Tomasz Indeka

Metody numeryczne

Zadanie 2.18– Sprawozdanie

**Pkt 1. Obliczanie wartości własnych macierzy nieosobliwych metodą QR**

Do obliczenia wartości własnych macierzy najpierw wyznaczyłem rozkład QR wg wzorów:

A później znormalizowałem:

Do obliczenia wartości własnych macierzy symetrycznych metodą bez przesunięć użyłem wzorów

I iterowałem aż do momentu kiedy wszystkie elementy macierzy A oprócz tych na diagonali były mniejsze od przyjętej tolerancji, osobiście przyjąłem 0,00001 lub do momentu przekroczenia maksymalnej ilości iteracji, którą wyznaczyłem na 1.000.000.

Do obliczenia wartości własnych macierzy symetrycznych i niesymetrycznych metodą z przesunięciami użyłem wzorów

Do wyznaczenia wartości własnej podmacierzy 2x2 z prawego dolnego rogu macierzy A

)

I jako wartość pk przyjąłem wartość x bliższą do wartości ak,k. Następnie użyłem wzorów:

Powtarzałem do momentu kiedy największa wartość bezwzględna (oprócz tej na diagonali) w k-tym wierszu będzie mniejsza od założonej tolerancji (0,00001). Wtedy zmniejszałem macierz A o ostatni wiersz i kolumnę i powtarzałem algorytm do momentu dojścia do pierwszego elementu macierzy.

Dla rozwiązanych macierzy porównałem wyniki z funkcją i wyniki były takie same z dokładnością do kolejności i do 4 miejsc po przecinku.

Średnia ilość iteracji dla różnych metod

|  |  |  |  |
| --- | --- | --- | --- |
| Wymiar macierzy | Symetryczna bez przesunięć | Symetryczna z przesunięciami | Dowolna |
| 5x5 | 45.8333 | 7.6667 | 11 |
|  | 280.0667 | 7.9667 | 10 |
|  | 74.7333 | 7.8000 | 10.1667 |
| 10x10 | 217.8000 | 16.3000 | 27.4667 |
|  | 1.8496e+03 | 16.0667 | 25 |
|  | 1.9667e+03 | 16.3667 | 25.9333 |
| 20x20 | 1.4409e+03 | 32.5667 | 57.6667 |
|  | 3.1137e+03 | 33.1333 | 58.6000 |
|  | 1.5847e+03 | 33.1333 | 57.5000 |

Zadanie 1 pkt a)

%

%Tomasz Indeka

%MNUM-Projekt 2.18

%zadanie 1

%

%Oblicznie wektorów własnych macierzy nieosobliwych metodą QR

%macierz symetryczna bez przesunięć

%

% A-macierz wejściowa

% n-wymiar macierzy

% s-wartość średniej liczby iteracji

% v-wektor wartości własnych macierzy A

function s = zad1a(n)

k=1;

suma=0;

while k<=30

A = rand(n);

A = A + A';

[i,v]=qr\_b\_prz (A,n);

suma = suma+i;

k=k+1;

end

s=suma/30;

end

Zadanie 1 pkt b)

%

%Tomasz Indeka

%MNUM-Projekt 2.18

%zadanie 1

%

%Oblicznie wektorów własnych macierzy nieosobliwych metodą QR

%macierz symetryczna z przesunięciami

%

% A-macierz wejściowa

% n-wymiar macierzy

% s-wartość średniej liczby iteracji

% v-wektor wartości własnych macierzy A

function s = zad1b(n)

k=1;

suma=0;

while k<=30

A = rand(n);

A = A + A';

[i,v]=qr\_prz (A,n);

suma = suma+i;

k=k+1;

end

s=suma/30;

end

Zadanie 1 pkt c)

%

%Tomasz Indeka

%MNUM-Projekt 2.18

%zadanie 1

%

%Oblicznie wektorów własnych macierzy nieosobliwych metodą QR

%macierz dowolna z przesunięciami

%

% A-macierz wejściowa

% n-wymiar macierzy

% s-wartość średniej liczby iteracji

% v-wektor wartości własnych macierzy A

function s = zad1c(n)

k=1;

suma=0;

while k<=30

A = rand(n);

[i,v]=qr\_prz (A,n);

suma = suma+i;

k=k+1;

end

s=suma/30;

end

Rozkład QR

function [Q,R] = qr\_moj(A,n)

% A-macierz wejściowa do rozkładu QR

% n-wymiar macierzy

% Q-macierz Q rozkładu QR

% R-macierz R rozkładu QR

% w-mnożnik do normalizacji macierzy QR

j=1;

