Michał Bernardelli

Kolegium Analiz Ekonomicznych
Szkoła Główna Handlowa w Warszawie

METODA SZYBKIEJ AKTUALIZACJI DEKOMPOZYCJI QR DLA MODELI LINIOWEJ REGRESJI

1. Wprowadzenie

Dane panelowe to często duże zbiory informacji. Wykonywanie obliczeń na takich zbiorach wiąże się z problemami, które nie są obserwowane dla mniejszej ilości danych. Stąd korzystanie z algorytmów bardziej wyrafinowanych, m.in. pod względem numerycznym, staje się koniecznością, jeśli chce się uzyskać wynik niewiele odbiegający od wyniku faktycznego. Jeszcze większe znaczenie ma zastosowanie odpowiednich algorytmów w przypadku zadań, dla których uzyskanie rozwiązania jest ściśle ograniczone czasowo.

Jedną z podstawowych technik modelowania ekonometrycznego jest regresja liniowa. Stanowi ona pierwsze – w wielu przypadkach mało dokładne, lecz szybkie – przybliżenie analizowanych danych. Używane od dziesięcioleci metody rozwiązywania zadania regresji liniowej opierają się m.in. na dekompozycji ortogonalno-trójkątnej QR macierzy. Dla metod tych wykazano ich dobre własności numeryczne[1], a istniejące implementacje tych algorytmów[2] są dopracowane pod kątem użycia pamięci i czasu. Mogą one jednak okazać się nieefektywne dla zbiorów danych, które szybko zmieniają się w czasie. Rozwiązanie mogą stanowić metody

[1] Patrz: J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, ed. 3rd, Springer, New York 2002.

[2] Patrz przede wszytkim: W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, ed. 3rd, Cambridge University Press, New York 2007.

dokonujące szybkiej aktualizacji dekompozycji QR macierzy zamiast wyznaczania jej każdorazowo.

Przykładem, w którym konieczna jest dokonywana w czasie rzeczywistym analiza danych, jest rynek walutowy Forex. Stanowi on doskonały test dokładności oraz szybkości działania różnych metod rozwiązywania zadania regresji liniowej. Symulacja komputerowa powinna przede wszystkim odzwierciedlać teoretyczne wyliczenia związane ze złożonością tych metod. Analizując wyniki, można i należy wziąć jednak pod uwagę również te czynniki, które w analizie teoretycznej zostały pominięte, jako mało znaczące lub zbyt trudne do uwzględnienia (takie jak specyficzna architektura pamięci komputera czy sposób działania systemu operacyjnego).

Na niniejsze opracowanie składa się sześć części. Po części wprowadzającej została przedstawiona krótka charakterystyka rynku kapitałowego Forex ze szczególnym uwzględnieniem dostępnych na tym rynku danych. Część trzecia zawiera podsumowanie informacji na temat liniowego zadania najmniejszych kwadratów oraz podstawowych metod jego rozwiązywania. Następnie opisano algorytmy szybkiej aktualizacji elementów Q i R dekompozycji macierzy: *qrdelete* oraz *qrinsert*. Dołączone zostały kody źródłowe tych algorytmów oraz analiza ich złożoności. Przedostatnia z części to opis symulacji komputerowej oraz jej wyników. Opracowanie jest zakończone podsumowaniem, w którym zostały wskazane kierunki dalszych badań.

2. Charakterystyka danych z rynku walutowego Forex

Forex (ang. *Foreign Exchange*) to największy międzynarodowy rynek wymiany walut. Ten zdecentralizowany rynek działa w każdy dzień tygodnia poza weekendami. Graczami na rynku Forex są zarówno banki, fundusze, klienci instytucjonalni, całe korporacje, jak i nawet rządy państw. Do największych graczy[3] na rynku należą m.in.: Deutsche Bank, Barclays Capital, UBS AG, Citigroup czy Goldman Sachs. Forex cieszy się jednak przede wszystkim niezwykłą popularnością wśród klientów indywidualnych. Wszystko to jest zasługą dwudziestoczterogodzinnego dostępu do rynku, zapewnionemu poprzez wiele dedykowanych systemów elektronicznych, do których dostępna jest łatwa obsługa za pośrednictwem stron internetowych. Natychmiastowa wymiana, tj. sprzedaż lub kupno, praktycznie dowolnej kwoty większości walut narodowych, a także np. srebra czy złota odbywa się poprzez zwykłe kliknięcie myszką lub częściej poprzez ustawienie opcji automatycznego zawierania transakcji przez dostępne albo napisane na zamówienie programy komputerowe.

