Rozwiązywanie układu równań liniowych $Ax = b$ z wykorzystaniem blokowej metody SOR. Wyznaczanie promienia spektralnego macierzy iteracji.

Julia Kaznowska
Piotr Wilczyński

IiAD MN gr.3

Styczeń 2022

1 Wstęp do zadania

Treść zadania:

14. **(2 osoby)** Rozwiązywanie układu równań liniowych $Ax = b$ z wykorzystaniem blokowej metody SOR, gdzie $A(n \times n)$ jest macierzą postaci

$$A = \begin{pmatrix} A_{11} & A_{12} & 0 \\ A_{12}^T & A_{11} & A_{23} \\ 0 & A_{23}^T & A_{11} \end{pmatrix},$$

gdzie $A_{ij}(p \times p)$ i $n = 3p$. Zakładamy, że A_{11} jest symetryczna i dodatnio określona. Do rozwiązania odpowiednich układów równań liniowych zastosować metodę Cholesky'ego–Banachiewicza (rozkład LDL^T). Wyznacz promień spektralny macierzy iteracji dla tej blokowej metody SOR.

Zadanie polega na rozwiązaniu układu równań liniowych $Ax = b$ z wykorzystaniem metody blokowej SOR oraz na wyznaczeniu promienia spektralnego macierzy iteracji dla tej blokowej metody. Dana macierz $A(n \times n)$ ma postać:

$$A = \begin{pmatrix} A_{11} & A_{12} & 0 \\ A_{12}^T & A_{11} & A_{23} \\ 0 & A_{23}^T & A_{11} \end{pmatrix}$$

gdzie każdy z bloków A_{ij} jest $(p \times p)$. Naturalnie $n = 3p$. Zakładamy również, że macierz A_{11} jest symetryczna i dodatnio określona. Do rozwiązywania późniejszych układów równań wynikających z blokowej metody SOR będziemy stosować metodę Cholsky'ego-Banachiewicza. Promień spektralny będziemy wyznaczać metodą potęgową.

2 Wstęp teoretyczny

Zadanie można podzielić na trzy części:

1. Rozwiązywanie układu równań liniowych $Ax = b$ metodą blokową SOR
2. Wyznaczanie macierzy iteracji
3. Wyznaczanie promienia spektralnego macierzy iteracji

2.1 Blokowa metoda SOR

Metoda SOR jest uogólnieniem metody Gaussa-Seidla. Występuje w niej parametr $\omega \in \mathbb{R}$, nazywany parametrem relaksacji. Aby metoda była zbieżna $\omega \in (0, 2)$. Wyprowadźmy wzory, które pomogą nam rozwiązać dany blokowy układ równań $Ax = b$.

$$\begin{pmatrix} A_{11} & A_{12} & 0 \\ A_{12}^T & A_{11} & A_{23} \\ 0 & A_{23}^T & A_{11} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

gdzie $x_1, x_2, x_3, b_1, b_2, b_3 \in \mathbb{R}^p$. Otrzymujemy:

$$\begin{pmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{12}^T x_1 + A_{11}x_2 + A_{23}x_3 \\ A_{23}^T x_2 + A_{11}x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Stąd:

$$\begin{cases} A_{11}x_1 + A_{12}x_2 = b_1 \\ A_{12}^T x_1 + A_{11}x_2 + A_{23}x_3 = b_2 \\ A_{23}^T x_2 + A_{11}x_3 = b_3 \end{cases}$$

Po przekształceniu:

$$\begin{cases} x_1 = A_{11}^{-1}(b_1 - A_{12}x_2) \\ x_2 = A_{11}^{-1}(b_2 - A_{12}^T x_1 - A_{23}x_3) \\ x_3 = A_{11}^{-1}(b_3 - A_{23}^T x_2) \end{cases}$$

Mnożąc wszystkie strony równań przez parametr ω oraz dodając stronami do i-tego równania x_i stronami:

$$\begin{cases} \omega x_1 + x_1 = \omega A_{11}^{-1}(b_1 - A_{12}x_2) + x_1 \\ \omega x_2 + x_2 = \omega A_{11}^{-1}(b_2 - A_{12}^T x_1 - A_{23}x_3) + x_2 \\ \omega x_3 + x_3 = \omega A_{11}^{-1}(b_3 - A_{23}^T x_2) + x_3 \end{cases}$$

