Dokumentacja

Portal Mathfinance
dla rynku FX kalkulatora

Marcin Szumski

Spis treści

1 Struktura katalogów

Poniżej opiszę strukturę katalogów w części, która jest powiązana z moją częścią projektu.

1. Katalog `mf\`– pliki pythona
   (a) `adapters.py` – plik z funkcjami odpowiedzialnymi za konwersję danych pomiędzy różnymi formatami (np. konwersję formatu daty, rozwijanie skrótów wykorzystywanych w funkcjach octave)
   (b) `enums.py` – plik z listami stałych; są to przede wszystkim elementy list wyboru dostępnych w aplikacji
   (c) `forms.py` – plik z funkcjami tworzącymi elementy stron w html (zwracającymi odpowiedni napis w html). Przede wszystkim są to różnego typu formy do wprowadzania danych
   (d) `helpers.py` – funkcje pomocnicze
   (e) `octave.py` – plik z funkcjami odpowiedzialnymi za kontakt z kodem octave. Do zadań tych funkcji należy między innymi mapowanie zmiennych octave, kontrola plików oraz odpalanie kodu octave
   (f) `views_fixed.py` – funkcje dla rynku fixed
   (g) `views_fx.py` – funkcje dla rynku FX. Jest to główny plik mojej części projektu i jego zawartość zostanie opisana przeze mnie później
   (h) `views_registration.py` – funkcje związane z zarządzaniem użytkownikami (rejestracja, zmiana hasła, usuwanie nieaktywnych)
   (i) `views_simple.py` – funkcje dla kalkulatora simple
   (j) `views.py` – ogólne funkcje związane z wyświetlaniem (np. logowanie, ładowanie stron startowych)
2. katalog octavecodes\
   (a) katalog `fx\` - pliki octave'a do rynku FX
      i. katalog `code\` - pliki z właściwym kodem octave'a
         A. `calculate_finish.m` – zapisuje wynik obliczeń parametrów opcji do pliku pośredniczącego (z którego zostaną one później pobrane przez kod pythona)
         B. `calculate_functions.m` – różne funkcje przydatne przy wywoływaniu obliczeń parametrów opcji
         C. `discount_functions.m` – funkcje generujące krzywą dyskontową
         D. `funkcje_kalendarzowe.m` – funkcje pomocnicze obliczające upływ czasu i dostosowujące daty wedle różnych konwencji
         E. `implied_volatility_ap.m` – funkcje tworzące krzywą zmienności implikowanej

F. `volatility_struct_hidden_vars.m` – plik ze zmiennymi do tworzenia krzywej zmienności implikowanej, które nie mogą być ustawiane przez użytkownika (i nie są dla niego widoczne)

G. `wroblewski_ap.m` – funkcje obliczające cenę oraz parametry greckie różnych opcji

ii. katalog `default\` - folder z domyślnymi danymi. Obecnie znajduje się tam tylko plik:

A. `fx_terminal.m` – domyślny plik służący do przekazywania danych pomiędzy kodem pythona a wywołaniami octave'a.

iii. katalog `triggers\` - katalog z plikami wywoływanymi bezpośrednio z kodu pythona (każdy plik wywoływany jest po naciśnięciu odpowiedniego przycisku)

A. `calculate_perform.m` – plik uruchamiany po naciśnięciu przycisku „*compute price and Greeks*" na etapie „calculate". Działanie tego etapu zostanie wyjaśnione później

B. `start_dom_curve_constr.m`, `start_for_curve_constr.m`, `start_vol_structure_constr.m` – pliki uruchamiane po naciśnięciu przycisku „*Calculate ...*" w zakładce *result* dla, odpowiednio, rynku domestic, foreign oraz struktury volatility; generuje dane będące finalnym efektem danego etapu wprowadzania danych

C. `start_ir.m`, `start_fx.m`, `start_vol.m` - pliki uruchamiane po naciśnięciu przycisku „*generate data*" w zakładce *verify* dla, odpowiednio, rynku domestic, foreign oraz struktury volatility; generuje tablicę z danymi wejściowymi do edycji

3. katalog `static\` - katalog z pomocniczymi plikami javascript i css oraz:

   (a) `models\fx` – katalog z przykładowymi danymi `csv` do załadowania do aplikacji