R=zeros(n);

while j<=n % rozkład qr macierzy A

k=1;

r=zeros(n,1);

R(j,j)=1;

while k<j

R(k,j)=(Q(:,k)'\*A(:,j))/(Q(:,k)'\*Q(:,k));

r(:,1)=r(:,1)+R(k,j)\*Q(:,k);

k=k+1;

end

Q(:,j)=A(:,j)-r(:,1);

j=j+1;

end

j=1;

while j<=n % normalizacja macierzy qr

w=0;

k=1;

while k<=n

w=w+Q(k,j)^2;

k=k+1;

end

w=sqrt(w);

k=1;

while k<=n

Q(k,j)=Q(k,j)/w;

R(j,k)=R(j,k)\*w;

k=k+1;

end

j=j+1;

end

end

Metoda QR bez przesunięć

function [i,v] = qr\_b\_prz(A,n)

% A-macierz wejściowa do rozkładu QR

% n-wymiar macierzy

% Q-macierz Q rozkładu QR

% R-macierz R rozkładu QR

% v-wektor wartości własnych macierzy A

i=0;

while i<=1000000

[Q,R]=qr\_moj(A,n);

A =R\*Q;

w=1;

j=1;

while j<=n %sprawdzanie warunku dokładności wyniku

k=1;

while k<n

if j~=k

if A(j,k)>0.00001

w=0;

end

end

k=k+1;

end

j=j+1;

end

if w==1

k=1;

while k<=n

v(k)=A(k,k);

k=k+1;

end

break

end

i=i+1;

end

end

Metoda QR z przesunięciami

function [i,v] = qr\_prz(A,n)

% A-macierz wejściowa do rozkładu QR

% n-wymiar macierzy

% Q-macierz Q rozkładu QR

% R-macierz R rozkładu QR

% v-wektor wartości własnych macierzy A

% b-ślad po podmacierzy 2x2 macierzy A

% c-wyznacznik podmacierzy 2x2 macierzy A

% d-delta równania l^2+bl+c=0

% x1,2 = pierwiastki równania l^2+bl+c=0

% e-wartość pk

v=0;

k=n;

i=0;

while k>1

while i<1000 & max(abs(A(k,1:k-1)))>0.00001

b=-(A(k,k)+A(k-1,k-1));

c=(A(k,k)\*A(k-1,k-1))-(A(k,k-1)\*A(k-1,k));

d=b^2-4\*c;

x1=(-b-sqrt(d))/2;

x2=(-b+sqrt(d))/2;

if (x1-A(k,k)) < (x2-A(k,k))

e = x1;

else

e = x2;

end

A=A-e\*eye(k);

[Q,R]=qr\_moj(A,k);

A = R\*Q+e\*eye(k);

i=i+1;

end

v(k)=A(k,k);

A = A(1:k-1,1:k-1);

k=k-1;

end

v(k)=A(k,k);

end

**Pkt 2. Aproksymacja danych do funkcji wielomianowych**

Do rozwiązania problemu użyłem następujących wzorów:

, gdzie a jest wektorem stałych przy kolejnych potęgach x

, gdzie n jest równe rozmiarowi danych wejściowych

Dane wejściowe:

|  |  |
| --- | --- |
| xi | yi |
| -5 | -14,2376 |
| -4 | -7,7256 |
| -3 | -4,1949 |
| -2 | -2,4815 |
| -1 | -1,2683 |
| 0 | -1,7885 |
| 1 | -1,7269 |
| 2 | -3,3830 |
| 3 | -8,9977 |
| 4 | -21,3130 |
| 5 | -42,6544 |

Wielomian przybliżający nie powinien mieć wyższej lub równej potęgi niż wejściowa ilość danych. Dlatego dla tych danych maksymalna potęga x wynosi 10.