Po przerzuceniu ωx_i na prawą stronę otrzymujemy:

$$\begin{cases} x_1 = (1-\omega)x_1 + \omega A_{11}^{-1}(b_1 - A_{12}x_2) \\ x_2 = (1-\omega)x_2 + \omega A_{11}^{-1}(b_2 - A_{12}^T x_1 - A_{23}x_3) \\ x_3 = (1-\omega)x_3 + \omega A_{11}^{-1}(b_3 - A_{23}^T x_2) \end{cases}$$

Powyższe wzory będą podstawą iteracji. Zaczynając od początkowego przybliżenia $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})^T \in \mathbb{R}^n$ obliczymy kolejne przybliżenia $x^{(k+1)}$ ($x_1^{(k)}, x_2^{(k)}, x_3^{(k)} \in \mathbb{R}^p$). Tak jak w metodzie Gaussa-Seidla, będziemy korzystać z ńajnowszych"dostępnych przybliżeń.
Dla $k = 0, 1, ...$:

$$\begin{cases} x_1^{(k+1)} = (1-\omega)x_1^{(k)} + \omega A_{11}^{-1}(b_1 - A_{12}x_2^{(k)}) \\ x_2^{(k+1)} = (1-\omega)x_2^{(k)} + \omega A_{11}^{-1}(b_2 - A_{12}^T x_1^{(k+1)} - A_{23}x_3^{(k)}) \\ x_3^{(k+1)} = (1-\omega)x_3^{(k)} + \omega A_{11}^{-1}(b_3 - A_{23}^T x_2^{(k+1)}) \end{cases}$$

Oznaczmy:

$$\begin{cases} y_1^{(k+1)} = A_{11}^{-1}(b_1 - A_{12}x_2^{(k)}) \\ y_2^{(k+1)} = A_{11}^{-1}(b_2 - A_{12}^T x_1^{(k+1)} - A_{23}x_3^{(k)}) \\ y_3^{(k+1)} = A_{11}^{-1}(b_3 - A_{23}^T x_2^{(k+1)}) \end{cases}$$

Wtedy otrzymujemy układy równań liniowych:

$$\begin{cases} A_{11}y_1^{(k+1)} = b_1 - A_{12}x_2^{(k)} \\ A_{11}y_2^{(k+1)} = b_2 - A_{12}^T x_1^{(k+1)} - A_{23}x_3^{(k)} \\ A_{11}y_3^{(k+1)} = b_3 - A_{23}^T x_2^{(k+1)} \end{cases}$$

Ponieważ, będziemy wielokrotnie rozwiązywać układy równań z macierzą A_{11}, efektywnie będzie rozłożyć tę macierz. Będziemy używać rozkładu Cholsky'ego-Banachiewicza LL^T (ten rozkład jest tożsamy z rozkładem LDL^T, gdzie D jest macierzą jednostkową). Wiemy, że ten istnieje, ponieważ z założeń zadania A_{11} jest symetryczna i dodatnio określona.

2.1.1 Wyznaczanie rozkładu Cholsky'ego-Banachiewicza LL^T

Rozkład Chlosky'ego Banachiewicza wyznaczamy znanym algorytmem. Załóżmy, że macierz $A(n \times n)$ o elementach a_{ij} jest symetryczna i dodatnio określona. Wtedy:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{12} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \cdot \begin{pmatrix} l_{11} & l_{21} & \dots & l_{n1} \\ 0 & l_{22} & \dots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & l_{nn} \end{pmatrix}$$

Analizując wynik mnożenia macierzy po prawej stronie otrzymujemy algorytm:
dla $k = 1, 2, ..., n$:
- $l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$
- dla $i = k+1, k+2, ..., n$
- $l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij}l_{kj})/l_{kk}$

2.1.2 Rozwiązywanie równań $LL^Tx = b$

Załóżmy, że mamy dany układ równań $LL^Tx = b$, gdzie macierz L jest macierzą dolnotrójkątną. Wtedy:

$$L\underbrace{L^Tx}_{y} = b$$

Najpierw rozwiązujemy układ równań $Ly = b$, a następnie $L^Tx = y$. W ten sposób wyznaczamy wektor x. Oba powyższe układy rozwiązujemy prosto i efektywnie, ponieważ zarówno L jak i L^T są macierzami trójkątnymi.