4. katalog `userdata\username\`

   (a) katalog `fx\`

   i. `fx_terminal.m` – plik przechowujący wartości podstawowych zmiennych wprowadzonych przez użytkownika

   ii. `octave.sh` – skrypt tworzony przy każdym wywołaniu triggera (zobacz sekcję „Triggery")

   iii. `octave.txt`, `octave.err` – pliki zapisujące standardowe wyjście oraz standardowy błąd z uruchomienia octave w pliku `octave.sh`

   iv. `calculate_selected_function.m` – w tym pliku umieszczane są dodatkowe parametry dla funkcji wyceniającej opcje (zobacz sekcję „Działanie wywoływania obliczeń parametrów opcji")

   v. `calculation_output.m` – tu zapisywany jest wynik obliczeń parametrów opcji

vi. reszta – pliki przechowujące rezultaty pośrednie ładowania danych

5. katalog `views\` - template'y html
   (a) katalog `fx\`
      i. `header.html` – dziedziczy po `layuot.html`; zawiera górny pasek z przyciskami pomocy i przyciskiem zmiany rynku
      ii. `load.html` – dziedziczy po `header.html`; zawiera treść taba *Select*
      iii. `load_dom.html`, `load_for.html`, `load_vol.html` – dziedziczą po `load.html`; każdy z tych plików składa się z 3 tabów (nie licząc jednego taba z pliku `load.html`): input, verify oraz result.
      iv. `calculate.html` – dziedziczy po `load.html`; składa się z jednego taba, posiadającego dwie kolumny: lewą do wprowadzania danych, prawą do wyświetlania wyniku
   (b) katalog `registration\`
      i. `password_change_form.html` – dziedziczy po `layuot.html`; formularz zmieniania hasła
      ii. `registration_form.html` – dziedziczy po `layuot.html`; formularz rejestracji użytkownika
   (c) `index_real.html` – dziedziczy po `layuot.html`; jeśli użytkownik jest zalogowany to wyświetla linki do rynku fixed oraz FX, a w przeciwnym przypadku wyświetla formularz logowania
   (d) `index.html` – dziedziczy po `layuot.html`; strona główna kalkulatora wyświetlająca linki do kalkulatorów *simple* oraz *real*

2 Schemat działania kalkulatora FX

W tym miejscu przedstawię typowy *use case* korzystania z kalkulatora FX.
Korzystanie z kalkulatora *real* możliwe jest tylko po zalogowaniu.

1. Użytkownik na głównej stronie wybiera link *real data calculator* i zostaje przekierowany na stronę logowania.
2. Jeśli użytkownik posiada już konto, to się loguje (jeśli nie, to najpierw korzysta z linka *Create Account* w celu stworzenia użytkownika).
3. Użytkownik wybiera linka *FX Market*, który otwiera stronę z menu dla tego rynku.
4. Użytkownik musi teraz wprowadzić dane. Robi to po kolei w trzech podobnie wyglądających etapach: *Domestic Interest Rates*, *Foreign Interest Rates* and *Exchange Rate i Volatility Structure.* Pokażemy to na przykładzie *Domestic Interest Rates.*

(a) Użytkownik klika link *Domestic Interest Rates*, który przekierowuje go na zakładkę *Input* dla tego etapu.

(b) W tej zakładce uzupełnia on kolejne dane. W szczególności musi uploadować pliki `.xml` ze stopami depozytów, FRA oraz IRSów. Sprawdzenie struktury tych plików oraz znaczenia odpowiednich pól możliwe jest w pomocy, która wyświetla się po naciśnięciu przycisków *Description*, *Instruction* oraz *About.*

(c) Po uzupełnieniu wszystkich pól użytkownik naciska przycisk *Save* na dole strony w celu zapisania wprowadzonych danych.

(d) Po jego naciśnięciu należy przejść do następnej zakładki, co jest możliwe zarówno poprzez wybór zakładki *Verify* (u góry strony), jak i poprzez naciśnięcie przycisku *Next* (na dole strony).

(e) Użytkownik naciska *Regenerate Data*, co powoduje wyświetlenie tabeli z danymi. Tabela ta może być zmodyfikowana, co użytkownik zatwierdza przyciskiem *Save corrected data.*

(f) Użytkownik przechodzi do następnej zakładki, co znowu możliwe jest na dwa sposoby: poprzez wybranie zakładki lub przyciskiem *Next.*

(g) W zakładce *Result* możliwa jest zmiana sposobu interpolacji (zapisuje się ją przyciskiem *Save interpolation method*).

(h) Użytkownik naciska przycisk *Calculate Domestic Discount Factor Table*, który oblicza i zapisuje wynik tego etapu wprowadzania danych.

