Dokumentacja

Funkcje kalendarzowe

Mateusz Jabłoński, Agnieszka Kaszkowiak
wersja 2 – Krzesimir Arodź

Spis treści

1 Lista zadań i realizujące je funkcje

Funkcje realizujące poszczególnezadania:

a) Construction of standard calendar functions:

weekday(date) – Wbudowana funkcja Octave'a służąca do identyfikacji dnia tygodnia,

day_diff(date1,date2,basis) – Podstawowa funkcja licząca różnicę (w dniach) między dwoma datami. Nie są dokonywane żadne operacje przesunięcia do dni roboczych,

add_days(date1,n) – Funkcja przesuwająca datę o n dni. Nie są dokonywane żadne operacje przesunięcia do dni roboczych;

b) Implementation of the End of Month Adjustment:

add_months(date1, mwect,EMA) – Funkcja tworząca ciąg dat na podstawie daty początkowej i wektora długości okresów w miesiącach rozumianych nominalnie

c) Construction of a "business days" function:

easter(year,FC) – Funkcja pomocnicza wykorzystywana przy zadaniach związanych z dniami wolnymi

is_business_day(date,FC) – Główna funkcja sprawdzająca, czy podana data jest dniem aktywności wybranej giełdy

is_business_day2(date,FCL) – Funkcja sprawdzająca, czy podana data jest dniem aktywności wybranych giełd (wersja wielowalutowa funkcji **is_business_day**)

d) Implementation of the Business Day Adjustment:

day_shift(date1,FCL,x) – Funkcja pomocnicza przydatna w bardziej złożonych procedurach, szczególnie przy zagadnieniach związanych z konwencjami dni i płatności

day_shift2(date1,FCL,n) – Funkcja pomocnicza przydatna w bardziej złożonych procedurach, szczególnie przy zagadnieniach związanych z value date. Przesuwa podaną datę o n dni roboczych do przodu

mod_day_shift(date1,FCL,x) – Funkcja pomocnicza wykorzystywana w date_rolling(). Dla podanej daty odnajduje najbliższy następny/poprzedni (x = 1 / x = -1) dzień roboczy w sensie konwencji Modified Following/Previous Business Day

date_rolling(date1,FCL,BDA) – Podstawowa funkcja odnajdująca rzeczywisty dzień płatności (bez przesunięcia związanego z value date);

e) Implementation of the Day Count Convention:

act_act_frac(date1,date2) – Funkcja pomocnicza wykorzystywana w year_frac(). Zwraca frakcję roku dla konwencji ACT/ACT.

year_frac(date1, date2, DCC) – Główna funkcja wyznaczająca frakcję roku;

f) Payments functions:

payments(date1,nwect,FCL,DCC,BDA,valuedays) – Funkcja ułatwiająca korzystanie z poprzednich procedur w przypadku ciągu płatności, w tej wersji tworząca szereg dat w następujący sposób: do date1 dodaje nwect(1) dni i otrzymuje date2, na którą działa funkcją date_rolling(), a następnie przesuwa o valuedays dni roboczych do przodu (tzn. odnajduje rzeczywisty dzień płatności) do date1 dodaje nwect(1)+nwect(2) i otrzymuje date3, na którą działa funkcją date_rolling(), a następnie przesuwa o valuedays dni roboczych do przodu, itd., w efekcie uzyskuje ciąg length(nwect)+1 dat.

payments2(dates,FCL,DCC,BDA,valuedays) – Funkcja ułatwiająca korzystanie z poprzednich procedur w przypadku ciągu płatności, w tej wersji pobierająca explicite listę z datami, działająca na nie date_rolling(), na końcu przesuwająca o valuedays dni roboczych do przodu.

payments3(date1,n,d,FCL,DCC,BDA,valuedays) – Modyfikacja payments(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora dni pobiera liczbę okresów oraz (taką samą) ilość dni w każdym z nich

payments4(date1,mwect,FCL,DCC,BDA,valuedays,EMA) – Modyfikacja payments(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora dni pobiera wektor miesięcy. Przesunięcie następuje w rozumieniu zasady działania add_months().

payments5(date1,n,m,FCL,DCC,BDA,valuedays,EMA) – Modyfikacja payments4(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora miesięcy pobiera liczbę okresów oraz (taką samą) ilość miesięcy w każdym z nich. Przesunięcie następuje w rozumieniu zasady działania add_months().

2 Zagadnienia wprowadzające

2.1 Day Count Convention

Day Count Convention ustala sposób wyznaczania ułamka roku pomiędzy dwiema danymi datami. Konwencje Day Count Convention można podzielić na 2 zasadnicze grupy:

- 30/360;

- actual.

Jeśli dane są dwie daty T_1 i T_2 to Day Count Convention ustala sposób wyznaczania ułamka roku od (włączając) T_1 do (wyłączając) T_2.

W dalszej części używać będziemy następujących oznaczeń:

- D_1, M_1, R_1 - odpowiednio dzień, miesiąc i rok dla daty T_1,
- D_2, M_2, R_2 - odpowiednio dzień, miesiąc i rok dla daty T_2,
- $d(T_1, T_2)$ - rzeczywista liczba dni między datami T_1 i T_2.