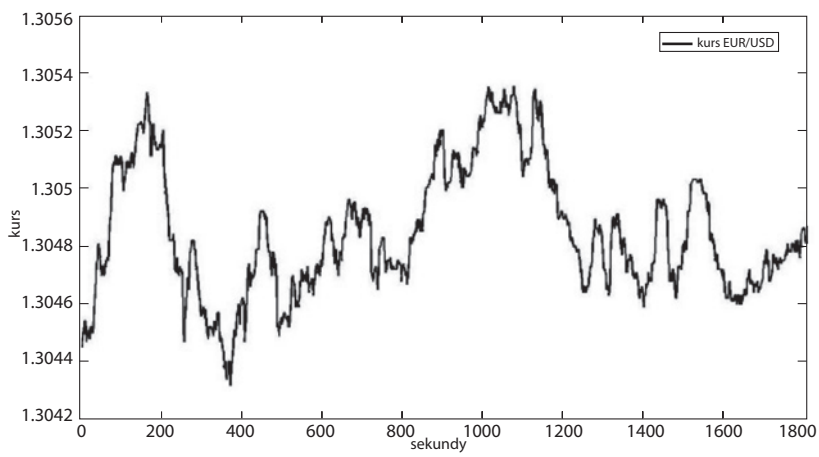
[3] http://www.euromoney.com/poll/3301/PollsAndAwards/Foreign-Exchange.html [dostęp 17.04.2012].

Najczęściej wymienianymi walutami są dolar amerykański (USD), euro (EUR) oraz japoński jen (JPY)[4].

Szybko rosnący popyt na usługi rynku Forex, przy niewielkim wzroście wiedzy grających na nim osób, zwiększa znacznie związane z transakcjami ryzyko utraty pieniędzy, jak również zaistnienia oszustw finansowych[5]. W Polsce instytucją nadzorującą działanie rynku Forex jest Komisja Nadzoru Finansowego[6].

Forex jest rynkiem dyskretnym. Aktualne kursy wymiany pomiędzy walutami obliczane są przez centralny system komputerowy, który uwzględnia wszystkie oferty kupna i sprzedaży w danej chwili. Każda zmiana ceny, zwana potocznie „tickiem", jest przekazywana do firm brokerskich grających na rynku Forex na całym świecie. Dane te są generowane w niewielkich odstępach czasowych – nie większych niż 2 sekundy. Na wielu stronach internetowych dostępne są archiwalne dane z rynku Forex[7] o różnej granulacji czasowej, np. 10 minut, 1 minuta, 10 sekund czy nawet 2 sekundy. Wykres dla przykładowych danych z rynku walutowego Forex przedstawiony jest na rysunku 1.

Ze względu na zmieniające się w tak szybkim tempie kursy wymiany walut praktycznie niemożliwe jest ręczne dokonywanie transakcji na rynku Forex. Niezbędne jest odpowiednie oprogramowanie, które w czasie rzeczywistym analizuje dane i prognozuje kursy wymiany walut. Ze względu na ścisłe ograniczenie czasowe istotny jest zatem rozwój efektywnych pod względem złożoności algorytmów analizy danych.



Rysunek 1. Wykres kursu EUR do USD dla jednej godziny z dnia 16.04.2012 r.

Źródło: opracowanie własne na podstawie: http://www.forexite.com.

4 http://www.bis.org/publ/rpfxf10t.htm [dostęp 17.04.2012].

5 http://www.knf.gov.pl/aktualnosci/2011/forex.html [dostęp 17.04.2012].

6 Patrz: http://www.knf.gov.pl [dostęp 17.04.2012].

7 Patrz np.: http://www.forexite.com [dostęp 17.04.2012].

3. Liniowe zadanie najmniejszych kwadratów i metody jego rozwiązywania

Jednym z podstawowych i najczęściej spotykanych zadań w analizie danych jest zadanie regresji liniowej, w którym estymacja parametrów strukturalnych modelu jest na ogół sprowadzana do liniowego zadania najmniejszych kwadratów (LZNK). Opis zadania najmniejszych kwadratów w wersji liniowej, jak również uogólnionej można znaleźć m.in. w książce Björcka[8]. LZNK polega na rozwiązaniu nadokreślonego układu równań z macierzą $A \in R^{m \times n}$, gdzie $m \geq n$, oraz z wektorem prawej strony $b \in R^m$, tj. na znalezieniu wektora $x^* \in R^n$ takiego, że

$$\left\| b - Ax^* \right\|_2 = \min_{x \in R^n} \left\| b - Ax \right\|_2,$$

gdzie $\left\| x \right\|_2 = \sqrt{x^T x}$ oznacza drugą normę wektorową.

