Wydział Matematyki i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Metody numeryczne

Projekt 2

Metoda Jacobiego

Arkadiusz Kniaź

Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska
12 stycznia 2022

Spis treści

1 Opis matematyczny

1.1 Polecenie

9. Rozwiązywanie układu równań liniowych $Ax = b$, gdzie $A \in \mathbb{R}^{n\times n}$ jest macierzą siedmiodiagonalną, $b \in \mathbb{R}^n$, metoda Jacobiego

   (Klasyczne algorytmy należy tak zaprogramować, aby wykorzystać strukturę macierzy układu i nie wykonywać zbędnych operacji arytmetycznych.)

1.2 Teoria

Metoda Jacobiego jest metodą iteracji prostej. Jej zadaniem jest przybliżenie x gdzie $Ax = b$, A - macierz kwadratowa nieosobliwa. Metoda startuje z pewnego przybliżenia początkowego, zazwyczaj wektor zer, a następnie oblicza kolejne przybliżenia wektora x do spełnienia jednoego z kryteriów np.: $\|x^{(k+1)} - x^{(k)}\| < d$. Kolejne przybliżenia można obliczyć według wzorów:

$$
\begin{aligned}
x_1^{(k+1)} &= (b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(k)})/a_{11}, \\
x_2^{(k+1)} &= (b_2 - \sum_{j=1, j\neq 2}^{n} a_{2j}x_j^{(k)})/a_{22}, \\
&\dots \\
x_n^{(k+1)} &= (b_n - \sum_{j=1}^{n-1} a_{nj}x_j^{(k)})/a_{nn}.
\end{aligned}
$$

Algorytm metody Jacobego:

$x^{(0)} = (x_1^{(0)}, \dots, x_n^{(0)})^T$ – przybliżenie początkowe

for $k = 0, 1, \dots,$ (dopóki nie będzie spełniony wybrany warunek stopu)

  for $i = 1, 2, \dots, n$

    $x_i^{(k+1)} = (b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)})/a_{ii}$

  end

end

Jeżeli macierz A jest dominująca przekątniowo to metoda Jacobiego jest zbieżna dla dowolnego wektora początkowego.

2 Opis programu

2.1 Działanie

Program w swoim działaniu wykorzystuje 6 funkcji przedstawionych w następnych podrozdziałach. Użytkownik kożystając z programu wpisuje dane do i wywołuje bezpośrenio tylko jedną funkcję - rys_wykres. Funkcja rys_wykres sama tylko rysuje wykresy: czas/n, błąd przybliżenia/n, liczba iteracji/n, czas/d, błąd przybliżenia/d, liczba iteracji/d, czas/r, błąd przybliżenia/r, liczba iteracji/r, gdzie n - rozmiar macierzy, d - parametr definiujący dokładność, r - promień spekralny macierzy iteracji w metodzie Jacobiego. Wywołuje funkcje spr_n, spr_d i spr_r do przygotowania danych do wykresów. Funkcja spr_n wywołuje funkcję uklad, jacoby i odnosnik i na podstawie ich wyników tworzy macierz z danymi do wykresów w odniesieniu do n. Funkcja spr_d wywołuje funkcję uklad, jacoby i odnosnik i na podstawie ich wyników tworzy macierz z danymi do wykresów w odniesieniu do d. Funkcja spr_r wywołuje funkcję uklad, jacoby i odnosnik i na podstawie ich wyników tworzy macierz z danymi do wykresów w odniesieniu do r. Funkcja uklad na podstawie danych wejściowych przy pomocy funkcji randi() tworzy macierz A i wektor b. Funkcja odnosnik na podstawie macierzy A i wektora b zwraca wektor obliczony przy pomocy funkcji A\b. Funkcja jacoby jest implementacją matody iteracyjnej Jacobiego, przyjmuje macierz A, wektor b, parametr definiujący dokładność i maksymalną liczbę iteracji. Funkcja zwraca średni czas obliczeń, średnia z 5 wywołań, liczba iteracji, promień spekralny macierzy iteracji w metodzie Jacobiego i wektor x, gdzie A*x=b.