Algorytm obliczania ułamka roku dla konwencji typu 30/360

Celem obliczenia ułamka roku pomiędzy T_1 i T_2 dla poszczególnych konwencji typu 30/360 należy:

1. zmodyfikować D_1 i D_2 zgodnie z niżej opisanym algorytmem dla odpowiedniej konwencji,
2. obliczyć liczbę dni pomiędzy otrzymanymi po zastosowaniu algorytmów datami (przy czym przyjmujemy, że każdy pełen miesiąc ma 30 dni, a każdy pełen rok ma 360 dni),
3. otrzymaną liczbę dni podzielić przez 360.

Algorytm modyfikowania D_1 i D_2 dla konwencji typu 30/360

30/360

- Jeśli D_1 i D_2 są ostatnimi dniami lutego, to przyjmujemy D_2=30.
- Jeśli D_1 jest ostatnim dniem lutego i D_2 jest ostatnim dniem miesiąca, to przyjmujemy D_1=30.
- Jeśli D_2=31 i (D_1=31 lub D_1=30), to przyjmujemy D_2=30.
- Jeśli D_1=31, to przyjmujemy D_1=30.

30E/360

- Jeśli D_1=31, to przyjmujemy D_1=30.
- Jeśli D_2=31, to przyjmujemy D_2=30.

30E/360 ISDA

- Jeśli D_1 jest ostatnim dniem miesiąca, to przyjmujemy D_1=30.
- Jeśli D_2 jest ostatnim dniem miesiąca, to przyjmujemy D_2=30 (z wyjątkiem przypadku, gdy jest to luty, zaś T_2 jest maturity date, wtedy zostawiamy D_2 bez zmiany)

30E+/360

- Jeśli D_1=31, to przyjmujemy D_1=30.

- Jeśli D_2=31, to zwiększamy M_2 o 1 i przyjmujemy D_2=1.

Algorytmy wyznaczania ułamka roku dla konwencji typu actual

ACT/360

- Frakcja roku wynosi $\frac{d(T_1,T_2)}{360}$.

ACT/ACT ICMA

- W przypadku, gdy T_1 i T_2 leżą w tym samym okresie odsetkowym, ułamek roku otrzymujemy ze wzoru $\frac{l(T_P,T_K)\cdot d(T_1,T_2)}{12\cdot d(T_P,T_K)}$, gdzie T_P jest datą początku okresu odsetkowego, T_K jest datą końca okresu odsetkowego, zaś $l(T_P,T_K)$ długością okresu odsetkowego wyrażoną w miesiącach.
- W przeciwnym wypadku, ułamek roku otrzymujemy ze wzoru $\frac{l(T_{P1},T_{K1})\cdot d(T_1,T_{K1})}{12\cdot d(T_{P1},T_{K1})}+l(T_{K1},T_{P2})+\frac{l(T_{P2},T_{K2})\cdot d(T_{P2},T_2)}{12\cdot d(T_{P2},T_{K2})}$, gdzie T_{Pi} i T_{Ki} są datami odpowiednio początku i końca okresu odsetkowego, w którym leży T_i, zaś $l(T_a,T_b)$ jest łączną długością (wyrażoną w miesiącach) wszystkich okresów odsetkowych zawierających się między datami T_a i T_b.

ACT/ACT

- Ułamek roku wynosi $\frac{D_{np}}{365}+\frac{D_p}{366}$, gdzie D_{np} jest liczbą dni w okresie między T_1 a T_2 wypadających w latach, które nie są przestępne, zaś D_p jest liczbą dni w tym okresie wypadających w latach przestępnych.

ACT/365

- Ułamek roku wynosi $\frac{d(T_1,T_2)}{365}$.

ACT/365L

- Jeśli długość okresu odsetkowego wynosi dokładnie jeden rok, to przyjmujemy, że *bazaodsetkowa* wynosi 366 jeśli w przedziale od T_1 (wyłączając) do T_2 (włączając) zawarty jest 29 lutego, zaś 365 w przeciwnym wypadku.
- Jeśli długość okresu odsetkowego jest różna od jednego roku, to przyjmujemy, że *bazaodsetkowa* wynosi 366 jeśli R_2 jest rokiem przestępnym, zaś 365 w przeciwnym przypadku.
- Ułamek roku wynosi $\frac{d(T_1,T_2)}{\text{baza odsetkowa}}$.

ACT/ACT AFB

- Jeśli długość okresu odsetkowego przekracza jeden rok, to obliczamy (od końca okresu) liczbę pełnych lat, którą następnie dodajemy do ułamka, przedstawiającego ułamek roku dla pozostałego okresu i obliczonego zgodnie z wzorem podanym poniżej. Jeśli T_2 jest ostatnim dniem lutego, to data otrzymana w skutek tego przesunięcia T_2 do tyłu będzie również ostatnim dniem lutego.
- Przyjmujemy, że *bazaodsetkowa* wynosi 366 jeśli w przedziale od T_1 (wyłączając) do T_2 (włączając) zawarty jest 29 lutego, zaś 365 w przeciwnym wypadku.
- Ułamek roku wynosi $\frac{d(T_1,T_2)}{\text{baza odsetkowa}}$.

2.2 Konwencja Business Day Adjustment

Konwencja płatności określa, kiedy nastąpi przepływ wynikający z danego instrumentu finansowego, wówczas gdy data tego przepływu (z warunków transakcji) przypada na dzień wolny od pracy.

Istnieje pięć podstawowych typów konwencji płatności:

1. Konwencja następnego dnia roboczego (Standard Following Business Day, sfbd) - Data płatności przesuwana jest na pierwszy dzień roboczy po dniu płatności.