2.1.3 Podsumowanie blokowego SOR

W ten sposób wyznaczamy $y_1^{(k+1)}, y_2^{(k+1)}, y_3^{(k+1)}$. Wracając do blokowego SOR mamy:

$$\begin{cases} x_1^{(k+1)} = (1-\omega)x_1^{(k)} + \omega y_1^{(k+1)} \\ x_2^{(k+1)} = (1-\omega)x_2^{(k)} + \omega y_2^{(k+1)} \\ x_3^{(k+1)} = (1-\omega)x_3^{(k)} + \omega y_3^{(k+1)} \end{cases}$$

Iterację wykonujemy dopóki nie zostanie spełniony warunek stopu. Jako warunek stopu, weźmiemy warunek Gilla, ponieważ bierze on pod uwagę zarówno bezwzględną jak i względną różnicę pomiędzy kolejnymi iteracjami.
Warunek Gilla:

$$||x^{(k+1)} - x^{(k)}|| < d_1||x^{(k)}|| + d_2$$

gdzie jako $||.||$ przyjmiemy normą euklidesową.
Wartości parametrów d_1 i d_2 ustalimy jako:

$$d_1 = 10^{-10}$$

$$d_2 = 10^{-20}$$

Kiedy zostanie spełniony warunek stopu w $x^{(k+1)} = (x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)})^T$ powinniśmy mieć bardzo dobrze przybliżone rozwiązanie początkowego równania $Ax = b$.

2.2 Wyznaczenie macierzy iteracji B_{SOR}

Macierz iteracji dla metody SOR przedstawia się wzorem:

$$B_{SOR} = (D + \omega L)^{-1} \cdot ((1-\omega)D - \omega U)$$

Przekształćmy ten wzór tak, aby móc stworzyć układ równań:

$$(D + \omega L) \cdot B_{SOR} = ((1-\omega)D - \omega U)$$

Przedstawmy ten wzór w postaci macierzowej:

$$\begin{pmatrix} A_{11} & 0 & 0 \\ \omega A_{12}^T & A_{11} & 0 \\ 0 & \omega A_{23}^T & A_{11} \end{pmatrix} \cdot B_{SOR} = \begin{pmatrix} (1-\omega)A_{11} & -\omega A_{12} & 0 \\ 0 & (1-\omega)A_{11} & -\omega A_{23} \\ 0 & 0 & (1-\omega)A_{11} \end{pmatrix}$$

Macierz B_{SOR} zapiszmy jako:

$$\begin{pmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{pmatrix}$$

gdzie $X_{ii} \in \mathbb{R}^{(p \times p)}$, $i \in \{1, 2, 3\}$

Stąd możemy wyprowadzić następujący układ równań:

$$\begin{cases} A_{11}X_{11} = (1-\omega)A_{11} \\ A_{11}X_{12} = -\omega A_{12} \\ A_{11}X_{13} = 0 \\ \omega A_{12}^T X_{11} + A_{11}X_{21} = 0 \\ \omega A_{12}^T X_{12} + A_{11}X_{22} = (1-\omega)A_{11} \\ \omega A_{12}^T X_{13} + A_{11}X_{23} = -\omega A_{23} \\ \omega A_{23}^T X_{21} + A_{11}X_{31} = 0 \\ \omega A_{23}^T X_{22} + A_{11}X_{32} = 0 \\ \omega A_{23}^T X_{23} + A_{11}X_{33} = (1-\omega)A_{11} \end{cases}$$

Po przekształceniu wygląda on następująco:

$$\begin{cases} A_{11}X_{11} = (1-\omega)A_{11} \\ A_{11}X_{12} = -\omega A_{12} \\ A_{11}X_{13} = 0 \\ A_{11}X_{21} = -\omega A_{12}^T X_{11} \\ A_{11}X_{22} = (1-\omega)A_{11} - \omega A_{12}^T X_{12} \\ A_{11}X_{23} = -\omega A_{23} - \omega A_{12}^T X_{13} \\ A_{11}X_{31} = -\omega A_{23}^T X_{21} \\ A_{11}X_{32} = -\omega A_{23}^T X_{22} \\ A_{11}X_{33} = (1-\omega)A_{11} - \omega A_{23}^T X_{23} \end{cases}$$

