# Dokumentacja

# Wycena opcji amerykańskich na rynku *equity* metodą Longstaffa-Schwartza

Dorota Toczydłowska
Piotr Kosewski
Katarzyna Mioduszewska
Piotr Garbuliński
Bartosz Głowinkowski

# Spis treści

# Opis zadania

Celem projektu jest stworzenie oprogramowania służacego wycenie opcji amerykańskiej przy pomocy symulacji Monte Carlo. Wykorzystujemy przybliżenie opcją bermudzką oraz metodę Longstaffa - Schwartza opisana w [1]. Metoda pozwala na stworzenie efektywnej implementacji wyceny opcji przy ograniczonej złożoności pamięciowej, co umożliwia przeprowadzenie symulacji z dużą liczbą ścieżek i punktów czasowych.

Wycena instrumentu składa się z dwóch etapów : sprowadzenie przypadku do zagadnienia jednowymiarowego oraz estymacji metodą najmniejszych kwadratów parametrów z kroku pierwszego.

# 1 Wstęp teoretyczny

Niech $(\Omega,\ \mathcal{A},\ \mathbf{P})$ będzie przestrzenią probabilistyczną z filtracją $(\mathcal{F})_{t\in[0,T]}$

Niech $(X_t,\ \mathcal{F}_t)$ będzie łańcuchem Markowa o przestrzeni stanów $(E, \epsilon)$. Załóżmy, że $X_0$ jest deterministyczne. Jest to proces odzwierciedlający wszystkie informacje o instrumencie podstawowym.

Wycena ciągłej opcji amerykańskiej może być sformułowana przez zdefiniowanie procesu $U_t$ dla $t \in [0, T]$, będącego procesem całkowalnym z kwadratem, adaptowanym do filtracji $(\mathcal{F})_{t\in[0,T]}$, który określa zdyskontowaną wypłatę z opcji z momentu $t$ postaci $U(t) = h(X_t)$, gdzie $h$ jest funkcją borelowską.

Zdefiniujmy dodatkowo klasę momentów stopu $\tau \in \mathcal{T}_{t,T}$ o wartościach z przedziału $[0, T]$. Niech $T$ będzie momentem wygaśnięcia opcji. Przez wartość opcji w chwili $0$ oznaczamy

$$V_0 = sup_{\tau\in\mathcal{T}_{0,T}}\ \mathbb{E}U_\tau$$

Przez wartość opcji w chwili $t$ oznaczamy

$$V_t = sup_{\tau\in\mathcal{T}_{t,T}}\ \mathbb{E}U_\tau$$

Podstawą metody Longstaffa - Schwartza jest zasada programowania dynamicznego

$$V_T = U_T$$

$$V_t = max(U_t,\ \mathbb{E}(V_{t+1}|\mathcal{F}_t)) \tag{1}$$

W języku momentów stopu $\tau_t = \{k \geq t|U_k = V_k\}$ zasadę tę możemy zapisać jako

$$\begin{aligned} &\tau_T = T, \\ &\tau_t = t\mathbf{1}_{U_t\geq\mathbb{E}(U_{\tau_{t+1}}|\mathcal{F}_t)} + \tau_{t+1}\mathbf{1}_{U_t<\mathbb{E}(U_{\tau_{t+1}}|\mathcal{F}_t)} \text{ dla } t \in [0, T-1] \end{aligned} \tag{2}$$

Między innymi z własności procesu Markowa procesu $X_t$ mamy równość (dowód [3])

$$\mathbb{E}(U_{\tau_{t+1}}|\mathcal{F}_t) = \mathbb{E}(U_{\tau_{t+1}}|X_t)$$

Wówczas

$$\begin{aligned}&\tau_T = T,\\&\tau_t = t\mathbf{1}_{U_t \geq \mathbb{E}(U_{\tau_{t+1}}|X_t)} + \tau_{t+1}\mathbf{1}_{U_t < \mathbb{E}(U_{\tau_{t+1}}|X_t)} \text{ dla } t \in [0, T-1]\end{aligned} \tag{3}$$