2. Zmodyfikowana konwencja następnego dnia roboczego (Modified Following Business Day, mfbd) - Data płatności przesuwana jest na pierwszy dzień roboczy po dniu płatności, jeżeli ów dzień znajduje się w tym samym miesiącu kalendarzowym co data płatności. W przeciwnym przypadku, data płatności przesuwana jest na pierwszy dzień roboczy przed dniem płatności.

3. Konwencja poprzedniego dnia roboczego(Standard Previous Business Day, spbd) - Data płatności przesuwana jest na pierwszy dzień roboczy przed dniem płatności.

4. Zmodyfikowana konwencja poprzedniego dnia roboczego (Modified Previous Business Day, mpbd) - Data płatności przesuwana jest na pierwszy dzień roboczy przed dniu płatności., jeżeli ów dzień znajduje się w tym samym miesiącu kalendarzowym co data płatności . W przeciwnym przypadku, data płatności przesuwana jest na pierwszy dzień roboczy po dniu płatności.

5. Konwencja końca miesiąca (Standard End of Month, seom) - Data płatności ustalana jest w ostatnim dniu roboczym miesiąca kalendarzowego.

2.3 Wyznaczanie dnia Wielkanocy

Algorytm Meeusa/Jonesa/Butchera służy do obliczania ruchomego święta Wielkanocy. Ten sposób został przedstawiony przez Jeana Meeusa w jego książce

Astronomical Algorithms w 1991 roku. Może być uznany za lepszy od algorytmu Gaussa, ponieważ nie wymaga żadnych cyfr dla określonego zakresu czasu i nie ma od niego wyjątków. Wystarczy podać dowolny rok.

Algorytm Meeusa/Jonesa/Butchera dla kalendarza gregoriańskiego:

1. Dzielimy liczbę roku na 19 i wyznaczamy resztę a.
2. Dzielimy liczbę roku przez 100, wynik zaokrąglamy w dół (odcinamy część ułamkową) i otrzymujemy liczbę b.
3. Dzielimy liczbę roku przez 100 i otrzymujemy resztę c.
4. Liczymy: b : 4 i wynik zaokrąglamy w dół i otrzymujemy liczbę d.
5. Liczymy: b : 4 i wyznaczamy resztę e.
6. Liczymy: (b + 8) : 25 i wynik zaokrąglamy w dół i otrzymujemy liczbę f.
7. Liczymy: (b - f + 1) : 3 i wynik zaokrąglamy w dół i otrzymujemy liczbę g.
8. Liczymy: (19 x a + b - d - g + 15) : 30 i wyznaczamy resztę h.
9. Liczymy: c : 4 i wynik zaokrąglamy w dół i otrzymujemy cyfrę i.
10. Liczymy: c : 4 i wyznaczamy resztę k.
11. Liczymy: (32 + 2 x e + 2 x i - h - k) : 7 i otrzymujemy resztę l.
12. Liczymy: (a + 11 x h + 22 x l) : 451 i wynik zaokrąglamy w dół i otrzymujemy liczbę m.
13. Liczymy: (h + l - 7 x m + 114) : 31 i otrzymujemy resztę p.
14. Dzień Wielkanocy = p + 1.
15. Miesiąc = Zaokrąglenie w dół dzielenia (h + l - 7 x m + 114) przez 31.

Święta ruchome

Poza dniem Wielkanocy, dla każdego centrum finansowego charakterystyczne są również powiązane z nim święta ruchome. Zestawienie tych świąt znajduje się poniżej:

Warszawa

- Wielki Piątek - 2 dni przed Niedzielą Wielkanocną
- Poniedziałek Wielkanocny - 1 dzień po Wielkanocy
- Boże Ciało - 60 dni po wielkanocy

Londyn

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy

Zurich

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy
- Ascension Day - 39 dni po Wielkanocy
- Whit Monday - 50 dni po Wielkanocy

Frankfurt

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy
- Ascension Thursday - 39 dni po Wielkanocy
- Whit Monday - 50 dni po Wielkanocy
- Corpus Christi - 60 dni po Wielkanocy

TARGET

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy

Sydney

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy

Toronto

- Good Friday - 2 dni przed Niedzielą Wielkanocną
- Easter Monday - 1 dzień po Wielkanocy

NYMEX

- Good Friday - 2 dni przed Niedzielą Wielkanocną

2.4 Święta nieruchome

Wykaz świąt (poza objętymi funkcją easter):

Warszawa

- Nowy Rok - 1 stycznia
- Trzech Króli - 6 stycznia (od roku 2011)
- Święto państwowe - 1 maja

- Święto Konstytucji 3 Maja - 3 maja
- Wniebowzięcie NMP - 15 sierpnia
- Wszystkich Świętych - 1 listopada
- Święto Niepodległości - 11 listopada
- Wigilia Bożego Narodzenia - 24 grudnia
- Boże Narodzenie - 25, 26 grudnia

Londyn

- New Year's Day - 1 stycznia
- Early May Bank Holiday - pierwszy poniedziałek maja
- Spring Bank Holiday - ostatni poniedziałek maja
- Summer Bank Holiday - ostatni poniedziałek sierpnia
- Christmas Day - 25 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)
- Boxing Day - 26 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)

Zurich

- New Year's Day - 1 stycznia
- Berchtoldstag - 2 stycznia
- Labour Day - 1 maja
- National Day 1 sierpnia
- Christmas Eve - 24 grudnia
- Christmas - 25 grudnia
- St. Stephen's Day - 26 grudnia
- New Year's Eve - 31 grudnia