2.2.1 Rozwiązywanie pojedynczego równania

Możemy zauważyć, że wszystkie równania w powyższym układzie można przedstawić w postaci

$$A_{11} \cdot X = B$$

Zapiszmy je zatem w postaci macierzowej:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Teraz możemy stworzyć odpowiadający temu działaniu układ równań:

$$\begin{cases} a_{11}x_{11} + a_{12}x_{21} + \ldots + a_{1n}x_{n1} = b_{11} \\ a_{21}x_{11} + a_{22}x_{21} + \ldots + a_{2n}x_{n1} = b_{21} \\ \ldots \\ a_{n1}x_{11} + a_{n2}x_{21} + \ldots + a_{nn}x_{n1} = b_{n1} \\ a_{11}x_{12} + a_{12}x_{22} + \ldots + a_{1n}x_{n2} = b_{12} \\ a_{21}x_{12} + a_{22}x_{22} + \ldots + a_{2n}x_{n2} = b_{22} \\ \ldots \\ a_{n1}x_{1n} + a_{n2}x_{2n} + \ldots + a_{nn}x_{nn} = b_{nn} \end{cases}$$

Możemy zauważyć, że równania da się połączyć w bloki, na przykład:

$$\begin{cases} a_{11}x_{1i} + a_{12}x_{2i} + \ldots + a_{1n}x_{ni} = b_{1i} \\ a_{21}x_{1i} + a_{22}x_{2i} + \ldots + a_{2n}x_{ni} = b_{2i} \\ \ldots \\ a_{n1}x_{1i} + a_{n2}x_{2i} + \ldots + a_{nn}x_{ni} = b_{ni} \end{cases}$$

Każdy taki blok posiada swoją macierzową reprezentację:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{ni} \end{pmatrix} = \begin{pmatrix} b_{1i} \\ b_{2i} \\ \vdots \\ b_{ni} \end{pmatrix}$$

gdzie i to numer kolumny macierzy X i B odpowiadającej danemu blokowi układu równań.
Dzięki tej obserwacji możemy zapisać p równań macierzowych (dla każdego bloku X_{ij}, gdzie $i, j \in \{1, 2, 3\}$) w postaci $A_{11}x = b$. Te natomiast możemy w łatwy sposób rozwiązać korzystając z omawianego wcześniej rozkładu Cholsky'ego-Banachiewicza LL^T.

2.3 Wyznaczanie promienia spektralnego macierzy

Promień spektralny jest maksymalnym modułem z wartości własnych macierzy. Wyznaczymy go korzystając z wbudowanych funkcji w MATLAB:

- `eig(A)` - zwraca wartości własne macierzy A
- `abs(x)` - zwraca wartość modułu x
- `max(T)` - zwraca maksymalny element tablicy T

Promień spektralny macierzy A wyznaczamy przez:

```
max(abs(eig(A)))
```

3 Program

Główną funkcją programu w MATLAB jest funkcja `zadanie14(`A`, `b`, `ω`)`. Ta funkcja jako argumenty przyjmuje:
A - macierz A spełniająca założenia zadania
b - wektor $b \in \mathbb{R}^n$
ω - współczynnik $\omega \in (0, 2)$ metody SOR
Funkcja wypisuje oraz zwraca rozwiązanie danego równania i promień spektralny macierzy iteracji metody SOR. Aby `zadanie14()` działała poprawnie zdefiniowaliśmy szereg funkcji pomocniczych:

- `matrixDivision(`A`)` - funkcja przyjmuje jako argument daną macierz A, a następnie zwraca określone w zadaniu bloki A_{11}, A_{12}, A_{23}.
- `cholskyBanachiewicz(`A`)` - funkcja przyjmuje jako argument symetryczną i dodatnio określoną macierz A. Zwraca macierz L, spełniającą: $A = LL^T$.
- `solveCholsky(`L`, `b`)` - funkcja przyjmuje jako argumenty macierz dolną trójkątną $L \in \mathbb{R}^{(p \times p)}$ oraz wektor $b \in \mathbb{R}^p$. Zwraca rozwiązanie równania $LL^T x = b$
- `conditionGill(Xold, Xnew)` - funkcja przyjmuje jako argumenty Xold, Xnew $\in \mathbb{R}^n$. Jeśli warunek Gilla dla danych wektorów jest spełniony, zwraca TRUE. W przeciwnym przypadku zwraca FALSE.
- `normEuklides(`x`)` - zwraca normę euklidesową podanego jako argument wektora x.
- `iterationMatrix(`$A11$`, `$A12$`, `$A23$`, `w`)` - funkcja zwraca macierz iteracji B_{SOR} wyznaczoną dla macierzy A określonej w zadaniu o blokach: A_{11}, A_{12}, A_{23}.
- `getXBlock(`M`, `L`, `p`)` - funkcja pomocnicza dla funkcji `iterationMatrix()`. Wyznacza blok macierzy iteracji.