2.2 Funkcja rys_wykres

```
function []=rys_wykres(k_max,n,n_iter,d,d_iter,r_iter,rodzaj,b_rand,A_rand,A_rand_d)
 %% rys_wykres
 % funkcja przyjmuje:
 % k_max - maksymalna liczba iteracji (@jacoby)
 % n - rozmiar macierzy(wektor poziomy) (@uklad)
 % d- parametr definiujacy dokladnosc(wektor poziomy) (@jacoby)
 % n_iter - liczba iteracji funkcji spr_n(ile razy ma sprawdzic uklad o
 % podanych rozmiarach) (@spr_n)
 % d_iter - liczba iteracji funkcji spr_d(ile razy ma sprawdzic uklad z
 % danym parametrem definujacym dokladnosc) (@spr_d)
 % r_iter - liczba iteracji funkcji spr_r(ile razy ma sprawdzic uk ad) (@spr_r)
 % rodzaj - rodzaj ukladu A*x=b ze wzgledu na liczbe rozwiazan (@uklad)
 % b_rand - przedzial z ktorego losowane sa dane do wektora b, (postac-[a,b])(@uklad)
 % A_rand - przedzial z ktorego losowane sa dane do macierzy A poza glowna przekatna, (postac-[a,b])(@uklad)
 % A_rand_d - przedzial z ktorego losowane sa dane do  glownej przekatnej macierzy A,(postac-[a,b])(@uklad)
 % funkcja rysuje wykresy

wynik_n=spr_n(@jacoby,@odnosnik,@uklad,k_max,n,d(1,floor(length(d)/2)),n_iter,rodzaj,b_rand,A_rand,A_rand_d);
xn=wynik_n(:,1);
y1=wynik_n(:,2);
```

```
y2=wynik_n(:,3);
y3=wynik_n(:,4);

wynikj_r=spr_r(@jacoby,@odnosnik,@uklad,k_max,n(1,floor(length(n)/2)),
   d(1,floor(length(d)/2)),r_iter,rodzaj,b_rand,A_rand,A_rand_d);
xr=wynikj_r(:,1);
y4=wynikj_r(:,2);
y5=wynikj_r(:,3);
y6=wynikj_r(:,4);

wynik_d=spr_d(@jacoby,@odnosnik,@uklad,k_max,n(1,floor(length(d)/2)),d
   ,d_iter,rodzaj,b_rand,A_rand,A_rand_d);
xd=wynik_d(:,1);
y7=wynik_d(:,2);
y8=wynik_d(:,3);
y9=wynik_d(:,4);

figure;

subplot(3,3,1);
scatter(xn,y1,'b');
h1 = lsline;
h1.Color = 'b';
title('Wykres 1','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('n');
ylabel('czas');

subplot(3,3,2);
scatter(xn,y2,'b');
h2 = lsline;
h2.Color = 'b';
title('Wykres 2','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('n');
ylabel('b   d  przybli enia');

subplot(3,3,3);
scatter(xn,y3,'b');
h3 = lsline;
h3.Color = 'b';
title('Wykres 3','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('n');
ylabel('liczba iteracji');

subplot(3,3,4);
scatter(xr,y4,'b');
h4 = lsline;
h4.Color = 'b';
title('Wykres 4','FontSize',14);
legend({'jacoby'},'Location','best');
```

```
xlabel('r');
ylabel('czas');

subplot(3,3,5);
scatter(xr,y5,'b');
h5 = lsline;
h5.Color = 'b';
title('Wykres 5','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('r');
ylabel('b   d  przybli enia');

subplot(3,3,6);
scatter(xr,y6,'b');
h6 = lsline;
h6.Color = 'b';
title('Wykres 6','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('r');
ylabel('liczba iteracji');

subplot(3,3,7);
scatter(xd,y7,'b')
set(gca,'xscale','log')
title('Wykres 7','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('d');
ylabel('czas');

subplot(3,3,8);
scatter(xd,y8,'b')
set(gca,'xscale','log')
set(gca,'yscale','log')
title('Wykres 8','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('d');
ylabel('b   d  przybli enia');

subplot(3,3,9);
scatter(xd,y9,'b')
set(gca,'xscale','log')
title('Wykres 9','FontSize',14);
legend({'jacoby'},'Location','best');
xlabel('d');
ylabel('liczba iteracji');

end
```