W praktyce używa się ścieżek *in - the- money*, co ma swoje uzasadnienie w większej efektywności obliczeniowej algorytmu. W takim wypadku, zdefiniowane wyżej momenty stopu mają postać

$$\begin{aligned}&\bar{\tau}_T = T,\\&\bar{\tau}_t = t\mathbf{1}_{\{U_t \geq \mathbb{E}(U_{\tau_{t+1}}|X_t)\}\cap\{U_t>0\}} + \bar{\tau}_{t+1}\mathbf{1}_{\{U_t < \mathbb{E}(U_{\tau_{t+1}}|X_t)\}\cup\{U_t>0\}} \text{ dla } t \in [0, T-1].\end{aligned} \tag{4}$$

## 1.1 Opis algorytmu

1. Krok 1.

   Pierwszym krokiem konstrukcji ceny opcji metody Longstaffa - Schwartza jest rzut ortogonalny procesu $X_t$ na przestrzeń funkcji mierzalnych od $X_t$, gdzie $e_k : E \to R$ jest ciągiem tych funkcji spełniających warunki dla $t \in \{0,\ 1, ...\ T-1\}$

   - $e_k(X_t)$ jest zupełny w $L^2(\sigma(X_t))$,
   - dla $m \geq 1$ $\sum_{k=1}^{m} \beta_{t,k} e_k(X_t) = 0$ p.n. $\Rightarrow \beta_{t,k} = 0$ dla $k \in \{1, ..., m\}$ .

   Główną ideą metody Longstaffa - Schwartza jest aproksymowanie wartości oczekiwanej z (1) używając ciągu $m+1$ funkcji $e_k$ w każdym z kroków $t$

   $$\mathbb{E}(V_{t+1}|\mathcal{F}_t) = \mathbb{E}(V_{t+1}|X_t) = \sum_{k=0}^{m} \beta_{t,k} e_k(X_t) \tag{5}$$

   Postulowana w pracy Longstaffa i Schwartza rodziną funkcji spełniającą powyższe warunki, są ważone wielomiany Laguerre'a. Ich postać do

   $$\begin{aligned}&e_0(x) = \exp^{-\frac{x}{2}}\\&e_1(x) = \exp^{-\frac{x}{2}}(1-x)\\&...\\&e_k = \exp^{-\frac{x}{2}} \frac{e^x}{k!}\frac{d^k}{dx^k}(x^k e^{-x})\end{aligned} \tag{6}$$

   W praktyce, używa się nieważonych wielomianów Laguerre'a w związku z szybszą zbieżnością ceny uwarunkowaną stopniem wielomianów, który jest ustalony. Opis zbieżności algorytmu w [3].

2. Krok 2.

   Krokiem drugim metody Logstaffa - Schwartza jest wyznaczenie parametrów $\beta_{t,k}$ (5) poprzez minimalizowanie wartości oczekiwanej kwadratu różnicy

   $$\mathbb{E}\left(\mathbb{E}(V_{t+1}|X_t) - \sum_{k=0}^{m} \beta_{t,k} e_k(X_t)\right)^2 \tag{7}$$

Po zminimalizowaniu otrzymywane są następujące macierze, niezbędne do algorytmu

$$\begin{aligned}(\mathbf{M}_{ee})_{k,l} &= \mathbb{E}(e_k(X_t)e_l(X_t)),\\ (\mathbf{M}_{Ve})_k &= \mathbb{E}(\mathbb{E}(V_{t+1}|X_t)e_k(X_t))\end{aligned} \tag{8}$$

W związku z mierzalnością $e_k(X_t)$ względem $X_t$ możemy zapisać

$$(\mathbf{M}_{Ve})_k = \mathbb{E}(\mathbb{E}(V_{t+1}e_k(X_t)|X_t))$$

Wektor $\beta_t$ równy $[\beta_{t,0},\ \beta_{t,1}, ...,\ \beta_{t,m}]$ jest postaci

$$\beta_t = \mathbf{M}_{ee}^{-1}\mathbf{M}_{Ve} \tag{9}$$

W algorytmie będziemy generować $N$ ścieżek aktywa bazowego. Wówczas estymatorami wyżej zdefiniowanych macierzy będą

$$\begin{aligned}(\hat{\mathbf{M}}_{ee})_{k,l} &= \frac{1}{N}\sum_{n=0}^{N} e_k(X_t^n)e_l(X_t^n),\\ (\hat{\mathbf{M}}_{Ve})_k &= \frac{1}{N}\sum_{n=0}^{N} V_{t+1}^n e_k(X_t^n)),\end{aligned} \tag{10}$$