Frankfurt

- New Year's Day - 1 stycznia
- Labour Day - 1 maja
- National Day - 3 października
- Christmas Eve - 24 grudnia
- Christmas - 25 grudnia
- Boxing Day - 26 grudnia
- New Year's Eve - 31 grudnia

NYMEX

- New Year's Day - 1 stycznia (możliwe przesunięcie na poniedziałek)
- Martin Luther King's birthday - trzeci poniedziałek stycznia
- Washington's birthday - trzeci poniedziałek lutego
- Memorial Day - ostatni poniedziałek maja
- Independence Day - 4 lipca (możliwe przesunięcie na piątek lub poniedziałek)
- Labour Day - pierwszy poniedziałek września
- Thanksgiving Day - trzeci czwartek listopada
- Christmas - 25 grudnia (możliwe przesunięcie na piątek lub poniedziałek)

Sydney

- New Year's Day - 1 stycznia (możliwe przesunięcie na poniedziałek)
- Australia Day - 26 stycznia (możliwe przesuniecie na poniedziałek)
- ANZAC Day - 25 kwietnia (możliwe przesunięcie na poniedziałek)
- Queen's Birthday - drugi poniedziałek czerwca
- Bank Holiday - pierwszy poniedziałek sierpnia
- Labour Day - pierwszy poniedziałek października
- Christmas - 25 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)
- Boxing Day - 26 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)

TARGET

- New Year's Day - 1 stycznia
- Labour Day - 1 maja (od roku 2000)
- Christmas - 25 grudnia
- Day of Goodwill - 26 grudnia (od roku 2000)
- 31 grudnia (lata 1998-1999)

Tokyo

- New Year's Day
- Bank Holiday - 2 i 3 stycznia
- Coming of Age Day - drugi poniedziałek stycznia (przed rokiem 2000 - 15 stycznia z możliwym przesunięciem na poniedziałek)
- National Foundation Day - 11 lutego (możliwe przesunięcie na poniedziałek)

- równonoc wiosenna [1]
- Showa Day - 29 kwietnia (możliwe przesunięcie na poniedziałek)
- Constitution Memorial Day - 3 maja
- Greenery Day - 4 maja
- Children's Day - 5 maja (możliwe przesunięcie na poniedziałek)
- Marine Day - trzeci poniedziałek lipca (od roku 1996; w latach 1996-2002 włącznie 20 lipca z możliwym przesunięciem na poniedziałek)
- Respect for the Aged Day - trzeci poniedziałek września (przed rokiem 2003 - 15 września z możliwym przesunięciem na poniedziałek)
- równonoc jesienna
- Health and Sports Day - drugi poniedziałek października (przed rokiem 2000 - 10 października z możliwym przesunięciem na poniedziałek)
- National Culture Day - 3 listopada (możliwe przesunięcie na poniedziałek)
- Labor Thanksgiving Day - 23 listopada (możliwe przesunięcie na poniedziałek)
- Emperor's Birthday - 23 grudnia (od 1989 roku, możliwe przesunięcie na poniedziałek)
- Bank Holiday - 31 grudnia
- Marriage of Prince Akihito - 10 kwietnia 1959
- Rites of Imperial Funeral - 24 lutego 1989
- Enthronement Ceremony - 12 listopada 1990
- Marriage of Prince Naruhito - 9 czerwca 1993
- w wypadku, gdy pomiędzy świętem nieruchomym Respect for the Aged Day wypada dokładnie jeden dzień, ten dzień jest również wolny.

Toronto

- New Year's Day - 1 stycznia (możliwe przesunięcie na poniedziałek)
- Victoria Day - poniedziałek wypadający 24 maja lub najbliższy poniedziałek przed 24 maja
- Canada Day - 1 lipca (możliwe przesunięcie na poniedziałek)
- Provincial Holiday - pierwszy poniedziałek sierpnia
- Labor Day - pierwszy poniedziałek września
- Thanksgiving Day - drugi poniedziałek listopada

[1] Funkcja zawiera implementację algorytmu, obliczającego dzień równonocy wiosennej i jesiennej, niemniej ze względu na jego niedokładność może on niekiedy dawać błędne wyniki.

- 11 listopada
- Christmas - 25 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)
- Boxing Day - 26 grudnia (możliwe przesunięcie na poniedziałek lub wtorek)

2.5 Wyznaczanie dni równonocy wiosennej i jesiennej

Poniższy algorytm służy do obliczania dni równonocy wiosennej i jesiennej. Ze względu na jego niedokładność[2], może on niekiedy dawać błędne wyniki.

Opis działania algorytmu:

1. W programie wprowadzone są jako stałe daty ekwinocjów w roku 2000. Przyjęto, że dokładna data ekwinocjum wiosennego (dzień jest przedstawiony jako ułamek; część ułamkowa reprezentuje godziny i minuty) to 20.69115 marca, a dokładna data ekwioncjum jesiennego to 23.09 września. Przyjęto, że długość roku astronomicznego wynosi 365.242194 dni.