2.3 Funkcja spr_n

```
function [wynik]=spr_n(jacoby,odnosnik,uklad,k_max,n,d,n_iter,rodzaj,b_rand,A_rand,A_rand_d)
 %% spr_n
 % funkcja przyjmuje:
 % jacoby - @jacoby
 % odnonnik - @odnosnik
 % uklad - @uklad
 % k_max - maksymalna liczba iteracji (@jacoby)
 % n - rozmiar macierzy(wektor poziomy) (@uklad)
 % d- parametr definiujacy dokladnosc (@jacoby)
 % n_iter - liczba iteracji funkcji spr_n(ile razy ma sprawdzic uklad o podanych rozmiarach)
 % rodzaj - rodzaj ukladu A*x=b ze wzgledu na liczbe rozwiazan (@uklad)
 % b_rand - przedzial z ktorego losowane sa dane do wektora b, (postac-[a,b])(@uklad)
 % A_rand - przedzial z ktorego losowane sa dane do macierzy A poza glowna przekatna, (postac-[a,b])(@uklad)
 % A_rand_d - przedzial z ktorego losowane sa dane do  glownej przekatnej macierzy A,(postac-[a,b])(@uklad)
 % funkcja zwraca:
 % wynik(row,1)=n(1,k) - rozmiar macierzy dla kt rej wywoluje funkcj  @jacoby
 % wynik(row,2)=j(1,1) - czas wykonywania oblicze  usredniony z 5 prob @jacoby
 % wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n(1,k) - blad wzgledny
 % uzyskanego wyniku @jacoby, @odnosnik
 % wynik(row,4)=j(1,2) - liczba wykonanych iteracji @jacoby

wynik=zeros(length(n)*n_iter,4);
for iter=1:n_iter
    for k=1:length(n)
        A_b=uklad(rodzaj,n(1,k),b_rand,A_rand,A_rand_d);
        A=A_b(:,1:n(1,k));
        b=A_b(:,n(1,k)+1);
        j=jacoby(A,b,d,k_max);
        odn=odnosnik(A,b);
        row=(iter-1)*length(n)+k;
        wynik(row,1)=n(1,k);
        wynik(row,2)=j(1,1);
        wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n(1,k);
        wynik(row,4)=j(1,2);
    end
end
end
```

2.4 Funkcja spr_d

```
function [wynik]=spr_d(jacoby,odnosnik,uklad,k_max,n,d,d_iter,rodzaj,b_rand,A_rand,A_rand_d)
 %% spr_d
```

```
 % funkcja przyjmuje:
 % jacoby - @jacoby
 % odnonnik - @odnosnik
 % uklad - @uklad
 % k_max - maksymalna liczba iteracji (@jacoby)
 % n - rozmiar macierzy (@uklad)
 % d- parametr definiujacy dokladnosc(wektor poziomy) (@jacoby)
 % d_iter - liczba iteracji funkcji spr_d(ile razy ma sprawdzic uklad z danym parametrem definujacym dokladnosc)
 % rodzaj - rodzaj ukladu A*x=b ze wzgledu na liczbe rozwiazan (@uklad)
 % b_rand - przedzial z ktorego losowane sa dane do wektora b, (postac-[a,b])(@uklad)
 % A_rand - przedzial z ktorego losowane sa dane do macierzy A poza glowna przekatna, (postac-[a,b])(@uklad)
 % A_rand_d - przedzial z ktorego losowane sa dane do  glownej przekatnej macierzy A,(postac-[a,b])(@uklad)
 % funkcja zwraca:
 % wynik(row,1)=d(1,k) - parametr definiujacy dokladnosc z ktorym wywo uje funkcje @jacoby
 % wynik(row,2)=j(1,1) - czas wykonywania obliczen usredniony z 5 prob @jacoby
 % wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n - blad wzgledny
 % uzyskanego wyniku @jacoby, @odnosnik
 % wynik(row,4)=j(1,2) - liczba wykonanych iteracji @jacoby
wynik=zeros(length(d)*d_iter,4);
for iter=1:d_iter
    for k=1:length(d)
        A_b=uklad(rodzaj,n,b_rand,A_rand,A_rand_d);
        A=A_b(:,1:n);
        b=A_b(:,n+1);
        j=jacoby(A,b,d(1,k),k_max);
        odn=odnosnik(A,b);
        row=(iter-1)*length(d)+k;
        wynik(row,1)=d(1,k);
        wynik(row,2)=j(1,1);
        wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n;
        wynik(row,4)=j(1,2);
    end
end
end
```

