Tomasz Indeka

Metody numeryczne

Zadanie 1.20 – Sprawozdanie

**Pkt 1. Dokładność maszynowa komputera**

Do obliczenia dokładności maszynowej został wykorzystany algorytm obliczający kolejne ujemne potęgi dwójki. Moment, w którym liczba aktualna i kolejna zostały uznane za równe lub moment, w którym do obliczonej niedokładności dodanie 1 daje wynik równy 1, oznacza że obliczona wartość jest mniejsza niż dokładność maszynowa.

Warunki : 2-t = 2-(t+1) , 1+2-t = 1

Wynik: eps = 2-t+1

Otrzymany wynik: eps = 2.2204e-16

Wynik ten odpowiada liczbie 2-52

Kod programu:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20, pkt 1

%Wyznaczanie dokladnosci maszynowej

%t - liczba znakow mantysy

%eps - dokladnosc maszynowa

t=0;

while 2^(-t)~=2^(-(t+1)) && (1+2^(-t))~=1

t=t+1;

end

eps = 2^(-t+1)

**Pkt 2. Rozwiązywanie równań metodą Cholesky’ego-Banachiewicza**

Do rozwiązania macierzy metodą Cholesky’ego-Banachowicza zostały wyznaczone macierze L i LT wyznaczone ze wzoru : A=LLT , gdzie:

, i = 1, 2, … , n, j = i+1, i+2, …, n

Wykorzystano równania Ly = b i LTx = y do obliczenia wektora x.

Norma residuum jest liczona jako norma Euklidesowa wektora r = Ax-b.

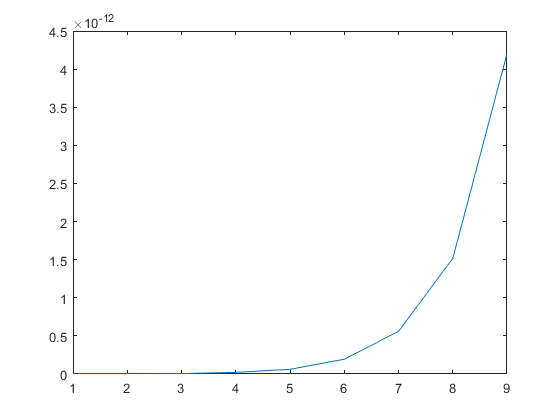
Dla danych:



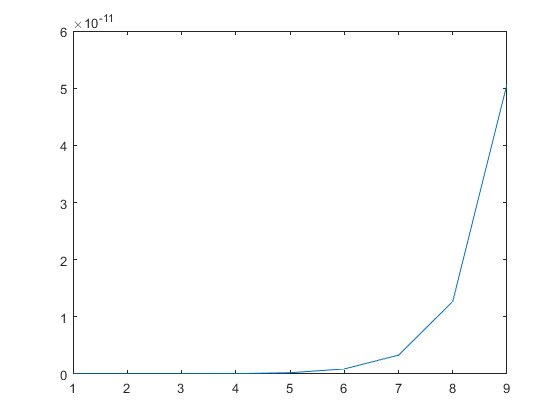
bi = 4 – 0.5i

Wyniki:

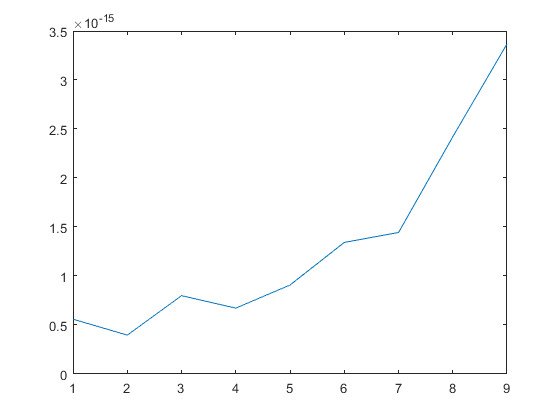
Dla mojej konfiguracji sprzętowej maksymalna ilość równań w czasie do 3 minut wynosiła 2560 i na podstawie obliczonych w tym czasie danych utworzyłem wykresy normy residuum od liczny równań :



b)



c)



Kod programu:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20, pkt 2

%Rozwiązywanie równań metodą Choleskiego – Banachowicza

%n-wielkość macierzy pierwotnej

%A-macierz rownań liniowych

%b-rozwiązanie rownania Ax=b

%x-wektor rozwiązań macierzy A

%m-mnożnik wielkości macierzy pierwornej

%r-rożnica między wynikiem obliczonym, a prawidłowym

%res-norma z rożnicy r

**a)**

m=1;

while m<=9 % warunek na wielkośc macierzy zależny od posiadanego sprzętu i czasu

n = 10\*2^(m-1);

A = zeros(n);

i=1;

while i<=n %obliczanie macierzy A za pomocą podanego w treści zadania wzoru

j=1;

while j<=n

if j==i

A(j,i)=10;

elseif i==(j+1) || i==(j-1)