a estymatorem wektora $\beta_t$ w kroku $t$ jest

$$\hat{\beta}_t = \hat{\mathbf{M}}_{ee}^{-1}\hat{\mathbf{M}}_{Ve}$$

Wówczas algorytm w pseudokodzie wyglada następująco

**Algorithm 1** Longstaff - Schwartz Algorithm

**Ensure:** $N,\ T,\ X_0,\ m$ is set
  generate N paths of T timesteps of underlying asset
  set $V_T := h(X_T)$
  **for** $t = T-1$ to 1 **do**
    $(\hat{\mathbf{M}}_{ee})_{k,l} := \frac{1}{N}\sum_{n=0}^{N} e_k(X_t^n)e_l(X_t^n)$
    $(\hat{\mathbf{M}}_{Ve})_k := \frac{1}{N}\sum_{n=0}^{N} V_{t+1}^n e_k(X_t^n)$
    $\hat{\beta}_t := \hat{\mathbf{M}}_{ee}^{-1}\hat{\mathbf{M}}_{Ve}$
    $C_t := \sum_{k=0}^{m} \beta_{t,k}e_k(X_t)$
    **if** $C_t > h(X_t)$ **then**
      set $V_t = e^{-r}C_t$
    **else**
      set $V_t = h(X_t)$
    **end if**
  **end for**

W przypadku obliczeń wykorzystujących liczby prawdziwie losowe, algorytm Longstaffa-Schwartza (jako opierający się o ruch wstecz w czasie) wymaga przechowywania pełnych ścieżek przeprowadzonej symulacji. Prowadzi to do oczywistego problemu z ilością wymaganej pamięci: przeprowadzenie symulacji wykorzystującej np. $10^6$ ścieżek

o $10^3$ punktów czasowych byłoby niemożliwe na przeciętnym komputerze osobistym (ponad dekadę od publikacji algorytmu).

W symulacjach komputerowych zazwyczaj wykorzystywany jest generator liczb pseudolosowych. Podstawową cechą takiego generatora jest możliwość powtórnego wygenerowania sekwencji liczb losowych, jeśli zainicjujemy go takim samym stanem początkowym. Można zatem odzyskać każdy wcześniejszy stan instrumentu bazowego przechowując jedynie pewną stałą liczbę danych (stan początkowy - *seed*).

Oczywiście przeprowadzanie pełnej symulacji w każdym kroku czasowym ($T$ razy) nie jest akceptowalne z punktu widzenia złożoności obliczeniowej. Prostym rozwiązaniem tego problemu jest zapisanie nie tylko jednego stanu początkowego, ale całego wektora stanów występujących w trakcie symulacji.

Implementacja została oparta o propozycję opisaną w [5]. Autorzy sugerują następujący schemat symulacji:

1. Krok 1.
   Stwórz wektor $U$ przechowujący stan losowanego procesu stochastycznego (sumę kolejnych liczb losowych w każdej ze ścieżek).
   Stwórz wektor $S$ przechowujący stany generatora. Umieść w nim początkowy *seed*.

2. Krok 2.
   Wylosuj liczby dla wszystkich ścieżek symulacji (cały krok czasowy) i dodaj je odpowiednio do $U$.
   Jeśli osiągnąłeś koniec symulacji (chwilę wygaśnięcia opcji) przejdź do kroku 4.

3. Krok 3.
   Dopisz aktualny stan generatora do $S$.
   Przejdź do kroku 2.

4. Krok 4.
   Zakończ generowanie liczb losowych.

Po zakończeniu losowania otrzymujemy wektor $U$ potrzebny do obliczenia ceny instrumentu bazowego w chwili wygaśnięcia opcji. Wykorzystując zapisane w $S$ stany generatora liczb pseudolosowych, możemy odtworzyć dowolną wartość w symulacji. W całym programie każde z losowań przeprowadzamy tylko 2-krotnie (optymalnie z punktu widzenia złożoności obliczeniowej [1] ).
W efekcie unikamy wyjściowej złożoności pamięciowej $O(TN)$ i otrzymujemy algorytm klasy $O(T + N)$ (przy czym zazwyczaj $T \ll N$).