2.5 Funkcja spr_r

```
function [wynik]=spr_r(jacoby,odnosnik,uklad,k_max,n,d,r_iter,rodzaj,b_rand,A_rand,A_rand_d)
%% spr_r
 % funkcja przyjmuje:
 % jacoby - @jacoby
 % odnonnik - @odnosnik
 % uklad - @uklad
```

```
 % k_max - maksymalna liczba iteracji (@jacoby)
 % n - rozmiar macierzy (@uklad)
 % d- parametr definiuj cy dokladnosc (@jacoby)
 % r_iter - liczba iteracji funkcji spr_r(ile razy ma sprawdzic uklad)
 % rodzaj - rodzaj ukladu A*x=b ze wzgledu na liczbe rozwiazan (@uklad)
  % b_rand - przedzial z ktorego losowane sa dane do wektora b, (postac-[a,b])(@uklad)
 % A_rand - przedzial z ktorego losowane sa dane do macierzy A poza glowna przekatna, (postac-[a,b])(@uklad)
 % A_rand_d - przedzial z ktorego losowane sa dane do  glownej przekatnej macierzy A,(postac-[a,b])(@uklad)
 % funkcja zwraca:
 % wynik(row,1)=j(1,3) - promien spekralny macierzy iteracji w metodzie Jacobiego @jacoby
 % wynik(row,2)=j(1,1) - czas wykonywania obliczen usredniony z 5 prob @jacoby
 % wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n - blad wzgledny
 % uzyskanego wyniku @jacoby, @odnosnik
 % wynik(row,4)=j(1,2) - liczba wykonanych iteracji @jacoby
wynik=zeros(r_iter,4);
for row=1:r_iter
    A_b=uklad(rodzaj,n,b_rand,A_rand,A_rand_d);
    A=A_b(:,1:n);
    b=A_b(:,n+1);
    j=jacoby(A,b,d,k_max);
    odn=odnosnik(A,b);
    wynik(row,1)=j(1,3);
    wynik(row,2)=j(1,1);
    wynik(row,3)=sum(abs((j(1,4:end)-odn)./odn))/n;
    wynik(row,4)=j(1,2);
end
end
```

2.6 Funkcja odnosnik

```
function [wynik]=odnosnik(A,b)
%% odnosnik
% A - macierz ta sama co do @jacoby
% b - wektor ten sam co do @jacoby
% wynik - wektor x, Ax=b
wynik=(A\b)';
end
```

