Dokumentacja

Portal Mathfinance
dla rynku Fixed Income kalkulatora

Magda Hubicz, Patrycja Pol, Rafał Rutkowski

Spis treści

1 Struktura katalogów

Poniżej opisana została struktura katalogu `mathfinance\` w zakresie rynku Fixed Income.

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. Jest to główny plik dotyczący tego rynku i jego zawartość zostanie dokładniej opisana w rozdziale 4.
   (g) `views_fx.py` - funkcje dla rynku FX
   (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 `fixed\` - pliki octave'a do rynku Fixed Income
      i. katalog `code\` - pliki z właściwym kodem octave'a
         A. `base.m` - plik z danymi dotyczącymi standardów obowiązujących na różnych rynkach
         B. `cap_swap_vol.m` - funkcje generujące strukturę volatility dla capów i swapcji
         C. `bond.m`, `irs.m`, `fixed_deriv.m` - funkcje obliczające ceny instrumentów
         D. `discount_functions.m` - funkcje generujące krzywą dyskontową
         E. `funkcje_kalendarzowe.m` - funkcje pomocnicze obliczające upływ czasu i dostosowujące daty wedle różnych konwencji

ii. katalog `default\` - katalog z terminalami wczytywanymi przy pierwszym wywołaniu stron

iii. katalog `triggers\` - katalog z plikami Octave'a uruchamianymi bezpośrednio z kodu pythona (więcej na ten temat w rozdziale "Triggery"):

A. `load_dom_verify_start.m`, `load_dom_result_start.m`, `load_cap_start.m`, `load_swap_start.m` - pliki wywołujące funkcje generujące krzywą dyskontową oraz strukturę volatility dla capów i swapcji

B. pliki typu `..._base_update.m` - pliki wczytujące z pliku `base.m` dane dotyczące standardów obowiązujących na różnych rynkach

C. pliki typu `...PriceSkrypt.m` - pliki wywołujące funkcje obliczające ceny instrumentów

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

   (a) `models\fixed\` - katalog z przykładowymi danymi `csv` do załadowania do aplikacji

4. katalog `userdata\username\` - katalog przechowujący dane wprowadzone przez użytkownika

   (a) katalog `fixed\`

      i. `terminal.m` - plik przechowujący dodatkowe dane związane z wczytanymi kwotowniami rynkowymi (dane te dotyczą standardów obowiązujących na rynku)

      ii. `depo_rates.m`, `fra_rates.m`, `irs_rates.m`, `EC.m`, `ES.m` - pliki zawierające wczytane kwotowania rynkowe

      iii. pliki typu `..._terminal.m` - pliki przechowujące dane dotyczące poszczególnych instrumentów

      iv. pliki typu `...Price.out` - pliki zawierające cenę instrumentu, jeśli obliczenia zakończyły się powodzeniem, lub komunikat błędu, w przeciwnym przypadku

      v. `octave.sh` - skrypt tworzony przy każdym wywołaniu triggera (więcej na ten temat w sekcji "Triggery")

      vi. `octave.txt`, `octave.err` - pliki zapisujące standardowe wyjście oraz standardowy błąd z uruchomienia octave w pliku `octave.sh` (więcej na ten temat w sekcji "Triggery")

5. katalog `views\` - template'y html

   (a) katalog `fixed\`

      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_cap.html`, `load_swap.html` - dziedziczą po `load.html` - strony wczytywania kwotowań rynkowych

iv. pliki typu `calc_....html` - strony wprowadzania danych dotyczących poszczególnych instrumentów oraz wyświetlające wyniki obliczeń

(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 Fixed Income

W tym rozdziale opisany został typowy *use case* korzystania z kalkulatora Fixed Income.

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 *Fixed Income 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*, *Cap/Floor Volatility* oraz *Swaption Volatility*. 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 (przykładowe pliki z kwotowaniami znajdują się w katalogu `mathfinance\static\models\fixed`).

   (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 instrument, który chciałby wycenić. Jego kliknięcie tworzy nową zakładkę: *Price calculation*.

6. W tej zakładce użytkownik wprowadza żądane parametry instrumentu i zatwierdza je przyciskiem *Save*. W przypadku pól dotyczących standardów obowiązujących na rynku użytkownik może kliknąć przycisk *Set Default Values*, co spowoduje wypełnienie się pól danymi z bazy (plik `base.m`). W przypadku instrumentów niestandardowych możliwe są parametry zależne od okresu - w tym przypadku dane wprowadzane są w tabeli, której długość (liczbę okresów) ustala się za pomocą przycisków *Add Row* oraz *Delete Row*.

7. Na zakończenie użytkownik naciska przycisk *Calculate Cash Flow Tables*, co powoduje wyświetlenie ceny oraz tabeli z informacjami dotyczącymi poszczególnych przepływów pieniężnych.

8. W zakładce *Select* można wybrać kolejny instrument, którego cenę chcielibyśmy policzyć.

3 Triggery

Triggery są to pliki Octave'a uruchamiane bezpośrednio z pythona. Wszystkie triggery znajdują się w katalogu `octavecodes\fixed\triggers`. Rodzaje triggerów zostały opisane w punkcie 2.iii struktury katalogów. Triggery uruchamiane są przez naciśnięcie odpowiedniego przycisku na stronie (np. *Calculate Domestic Discount Factor Table*, *Calculate Cash Flow Tables*, *Set Default Values*):

1. Wywoływana jest funkcja `execute_wait_reload`, która:

(a) wywołuje funkcję odpalającą kod octave'a,

(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 (`model_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\fixed\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` lub plik `.out` z wynikami

4 Plik views_fixed.py

W tym rozdziale opisane zostały funkcje z pliku `views_fixed.py`.

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

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

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

2. `load_cap()`, `load_swap()` - analogiczne funkcje dla struktury volatility dla capów oraz swapcji

3. funkcje typu `calc_...()` - funkcje zwracające strony odpowiedzialne za wprowadzanie danych dotyczących poszczególnych instrumentów i wyświetlające wyniki obliczeń. Funkcje działają analogicznie do funkcji `load_dom()`. Ponadto, w przypadku instrumentów niestandardowych, jeśli wywołanie strony nastapiło przez kliknięcie przycisku *Add Row* lub przycisku *Delete Row*, funkcja modyfikuje pliki zwaierające tabelę.

4. `model_exec()` - funkcja do odpalania triggerów (jej działanie zostało opisane w rozdziale "Triggery")