## 1.2 Wyznaczenie parametrów greckich

Poza ceną opcji, zwracane są także wartości parametrów greckich, które obliczamy wraz z ceną opcji.

[1] Rzeczywistą szybkość algorytmu można poprawić przy wykorzystaniu obliczeń równoległych. Jeśli podczas pierwszego losowania $N$ ścieżek w pewnym kroku czasowym $t$ zapiszemy więcej stanów generatora, w drugim losowaniu będziemy mogli skorzystać z kilku procesorów.

Korzystamy przy tym z wniosków pochodzących z [2]. Dla ceny aktywa bazowego w chwili $t$ danego funkcją $S_t$, odpowiedniej ceny opcji $V_t$, funkcji wypłaty $f$ oraz pewnego parametru $\theta$ mamy:

$$\frac{\partial V_0(S_0)}{\partial \theta} = \mathbb{E}\left(\exp(-r\tau)\frac{\partial f(S_\tau(\theta))}{\partial S_\tau}\frac{\partial S_\tau(\theta)}{\partial \theta}\right), \tag{11}$$

gdzie $\tau$ nazywamy czasem zatrzymania (długością czasu do momentu realizacji opcji). Możliwe jest zatem obliczenie parametru greckiego poprzez uśrednienie pewnej funkcji (iloczynu 2 pochodnych cząstkowych) po realizacjach na ścieżkach symulacji Monte Carlo. Wystarczy *wycenić* tę funkcję tak, jak wyceniamy wartości realizacji opcji w kolejnych ścieżkach symulacji. Druga pochodna w tym iloczynie nie zależy od funkcji wypłaty i może być obliczona analitycznie:

$$\frac{\partial S_\tau(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}\left(S_0 \exp\left(r - \frac{1}{2}\sigma^2\right)\tau + \sigma W_t\right). \tag{12}$$

Z kolei pierwsza pochodna jest zależna nie od tego, który parametr grecki obliczamy, a jedynie od funkcji wypłaty. Może być zatem zaimplementowana niezależnie od algorytmu przeprowadzającego symulację.
Ponadto, metoda jest właściwa także dla parametrów greckich będących pochodnymi ceny opcji rzędu wyższego niż 1, ale wymagałoby to obliczenia odpowiedniej pochodnej funkcji wypłaty. Jednak ze względu na zazwyczaj nieciągłą pochodną funkcji wypłaty po cenie aktywa, nie da się w ten sposób obliczyć np. parametru gamma.
W tym celu skorzystamy z drugiego pomysłu opisanego w [2] - metody współczynnika wiarygodności (ang. "likelihood radio"). W tej metodzie obliczenie parametru greckiego sprowadza się do wyceny opcji w pierwszym kroku czasowym oraz obliczenia ważonej średniej tych wycen. W ten sposób otrzymujemy:

$$\Delta(S_0) = \frac{\partial V_0}{\partial S_0} = \mathbb{E}\left(F(S_1)\frac{\partial log g_1}{\partial S_0}\right), \tag{13}$$

$$\Gamma(S_0) = \frac{\partial \Delta(S_0)}{\partial S_0} = \mathbb{E}\left(F(S_1)\left(\frac{\partial^2 log g_1}{\partial S_0^2} + \left(\frac{\partial log g_1}{\partial S_0}\right)^2\right)\right), \tag{14}$$

gdzie $g_1$ jest gęstością przejścia aktywa ze stanu $S_0$ do $S_1$, a $F(S_1) = \exp(-r\Delta t)V_1(S_1)$ jest zdyskontowaną na chwilę 0 wartością opcji w chwili 1. Ponadto, gęstość przejścia $g_1$ zależy (poza cenami aktywa w chwilach 0 i 1) wyłącznie od parametrów rynku i symulacji: $r$,$\sigma$,$\Delta t$.

## 1.3 Typy opcji

Zaimplementowane typy opcji są wymienione poniżej. Dokładny opis funkcji wypłat poniższych opcji jest opisany w dokumentacji [4].