2.7 Funkcja jacoby

```
function [wynik]=jacoby(A,b,d,kmax)
%% jacoby
% funkcja przyjmuje:
```

```
% A - macierz siedmiodiagonalna kwadratowa, dzia a dla macierzy size(
    A)>=6
% b - pionowy wektor
% d - parametr definiujacy dokladnosc, w warunku z bledem bezwzglednym
% kmax - maksymalna liczba iteracji
% funkcja zwraca:
% wynik(1,1)=t_sredni - czas wykonywania obliczen usredniony z 5 prob
% wynik(1,2)=itr - liczba wykonanych iteracji
% wynik(1,3)=r - promien spekralny macierzy iteracji w metodzie
    Jacobiego
% wynik(1,4:end)=x(1,:) - wektor wynikowy, A*x=b

n=size(A,1);
t_sr=zeros(1,5);
Aj=zeros(n,n);
for i=1:n
    Aj(i,:)=-A(i,:)/A(i,i);
    Aj(i,i)=0;
end
r=max(abs(eig(Aj)));
for j=1:5
    tic
    itr=1;
    x=zeros(1,n);
    prep_x=ones(1,n);
    for k=1:kmax
        if abs(max(prep_x-x))<d
            break
        end
        suma=sum(A(1,2:4).*x(1,2:4));
        prep_x(1)=x(1);
        x(1)=(b(1)-suma)/A(1,1);
        suma=sum(A(2,[1 3:5]).*x(1,[1 3:5]));
        prep_x(2)=x(2);
        x(2)=(b(2)-suma)/A(2,2);
        suma=sum(A(3,[1:2 4:6]).*x(1,[1:2 4:6]));
        prep_x(3)=x(3);
        x(3)=(b(3)-suma)/A(3,3);
        for i=4:n-3
            suma=sum(A(i,[i-3:i-1 i+1:i+3]).*x(1,[i-3:i-1 i+1:i+3]));
            prep_x(i)=x(i);
            x(i)=(b(i)-suma)/A(i,i);
        end
        suma=sum(A(n-2,[n-5:n-3 n-1:n]).*x(1,[n-5:n-3 n-1:n]));
        prep_x(n-2)=x(n-2);
        x(n-2)=(b(n-2)-suma)/A(n-2,n-2);
        suma=sum(A(n-1,[n-4:n-2 n]).*x(1,[n-4:n-2 n]));
        prep_x(n-1)=x(n-1);
        x(n-1)=(b(n-1)-suma)/A(n-1,n-1);
        suma=sum(A(n,n-3:n-1).*x(1,n-3:n-1));
        prep_x(n)=x(n);
        x(n)=(b(n)-suma)/A(n,n);
```

```
        itr=itr+1;
    end
    t_sr(1,j)=toc;
end
t_sredni=sum(t_sr)/5;
wynik=zeros(1,n+3);
wynik(1,1)=t_sredni;
wynik(1,2)=itr;
wynik(1,3)=r;
wynik(1,4:end)=x(1,:);
end
```

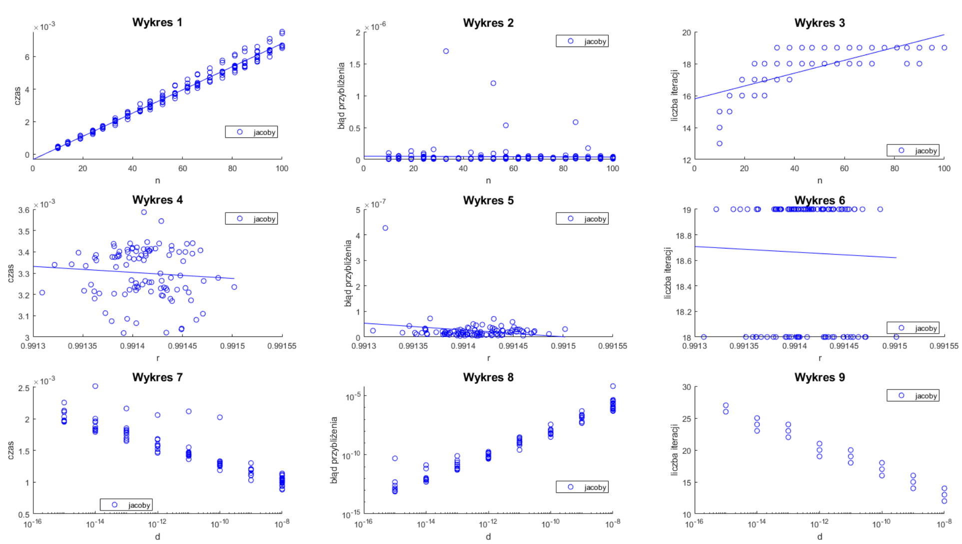
3 Przykłady

3.1 Przykład 1

Przykład pokazuje jak zachowuje się metoda Jacobiego, gdy r - promień spekralny macierzy iteracji w metodzie Jacobiego jest "bliski" 1 ale mniejszy niż 1 .

```
>> n=floor(linspace(10,100,20));
>> d=[10^(-8),10^(-8),10^(-9),10^(-10),10^(-11),10^(-12),10^(-13),10^(-14),10^(-15)];
>> rys_wykres(500,n,10,d,10,100,"tak",[-10,10],[999,1000],[6001,6002])
```