A(j,i)=2.5;

end

j=j+1;

end

i=i+1;

end

b = zeros(n,1);

i=1;

while i<=n %obliczanie wektora b za pomocą podanego w treści zadania wzoru

b(i,1)=4-(i/2);

i=i+1;

end

x = Choleski(n,A,b); % wywolanie funkcji obliczającej x

r=A\*x-b;

res(m) = residuum(n,r); % wywolanie funkcji obliczającej residuum

m=m+1;

end

plot(res)

**b)**

m=1;

while m<=9 % warunek na wielkośc macierzy zależny od posiadanego sprzętu i czasu

n = 10\*2^(m-1);

A = zeros(n);

i=1;

while i<=n %obliczanie macierzy A za pomocą podanego w treści zadania wzoru

j=1;

while j<=n

if j==i

A(j,i)=3\*n\*n-i;

else

A(j,i)=i+j+1;

end

j=j+1;

end

i=i+1;

end

b = zeros(n,1);

i=1;

while i<=n %obliczanie wektora b za pomocą podanego w treści zadania wzoru

b(i,1)=2.5+0.6\*i;

i=i+1;

end

x=Choleski(n,A,b); % wywolanie funkcji obliczającej x

r=A\*x-b;

res(m) = residuum(n,r); % wywolanie funkcji obliczającej residuum

m=m+1;

end

plot(res)

**c)**

m=1;

while m<=9 % warunek na wielkość macierzy zależny od posiadanego sprzętu i czasu

n = 10\*2^(m-1);

A = zeros(n);

i=1;

while i<=n %obliczanie macierzy A za pomocą podanego w treści zadania wzoru

j=1;

while j<=n

if j==i

A(j,i)=0.1\*n+0.3\*i;

else

A(j,i)=1/(6\*(i+j+1));

end

j=j+1;

end

i=i+1;

end

b = zeros(n,1);

i=1;

while i<=n %obliczanie wektora b za pomocą podanego w treści zadania wzoru

b(i,1)=5/(3\*i);

i=i+1;

end

x=Choleski(n,A,b); % wywolanie funkcji obliczającej x

r=A\*x-b;

res(m) = residuum(n,r); % wywolanie funkcji obliczającej residuum

m=m+1;

end

plot(res)

Kod funkcji Choleski:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20

%Algorytm Choleskiego-Banachowicza

%n-wielkość macierzy pierwotnej

%A-macierz rownań liniowych

%b-rozwiązanie rownania Ax=b

%x-wektor rozwiązań macierzy A

%L-macierz poddiagonalna rozkładu A=LL'

%y-wektor pośredni rozwiązania Ly=b

function x = Choleski(n,A,b)

L = zeros(n);

i=1;

while i<=n %utworzenie macierzy L rozkładu LL'

j=i;

while j<=n

if j==i

k=1;

l=0;

while k<=i-1

l=l+L(i,k)^2;

k=k+1;

end

L(i,j) = sqrt(A(j,i)-l);

else

k=1;

l=0;

while k<=i-1

l=l+L(j,k)\*L(i,k);

k=k+1;

end

L(j,i) = (A(j,i)-l)/L(i,i);

end

j=j+1;

end

i=i+1;

end

y = zeros(n,1); %obliczenie wektora y ze wzoru Ly=b

j=1;

while j<=n

i=1;

l=0;

while i<j

l = l + y(i,1)\*L(j,i);

i=i+1;

end

y(j,1)=(b(j,1)-l)/L(j,i);

j=j+1;

end

x = zeros(n,1); %obliczenie wektora x, będącego rozwiązaniem układu) ze wzoru L'x=y

i=n;

while i>=1

j=n;

l=0;

while i<j

l = l + x(j,1)\*L(j,i);

j=j-1;

end

x(j,1)=(y(j,1)-l)/L(j,i);

i=i-1;

end

end

Kod funkcji residuum:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20

%obliczanie residuum z rozwiązania

%n-wielkosc macierzy pierwotnej

%r-rożnica między wynikiem obliczonym, a prawidłowym

%res-norma z rożnicy r

function res = residuum (n,r)

i=1;

res=0;

while i<=n %obliczanie normy euklidesowej z wektora r

res = res + r(i)^2;

i = i+1;

end

res= sqrt(res);

end

**Pkt 3. Rozwiązywanie równań metodą Gaussa-Seidela**

Do rozwiązania macierzy metodą iteracyjną Gaussa-Seidela zostały wyznaczone macierze D, U i L wyznaczone ze wzoru : A=U+D+L , gdzie:  
 , j = i

, j < i

, j > i

a reszta macierzy uzupełniona jest zerami. Wykorzystano równanie: do obliczenia wektora xi+1, przyjmując jako x0 wektor zerowy. Błąd przybliżenie był liczony jako norma euklidesowa różnicy między kolejnymi iteracyjnymi przybliżeniami rozwiązania.