(i) Na koniec etapu użytkownik przechodzi z powrotem do zakładki *Select* (a konkretnie w tym wypadku *Selected: Domestic Interest Rates*), gdzie możliwy jest wybór kolejnego etapu.

5. Po zakończeniu wprowadzania danych użytkownik w zakładce *Select* wybiera opcję, którą chciałby wycenić. Jej kliknięcie tworzy nową zakładkę: *Option calculation.*

6. W tej zakładce użytkownik wprowadza żądane parametry opcji i zatwierdza je przyciskiem *Save.*

7. Na zakończenie użytkownik naciska przycisk *Compute price and Greeks*, co powoduje wyświetlenie tabeli z wynikami w prawej części zakładki.

8. W zakładce *Select* można wybrać kolejną opcję, której parametry chcielibyśmy policzyć.

3 Działanie pomocy

Pomoc wyświetlana jest po naciśnięciu przycisków *Description*, *Instruction* lub *About*, znajdujących się na górnym pasku aplikacji. Dzieje się to poprzez wyświetlenie javascriptowego okna dialogowego (funkcje w pliku `pasek.js`).

Aplikacja umożliwia stworzenie różnych helpów dla każdej zakładki na każdym etapie wprowadzania danych, a także dla każdej opcji. Robi się to poprzez uzupełnienie odpowiednich słowników w pliku `views_fx.py`. Są to słowniki:

1. `help_texts_dom` dla wprowadzania danych na etapie *domestic,*
2. `help_texts_for` dla wprowadzania danych na etapie *foreign*,
3. `help_texts_vol` dla wprowadzania danych na etapie *volatility structure.*

Dla każdego taba konieczne jest uzupełnienie 3 tekstów w powyższej tabeli. Są to: `desc0`, `instr0`, `about0`, gdzie w miejsce 0 wstawiany jest numer taba (czyli tab *select* to 0, *input data* to 1, *verify* to 2 i *result* to 3).

Dodatkowym elementem helpów są ich tytuły (wyświetlane powyżej tekstu helpa w oknie dialogowym). Są one jednakowe dla wszystkich etapów wprowadzania danych i znajdują się w słowniku `help_titles`.

Specjalnym rodzajem helpów są te dla etapu obliczania parametrów opcji. W ich przypadku możliwe jest stworzenie jednego domyślnego helpa dla wszystkich opcji danego typu (czyli zwykłych, z barierą europejską lub z barierą amerykańską), który może być dla wybranych opcji zastępowany przez helpy specjalizowane.

Zorganizowane jest to w ten sposób, że w słowniku `help_texts_calc` dla każdego typu helpa (czyli `desc0`, `instr0`, `about0`, gdzie numer taba może być 0 – *select* lub 1 – *calculate*) mamy kolejny słownik. Zawiera on zawsze teksty dla 3 typów zwykłych (pod kluczami *simple*, *european* oraz *american*) oraz może zawierać teksty dla konkretnego typu opcji (kluczem jest tu skrótowa nazwa opcji, np. *uoeuput*). W momencie poszukiwania tekstu dla helpa program sprawdza najpierw, czy istnieje tekst specjalizowany, a jeśli go nie ma, to korzysta z odpowiedniego tekstu domyślnego.

Na podstawie tych tekstów przy pomocy funkcji *help_javascript* (z pliku `forms.py`) tworzony jest odpowiedni element html (`div`) do wyświetlania okna dialogowego, który następnie przekazywany jest do odpowiedniego template'a (czyli dodawany jest do odpowiedniej strony).

4 Triggery

Jak zostało już wspomniane wcześniej, obliczenia octave'a wywoływane są przy pomocy triggerów znajdujących się w katalogu `octavecodes/fx/triggers`. Po naciśnięciu przez użytkownika odpowiedniego przycisku (powiązania przycisków zostały opisane w punkcie 2.3 struktury katalogów):

1. wywoływana jest funkcja `execute_wait_reload`, która:
   (a) wywołuje funkcję odpalającą kod octave,
   (b) zmienia zawartość wskazanego diva na kółeczko oczekiwania,
   (c) aktywnie oczekuje na pojawienie się znacznika zakończenia obliczeń (czyli aż odpowiedni adres przestanie zwracać błąd braku strony),

(d) po zakończeniu obliczeń przeładowuje stronę.

2. Funkcja odpalająca kod octave'a (`calculate_exec`) tworzy skrypt bashowy, który:

   (a) kasuje wszystkie pliki tymczasowe `octave.*`,

   (b) uruchamia octave'a przekazując mu jako parametr odpowiedni skrypt (wszystkie takie skrypty znajdują się w katalogu `octavecodes/fx/triggers`),

   (c) po wykonaniu poprzednich zadań tworzy plik `octave.stamp`.