Rysunek 1: Dane wejściowe dla przykładu 1



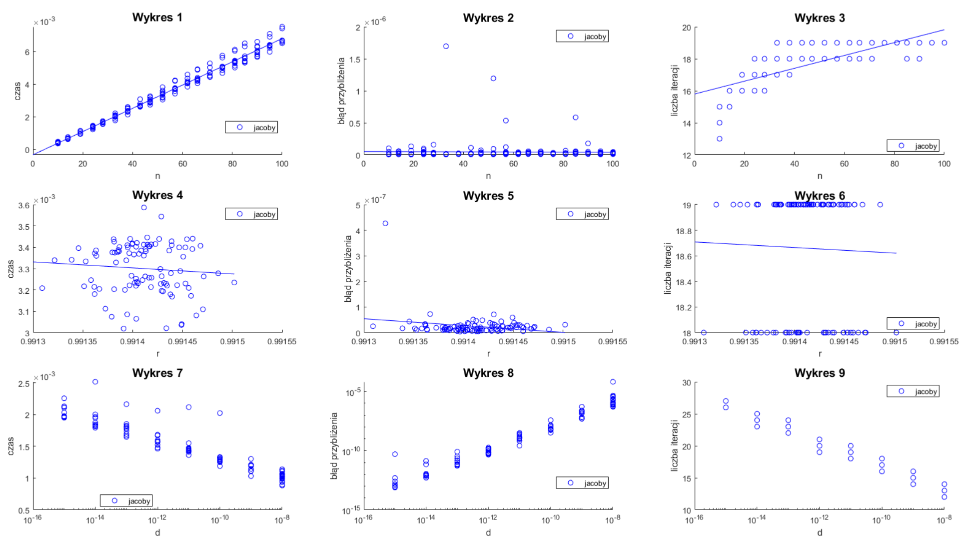
Rysunek 2: Wykresy dla przykładu 1

3.2 Przykład 2

Przykład pokazuje jak zachowuje się metoda Jacobiego, gdy r - promień spekralny macierzy iteracji w metodzie Jacobiego jest "bliski" 0.

```
>> n=floor(linspace(10,100,20));
>> d=[10^(-8),10^(-8),10^(-9),10^(-10),10^(-11),10^(-12),10^(-13),10^(-14),10^(-15)];
>> rys_wykres(500,n,10,d,10,100,"tak",[-10,10],[1,2],[1001,1002])
```

Rysunek 3: Dane wejściowe dla przykładu 2



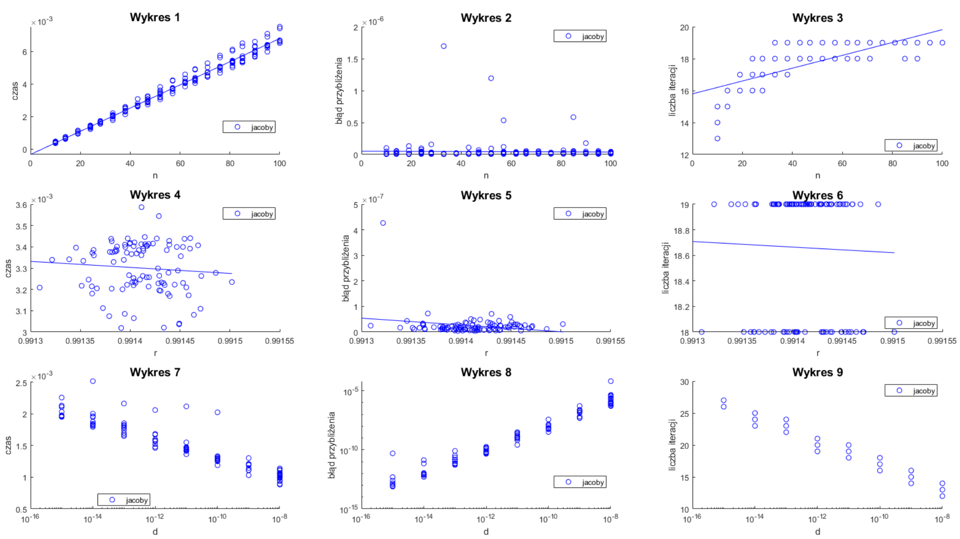
Rysunek 4: Wykresy dla przykładu 2

3.3 Przykład 3

Przykład pokazuje jak zachowuje się metoda Jacobiego, gdy n - rozmiar macierzy kwadratowej A jest "duży".

```
>> n=floor(linspace(1000,2000,20));
>> d=[10^(-8),10^(-8),10^(-9),10^(-10),10^(-11),10^(-12),10^(-13),10^(-14),10^(-15)];
>> rys_wykres(500,n,10,d,10,100,"tak",[-10,10],[1,10],[61,75])
```