Na przestrzeni lat powstało wiele metod rozwiązywania LZNK, jedna z nich to sprowadzenie LZNK do układu równań normalnych $A^T Ax = A^T b$ i rozwiązanie go[9] przy wykorzystaniu macierzy pseudoodwrotnej A^+:

$$x^* = A^+ b = (A^T A)^{-1} A^T b.$$

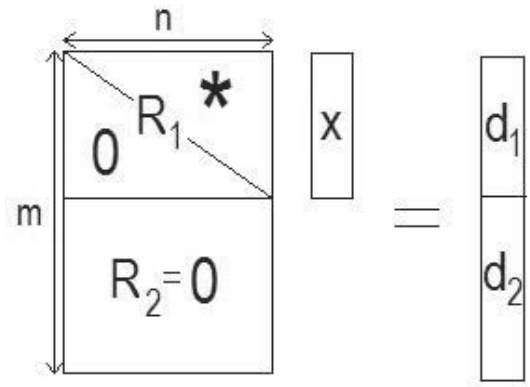
Ten sposób rozwiązywania LZNK nie jest polecany przede wszystkim ze względu na błędy numeryczne[10] i nie należy z niego korzystać w praktyce obliczeniowej.

Najbardziej rozpowszechnione metody rozwiązywania LZNK sprowadzają się do wykorzystania dekompozycji ortogonalno-trójkątnej QR macierzy A. Dekompozycja ta istnieje dla każdej macierzy wymiaru m na n o maksymalnym rzędzie. Macierz $Q \in R^{m \times m}$ jest macierzą ortogonalną, tj. macierzą, której kolumny są ortonormalne, $R \in R^{m \times n}$ zaś uogólnioną macierzą trójkątną górną, tj. taką, której elementy $r_{ij} = 0$ dla $i > j$. Mając daną dekompozycję QR, sprowadzamy zadanie do rozwiązania równania $Rx = Q^T b = d$. Ze względu na specjalną postać macierzy R (patrz rysunek 2) wymaga to tylko rozwiązania prostego układu równań liniowych $R_1 x = d_1$ z macierzą górną trójkątną wymiaru n.

[8] A. Björck, *Numerical methods for least squares problems*, SIAM, Philadelphia (USA) 1996.

[9] Rozwiązanie LZNK istnieje i jest jednoznaczne, jeżeli kolumny macierzy A są liniowo niezależne (rząd macierzy A jest równy n).

[10] Patrz: A. Kiełbasiński, H. Schwetlick, *Numeryczna algebra liniowa*, Wydawnictwa Naukowo-Techniczne, Warszawa 1992.



Rysunek 2. Układ równań LZNK z uwzględnieniem dekompozycji QR

Źródło: opracowanie własne.

Kwestią pozostającą do rozważenia jest znalezienie efektywnej pod względem złożoności obliczeniowej oraz poprawności numerycznej metody uzyskania dekompozycji ortogonalno-trójkątnej macierzy. Do podstawowych metod należą:

- ortogonalizacja Grama–Schmidta,
- odbicia Householdera,
- obroty Givensa.

Pierwsza z metod, a właściwie jej stabilna numerycznie wersja znana pod nazwą zmodyfikowanego algorytmu Grama–Schmidta, opisana m.in. przez Kincaida i Cheneya[11], jest używana rzadziej ze względu na jej gorszą złożoność obliczeniową. Pominiemy ją zatem w dalszych rozważaniach.

Druga z metod charakteryzuje się bardzo dobrymi własnościami numerycznymi, przy czym jej implementacje są efektywne pod względem czasowym i pamięciowym[12]. Opiera się ona na wykonaniu serii tzw. odbić Householdera, które przyporządkowują wektorowi jego lustrzane odbicie względem pewnej hiperpłaszczyzny. Stosując odbicia Householdera dla wektorów uzyskanych z kolejnych kolumn macierzy A, przeprowadzamy je na wektory postaci $[\alpha, 0, 0, \ldots, 0]$, dla pewnej stałej α. Dokładniej dla j-tej kolumny ($j = 1, 2, \ldots, n$) wybieramy wektor złożony z $m - j + 1$ ostatnich jej współrzędnych i taki wektor przekształcamy na $[r_{jj}, 0, 0, \ldots, 0]$. Tak obliczone wektory utworzą macierz:

[11] D. Kincaid, W. Cheney, *Analiza numerycza*, Wydawnictwa Naukowo-Techniczne, Warszawa 2006.

[12] Patrz: G.H. Golub, Ch.F. Van Loan, *Matrix Computations*, ed. 2nd, Johns Hopkins University Press, New York 1990.

