Pracownia z analizy numerycznej

Sprawozdanie do zadania **P2.20.**
Sprawdzający: dr Witold Karczewski

Aleksander Balicki nr indeksu: 220989
Dominika Rogozińska nr indeksu: 221094

Wrocław, 3 grudnia 2010 r.

1. Wstęp

Definicja 1. *Macierz o wymiarach m na n (macierzą o m wierszach i n kolumnach), nad ciałem K nazywamy każdą funkcję typu* $\{1, \cdots, m\} \times \{1, \cdots, n\} \to K$.

Macierz zapisujemy tak:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

Rząd macierzy definiujemy [1] definiujemy jako:

Definicja 2. *Rzędem macierzy nazywamy wymiar przestrzeni generowanej przez jej wiersze.*

Alternatywna definicja [6]:

Definicja 3. *Rzęd macierzy to maksymalna liczba liniowo niezależnych wektorów tworzących kolumny danej macierzy.*

W powyższych definicjach możemy używać wierszy i kolum naprzemiennie, ponieważ wiemy [1], że:

Twierdzenie 1. *Rząd macierzy A jest równy rzędowi macierzy transponowanej do A.*

Obliczanie rzędu macierzy przydaje się do sprawdzania własności układów równań. Dzięki rzędowi macierzy możemy dowiedzieć się czegoś o ilości rozwiązań takiego układu. Kolejnym zastosowaniem jest sprawdzanie niezależności danych współczynników, co może być przydatne np. w statystyce.

Weźmy przykładowy układ równań:

$$\begin{cases} a_{1,1}x_1 & + & a_{1,2}x_2 & + & \cdots & + & a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 & + & a_{2,2}x_2 & + & \cdots & + & a_{2,n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ a_{m,1}x_1 & + & a_{m,2}x_2 & + & \cdots & + & a_{m,n}x_n & = & b_n \end{cases}$$

Z tym układem wiążemy macierz układu - A:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

oraz macierz rozszerzoną układu - A':

$$A' = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & b_m \end{bmatrix}$$

Z tymi macierzami związane są pewne twierdzenia [1].

Twierdzenie 2 (Kronecker-Capelli)**.** *Układ równamń liniowych ma rozwiązanie wtedy i tylko wtedy, gdy rząd jego macierzy jest równy rzędowi jego macierzy rozszerzonej.*

Twierdzenie 3. *Układ ma dokładnie jedno rozwiązanie, gdy rzędy macierzy i macierzy rozszerzonej są równe liczbie niewiadomych.*

2. Metoda eliminacji Gaussa

Metoda ta została obmyślona przez Carla Friedricha Gaussa. Daje ona algorytm do rozwiązania układu równań liniowych, obliczenia rzędu macierzy i znalezienia macierzy odwrotnej do danej. Algorytm składa się z 2 kroków, najpierw doprowadzamy macierz do postaci schodkowej, a potem znajdujemy wynik układu poprzez podstawienie w tył. W metodzie Gaussa stososuje się 3 operacje elementarne na wierszach macierzy. Te operacje to:

1. Zamiana kolejności wierszy
2. Pomnożenie wszystkich wartości w wierszu przez niezerowy skalar λ
3. Dodanie do dowolnego wiersza kombinacji liniowej pozostałych wierszy

Operacje elementarne mają ciekawe własności, mianowicie:

- Nie zmieniają rzędu macierzy
- Dowolną macierz można za pomocą skończonej liczby kroków doprowadzić do macierzy w postaci schodkowej

Rząd macierzy to liczba jej wierszy w postaci schodkowej, więc metoda obliczenia rzędu będzie polegać na sprowadzeniu macierzy do postaci schodkowej, a potem policzeniu jej niezerowych wierszy. Dla omówionego wyżej przypadku algorytm dla macierzy M o rozmiarze $N \times N$ wyglądałby tak:

```
for k from 1 to N-1
    for i from k+1 to N
        odejmij od i-tego wiersza k-ty wiersz (M[i,k]/M[k,k]) razy
zwróć ilość niezerowych wierszy
```