Rysunek 5: Dane wejściowe dla przykładu 3



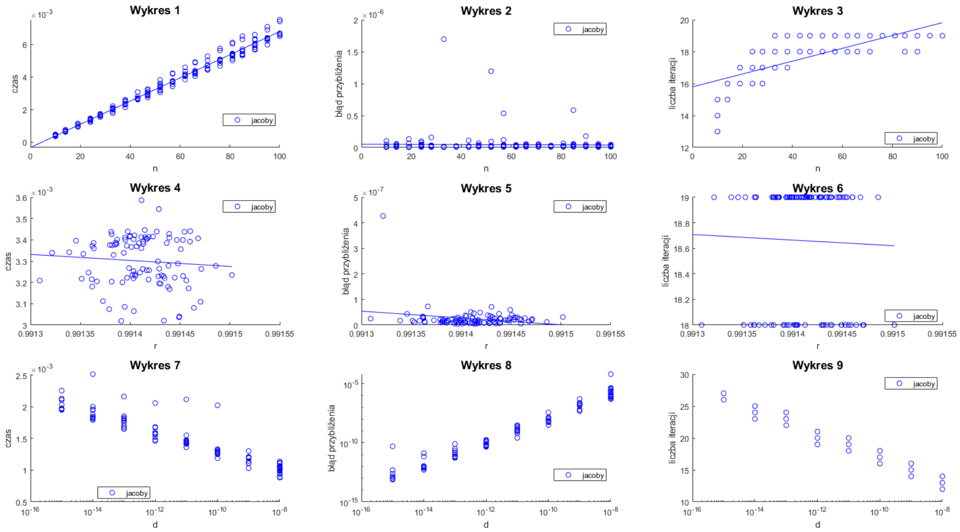
Rysunek 6: Wykresy dla przykładu 3

3.4 Przykład 4

Przykład pokazuje jak zachowuje się metoda Jacobiego, gdy d - parametr definiujący dokładność jest "mały".

```
>> n=floor(linspace(10,100,20));
>> d=[10^(-12),10^(-13),10^(-14),10^(-15),10^(-16),10^(-17),10^(-18),10^(-19),10^(-20)];
>> rys_wykres(500,n,10,d,10,100,"tak",[-10,10],[1,10],[61,75])
```