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in R^{m \times m}.$$

Macierz Q wyznaczamy poprzez złożenie macierzy odbić Householdera[13]. Wzorując się na implementacji algorytmu dekompozycji QR metodą odbić Householdera za Trefethenem i Bau[14], podajemy kod tego algorytmu[15] zapisany w notacji programu Matlab[16].

```
1.  function [q,r] = householderQR(A,m,n)
2.          q = eye(m);
3.          r = A;
4.          for k=1:n
5.                  ak = r(k:m,k);
6.                  vk = ak + [sign(ak(1))*norm(ak); zeros(m–k,1)];
7.                  vk = vk/norm(vk);
8.                  r(k:m,k:n) = r(k:m,k:n) – 2*vk*vk'*r(k:m,k:n);
9.                  Hk = eye(m–k+1) – 2*vk*vk';
10.                 q = q*[eye(k–1) zeros(k–1,m–k+1); zeros(m–k+1,k–1) Hk];
11.         end
12. end
```

Złożoność efektywnej implementacji algorytmu[17] mierzony liczbą operacji zmiennoprzecinkowych to mniej więcej $2mn^2$. W przypadku, gdy chcemy otrzymać macierz Q *explicite* (linie kodu 9–10), koszt czasowy wzrasta drastycznie o czynnik rzędu nawet m^4. Przypadek n=2, który będzie rozważany w opracowaniu, jest jednak szczególny, gdyż mnożenie w linii 10 wykonywane jest tylko raz (dla $k = 2$), i w związku z tym koszt przedstawionego algorytmu to jedynie ok. $4m^2$ operacji zmiennoprzecinkowych[18].

[13] Odbicie Householdera nie zmienia długości wektora i jego macierz jest macierzą ortogonalną.

[14] L.N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia (USA) 1997.

[15] Przedstawiony kod nie stanowi efektywnej implementacji algorytmu dekompozycji QR metodą Householdera i został przytoczony dla czytelności opisu.

[16] Matlab jest programem firmy MathWorks, patrz: http://www.mathworks.com/products/matlab.

[17] Patrz np.: L.N. Trefethen, D. Bau, op.cit.

[18] Rozważane są dominujące operacje arytmetyczne, tj. dzielenie, mnożenie czy pierwiastkowanie z pominięciem różnic w szybkości obliczeń pomiędzy nimi.

Trzecia z metod uzyskania dekompozycji QR macierzy A wykorzystuje obroty, które są przykładem przekształceń unitarnych[19]. Obrót Givensa T_{ij} dla zadanego m-wymiarowego wektora zeruje jego j-tą współrzędną. Postać macierzy T_{ij} jest następująca:

$$T_{ij} = \begin{pmatrix} I & & & & \\ & c & & s & \\ & & I & & \\ & -s & & c & \\ & & & & I \end{pmatrix},$$

przy czym I to macierz jednostkowa, $c = cos(\varphi)$ oraz $s = sin(\varphi)$ dla pewnego kąta φ, wartości $\pm s$ zaś występują w macierzy T_{ij} na miejscach (i,j) oraz (j,i). Kod wyznaczający wartości c oraz s, definiujące obrót Givensa, można znaleźć np. w książce Dryi i Jankowskich[20]. Koszt tej, odpornej na błędy nadmiaru i niedomiaru, implementacji jest stały i równa się czterem operacjom zmiennoprzecinkowe[21]. Mnożąc kolumny a_i macierzy A dla $i = 1, 2, \ldots, n$ przez odpowiednie obroty Givensa $T_{i,i+1}$, $T_{i,i+2}$, …, $T_{i,n}$, zerujemy kolejno współrzędne $a_{i+1,i}$, $a_{i+2,i}$, …, $a_{n,i}$. Koszt tej metody dla pełnej macierzy A jest wyższy od metody z odbiciami Householdera[22]. Jest ona jednak szybsza dla macierzy rzadkich, tj. tych, które składają się z wielu zer.

4. Algorytmy aktualizacji dekompozycji QR

W sytuacji, gdy smamy do czynienia z koniecznością rozwiązywania wielu liniowych zadań najmniejszych kwadratów dla bardzo podobnych macierzy A oraz prawych stron b, nie jest konieczne każdorazowe wyznaczanie dekompozycji ortogonalno-trójkątnej. Istnieją algorytmy aktualizacji dekompozycji w zależności od różnic w macierzach wejściowych. Typowym przykładem jest ciąg napływających na bieżąco danych, do których należy dobrać najlepsze (w sensie najmniejszych kwadratów) estymatory parametrów regresji liniowej. Dwa kolejne modele różnią się wówczas tylko nowo odczytanym wierszem[23] macierzy A oraz wektora prawej

[19] Przekształcenia unitarne zachowują długość wektorów.

[20] M. Dryja, J. i M. Jankowscy, *Przegląd metod i algorytmów numerycznych*, cz. 2, Wydawnictwa Naukowo-Techniczne, Warszawa 1988.