Dane macierzy były takie same jak dane w p(b) zadania powyżej, czas także był ograniczony do 3 minut.

Warunki ciągłości pętli iterowania:

i<1000

e>2-50

Dodatkowo zastosowałem warunek, aby wykonał co najmniej 5 iteracji, aby nie zakończył od razu (wartość e była inicjowana zerem, a to nie spełniało warunku)

Wyniki:

n – ilość równań w macierzy

i – ilość iteracji, po których uzyskano wynik

e – uzyskany błąd przybliżenia

n = 10

i = 14

e = 4.6931e-16

n = 20

i = 14

e = 1.2945e-16

n = 40

i = 14

e = 1.1827e-16

n = 80

i = 13

e = 7.3252e-16

n = 160

i = 13

e = 4.1899e-16

n = 320

i = 13

e = 2.7090e-16

n = 640

i = 13

e = 1.8442e-16

n = 1280

i = 13

e = 1.2843e-16

n = 2560

i = 13

e = 9.0129e-17

Wnioski:

Taka zadziwiająco niska ilość iteracji najprawdopobniej wynika ze szczególnego rozkładu wartości w macierzy, gdzie wartości na diagonali były wielokrotnie większe niż w reszcie macierzy. Poskutkowało to bardzo szybkim zbieganiem wyniku do wyznaczonej dokładności czyli do wartości 2-50.

Algorytm sprawiał wrażenie wydajniejszego od metody Cholesky’ego-Banachowicza, ponieważ dla tej samej liczby równań osiągał trochę niższy czas.

Kod programu:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20, pkt 2

%Rozwiązywanie równań metodą Gaussa-Seidela

%i-ilość powtorzeń iteracyjnego poprawiania rozwiązania

%n-wielkość macierzy pierwotnej

%A-macierz rownań liniowych

%b-rozwiązanie rownania Ax=b

%x-wektor rozwiązań macierzy A

%D-macierz diagonalna macierzy A

%U-macierz naddiagonalna macierzy A

%L-macierz poddiagonalna macierzy A

%w-wektor Ux-b

%m-mnożnik wielkości macierzy pierwornej

%e-bląd przybliżenia

m=1;

while m<=9 % warunek na wielkośc macierzy zależny od posiadanego sprzętu i czasu

n = 10\*2^(m-1);

A = zeros(n);

k=1;

while k<=n %obliczanie macierzy A za pomocą podanego w treści zadania wzoru

j=1;

while j<=n

if j==k

A(j,k)=3\*n\*n-k;

else

A(j,k)=k+j+1;

end

j=j+1;

end

k=k+1;

end

b = zeros(n,1); %obliczanie wektora b za pomocą podanego w treści zadania wzoru

k=1;

while k<=n

b(k,1)=2.5+0.6\*k;

k=k+1;

end

x = Gauss\_Seidel(n,A,b); % wywolanie funkcji obliczającej x

m=m+1;

end

Kod funkcji Gauss\_Seidel:

%Tomasz Indeka

%MNUM-Projekt

%zadanie 1.20, pkt 2

%Algorytm Gaussa-Seidela

%i-ilość powtorzeń iteracyjnego poprawiania rozwiązania

%n-wielkość macierzy pierwotnej

%A-macierz rownań liniowych

%b-rozwiązanie rownania Ax=b

%x-wektor rozwiązań macierzy A

%D-macierz diagonalna macierzy A

%U-macierz naddiagonalna macierzy A

%L-macierz poddiagonalna macierzy A

%w-wektor Ux-b

%e-bląd przybliżenia

function x = Gauss\_Seidel(n,A,b)

D = zeros(n);

L = zeros(n);

U = zeros(n);

k=1;

while k<=n %obliczanie macierzy D, L i U na podstawie macierzy A

j=1;

while j<=n

if j==k

D(j,k)=A(j,k);

elseif j>k

L(j,k)=A(j,k);

else

U(j,k)=A(j,k);

end

j=j+1;

end

k=k+1;

end

i=0;

e=0;

x = zeros(n,1); %zalożenie że rozwiązanie zerowe jest rowne wektorowi zerowemu

while (i<1000 && e>2^(-50)) || i<5 % obliczanie kolejnych przybliżeń rozwiązania

xpom=x; % chwilowe zapamiętanie wektora rozwiązań do poźniejszych porownań

w=U\*x - b;

j=1;

while j<=n

k=1;

l=-w(j);

while k<=j

l=l-L(j,k)\*x(k);

k=k+1;

end

x(j)=l/D(j,j);

j=j+1;

end

xpom=xpom-x; %obliczenie rożnicy pomiędzy rozwiązaniami

k=1;

e=0;

while k<=n % obliczenie blędu przybliżenia

e = e + xpom(k)^2;

k = k+1;

end

e = sqrt(e);

i=i+1;

end

n

i

e

end