W tak naiwnie zaimplementowanym algorytmie widać od razu, gdzie możemy napotkać błędy. Jeżeli istnieje takie n, że $M_{n,n} = 0$, to występuje dzielenie przez zero. Aby rozwiązać ten problem po prostu zamieniamy wiersz z zerem na pierwszy wiersz z niezerową wartością. Jeżeli każda wartość w kolumnie jest zerowa, to omijamy kolumnę. Metoda ta nazywa się eliminacją Gaussa z wyborem elementów głównych. Po zmodyfikowaniu naszego algorytmu, aby radził sobie z dzieleniem przez zero, rozważmy następującą macierz:

$$Z = \begin{bmatrix} 0 & 5 & 23 \\ 0.0000000000001 & 20 & 55 \\ 45 & 3 & 5 \end{bmatrix}$$

Nasz algorytm zauważy zero na pozycji $Z_{1,1}$. Zamieni pierwszy wiersz z drugim. Uzna, że wartość $Z_{1,1}$ jest niezerowa i przejdzie do następnego kroku. W trakcie algorytmu wykona się dzielenie 45/0.0000000000001. Wiemy, z [2], że przy dzieleniu przez liczby bliskie zeru następuje utrata cyfr dokładnych wyniku. Algorytm zachowywałby się zdecydowanie lepiej, jeżeli dzielilibyśmy przez większą liczbę. Algorytm wyboru elementu głównego w kolumnie:

```
for k from 1 to N-1
    znajdz wiersz, w którym jest maksymalna wartość |M[i,k]| dla wszystkich i > k
    zamień wiersz z tą maksymalną wartością z k-tym wierszem
    for i from k+1 to N
        odejmij od i-tego wiersza k-ty wiersz (M[i,k]/M[k,k]) razy
zwróć ilość niezerowych wierszy
```

Można też szukać elementu o największym module w całej podmacierzy, wydłuża to czas obliczeń, ale poprawia własności numeryczne. Algorytm z wyborem pełnym:

```
for k from 1 to N-1
    znajdz i,l takie, że |M[i,l]| jest maksymalne dla wszystkich i > k, l > k
    zamień l-tą kolumnę z k-tą kolumną
    zamień i-ty wiersze z k-tym wierszem
    zamień wiersz z tą maksymalną wartością z k-tym wierszem
    for i from k+1 to N
        odejmij od i-tego wiersza k-ty wiersz (M[i,k]/M[k,k]) razy
zwróć ilość niezerowych wierszy
```

3. Rozkład QR

W tej metodzie rozkładamy macierz A na

$$A = QR \tag{3.0.1}$$

gdzie Q to macierz ortogonalna, czyli $Q^TQ = I$, gdzie I to macierz identycznościowa, a R to macierz górnotrójkątna. Rząd macierzy w rozkładzie QR to liczba różnych rzędów w macierzy Q.

3.1. Ortogonalizacja Grama-Schmidta

Rozważmy najpierw metodę ortogonalizacji Grama-Schmidta na kolumnach macierzy $A = [a_1, \cdots, a_n]$, z działaniem zdefiniowanym tak $< v, w >= v^T w$. Zdefiniujmy projekcję:

$$proj_e a = \frac{< e, a >}{< e, e >} e \tag{3.1.2}$$

Wtedy:

$$\begin{aligned}
u_1 &= a_1 \\
e_1 &= \frac{u_1}{||u_1||} \\
u_2 &= a_2 - proj_{e_1} a_2 \\
e_2 &= \frac{u_2}{||u_2||} \\
u_3 &= a_3 - proj_{e_1} a_3 - proj_{e_2} a_3 \\
e_3 &= \frac{u_3}{||u_3||} \\
&\vdots
\end{aligned}$$

$$
\begin{aligned}
u_k &= a_k - \sum_{j=1}^{k-1} proj_{e_k} a_k \\
e_k &= \frac{u_k}{||u_k||}
\end{aligned}
$$

Teraz przekształćmy równania:

$$
\begin{aligned}
a_1 &= < e_1, a_1 > e_1 \\
a_2 &= < e_1, a_2 > e_1 + < e_2, a_2 > e_2 \\
&\vdots \\
a_k &= \sum_{j=1}^{k} < e_j, ak > e_j
\end{aligned}
$$

gdzie $< e_i, a_i >= ||u_i||$ Można to zapisać jako macierz $A = QR$:

$$
Q = [e_1, \cdots, e_n] \text{ i } R = \begin{bmatrix} < e_1, a_1 > & < e_1, a_2 > & < e_1, a_3 > & \cdots \\ 0 & < e_2, a_2 > & < e_2, a_3 > & \cdots \\ 0 & 0 & < e_3, a_3 > & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.1.3}
$$