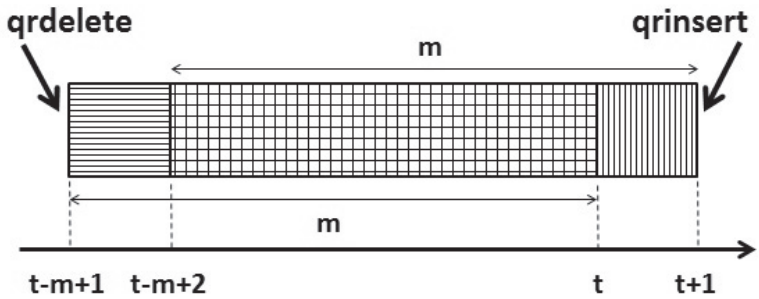
[21] A dokładniej jedno dzielenie, dwa mnożenia oraz obliczenie jednego pierwiastka kwadratowego.

[22] Istnieje pewna bardziej efektywna implementacja modyfikacja metody wykorzystującej obroty Givensa znana pod nazwą algorytmu Gentelmana.

[23] Analogicznie można rozpatrywać nowe kolumny zamiast wierszy.

strony *b*. Jeżeli dodatkowo chcemy ograniczyć się do danych z ostatnich *m* chwil, to należy kolejno (patrz rysunek 3):

- zaktualizować dekompozycję QR po usunięciu najstarszej obserwacji (procedura *qrdelete*),
- zaktualizować dekompozycję QR po dodaniu najnowszej obserwacji (procedura *qrinsert*).



Rysunek 3. Idea wykorzystania procedur *qrdelete* oraz *qrinsert*

Źródło: opracowanie własne.

Analiza złożoności autorskich kodów źródłowych obu procedur ma na celu porównanie czasu ich działania z czasem działania metod pełnej dekompozycji.

```
1.  function [q,r] = qrdelete(q,r,m,n)
2.      for j=m–1:–1:2
3.          [c, s] = givens(q(1,j), q(1,j+1));
4.          q(1,j) = c*q(1,j)–s*q(1,j+1);
5.          if (j<=n)
6.          r(j:j+1,j:n) = [c –s; s c]*r(j:j+1,j:n);
7.          end
8.          q(2:m,j:j+1) = q(2:m,j:j+1)*[c s; –s c];
9.      end
10.         % obliczenia dla pierwszego wiersza
11.     [c, s] = givens(q(1,1), q(1,2));
12.     q(1,1) = c*q(1,1)–s*q(1,2);
13.     r(1:2,1:n) = [c –s; s c]*r(1:2,1:n);
14.     q(2:m,2) = s*q(2:m,1)+c*q(2:m,2);
15.         % usunięcie pierwszych wierszy
16.     r = r (2:m,:);
17.     q = q(2:m,2:m);
18. end
```

Procedura *qrdelete* otrzymuje jako dane wejściowe macierze *Q* i *R* dekompozycji QR macierzy *A* w chwili *t*, jako dane wyjściowe zwraca natomiast macierze

$\widetilde{Q}$ i $\widetilde{R}$ dekompozycji QR macierzy A bez najstarszej obserwacji, czyli bez pierwszego wiersza. Wyodrębniając pierwsze wiersze macierze A, Q i R, można przedstawić zależność $A = QR$ w postaci:

$$\begin{pmatrix} a_1 \\ A_{reszta} \end{pmatrix} = \begin{pmatrix} q_1 \\ Q_{reszta} \end{pmatrix} \begin{pmatrix} r_1 \\ R_{reszta} \end{pmatrix}.$$

Następnie wystarczy przekształcić pierwszy wiersz macierzy Q na wektor jednostkowy e_1, wykorzystując serie obrotów Givensa

$$\begin{pmatrix} a_1 \\ A_{reszta} \end{pmatrix} = \begin{pmatrix} q_1 \\ Q_{reszta} \end{pmatrix} \underbrace{T_{m-1,m} T_{m-2,m-1} \ldots T_{3,4} T_{2,3} T_{1,2}}_{T} \underbrace{T_{1,2}^T T_{2,3}^T T_{3,4}^T \ldots T_{m-2,m-1}^T T_{m-1,m}^T}_{T^T} \begin{pmatrix} r_1 \\ R_{reszta} \end{pmatrix}$$

i otrzymać

$$\begin{pmatrix} a_1 \\ \widetilde{A} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} \begin{pmatrix} \widetilde{r}_1 \\ \widetilde{R} \end{pmatrix},$$

gdzie

$$\widetilde{A} = A_{reszta}, \qquad \begin{pmatrix} q_1 \\ Q_{reszta} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} T \qquad \text{i} \qquad \begin{pmatrix} \widetilde{r}_1 \\ \widetilde{R} \end{pmatrix} = T^T \begin{pmatrix} r_1 \\ R_{reszta} \end{pmatrix}.$$