1. Opcja waniliowa
2. Strategia *cash-or-nothing*

3. Strategia *asset-or-nothing*
4. Strategia *Straddel*
5. Strategia *Risk Reversall*
6. Strategia *Butterfly*
7. Strategia *Strangle*

W przypadku opcji europejskich można dodatkowo wycenić opcje z barierą europejską (dla opcji amerykańskich wycena opcji z barierą europejską nie ma sensu).

1. Opcja z barierą europejską *down-and-out*
2. Opcja z barierą europejską *up-and-in*
3. Opcja z barierą europejską *up-and-out*
4. Opcja z barierą europejską *down-and-in*

# 2 Wynik projektu

Wynikiem tej części projektu jest stworzenie silnika do symulacji Monte Carlo według modelu Longstaffa - Schwarza, mającego na celu wycenę opcji europejskich i amerykańskich opisanych w Rozdziale 1. Wycena ta ma używać danych rynkowych takich jak czynniki dyskontowe, kalendarz różnych giełd i lat oraz status dnia - roboczy bądź nie.

# 3 Dokumentacja

W tej części przedstawimy opisy zaimplementowanych funkcji wyliczających ceny poszczególnych opcji amerykańskich i europejskich w modelu Longstaffa-Schwartza. Części dokumentacji podzielone są na rozdziały : *Inicjalizacja*, gdzie opiszemy parametry wchodzące (inputy), ustalane przez użytkownika i administratora, *Plik pomocniczy* `calendarBasedFunctions.m`, gdzie zostają wyliczane parametry oparte na funkcjach kalendarzowych, *Program główny plik* `pricingFunctions.m` będący opisem wywoływanych funkcji głównych i pomocniczych wyliczających wspomniane ceny opcji: `Payoff` funkcja określająca funkcje wypłaty z opcji, `LSMmodel` i jej funkcje pomocnicze, oraz `priceCalculate` będąca funkcją główną obliczająca ceny opcji amerykańskich i europejskich przez wywołanie potrzebnych funkcji.

## 3.1 Inicjalizacja

Przed rozpoczęciem wczytywania funkcji podanych w poniższych programach, należy wczytać pliki `funkcje_kalendarzowe.m` oraz `discount_functions.m`. Następnie, jest wskazane sprawdzenie, czy wszystkie zmienne określona globalnie i wskazane w poniższym podrozdziale zostały zdefiniowane.

### 3.1.1 Opis danych wejściowych określonych *globalnie*

- *start_date* - data otrzymania danych rynkowych, *string* o formacie 'dd-mmm-yyy'.
- *expire_date* - data wygaśnięcia kontraktu opcyjnego, *string* o formacie 'dd-mmm-yyy'.
- *issue_date* - data podpisania kontraktu opcyjnego, *string* o formacie 'dd-mmm-yyy', nie może wypadać w dzień roboczy.
- $S_0$ - wartość początkowa aktywa bazowego na *issue_date*, numeric.
- K - ceny wykoniania - w zależności od typu opcji może to być skalar (opcja cechująca się tylko jedną ceną wykonania) albo może to być wektor w przypadku np. opcji z dwiema barierami, numeric.
- B - dodatkowy parametr, numeric, będący wykorzystywany w funkcjach barierowych, numeric.
- $d$ - intensywność wypłaty dywidendy, numeric.
- *Mt* - ilość kroków gridu czasu w ciągu jednego dnia życia opcji, bardziej szczegółowo opisany w podrodziale *Funkcje Payoff*, integer.
- *Mx* - ilość ścieżek aktywa bazowego w symulacji Monte Carlo, integer.
- *order* - stopień wielomianu Laguerre, integer.
- *type* - '1' - call lub sprzedaż długa (long), '$-1$' - put lub sprzedaż krótka (short).