2. Następnie obliczamy, ile spośród lat pomiędzy rokiem 2000 a rokiem, dla którego liczymy daty ekwinocjów (ozn *rok*), było przestępnych. Liczba tych lat (ozn *przestepne*) to część całkowita z następujacej liczby:

$$(rok - 2000) : 4 + (rok - 2000) : 100 - (rok - 2000) : 400$$

3. Ekwinocjum wiosenne wypada w n-tym dniu marca, gdzie n jest częścią całkowitą z następujacej liczby:

$$20.69115 + (rok - 2000) \cdot (0.242194) - przestepne$$

Zaś ekwinocjum jesienne w n-tym dniu września, gdzie n jest częścią całkowitą z następującej liczby:

$$23.09 + (rok - 2000) \cdot (0.242194) - przestepne$$

3 Zakres projektu

Przygotowana dokumentacja dotyczy części Dużego Projektu 2010 oraz części Dużego Projektu 2011, przygotowanych w Octave-3.2.3. Zawiera 22 funkcje kalendarzowe do wykorzystania w pozostałych częściach Dużego Projektu, bądź niezależnie.

Dopuszczalny zakres dat obejmuje okres 1 Stycznia 1970 – 19 stycznia 2038.

Wykorzystane oznaczenia w dalszym opisie projektu:

[2]Por np http://aom.giss.nasa.gov/srvernal.html; wynika ona z tego, że nie znamy dostatecznie dokładnego przybliżenia długości roku astronomicznego. Jeden z wyjątków w podanym tu algorytmie to równonoc jesienna w 1997, która wg algorytmu wypada 23.09, a w rzeczywistości był to 22.09 o godzinie 23:56.

BDA – dostępne identyfikatory BDA:

- sfbd = Standard Following Business Day,
- mfbd = Modified Following Business Day,
- spbd = Standard Previous Business Day,
- mpbd = Modified Previous Business Day,
- seom = Standard End of Month,
- actu = Actual (formalny parametr oznaczający brak konwencji tzn. datę faktyczną i brak przesunięcia);

DCC – dostępne konwencje:

- 30/360,
- 30E/360,
- 30E+/360,
- ACT/360,
- ACT/ACT,
- ACT/365,
- ACT/365L,
- ACT/ACT AFB
- 30E/360 ISDA i ACT/ACT ICMA, ale ponieważ ich wywołanie wymaga podania dodatkowych danych, są one dostępne wyłącznie w funkcjach isda_30e_360_frac i act_act_icma_frac odpowiednio.

FC – centra finansowe, dostępne listy dni roboczych wg następujących kalendarzy:

- warsaw, london, zurich, frankfurt, NYMEX, sydney, TARGET, toronto, tokyo.

DC360 – identyfikator dotyczące sposobu liczenia liczby dni miedzy dwiema datami dla metody 30/360; dostępne identyfikatory:

- ACT,
- 30,
- 30E,
- 30E+;

lista rynków (FCL) – cell array zawierający kolejno: $n > 0$ parametrów FC oraz opcjonalny parametr FCi, będący też cell array i zawierający liczby ze zbioru $\{1, ..., n\}$ lub liczbę 0. Zadany dzień będzie traktowany przez funkcje kalendarzowe jako roboczy w zależności od wartości opcjonalnego parametru:

- Jeśli FCi nie został podany, to zadany dzień uznajemy za roboczy, jeśli jest roboczy w każdym z wymienionych w FCL centrów finansowych,
- Jeśli FCi = $\{0\}$, to zadany dzień uznajemy za roboczy, jeśli jest roboczy w co najmniej jednym z wymienionych w FCL centrów finansowych,
- Jeśli FCi = $\{a_1, a_2, ..., a_k\}$, gdzie a_i jest jedną z liczb $\{1, ..., n\}$ to zadany dzień uznajemy za roboczy, jeśli jest roboczy w każdym z centrów finansowych o numerach podanych w FCi (przyjmujemy numerację wg kolejności centrów finansowych w FCL).

4 Funkcje kalendarzowe

4.1 weekday(date)

Opis funkcji:

Wbudowana funkcja Octave'a służąca do identyfikacji dnia tygodnia.

Dane wejściowe:

date - string interpretowany jako data formatu "dd-mmm-yyyy"

Dane wyjściowe:

integer z zakresu [1,7] (od: 1=niedziela do: 7=sobota)

Przykład wywołania:

weekday("01-Jan-2001")

4.2 day_diff(date1,date2,basis)

Opis funkcji:

Podstawowa funkcja licząca różnicę (w dniach) między dwoma datami. Nie są dokonywane żadne operacje przesunięcia do dni roboczych.

Dane wejściowe:

date1 – string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 – string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa), date1<date2,

basis – string, identyfikator konwencji DC360.

Dane wyjściowe:

integer – liczba dni pomiędzy datami w sensie danej konwencji.

Przykład wywołania:

day_diff("01-Jan-2001","01-Jan-2002","ACT").

4.3 add_days(date1,n)

Opis funkcji:

Funkcja przesuwająca datę o n dni. Nie są dokonywane żadne operacje przesunięcia do dni roboczych.

Dane wejściowe:

date1 – string interpretowany jako data formatu "dd-mmm-yyyy"*

n – liczba całkowita, liczba dni (0 = data bez zmian)

Dane wyjściowe:

string interpretowany jako data formatu "dd-mmm-yyyy"* (date1 plus n dni)

Przykład wywołania:

add_days("01-Jan-2001",5)

4.4 add_months(date1, mwect,EMA)

Opis funkcji:

Funkcja tworząca ciąg dat na podstawie daty początkowej i wektora długości okresów w miesiącach rozumianych nominalnie np. 1 stycznia 2000 + 1M = 1 lutego 2000, 1 stycznia 2000 + 1M + 3M = 1 maja 2000.