Na podstawie zamieszczonego kodu procedury *qrdelete* zostanie wyznaczony koszt jej działania. W j-tej iteracji pętli *for* dla $j = 2, 3, \ldots, min(m - 1, n)$ koszt K_j jest równy:

$$K_j = 4 \text{ (linia 3)} + 2 \text{ (linia 4)} + 4^*(n - j + 1) \text{ (linia 6)} + 4^*(m - 1) \text{ (linia 8)}.$$

Dla j-tej iteracji pętli *for* dla $j = n, n + 1, \ldots, m - 1$ koszt K_j jest równy:

$$K_j = 4 \text{ (linia 3)} + 2 \text{ (linia 4)} + 4^*(m - 1) \text{ (linia 8)},$$

natomiast liczba operacji zmiennoprzecinkowych wykonywanych po zakończeniu działania pętli *for* wyraża się wzorem:

$$K_1 = 4 \text{ (linia 11)} + 2 \text{ (linia 12)} + 4^*n \text{ (linia 13)} + 4^*(m - 1) \text{ (linia 14)}.$$

Sumując po wszystkich j, otrzymujemy $4m^2 + 2n^2 - 2m + 6n - 2$ operacji zmiennoprzecinkowych, co dla $n = 2$ daje w przybliżeniu $4m^2$ operacji.

```
1.  function [q,r] = qrinsert(q,r,m,n)
2.          for j = 1:n
3.                  [c, s] = givens(r(j,j), r(m,j));
4.                          % aktualizacja macierzy r
5.                  r(j,j) = c*r(j,j) – s*r(m,j);
6.                  r(m,j) = 0;
7.                  tmp1 = r(j,j+1:n);
8.                  tmp2 = r(m,j+1:n);
9.                  r(j,j+1:n) = c*tmp1 – s*tmp2;
10.                 r(m,j+1:n) = s*tmp1 + c*tmp2;
11.                         % aktualizacja macierzy q
12.                 tmp1 = q(1:m,j);
13.                 tmp2 = q(1:m,m);
14.                 q(1:m,j) = c*tmp1 – s*tmp2;
15.                 q(1:m,m) = s*tmp1 + c*tmp2;
16.         end
17. end
```

Procedura *qrinsert* otrzymuje jako dane wejściowe macierze Q i R dekompozycji QR macierzy A w chwili t oraz dane z chwili $t + 1$, czyli nowy wiersz r_{t+1} macierzy A, jako dane wyjściowe zwraca natomiast macierze $\widetilde{Q}$ i $\widetilde{R}$ dekompozycji QR macierzy A w chwili $t + 1$. Idea wyznaczenia aktualizacji dekompozycji QR polega na wykorzystaniu zależności $A = QR$ przedstawionej w postaci:

$$\begin{pmatrix} A \\ r_{t+1} \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R \\ r_{t+1} \end{pmatrix},$$

a następnie wyzerowaniu ostatniego wiersza r_{t+1} przy wykorzystaniu obrotów Givensa:

$$\begin{pmatrix} A \\ r_{t+1} \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} \underbrace{T_{1,m+1} T_{2,m+1} \ldots T_{n-1,m+1} T_{n,m+1}}_{T} \underbrace{T^T_{n,m+1} T^T_{n-1,m+1} \ldots T^T_{2,m+1} T^T_{1,m+1}}_{T^T} \begin{pmatrix} R \\ r_{t+1} \end{pmatrix} = \widetilde{Q}\widetilde{R},$$

gdzie

$$\widetilde{Q} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} T \qquad \text{i} \qquad \widetilde{R} = T^T \begin{pmatrix} R \\ r_{t+1} \end{pmatrix}.$$

Na podstawie zamieszczonego kodu procedury *qrinsert* zostanie wyznaczony koszt jej działania. W j-tej iteracji pętli *for* dla $j = 1, 2, \ldots, n$ koszt K_j jest równy:

K_j = 4 (linia 3) + 2 (linia 5) + 4*($n - j + 1$) (linie 9–10) + 4*m (linie 14–15).

Sumując po wszystkich j, otrzymujemy $2n^2 + 4mn + 8n$ operacji zmiennoprzecinkowych, co dla $n = 2$ daje około $8m$ operacji.