- *name* - string, nazwa funkcji wypłaty danej opcji. Wyróżniamy 'PayoffCallPut', 'PayoffStraddel', 'PayoffCashOrNot', 'PayoffAssetOrNot', 'PayoffRiskReversall', 'PayoffButterfly','PayoffStrangle' i inne funkcje wypłat bardziej szczegółowo opisane w podrodziale *Funkcje postaci* `PayoffOptionName`.
- *vol_DDC* - string, typ konwencji na jakiej pracujemy, czyli np. "ACT/360", "ACT/365", "30/360", "30E/360".
- *FC_DOM* - string, nazwa giełdy na której wyceniamy opcję.
- *CURR_DOM* - string, nazwa waluty na której wyceniamy opcję.
- *FC_F* - string, nazwa giełdy na której wyceniany jest instrument bazowy.
- *CURR_F* - string, nazwa waluty na której wyceniany instrument bazowy.
- *Depo_BDA* - string, identyfikator płatności, np. *sfbd* - Standard Following Business Day, *mfbd* - Modified Following Business Day.
- *DF_type* - string, rodzaj czynnika dyskontowego, czyli : 'Ave', 'Ask', 'Bid'.
- *PPO* - Premium Payment Offset; liczba dni od *start_date* (to znaczy dnia zawarcia kontraktu opcyjnego) do *issue_date* (transaction date, czyli daty przepływu pieniędzy).
- *OSO* - Option Settlement Offset; liczba dni od *expire_date* do settlement date (czyli daty realizacji świadczenia z opcji).
- *interp_method* - string, metoda interpolacji czynników dyskontowych, np. 'linear on df'.
- *checkWorkDay* - '1' - realizujemy opcję tylko w dni robocze, '0' realizujemy opcje również w dni nierobocze.

### 3.1.2 Opis danych wejściowych będących *outputami* min. funkcji pomocniczych `calendarBasedFunctions.m`

- *daysNumber* - integer, liczba pełnych dni do wygaśnięcia opcji,
- *workingGrid* - wektor wartości binarnych dla każdego gridu czasowego ( długość wektora *daysNumber* razy *Mt* ) : '1' - dzień roboczy, '0' - dzień wolny,
- *datesVector* - wektor stringów, dat od *issue_date* do *expire_date*, format 'dd-mmm-yyyy',
- *grid* - integer, długość wektora *daysNumber* razy *Mt*,
- *properGrid* - integer, długość wektora *daysNumber* razy *Mt* - Mt + 1, czyli ilość symulacji z włącznie z jedną symulacją w *expire_date*,
- *dt* - delta czasu między jednym gridem a drugim, jako ułamek roku,

- *tgrid* - wektor delt czasu narastająco,
- *discountCurve* - wektor czynników dyskontowych dla narastających delt czasu,
- *discountDelta* - wektor czynników dyskontowych dla delt czasu,
- *PayoffFunction* - funkcja wypłaty określona przy wyborze przez użytkownika rodzaju opcji (*name* globalnie)
- *inTheMoney* - indeks określający, który z wierszy danego wektora jest 'in the money'.

## 3.2 Plik pomocniczy `calendarBasedFunctions.m`

Funkcje znajdują się w pliku `calendarBasedFunctions.m` i przed włączeniem wymagają załadowania plików `discount_functions.m` oraz `funkcje_kalendarzowe.m`

### `datesCalculate`

1. **Opis**: Funkcja wylicza *daysNumber*, *workingGrid*, *datesVector*, *grid* na podstawie danych określonych globalnie.
2. **Input**: *vol_DCC*,*FM_DOM*, *Mt*, *start_date*, *issue_date*, *maturity_date* , *checkWorkDay,*
3. **Output**: *daysNumber*, *workingGrid*, *datesVector*, *grid*.

### `tgridCalculate`

1. **Opis**: Funkcja wylicza *dt* i *tgrid* na podstawie danych określonych globalnie i outputow funkcji `datesCalculate`.
2. **Input**: *vol_DCC*, *Mt*, *datesVector*, *grid*,
3. **Output**: *dt*, *tgrid*.

### `discountsCalculate`

1. **Opis**: Funkcja wylicza krzywą dyskontową dla wektora *tgrid* na podstawie wektora *datesVector* oraz dodatkowo wylicza krzywą delt dyskontowych dla uprzednio określonej krzywej.
2. **Input**: *DF_type*, *start_date*, *Mt*, *datesVector*, *grid*, *Mt*,
3. **Output**: *discountCurve* , *discountDelta*.

**`discountsCalculateF`**

1. **Opis**: Funkcja wylicza krzywą dyskontową dla wektora *tgrid* na podstawie wektora *datesVector* oraz dodatkowo wylicza krzywą delt dyskontowych dla uprzednio określonej krzywej, ta funkcja wkonuje to dla rynku instrumentu bazowego.