W przypadku odwołania do nieistniejącego dnia np. 31 lutego, parametr EMA koduje cofnięcie do ostatniego dnia miesiąca (x = -1) lub przesunięcie do pierwszego dnia miesiąca następnego (x = 1). Poza tym nie są dokonywane inne przesunięcia dat.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)*

nwect - wektor liczb naturalnych, liczba miesięcy w poszczególnych okresach

EMA - integer o wartości 1 lub -1

Dane wyjściowe:

lista length(mwect)+1 stringów interpretowanych jako daty

(formatu "dd-mmm-yyyy"*) wyznaczone z danych wejściowych

Przykład wywołania:

dat_list = add_months("01-Jan-2001",[3,6,3,12],-1)

4.5 easter(year,FC)

Opis funkcji:

Funkcja pomocnicza wykorzystywana przy zadaniach związanych z dniami wolnymi. Za pomocą algorytmu Meeusa/Jonesa/Butchera odnajduje datę Wielkanocy i oblicza powiązane z tym świętem daty świąt ruchomych na danej giełdzie.

Dane wejściowe:

date - string interpretowany jako data formatu "dd-mmm-yyyy"

FC - string, identyfikator centrum finansowego

Dane wyjściowe:

lista ze stringami interpretowanymi jako daty (formatu "dd-mmm-yyyy") bank holidays wyznaczonych przez Wielkanoc na danej giełdzie

Przykład wywołania:

easter_hol = easter(2001,"warsaw")

4.6 is_business_day(date,FC)

Opis funkcji:

Główna funkcja sprawdzająca, czy podana data jest dniem aktywności wybranej giełdy. Funkcja zawiera sprawdzenie, czy dany dzień nie jest sobotą albo niedzielą, a następnie sprawdza, czy nie jest jednym ze świąt charakterystycznych dla danego rynku.

Dane wejściowe:

date - string interpretowany jako data formatu "dd-mmm-yyyy"

FC - string, identyfikator centrum finansowego

Dane wyjściowe:

logiczny 0 lub 1 (dany dzień nie jest/jest dniem roboczym)

Przykład wywołania:

is_business_day("01-Jan-2001","warsaw")

4.7 is_business_day2(date,FCL)

Opis funkcji:

Główna funkcja sprawdzająca, czy podana data jest dniem aktywności wybranych giełd.

Dane wejściowe:

date - string interpretowany jako data formatu "dd-mmm-yyyy"

FCL - cell array, identyfikator centrów finansowych

Dane wyjściowe:

logiczny 0 lub 1 (dany dzień nie jest/jest dniem roboczym)

Przykłady wywołania:

is_business_day2("01-Jan-2001",{"warsaw","london"})

is_business_day2("01-Jan-2001",{"warsaw","london",{0}})

is_business_day2("01-Jan-2001",{"warsaw","london","zurich",{1,3}})

4.8 day_shift(date1,FCL,x)

Opis funkcji:

Funkcja pomocnicza przydatna w bardziej złożonych procedurach, szczególnie przy zagadnieniach związanych z konwencjami dni i płatności. Dla podanej daty odnajduje najbliższy następny (x = 1) lub najbliższy poprzedni (x = -1) dzień roboczy na danej giełdzie.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy",

FCL - cell array, identyfikator centrów finansowych,

x - integer o wartości 1 lub -1,

Dane wyjściowe:

string interpretowany jako data formatu "dd-mmm-yyyy" (x = 1 najbliższy następny, x = -1 najbliższy poprzedni dzień roboczy)

Przykład wywołania:

day_shift("01-Jan-2001",{"warsaw","frankfurt"},1)

4.9 day_shift2(date1,FCL,n)

Opis funkcji:

Funkcja pomocnicza przydatna w bardziej złożonych procedurach, szczególnie przy zagadnieniach związanych z value date. Przesuwa podaną datę o n dni roboczych do przodu/do tyłu.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy"

FCL - cell array, identyfikator centrów finansowych

n - integer, liczba dni roboczych

Dane wyjściowe:

string interpretowany jako data formatu "dd-mmm-yyyy", data oddalona o n dni roboczych od daty wejściowej

Przykład wywołania:

day_shift2("01-Jan-2001",{"warsaw","zurich",{2}},-5)

4.10 mod_day_shift(date1,FCL,x)

Opis funkcji:

Funkcja pomocnicza wykorzystywana w date_rolling(). Dla podanej daty odnajduje najbliższy następny/poprzedni (x = 1 / x = -1) dzień roboczy w sensie konwencji Modified Following/Previous Business Day.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy"

FCL - cell array, identyfikator centrów finansowych

x - integer o wartośi 1 lub -1

Dane wyjściowe:

string interpretowany jako data formatu "dd-mmm-yyyy" (x = 1 najbliższy następny, x = -1 najbliższy poprzedni dzień roboczy w sensie konwencji Modified Following/Previous Business Day)

Przykład wywyłania:

mod_day_shift("01-Jan-2001",{"warsaw"},1)

4.11 date_rolling(date1,FCL,BDA)

Opis funkcji:

Podstawowa funkcja odnajdująca rzeczywisty dzień płatności (bez przesunięcia związanego z value date).