Porównując czasy działania dekompozycji QR dla $n = 2$ metodą odbić Householdera oraz z użyciem procedur *qrdelete* i *qrinsert*, otrzymujemy wyrażenia tego samego rzędu, tj. m^2 z tą samą stałą przy najwyższej potędze. W celu określenia zależności szybkości działania od czynników niższych rzędów można policzyć osobno każdą z operacji zmiennoprzecinkowych: dodawanie, odejmowanie, mnożenie, dzielenie, pierwiastkowanie[24]. Niestety, szybkość działania każdego z tych działań arytmetycznych zależy od budowy konkretnego procesora/procesorów, stąd takie porównanie mogłoby nie być miarodajne. Rozsądne jednak wydaje się założenie, iż procesory współczesnych komputerów osobistych nie różnią się diametralnie, jeżeli chodzi o obsługę podstawowych operacji arytmetycznych. Ważniejszą kwestią okazuje się jednak wpływ pozostałych działań, które w przypadku wyraźnych różnic w liczbie operacji zmiennoprzecinkowych były pomijane. Chodzi tutaj o takie rzeczy, jak: przypisanie, kopiowanie zmiennych, zwracanie wartości funkcji czy organizacja i alokacja pamięci. Czas wymagany przez te dodatkowe i zwykle pomijane przy badaniu złożoności operacje zależy od tak wielu czynników, iż w praktyce jest niemożliwy do policzenia. Odpowiedź na pytanie o porównanie szybkości działania obu metod może dać natomiast symulacja komputerowa.

5. Wyniki symulacji komputerowej

Symulacja komputerowa została przeprowadzona w oparciu o przedstawione kody źródłowe na danych z rynku Forex w granulacji dwusekundowej. Wykorzystane zostały kursy walutowe euro do franka szwajcarskiego (EUR/CHF), euro do korony norweskiej (EUR/NOK), euro do dolara amerykańskiego (EUR/USD), srebra do dolara amerykańskiego (AG/USD) oraz złota do dolara amerykańskiego (AU/USD). Dla liczb wierszy m macierzy regresji liniowej w zakresie od 5 do 1000 oraz kolejnych 25 000 danych każdego z szeregów czasowych wyznaczone zostały:

- dekompozycje QR metodą odbić Householdera (metoda M_1),
- kolejne aktualizacje dekompozycji, tj. *qrdelete* i *qrinsert* (metoda M_2).

Uśrednione dla wybranych wartości m zmierzone czasy obliczeń zostały przedstawione w tabeli 1. Warto podkreślić, iż jeśli chcemy oprzeć strategię gry na rynku Forex na większej liczbie prognoz z regresji liniowej, suma czasów koniecznych do ich wyznaczenia nie może w tym przypadku przekraczać dwusekundowego przedziału. W przypadku obliczeń tylko dla rozważanych pięciu szeregów czasowych dla m większych od ok. 700 uzyskano w przypadku metody M_1 czasy niespełniające tego ograniczenia.

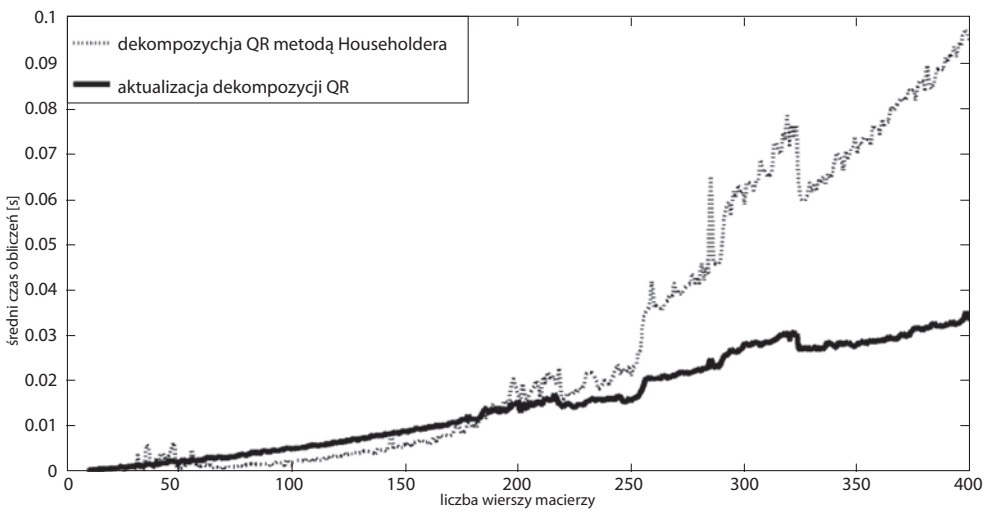
[24] Jeśli wziąć pod uwagę najwolniejsze z działań, tj. pierwiastkowanie teoretyczne, wyliczenia wypadają na korzyść algorytmu z aktualizacją dekompozycji QR.