3. Plik `.sh` jest odpalany z pythona (a strona oczekuje na pojawienie się pliku `octave.stamp`).

4. Gdy pojawia się plik `octave.stamp` ładowane są z odpowiedniego pliku wyniki działania skryptu octave'a.

5. Każde takie obliczenie pozostawia po sobie plik `.m` z wynikami i plik ten jest później wykorzystywany przez następne skrypty octave'a.

5 Działanie wywoływania obliczeń parametrów opcji

Mechanizm obliczania parametrów opcji jest bardzo podobny do mechanizmu triggerów. Jedyną różnicą jest, że przed wywołaniem skryptu bashowego tworzony jest dodatkowo plik z informacjami o wywoływanej funkcji. Te informacje to: nazwa funkcji do wywołania, ilość strike'ów oraz informacja czy opcja posiada barierę.

Plik ten wykorzystywany jest później przez `trigger calculate_perform.m`, który weźmie pod uwagę te dane przy odpalaniu funkcji octave'a.

6 Opis funkcji

W tej sekcji przejdę do opisu podstawowych wykorzystywanych przeze mnie funkcji. Większość z nich znajduje się w pliku `views_fx.py`. Pozostałe zostaną specjalnie zaznaczone:

1. `load_dom()` - funkcja zwracająca stronę odpowiedzialną za wprowadzanie danych na etapie domestic. Wykonuje ona kilka operacji:

   (a) przygotowuje formularze do wprowadzania danych wraz z domyślnymi/ostatnio używanymi wartościami,

   (b) w zależności od istnienia plików z poprzednich kroków ustawia czy powinny być widoczne przyciski dla kolejnych aktywności,

   (c) tworzy divy z helpami,

(d) jeśli zapytanie zawierało przekazanie danych (metoda `POST`), to zapamiętuje odpowiednie dane (to jakie dane to są zależy od tego, z jakiego taba nastąpiło wywołanie),

(e) przekazuje nazwy opcji do stworzenia listy opcji w tabie *select*,

(f) wywołuje tworzenie strony (`mf_render`).

2. `load_for()`, `load_vol()` - analogiczne funkcje dla etapów *foreign* oraz *volatility*.

3. `calculate()` - funkcja działająca podobnie do poprzednich, tylko do tworzenia strony obliczającej parametry opcji. Dodatkowe jej zadania to:

   (a) sprawdzanie, czy wszystkie potrzebne dane zostały załadowane,

   (b) ustawianie, wprowadzanie których parametrów ma być możliwe dla użytkownika – jako, że opcje różnią się niektórymi parametrami,

   (c) ładowanie wyniku obliczeń – jeśli obliczenia zostały przeprowadzone dla obecnie wyświetlanej funkcji.

4. `get_option_names()` - tworzy tablicę par (kodowa/skrótowa nazwa opcji + pełna nazwa).

5. `calculate_strikes()` - korzystając z tabel `calculate_functions_2strikes_no_barrier` oraz `calculate_functions_3strikes_no_barrier` zwraca ile dana opcja ma strike'ów.

6. `calculate_parameters()` - obecnie nieużywana.

7. `is_barrier()` - sprawdza czy dana opcja jest opcją barierową (korzystając z tabeli `calculate_functions_1strike_barrier`).

8. `calculate_help()` - wyznacza helpy dla poszczególnych opcji. Sposób wyznaczania tych helpów został szczegółowo opisany w sekcji „Działanie pomocy".

9. `model_exec()` - funkcja do odpalania trigerrów (jej działanie zostało opisane w sekcji „Triggery").

10. `calculate_exec()` - funkcja do odpalania obliczeń parametrów opcji (jak wyżej).

11. *forms.py*: `help_javascript()` - tworzy diva zawierającego helpa o określonym tytule i tekście.

12. *views_ registration.py* – `register()` - wyświetla formularz rejestracji użytkownika.

13. *views_ registration.py* – `password_change()` - wyświetla formularz zmiany hasła.

14. *views_ registration.py* – `removeInactiveUsers()` - funkcja usuwająca nieaktywnych użytkowników. Przy czym użytkownik jest uznawany za nieaktywnego, jeśli jego ostatnie logowanie miało miejsce dawniej niż `[7 + (czas_od_zalogowania / 7)]` dni temu. Usuwanie to jest przeprowadzane tylko w momencie otwarcia formularza rejestracji nowego użytkownika.