Rysunek 7: Dane wejściowe dla przykładu 4



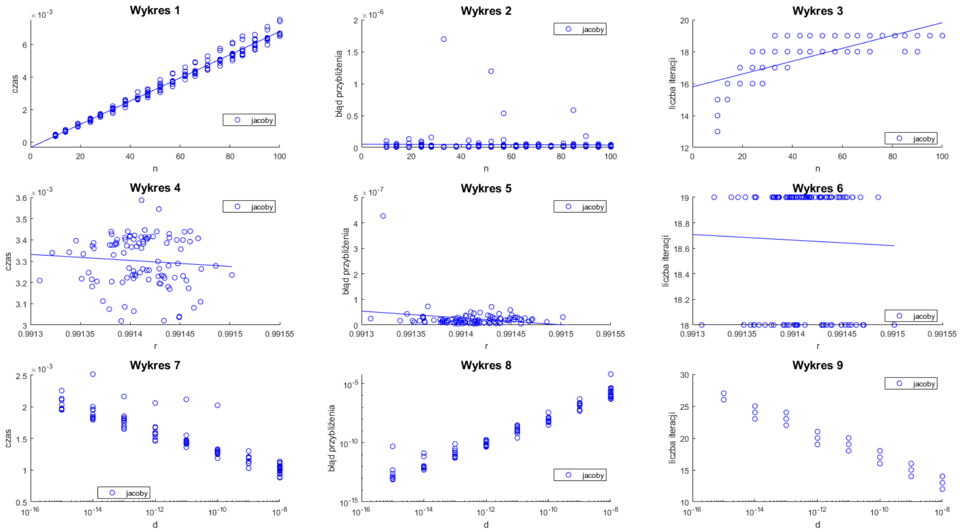
Rysunek 8: Wykresy dla przykładu 4

3.5 Przykład 5

Przykład pokazuje jak zachowuje się metoda Jacobiego, układ A*x=b jest sprzeczny.

```
>> n=floor(linspace(10,100,20));
>> d=[10^(-8),10^(-8),10^(-9),10^(-10),10^(-11),10^(-12),10^(-13),10^(-14),10^(-15)];
>> rys_wykres(500,n,10,d,10,100,"sprzeczny",[-10,10],[1,10],[1,10])
```

Rysunek 9: Dane wejściowe dla przykładu 5



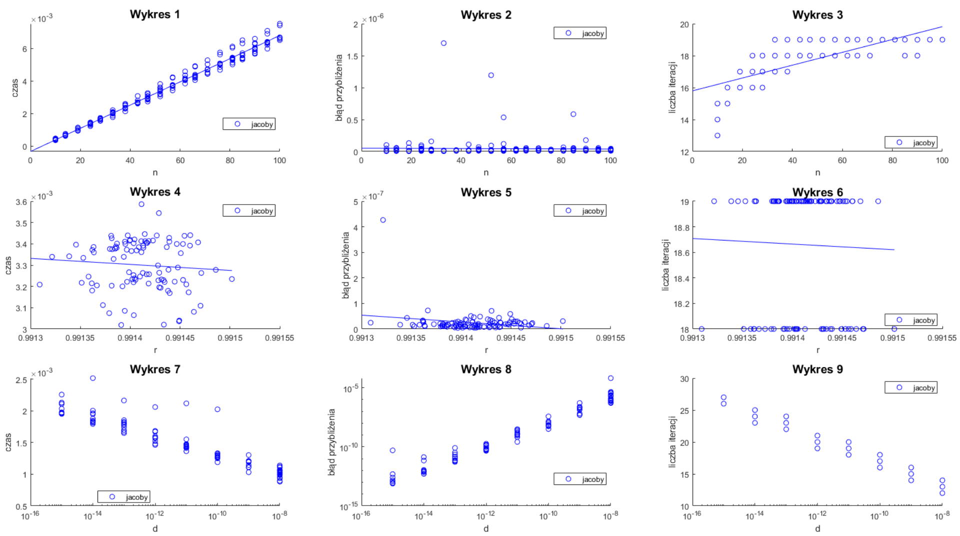
Rysunek 10: Wykresy dla przykładu 5

3.6 Przykład 6

Przykład pokazuje jak zachowuje się metoda Jacobiego, układ A*x=b ma nieskończenie wiele rozwiązań.

```
>> n=floor(linspace(10,100,20));
>> d=[10^(-8),10^(-8),10^(-9),10^(-10),10^(-11),10^(-12),10^(-13),10^(-14),10^(-15)];
>> rys_wykres(500,n,10,d,10,100,"nieskonczony",[-10,10],[1,10],[1,10])
```

Rysunek 11: Dane wejściowe dla przykładu 6



Rysunek 12: Wykresy dla przykładu 6

4 Analiza

Podsumowując na podstawie powyższych przykładów możemy zauważ, że metoda Jacobiego zachowuje się tak jak można by się tego spodziewać czyli czas obliczeń rośnie wraz z rozmiarem macierzy "n" i zadaną dokładnośćią "d", błąd przybliżenia maleje wraz zadaną dokładnośćią "d" ale mniej więcej do 10^{-16}, a liczba iteracji rośnie wraz z rozmiarem macierzy "n", promieniem spekralnym macierzy iteracji w metodzie Jacobiego "r" i zadaną dokładnośćią "d", oraz zdaje się być rozbieżna gdy układ $Ax = b$ jest sprzeczny lub ma nieskończenie wiele rozwiązań.