Dodatkowo w programie załączone są funkcje: `testMatrixGenerator(p)`, która generuje macierz $A^{(3p \times 3p)}$ spełniającą założenia zadania i `exampleHandler(A,b)`, która jest funkcją pomocniczą do opisywania przykładów.

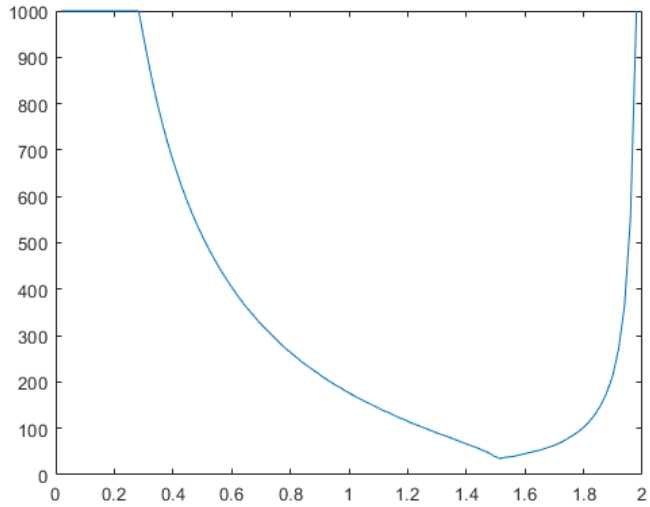
4 Przykłady

Zaprezentujemy przykładowe działanie naszej funkcji dla sześciu różnych macierzy. Pierwsze dwie będą macierzami (6×6), kolejne dwie (9×9), piąta (30×30), a ostatnia (300×300). Dla każdej macierzy będziemy wyznaczali zależność liczby iteracji i promienia spektralnego macierzy iteracji od współczynnika relaksacji (na osi x będzie ω). Dodatkowo założymy, że wyniki dawane przez standardowe, wbudowane rozwiązywanie równań macierzowych w MATLAB (A\b) jest dokładne i wyliczymy odchylenie standardowe. Wszystkie macierze i obliczenia są zapisane w pliku examples.m.

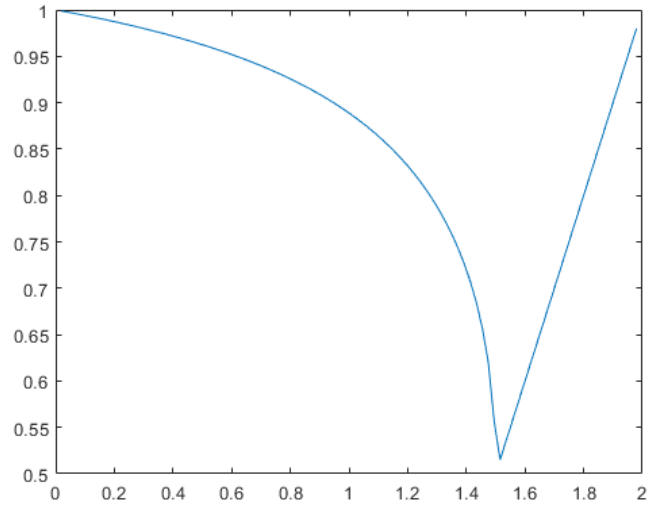
4.1 Przykład 1

Macierz zapisana w examples.m w zmiennej: Example1
Odchylenie standardowe = 7.1347e-09
Optymalny współczynnik relaksacji = 1.5152
Liczba iteracji przy optymalnym współczynniku relaksacji = 35

Wykres zależności liczby iteracji od współczynnika relaksacji:



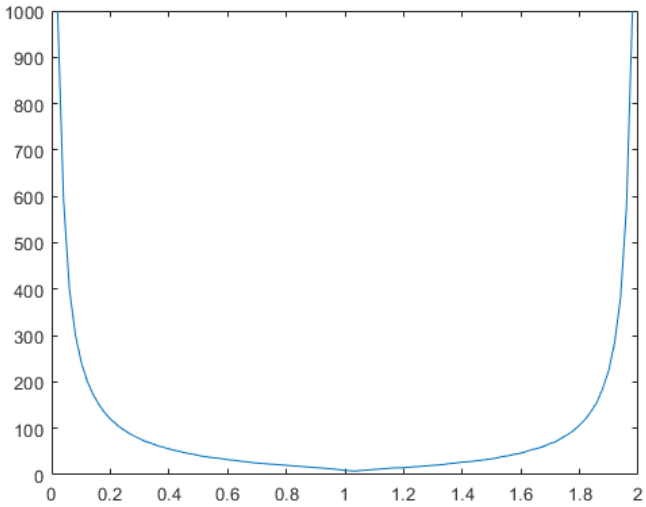
Wykres zależności promienia spektralnego od współczynnika relaksacji:



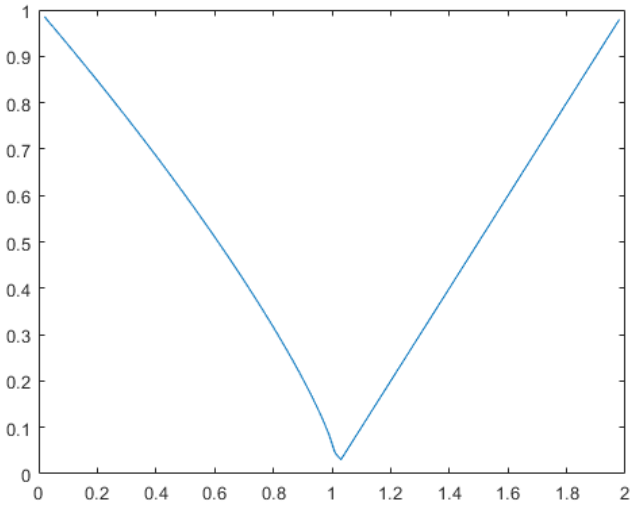
4.2 Przykład 2

Macierz zapisana w examples.m w zmiennej: Example2
Odchylenie standardowe = 4.5700e-13
Optymalny współczynnik relaksacji = 1.0303
Liczba iteracji przy optymalnym współczynniku relaksacji = 8

Wykres zależności liczby iteracji od współczynnika relaksacji:



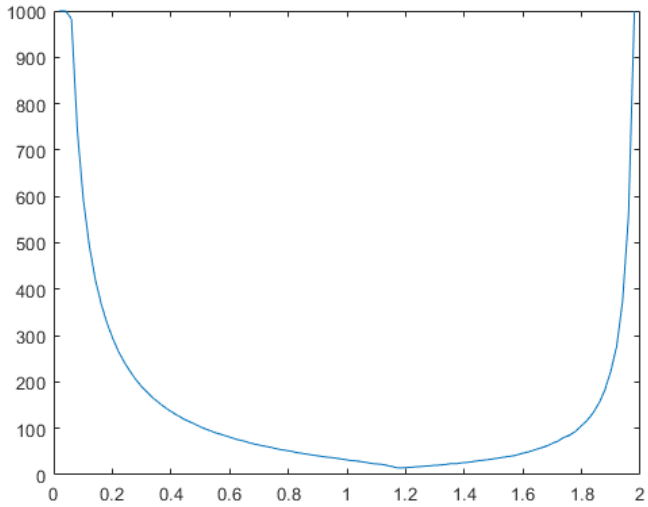
Wykres zależności promienia spektralnego od współczynnika relaksacji:



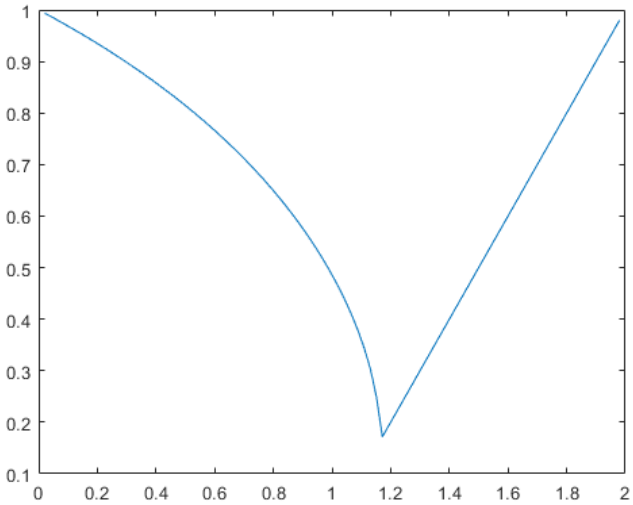
4.3 Przykład 3

Macierz zapisana w examples.m w zmiennej: Example3
Odchylenie standardowe = 2.4792e-11
Optymalny współczynnik relaksacji = 1.1717
Liczba iteracji przy optymalnym współczynniku relaksacji = 15