Tabela norm dla obliczonych stopni wielomianów

|  |  |  |
| --- | --- | --- |
| Maksymalna potęga x w wielomianie | Typ normy | |
| Czebyszewa | Euklidesowa |
| 0 | 32.6752 | 39.5544 |
| 1 | 22.9887 | 33.9368 |
| 2 | 6.6532 | 11.5811 |
| 3 | 2.6996 | 6.2604 |
| 4 | 0.5208 | 0.8837 |
| 5 | 0.5008 | 0.8117 |
| 6 | 0.4669 | 0.6298 |
| 7 | 0.3139 | 0.5013 |
| 8 | 0.2975 | 0.4627 |
| 9 | 0.2254 | 0.3845 |
| 10 | 1.0027e-08 | 1.2402e-08 |

Jak można zauważyć najdokładniejszą funkcję otrzymaliśmy w przypadku zastosowania wielomianu 10 stopnia. Jak można zauważyć nie oddaje on jednak całkowicie funkcji jakiej mogliśmy się na początku spodziewać, ponieważ posiada ekstremum lokalne pomiędzy x=-5, a x=-4, czego nie jest się w stanie przewidzieć patrząc na rozkład punktowy funkcji aproksymowanej. Dlatego też oprócz przybliżenia wielomianem 10 stopnia przybliżyłem również wielomianem 8 stopnia, który również ma dobre właściwości

Wykres funkcji aproksymującej 10 stopnia:



|  |  |
| --- | --- |
| Maksymalna potęga x w wielomianie | Wartość stałej przy x |
| 0 | -1.7885 |
| 1 | -0.3412 |
| 2 | 0.6211 |
| 3 | 0.1466 |
| 4 | -0.3737 |
| 5 | -0.0366 |
| 6 | 0.0459 |
| 7 | 0.0019 |
| 8 | -0.0025 |
| 9 | -3,4973e-05 |
| 10 | 4,5545e-05 |

Wykres funkcji aproksymującej 8 stopnia:



|  |  |
| --- | --- |
| Maksymalna potęga x w wielomianie | Wartość stałej przy x |
| 0 | -1.5631 |
| 1 | -0.0900 |
| 2 | -0.0492 |
| 3 | -0.0225 |
| 4 | -0.0756 |
| 5 | -0.0074 |
| 6 | 0.0024 |
| 7 | 0.0002 |
| 8 | -4,1509e-05 |

Zadanie 2

%

%Tomasz Indeka

%MNUM-Projekt 2.18

%zadanie 2

%

%Metoda najmniejszych kwadratów przy wyznaczaniu funkcji

%układ równań normalnych

%

% x-wektor danych x

% y-wektor danych y

% a-wektor aproksymacji w bazie wielomianów

% h-rozmiar wektorów danych

% m-maksymalny stopień wielomianu przybliżającego

function a = zad2a()

x = [-5:1:5];

y = [-14.2376 -7.7256 -4.1949 -2.4815 -1.2683 -1.7885 -1.7269 -3.3830 -8.9977 -21.3130 -42.6544];

h=11;

m=9;

a = aproksymacja(x,y,h,m);

end

Kod części aproksymującej

function a = aproksymacja(x,y,h,m)

% x-wektor danych x

% y-wektor danych y

% a-wektor aproksymacji w bazie wielomianów

% G-macierz pomocnicza

% q-wektor pomocniczy

% xa-wektor x funkcji aproksymującej

% ya-wektor y funkcji aproksymującej

% h-rozmiar wektorów danych

% cz-norma Czebyszewa (maksimum) obliczonej funkcji

% e-norma euklidesowa obliczonej funkcji

% m-maksymalny stopień wielomianu przybliżającego

n=1;

while n<=m % obliczanie wektora aproksymacji dla kolejnych h-1 maksymalnych potęg x

k=1;

while k<=n % obliczenie macierzy pomocniczej G

i=1;

while i<=n

j=1;

G(i,k) = 0;

while j<=h

G(i,k) = G(i,k) + x(j)^(i+k-2);

j=j+1;

end

i=i+1;

end

q(k)=0;

j=1;

while j<=h % obliczenie wektora pomocniczego q

q(k)=q(k)+y(j)\*x(j)^(k-1);

j=j+1;

end

k=k+1;

end

a=q\*G^(-1);

xa = [-5:0.1:5];

i=1;

while i<=(10/0.1)+1 % obliczanie wartości funkcji aproksymującej do narysowania wykresu

j=1;

ya(i)=0;

while j<=n

ya(i)= ya(i) + a(j)\*xa(i)^(j-1);

j=j+1;

end

i=i+1;

end

plot (x,y,'o',xa,ya);

i=1;

while i<=h % obliczanie wartości funkcji aproksymującej do obliczenia odpowiednich norm

j=1;

yb(i)=0;

while j<=n

yb(i)= yb(i) + a(j)\*x(i)^(j-1);

j=j+1;

end

i=i+1;

end

n=n+1;

cz = max(abs(y-yb)) % norma Czebyszewa (maksimum) obliczonej funkcji

e = norm(y-yb) % norma euklidesowa obliczonej funkcji

end

end