mais complexo foi o modelo dos mínimos quadrados de Monte Carlo (*LSM*). Foi mais robusto pois apresentou um erro quadrático médio menor do que os demais comparando com o preço de tela da *Bloomberg*, e mais complexo pelas quantidades de trajetórias e números de passos necessários para se obter um resultado eficaz.

O algoritmo de otimização da convergência de extrapolação de Richardson para o modelo de Árvore Binomial trouxe ganhos efetivos em tempo de processamento e facilidade de implementação, pois obteve-se o mesmo resultado que o modelo padrão no exemplo dado com aproximadamente 17 vezes menos períodos.

Em relação às plataformas, podemos destacar a facilidade do uso da plataforma *TPL* na IDE Visual Studio e de sua eficiência gerada pelo paralelismo da CPU em suas execuções, porém a performance da tecnologia *CUDA* foi superior, principalmente nas simulações que necessitam processar um grande volume de cálculos, ao qual oferece ganhos expressivos em determinadas configurações chegando à ordem de 10000 vezes. Essa observação pode ser vista no uso do modelo dos mínimos quadrados de Monte Carlo (*LSM*) devido ao enorme desempenho da biblioteca *cuRAND* de geração de números aleatórios e ao paralelismo nativo de múltiplos processadores da GPU.

Por outro lado, a plataforma *CUDA* possui alguns aspectos limitantes para os seus desenvolvedores, como a compatibilidade do *software* *CUDA* sendo restrito às placas gráficas *NVIDIA*. Outro ponto negativo observado na fase de desenvolvimentos da aplicação foi a necessidade de reiniciar a máquina toda vez que ocorria um erro na execução do

pacote. A impressão que se teve é a não correção na hora da compilação de alguns componentes da placa GPU ocasionado falhas.

Existem diversos caminhos para a continuidade do trabalho proposto, como: construir a aplicação CUDA em *Cloud Computing* de modo a aumentar a capacidade de processamento e escalabilidade da solução, além disso poder integrar com os *frameworks* de aprendizado de máquina, tecnologia recente, de grande futuro nas soluções. Do ponto de vista dos modelos de precificação, seria avaliar distintas combinações de indexadores para as carteiras de opções *call* e *put* de modo a “estressar” os modelos.

Referência Bibliográfica

LONGSTAFF. F.; SCHWARTZ. E. **Valuing American Options by Simulation: A Simple Least-Squares Approach**. The Review of Financial Studies, Vol. 14, No. 1. (Spring, 2001), pp. 113-147.

BAIDYA. T.; CASTRO. A. **Convergência dos Modelos de Árvores Binomiais para Avaliação de Opções**. Vol. 21, Nº 1, p. 17-30, junho de 2001

BAIDYA.T.; ARAUJO.R. **Avaliação de opções reais através do método dos mínimos quadrados de monte Carlo**. XXIV Encontro Nac.de Eng. de Produção - Florianópolis, SC, Brasil, 03 a 05 de novembro de 2004.

CAMPBELL.C.; JOHNSON.R.; MILLER.A.; TOUB.S., **Design Patters for Decomposition and Coordinator on Multicore Architectures**. Parallel Programming with Microsoft® .NET (2010).

BLACK.F.; SCHOLES.M. **The Pricing of Options and Corporate Liabilities**. Journal of Political Economy 1973 81 (3): 637-654 .

ITO.K. **Stochastic integral**. Proc. Imp. Acad. Tokyo 20, 519–524 (1944).

BRANKA.M; ZDRAVKA.A; TEA.P, **Numerical Methods versus Bjerk Sund and Stensland Approximations for American Options Pricing**. World Academy of Science, Engineering and Technology International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering Vol. 8, Nº 4, 2014.

BJERKSUND.P; STENSLAND.G, **Closed Form Valuation of American Options**. Department of Finance and Management Science, NHH, Helleveien 30, N-5045 Bergen, Norway (2002).

HULL,J.C. **Options, Futures and other derivatives**. 9 Edition – (2016)

BROADIE.M.; DETEMPLE.J. **American Option Valuation: New Bounds, Approximations, and a Comparison of Existing Methods** – Review of Financial Studies, Volume 9 – (Winter,1996), 1211-1250.

BLOOMBERG, Disponível em: <https://www.bloomberg.com.br/> Acesso em :17/07/2017

MICROSOFT, **Parallel Programming in the .NET Framework**. (2017)

ANGERER, C. **GPU Programming Introduction**. NVIDIA Corporation 2015.

NVIDIA KEPLER, **NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK 110/210** (2012), Disponível em: <http://international.download.nvidia.com/pdf/kepler/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf> Acesso em: 23/07/2017

NVIDIA FERMI, **NVIDIA's Next Generation CUDA Compute Architecture: Fermi** (2010).

Disponível em:

https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf Acesso em: 28/07/2017

NVIDIA MAXWELL, (2014) Disponível em: <https://developer.nvidia.com/maxwell-compute-architecture> Acesso em: 28/07/2017

NVIDIA PASCAL, (2016) Disponível em: <http://www.nvidia.com/object/gpu-architecture.html> Acesso em: 28/07/2017

NVIDIA, Disponível em: <http://www.nvidia.com> Acesso em: 23/07/2017

NVIDIA cuRAND, **CUDA Toolkit 4.1 CURAND Guide** - January 2012. Disponível em: https://hpc.oit.uci.edu/nvidia-doc/sdk-cuda-doc/CUDALibraries/doc/CURAND_Library.pdf

Acesso em 23/07/2017

NVIDIA NSIGHT, Disponível em: <http://www.nvidia.com.br/object/parallel-nsight-br.html> Acesso em: 24/07/2017

Glossário

DATA FIXING Data de apuração dos preços e parâmetros que irão definir o valor a ser liquidado na Data de Vencimento.

STRIKE Preço do exercício vinculado ao contrato da opção.

PAYOFF Valor a debitar ou creditar de um contrato no seu vencimento.

IN THE MONEY Termo utilizado para expressar um contrato de opção dando lucro, “no dinheiro”.

ATIVO SUBJACENTE Ativo referente ao papel ao qual a opção esteja indexada.

SPOT Cotação de mercado do Ativo Subjacente

THREAD Processamento de uma atividade do Sistema Operacional.

MULTITHREAD Mais de uma ação por vez com mais de um *thread*.

DRIFT Variável relacionado ao comportamento da tendência do ativo subjacente.

MARTINGALE Processo estocástico de *drift* zero. Uma de suas propriedades é de seu valor esperado em qualquer data futura é igual ao seu valor de hoje.

KERNEL Nome dado para uma função que é executada na GPU.

WARPS Número de *threads* que possam rodar em paralelo em uma máquina.

CLOUD COMPUTING Refere-se à utilização da memória e da capacidade de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio de uma rede de dados disponível (internet), seguindo o princípio da computação em *grid*.

VOLATILIDADE Refere-se à taxa de variação do ativo subjacente. Existem diversos métodos para calcular a volatilidade, sendo os mais conhecidos como simplesmente o desvio padrão no período (volatilidade histórica), ou com diferentes pesos entre as amostras mais recentes e passadas (EWMA).

VOLATILIDADE IMPLÍCITA Nome dado ao valor real da volatilidade que justifica o prêmio de mercado de uma determinada opção.