2. **Input**: *DF_type*, *start_date*, *Mt*, *datesVector*, *grid*, *Mt*,

3. **Output**: *discountCurve* , *discountDelta*.

## 3.3 Program główny plik `pricingFunctions.m`

Program główny wywołuje się za pomocą pliku `pricingFunctions.m`. W jego skład wchodzi szereg funkcji : główna funkcja wykonująca procedurę Longstaffa-Schwarza (`LSMmodel`), funkcja payoffu, czyli określająca rodzaj funkcji wypłaty z opcji (`pricingFunctions.m`) oraz szereg funkcji pomocniczych.

### 3.3.1 Funkcja wypłaty `Payoff`

**`Payoff`**

1. **Opis** : funkcja przyporządkowuje funkcję wypłaty do opcji wybranej przez użytkownika,

2. **Input** : *name*, *type*

3. **Output** : funkcja `payoff` wypłaty do opcji wybranej przez użytkownika

**Funkcje postaci `PayoffOptionName`**

**Inputami** wszystkich opcji jest wektor stanu ewolucji aktywa bazowego w czasie $S$, $K$ oraz *type*. Dodatkowo istnieje możliwość podania wartości argumentu *dS*, decydującego o tym, czy zwrócona zostanie funkcja wypłaty ($dS = 0$ - wartość domyślna), czy jej pochodna po *X* ($dS = 1$).

**Outputem** jest wektor *Payoff* o długości wektora *assetPrice*. .

- `PayoffCallPut` - funkcja wypłaty z opcji waniliowej, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffStraddel` - funkcja wypłaty ze strategii *Straddel*, *type*$= 1$ - long Straddel, *type*$= -1$ - short Straddel,
- `PayoffCashOrNot` - funkcja wypłaty ze strategii *cash-or-nothing*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffRiskReversall` - funkcja wypłaty ze strategii *Risk Reversall*,
- `PayoffAssetOrNot` - funkcja wypłaty ze strategii *asset-or-nothing*, *type*$= 1$ - call, *type*$= -1$ - put,

- `PayoffButterfly` - funkcja wypłaty ze strategii *Butterfly*
- `PayoffStrangle` - funkcja wypłaty ze strategii *Strangle*, *type*$= 1$ - long Strangle, *type*$= -1$ - short Strangle,
- `PayoffDownAndOutEur` - funkcja wypłaty z opcji z barierowej europejskiej *down-and-out*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffUpAndInEur` -funkcja wypłaty z opcji z barierowej europejskiej *up-and-in*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffUpAndOutEur` -funkcja wypłaty z opcji z barierowej europejskiej *up-and-out*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffDownAndInEur` - funkcja wypłaty z opcji z barierowej europejskiej *down-and-in*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffDownAndOutAm` -funkcja wypłaty z opcji z barierowej amerykańskiej *down-and-out*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffUpAndInAm` - funkcja wypłaty z opcji z barierowej amerykańskiej *up-and-in*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffUpAndOutAm` - funkcja wypłaty z opcji z barierowej amerykańskiej *up-and-out*, *type*$= 1$ - call, *type*$= -1$ - put,
- `PayoffDownAndInAm` - funkcja wypłaty z opcji z barierowej amerykańskiej *down-and-in*, *type*$= 1$ - call, *type*$= -1$ - put,

#### 3.3.2 `LSMmodel`

1. **Opis**: Funkcja wylicza cenę oraz wartości wybranych parametrów greckich dla opcji amerykańskiej i europejskiej wg modelu Longstaffa-Schwartza dla wybranego rodzaju opcji,
2. **Input**: $S0, K, r, d, sigma, T, Mt, Mx, order, PayoffFunction$
3. **Output** : *AmOptionPrice, EurOptionPrice* - ceny opcji, *amGreeksValue,eurGreeksValue* - wektory wartości parametrów greckich, *greeksNames* - wektor nazw zwracanych parametrów greckich.