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy"

FCL - cell array, identyfikator centrów finansowych

BDA - string, identyfikator konwencji płatności

Dane wyjściowe:

string interpretowany jako data formatu "dd-mmm-yyyy" (dzień płatności odpowiadający date1 wyznaczony przez konwencje)

Przykład wywołania:

date_rolling("01-Jan-2001",{"warsaw","zurich"},"sfbd")

4.12 act_act_frac(date1,date2)

Opis funkcji:

Funkcja pomocnicza wykorzystywana w year_frac(). Zwraca frakcję roku dla konwencji ACT/ACT.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

Dane wyjściowe:

liczba rzeczywista, frakcja roku (w konwencji ACT/ACT) wyznaczona przez daty date1 i date2

Przykład wywołania:

act_act_frac("01-Jan-2001","01-Jan-2002")

4.13 act_act_afb_frac(date1,date2)

Opis funkcji:

Funkcja pomocnicza wykorzystywana w year_frac(). Zwraca frakcję roku dla konwencji ACT/ACT AFB.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

Dane wyjściowe:

liczba rzeczywista, frakcja roku (w konwencji ACT/ACT AFB) wyznaczona przez daty date1 i date2

Przykład wywołania:

act_act_afb_frac("01-Jan-2001","01-Jan-2002")

4.14 act_365l_frac(date1, date2)

Opis funkcji:

Funkcja pomocnicza wykorzystywana w year_frac(). Zwraca frakcję roku dla konwencji ACT/365L.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

Dane wyjściowe:

liczba rzeczywista, frakcja roku (w konwencji ACT/365L) wyznaczona przez daty date1 i date2

Przykład wywołania:

act_365l_frac("01-Jan-2001","01-Jan-2002")

4.15 act_act_icma_frac(date1, date2, date_start1, date_end1, date_start2, date_end2, period1, period2, period_mid)

Opis funkcji:

Funkcja obliczająca frakcję roku w konwencji ACT/ACT ICMA. Nie jest dostępna z poziomu innych funkcji.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

date_start1 - string interpretowany jako data formatu "dd-mmm-yyyy" (początek okresu odsetkowego, w którym leży date1)

date_end1 - string interpretowany jako data formatu "dd-mmm-yyyy" (koniec okresu odsetkowego, w którym leży date1)

date_start2 - string interpretowany jako data formatu "dd-mmm-yyyy" (początek okresu odsetkowego, w którym leży date2)

date_end2 - string interpretowany jako data formatu "dd-mmm-yyyy" (koniec okresu odsetkowego, w którym leży date2)

period1 - dodatni integer interpretowany jako długość okresu (date_start1, date_end1) wyrażona w miesiącach

period2 - dodatni integer interpretowany jako długość okresu (date_start2, date_end2) wyrażona w miesiącach

period_mid - nieujemny integer interpretowany jako długość okresu (date_end1, date_start2) wyrażona w miesiącach; w wypadku gdy date1 i date2 leżą w tym samym okresie odsetkowym, należy podać liczbę 0

Dane wyjściowe:

liczba rzeczywista, frakcja roku (w konwencji ACT/ACT ICMA) wyznaczona przez daty date1 i date2 leżące w okresach odsetkowych (date_start1, date_end1) i (date_start2, date_end2) odpowiednio, gdzie długość tych okresów wynosi odpowiednio period1 i period2, zaś okresy odsetkowe pomiędzy nimi mają łączną długość period_mid

Przykład wywołania:

act_365l_frac("03-Jan-2001","03-Dec-2001","01-Jan-2001","31-Mar-2001","01-Oct-2001","31-Dec-2001",3,3,6)

4.16 isda_30e_360_frac(date1, date2, maturity)

Opis funkcji:

Funkcja obliczająca frakcję roku w konwencji 30E/360 ISDA. Nie jest dostępna z poziomu innych funkcji.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

maturity - string interpretowany jako data formatu "dd-mmm-yyyy" (maturity date)

Dane wyjściowe:

liczba rzeczywista, frakcja roku (w konwencji 30E/360 ISDA) wyznaczona przez daty date1 i date2, gdzie maturity jest datą zapadalności rozważanego instrumentu finansowego

Przykład wywołania:

isda_30e_360_frac("01-Jan-2001","01-Jan-2002","01-Jan-2002")

4.17 year_frac(date1, date2, DCC)

Opis funkcji:

Główna funkcja wyznaczająca frakcję roku.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

date2 - string interpretowany jako data formatu "dd-mmm-yyyy" (data końcowa)

DCC - string, identyfikator konwencji DCC

Dane wyjściowe:

liczba rzeczywista interpretowana jako frakcja roku (w zadanej konwencji) wyznaczona przez daty date1 i date2

Przykład wywołania:

year_frac("01-Jan-2001","01-Jan-2002","ACT/360")

4.18 payments(date1,dwect,FCL,DCC,BDA,CPO)

Opis funkcji:

Funkcja ułatwiająca korzystanie z poprzednich procedur w przypadku ciągu płatności, w tej wersji tworząca szereg dat w następujący sposób:

do date1 dodaje dwect(1) dni i otrzymuje date2, na którą działa funkcją date_rolling(), a następnie przesuwa o valuedays dni roboczych do przodu (tzn. odnajduje rzeczywisty dzień płatności)

do date1 dodaje dwect(1)+dwect(2) i otrzymuje date3, na którą działa funkcją date_rolling(), a następnie przesuwa o valuedays dni roboczych do przodu, ..., itd

w efekcie uzyskuje ciąg length(dwect)+1 dat.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

dwect - wektor liczb naturalnych, liczba dni w poszczególnych okresach

FCL - cell array, identyfikator centrów finansowych

DCC - string, identyfikator konwencji liczenia dni

BDA - string, identyfikator konwencji płatności

CPO - integer nieujemny, liczba dni roboczych od contract date do value date

Dane wyjściowe:

lista length(dwect)+1 stringów interpretowanych jako daty (formatu "dd-mmm-yyyy" , dni robocze) wyznaczone z danych wejściowych

lista length(dwect) liczb rzeczywistych, frakcji roku (w zadanej konwencji) pomiędzy sąsiednimi datami z poprzedniej listy