Tabela 1. Średnie czasy działania metody odbić Householdera (M_1) oraz metody aktualizacji dekompozycji QR (M_2) w zależności od liczby danych (m)

m / *dane*	10		100		200		300	
	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
EUR/CHF	.0002	.0006	.0091	.0056	.0250	.0117	.1122	.0333
EUR/NOK	.0002	.0006	.0032	.0060	.0108	.0114	.0465	.0238
EUR/USD	.0002	.0005	.0016	.0047	.0156	.0107	.0425	.0216
AG/USD	.0002	.0006	.0021	.0054	.0092	.0114	.0410	.0230
AU/USD	.0002	.0006	.0023	.0047	.0119	.0111	.0402	.0217
SUMA	.0010	.0029	.0183	.0264	.0725	.0563	.2824	.1234
ŚREDNIA	.0002	.0006	.0037	.0053	.0145	.0113	.0565	.0247

Źródło: opracowanie własne.

Na rysunku 4 została pokazana większa szybkość metody aktualizacji dekompozycji QR (metoda M_2) w porównaniu z pełną dekompozycją (metoda M_1) dla większych od ok. 250 wartości *m*.



Rysunek 4. Porównanie średnich czasów działania metod M_1 i M_2 w zależności od liczby wierszy macierzy LZNK

Źródło: opracowanie własne.

6. Podsumowanie i kierunki dalszych badań

Analiza teoretyczna badanych algorytmów wyznaczania dekompozycji QR wraz z przeprowadzonymi symulacjami komputerowymi prowadzą do następujących wniosków:

- szybkość działania metody M_1 dla zadań o niewielkich wymiarach ($m < 200$) jest wyższa od metody M_2; dla prognoz opartych na regresji liniowej małej liczby danych lepiej jest wykonać każdorazowo dekompozycję QR metodą odbić Householdera,
- dla większych rozmiarów liniowych zadań najmniejszych kwadratów efektywniejsza jest metoda wykorzystująca aktualizację istniejącej dekompozycji QR,
- decydujący koszt czasowy metody M_2 jest związany przede wszystkim z liczbą operacji zmiennoprzecinkowych algorytmu *qrdelete*; przewaga szybkości metody prognozowania opartej tylko na dodawaniu nowych danych bez usuwania starych nad metodą M_1 jest jeszcze większa.

Algorytmy opisane w niniejszym opracowaniu należy traktować jako podstawę do budowy bardziej wyrafinowanych metod prognozowania zmian kursów walutowych na rynku Forex. Z pewnością można rozważyć następujące uogólnienia i modyfikacje:

- rozszerzenie na modele liniowej regresji dla wielomianów wyższych stopni oraz dla większej liczby zmiennych,
- rozpatrzenie przypadku regresji nieliniowej,
- zastosowanie metod, które nie wymagają obliczania *explicite* macierzy Q oraz porównanie ich szybkości z szybkością działania metod M_1 i M_2.

Bibliografia

Björck A., *Numerical methods for least squares problems*, SIAM, Philadelphia (USA) 1996.

Dryja M., Jankowscy J. i M., *Przegląd metod i algorytmów numerycznych*, cz. 2, Wydawnictwa Naukowo-Techniczne, Warszawa 1988.

Golub G.H., Van Loan Ch.F., *Matrix Computations*, ed. 2nd, Johns Hopkins University Press, New York 1990.

Kiełbasiński A., Schwetlick H., *Numeryczna algebra liniowa*, Wydawnictwa Naukowo-Techniczne, Warszawa 1992.

Kincaid D., Cheney W., *Analiza numerycza*, Wydawnictwa Naukowo-Techniczne, Warszawa 2006.

Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P., *Numerical Recipes: The Art of Scientific Computing*, ed. 3rd, Cambridge University Press, New York 2007.

Stoer J., Bulirsch R., *Introduction to Numerical Analysis*, ed. 3rd, Springer, New York 2002

Trefethen L.N., Bau D., *Numerical Linear Algebra*, SIAM, Philadelphia (USA) 1997.

Źródła sieciowe

http://www.bis.org/publ/rpfxf10t.htm [dostęp 17.04.2012].

http://www.euromoney.com/poll/3301/PollsAndAwards/Foreign-Exchange.html [dostęp 17.04.2012].

http://www.forexite.com [dostęp 17.04.2012].

http://www.knf.gov.pl/aktualnosci/2011/forex.html [dostęp 17.04.2012].

Summary

Method of QR decomposition's fast updates for linear regression models

In the paper the description of algorithms of solving the linear least square problem with QR decomposition method and their analysis of usefulness to the fast changing panel data, is presented. The classical algorithm of QR decomposition using Householder reflections and the algorithm of fast updates of QR decomposition using Givens rotations are compared. Results of the computer simulation for the data from the foreign exchange market are presented.

Keywords: Forex, algorithm, linear least square problem, QR decomposition, Givens rotation, Householder reflection, QR factorization update

JEL Classification: C63, C23