#### 3.3.3 Funkcje pomocnicze

**`Vol`**

1. **Opis**: Funkcja zaprojektowana do obliczania *implied volatility* i w przyszłości *local volatility*.
2. **Input** : $S$ - wektor wartości aktywa podstawowego w czasie $t$, czas $t$ jako ułamek roku, $T$ czas wygaśnięcia opcji jako ułamek roku, $K$,
3. **Output** : $sigma$, czyli wartość volatility dla czasu $t$, będąca liczba rzeczywistą;

**`LaguerrePolyMatrix`**

1. **Opis** : Funkcja pomocnicza obliczająca macierz wielomianów Laguerre dla zadanego stopnia $order$.
2. **Input** : wektor $X$, *order*,
3. **Output** : wynikiem jest macierz *PolyMatrix*, gdzie w kolumnach znajdują się wartości dla każdego wielomianu Laguerre stopnia od $0$ do $order$ a argumentem każdego z nich jest wektor $X$;

**`generateAssetBase`**

1. **Opis** : Funkcja pomocnicza obliczająca ewolucję aktywa w czasie $T$ dla $Mx$ niezależnych ścieżek.
2. **Input** : $Mt$, $Mx$,
3. **Output** : wynikiem jest: macierz (traktowana jako wektor wektorów) *RNGSeeds*, w której zapisane są stany generatora pozwalające na szybkie ponowne wylosowanie odpowiedniego kroku czasowego (na wszystkich ścieżkach) oraz wektor $z$ zawierający końcowy stan symulacji na wszystkich ścieżkach. Procesem jest błądzenie losowe o średniej 0 i wariancji 1.

**`getAssetPrice`**

1. **Opis** : Funkcja pomocnicza obliczająca wartość aktywa bazowego przy zadanych parametrach aktywa oraz stanie procesu w momencie wyceny,
2. **Input** : $S0$, $r$, $d$, $\sigma$, $T$, $Mx$, $dt$ - krok czasowy symulacji, $w$ - stan procesu w kroku symulacji,
3. **Output** : wynikiem jest wektor ceny aktywa bazowego o wymiarze zgodnym z $w$.

**`backwardStep`**

1. **Opis** : Funkcja cofająca stan aktywa bazowego do poprzedniego kroku czasowego symulacji,
2. **Input** : $RNGSeed$ - stan generatora liczb losowych przed losowaniem docelowego kroku czasowego, $baseAssetPrice$ - wektor ceny aktywa w kroku wyjściowym, $r$, $d$, $\sigma$, $dt$ - ktok czasowy symulacji, $Mx$.
3. **Output** : wynikiem jest wektor ceny aktywa bazowego o wymiarze zgodnym z $baseAssetPrice$.

**`calculateExpectedValue`**

1. **Opis** : Funkcja pomocnicza tworząca wektor wartości oczekiwanej opcji.
2. **Input** : *inTheMoney*, *assetPrice*, *order*, *optionPrice*,
3. **Output** : wynikiem jest wektor *expectedValue* wyliczana wg algorytmu zaproponowanego przez Longstaffa- Schwarza opisanego w Rozdziale 1

### 3.3.4 Funkcja główna `priceCalculate`

1. **Opis** : Funkcja główna wywołująca potrzebne funkcje pomocnicze i funkcje wyceniające opcję wg modelu Longstaffa-Schwartza
2. **Input** : brak, wywołuje funkcje `LSMmodel`, `Payoff`, `discountsCalculate`, `tgridCalculate`, `datesCalculate`,
3. **Output** : *AmOptionPrice, EurOptionPrice* będące liczbami rzeczywistymi

# Literatura


[1] Francis A. Longstaff, Eduardo S. Schwartz *Valuing American Options by Simulation: A Simple Least-Squares Approach*. The Review of Financial Studies, 2001.

[2] Howard Thom, Wolfson College, *Longstaff Schwartz Pricing of Bermudan Options and their Greeks*. University of Oxford, Institute for Mathematical Finance, 2009.

[3] Anna Zaremba, *Metoda Longstaffa - Schwartza wyceny opcji amerykanskich*. Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki 2005.

[4] Bogusław Wróblewski, *Raport i Dokumentacja : Obliczanie cen i parametrów greckich opcji walutowych w modelu Blacka-Scholesa*. Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki 2011.

[5] Raymond H. Chan, Chi-Yan Wong, Kit-Ming Yeung *Pricing Multi-asset American-Style Options by Memory Reduction Monte Carlo Methods*. Applied Mathematics and Computation 2006.