Przykład wywołania:

[data_list,frac_list] = payments("01-Jan-2001", [30,30,30,365], {"warsaw","zurich",{1,2}}, "ACT/360","spbd",2)

4.19 payments2(dates,FCL,DCC,BDA,CPO)

Opis funkcji:

Funkcja ułatwiająca korzystanie z poprzednich procedur w przypadku ciągu płatności, w tej wersji pobierająca explicite listę z datami, działająca na nie date_rolling(), na końcu przesuwająca o valuedays dni roboczych do przodu.

Dane wejściowe:

dates - lista stringów interpretowanych jako daty (formatu "dd-mmm-yyyy")

FCL - cell array, identyfikator centrów finansowych

DCC - string, identyfikator konwencji liczenia dni

BDA - string, identyfikator konwencji płatności

CPO - integer nieujemny, liczba dni roboczych od contract date do value date

Dane wyjściowe:

lista stringów interpretowanych jako daty (formatu "dd-mmm-yyyy", dni robocze) wyznaczonych z listy wejściowej

lista liczb rzeczywistych, frakcji roku (w zadanej konwencji) pomiędzy sąsiednimi datami z poprzedniej listy

Przykład wywołania:

[data_list,frac_list] = payments2({"01-Jan-2001", "01-Jan-2002", "01-Jan-2003"}, {"warsaw"}, "ACT/360", "sfbd", 2)

4.20 payments3(date1,n,d,FCL,DCC,BDA,CPO)

Opis funkcji:

Modyfikacja payments(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora dni pobiera liczbę okresów oraz (taką samą) ilość dni w każdym z nich.

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

n - liczba naturalna, liczba okresów

d - liczba naturalna, liczba dni w pojedynczym okresie

FCL - cell array, identyfikator centrów finansowych

DCC - string, identyfikator konwencji liczenia dni

BDA - string, identyfikator konwencji płatności

CPO - integer nieujemny, liczba dni roboczych od contract date do value date

Dane wyjściowe:

lista n+1 stringów interpretowanych jako daty (formatu "dd-mmm-yyyy" , dni robocze) wyznaczone z danych wejściowych

lista n liczb rzeczywistych, frakcji roku (w zadanej konwencji) pomiędzy sąsiednimi datami z poprzedniej listy

Przykład wywołania:

[data_list,frac_list] = payments3("01-Jan-2001", 5, 30, {"warsaw", "zurich", {1}},"ACT/360","sfbd",2)

4.21 payments4(date1,mwect,FCL,DCC,BDA,CPO,EMA)

Opis funkcji:

Modyfikacja payments(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora dni pobiera wektor miesięcy. Przesunięcie następuje w rozumieniu zasady działania add_months().

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

mwect - wektor liczb naturalnych, ilość miesięcy w poszczególnych okresach

FCL - cell array, identyfikator centrów finansowych

DCC - string, identyfikator konwencji liczenia dni

BDA - string, identyfikator konwencji płatności

CPO - integer nieujemny, liczba dni roboczych od contract date do value date

EMA - parametr +/-1 ("w przód"/"w tył") do add_months()

Dane wyjściowe:

lista length(mwect)+1 stringów interpretowanych jako daty (formatu "dd-mmm-yyyy", dni robocze) wyznaczone z danych wejściowych

lista length(mwect) liczb rzeczywistych, frakcji roku (w zadanej konwencji) pomiędzy sąsiednimi datami z poprzedniej listy

Przykład wywołania:

[data_list,frac_list] = payments4("01-Jan-2001", [3,3,6,6], {"warsaw"}, "ACT/360", "sfbd",2,-1)

4.22 payments5(date1,n,m,FCL,DCC,BDA,CPO,EMA)

Opis funkcji:

Modyfikacja payments4(), różniąca się tylko sposobem przyjmowania parametrów - zamiast wektora miesięcy pobiera liczbę okresów oraz (taką samą) ilość miesięcy w każdym z nich. Przesunięcie następuje w rozumieniu zasady działania add_months().

Dane wejściowe:

date1 - string interpretowany jako data formatu "dd-mmm-yyyy" (data początkowa)

n - liczba naturalna, liczba okresów

m - liczba naturalna, liczba miesięcy w pojedynczym okresie

FCL - cell array, identyfikator centrów finansowych

DCC - string, identyfikator konwencji liczenia dni

BDA - string, identyfikator konwencji płatności

CPO - integer nieujemny, liczba dni roboczych od contract date do value date

EMA - parametr +/-1 ("w przód"/"w tył") do add_months()

Dane wyjściowe:

lista n+1 stringów interpretowanych jako daty (formatu "dd-mmm-yyyy", dni robocze) wyznaczone z danych wejściowych

lista n liczb rzeczywistych, frakcji roku (w zadanej konwencji) pomiędzy sąsiednimi datami z poprzedniej listy

Przykład wywołania:

[data_list,frac_list] = payments5("01-Jan-2001", 5, 6, {"warsaw"}, "ACT/360", "sfbd", 2, -1)