Podczas obliczeń u_k występuje dużo odejmowań, przez co macierz R może nie być dokładnie policzona. Aby poprawić numeryczną stabilność wprowadzamy usprawnienia przedstawione przez Schwarza i Rutishausera. Zamiast $u_k = a_k - \sum_{j=1}^{k-1} proj_{e_k} a_k$, wprowadzamy taki proces:

$$
\begin{aligned}
u_k^{(1)} &= u_k \\
u_k^{(j)} &= u_k^{(j-1)} - proj_{e_{j-1}} u_k^{(j-1)}
\end{aligned}
$$

3.2. Odbicia Householdera

Innym podejściem do rozkładu QR jest użycie odbić Householdera. Odbicie Householdera to przekształcenie opisujące odbicie o płaszczyznę lub hiperpłaszczyznę.

Dla danego wektora w^m o normie $||w||_2$, odbicia Householdera zdefiniowane jest jako macierz:

$$
H = I - 2ww^T \tag{3.2.4}
$$

Zauważamy, że spełnia ona warunek:

$$
H_x = x - 2(w^T x)w \tag{3.2.5}
$$

więc nie trzeba tej macierzy przy obliczeniach trzymać w pamięci.

Odpowiednia macierz Householdera ma taką własność, że jeżeli pomnożymy ją przez naszą macierz, zeruje ona elementy w danej kolumnie poniżej wybranego. Elementy zerują się, ponieważ znalezliśmy odpowiednią hiperpłaszczyznę, od której wykonujemy odbicie. Dla pierwszej kolumny macierzy A znajdujemy taką macierz Householdera Q_1, że zeruje ona wszystkie elementy poniżej pierwszego. Znajdujemy odpowiednie macierze aż do Q_{n-1}. Więc po złożeniu wszystkich macierzy w postaci Q_k z macierzą A otrzymujemy nasze R.

$$
R = Q_{n-1} \cdots Q_1 A \tag{3.2.6}
$$

Wiemy, że macierze Householdera są ortogonalne, więc:

$$
Q = (Q_{n-1} \cdots Q_1)^{-1} = Q_1 \cdots Q_{n-1} \tag{3.2.7}
$$

4. Program

Program testujący jest napisany w języku C++. Użyto typu podwójnej precyzji (double). Do generowania losowych macierzy wymyśliłem algorytm.

```
Pobierz ilość_kolumn, ilość_rzędów, rząd
ilość_liniowo_zależnych = rząd - min( ilość_kolumn, ilość_rzędów)
Losujemy 0-rowy rząd macierzy
Dla r = 1 do ilość_rzędów
    Przepisz rząd zerowy do rzędu r
    Jeśli( ilość_liniowo_zależnych > 0 )
        ilość_liniowo_zależnych--
    jeśli nie
        zastąp M[r][c] losową inną liczbą
    Pomnóż rząd przez losowy skalar
```

Tym sposobem wygenerujemy odpowiednią liczbę liniowo zależnych wierszy, a pozostałe na pewno będą niezależne.

5. Wnioski

Metoda Gaussa z wyborem elementów z podmacierzy jest lepsza, niż z wyborem z kolumny. Jest zwykle bliższa wyniku, ale oba te algorytmy często źle obliczają rząd. Metoda QR z ortogonalizacją Grama-Schmidta jest gorsza od obu wariantów metod Gaussa.

Literatura

[1] Notatki z wykładu Algebra Emanuela Kierońskiego

[2] Notatki z wykładu Analiza Numeryczna Stanisława Lewanowicza

[3] Kincaid David, Cheney Ward, Analiza numeryczna

[4] J. M. Jankowscy, Przegląd metod i algorytmów numerycznych

[5] http://wolframalpha.com/

[6] http://en.wikipedia.org/

[7] http://wazniak.mimuw.edu.pl/