Wykres zależności liczby iteracji od współczynnika relaksacji:



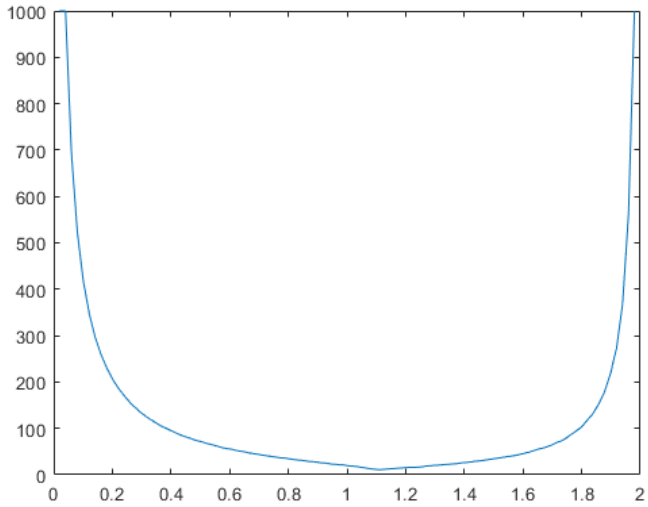
Wykres zależności promienia spektralnego od współczynnika relaksacji:



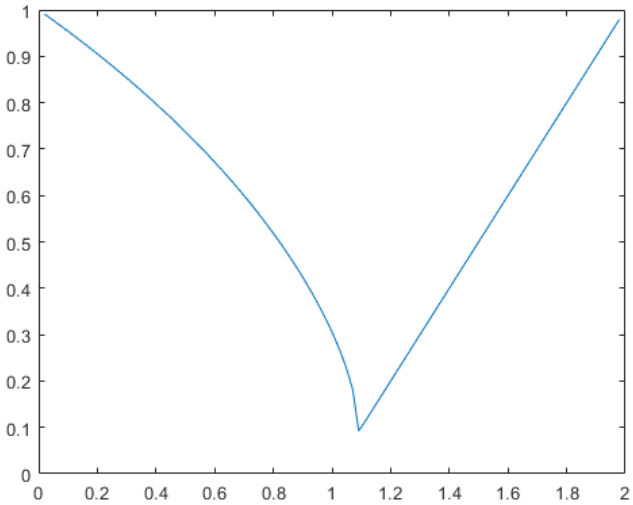
4.4 Przykład 4

Macierz zapisana w examples.m w zmiennej: Example4
Odchylenie standardowe = 2.1572e-10
Optymalny współczynnik relaksacji = 1.1111
Liczba iteracji przy optymalnym współczynniku relaksacji = 11

Wykres zależności liczby iteracji od współczynnika relaksacji:



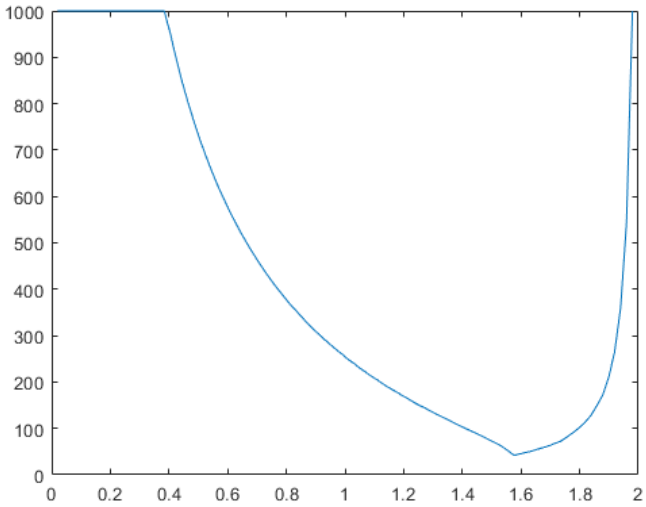
Wykres zależności promienia spektralnego od współczynnika relaksacji:



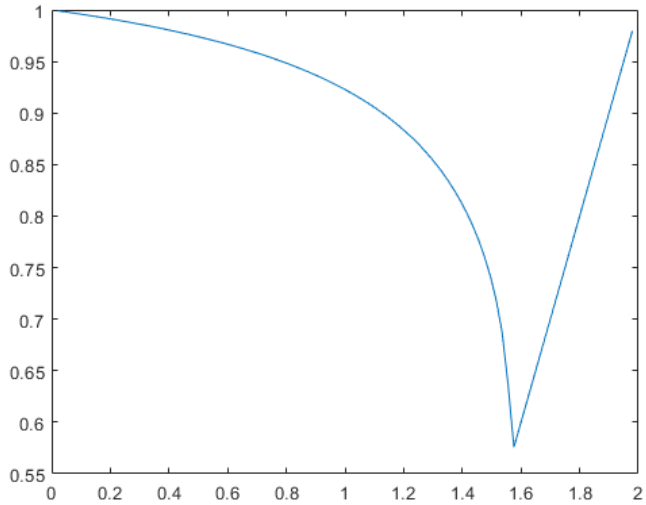
4.5 Przykład 5

Macierz zapisana w examples.m w zmiennej: Example5
Odchylenie standardowe = 2.2021e-10
Optymalny współczynnik relaksacji = 1.5758
Liczba iteracji przy optymalnym współczynniku relaksacji = 42

Wykres zależności liczby iteracji od współczynnika relaksacji:



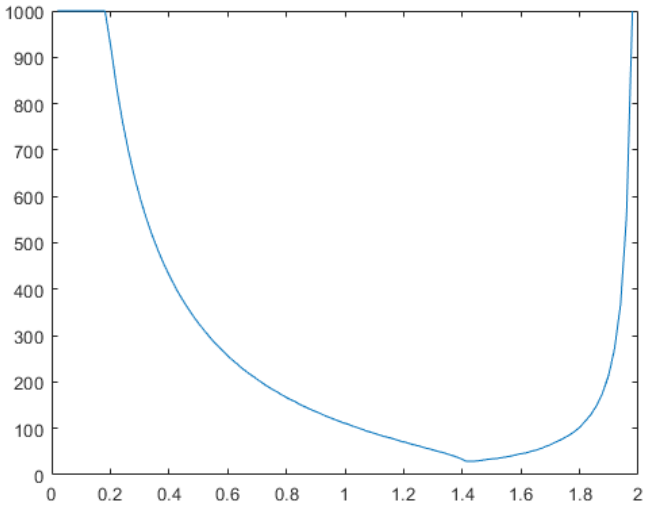
Wykres zależności promienia spektralnego od współczynnika relaksacji:



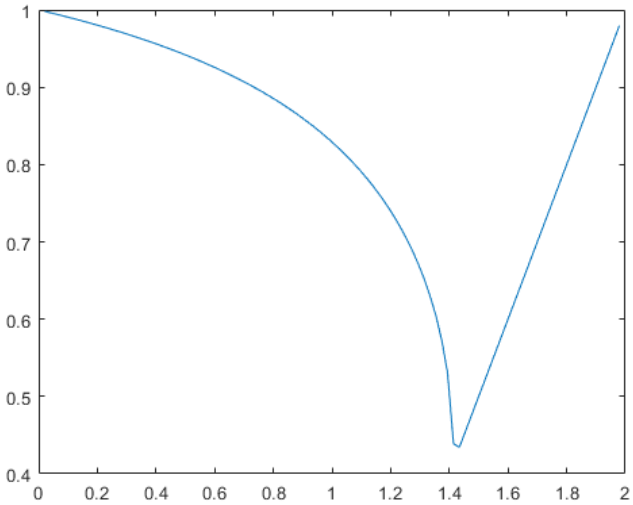
4.6 Przykład 6

Macierz zapisana w examples.m w zmiennej: Example6
Odchylenie standardowe = 1.5934e-10
Optymalny współczynnik relaksacji = 1.4141
Liczba iteracji przy optymalnym współczynniku relaksacji = 29

Wykres zależności liczby iteracji od współczynnika relaksacji:



Wykres zależności promienia spektralnego od współczynnika relaksacji:



5 Podsumowanie

W każdym z powyższych przykładów odchylenie standardowe jest rzędu równego lub mniejszego niż 10^{-9}. Dokładność można zwiększyć zmniejszając stałe w warunku stopu iteracji (warunku Gilla). Optymalny współczynnik relaksacji waha się pomiędzy 1, a 1.6. Wykres zależności zarówno potrzebnej liczby iteracji jak i promienia spektralnego w każdym z przykładów wygląda podobnie. Liczba iteracji dla optymalnego współczynnika relaksacji nie przekracza 50. Blokowy SOR przy powyższych założeniach wydaję się być bardzo skuteczną metodą wyznaczania rozwiązań układu równań liniowych $Ax = b$.