Lab 5c,
Programowanie matematyczne

Piotr Onyszczuk, gr. C

16 XII 2021

1 Treść zadania

Celem zadania jest rozwiązanie za pomocą różnych algorytmów układu równań liniowych $Ax = b$, gdzie $A = A^T$, dodatnio określona o współczynnikach całkowitych. Rozwiązywanie układu sprowadzi się do minimalizacji pewnej funkcji kwadratowej

1.1 Dane testowe

Do testów należy użyte zostały losowe macierze i wektory. Niestety dużym kłopotem okazało się wygenerowanie macierzy o wartościach z danego przedziału, spełniającej założenia zadania. W związku z tym warunek na przedział został de facto pominięty. Macierz spełniająca założenia jest generowana jako iloczyn pewnej losowej macierzy ze swoją transpozycją. Macierz używana w iloczynie zawiera wartości z zadanego przedziału. Po przemnożeniu zakres jest gubiony.

W celu zachowania rzędu wielkości, wektor b jest losowany z przedziału od najmniejszej wartości w macierzy A do największej wartości z tejże macierzy.

1.1.1 Macierz z narzuconymi wartościami własnymi

Macierz z narzuconymi wartościami własnymi jest generowana zgodnie z opisem z zadania:

1. Utwórz macierz diagonalną D z podanymi wartościami własnymi (wektor Q, n - długość wektora)
2. Wylosuj nieosobliwą macierz V
3. $V = orth(V)$
4. $A = VDV^T$

2 Definicja funkcji do minimalizacji

Minimalizowaną funkcją jest $f(x) = \frac{1}{2} * x * A * x - b * x$.
Jej gradient to $\nabla(x) = A * x - b$.
Jej hesjan to $\nabla^2(x) = A$.

3 Algorytmy

3.1 Wyznaczanie α_{max}

Algorytm ten nie będzie w pełni przytoczony. Szkic tego algorytmu znajduje się na stronie 22. wykładu nr 6. W kontekście wspomnianego szkicu:

- Na wejściu znamy $a_1 = 0$.
- Szukamy a_3, czyli końca przedziału.
- W algorytmie z wykładu brakuje jednego kroku. Brakujące miejsce opisane jest na slajdzie jako "(a_3 =?)". W tym miejscu do a_3 przypisywana jest poprzednia wartość a_2.

3.2 Algorytm złotego podziału

Algorytm ten jest w pełni opisany krok po kroku na stronie 28. wykładu nr 6.

3.3 Algorytm Armijo z kontrakcją

Algorytm ten jest opisany na stronie 9. i 10. wykładu nr 7.

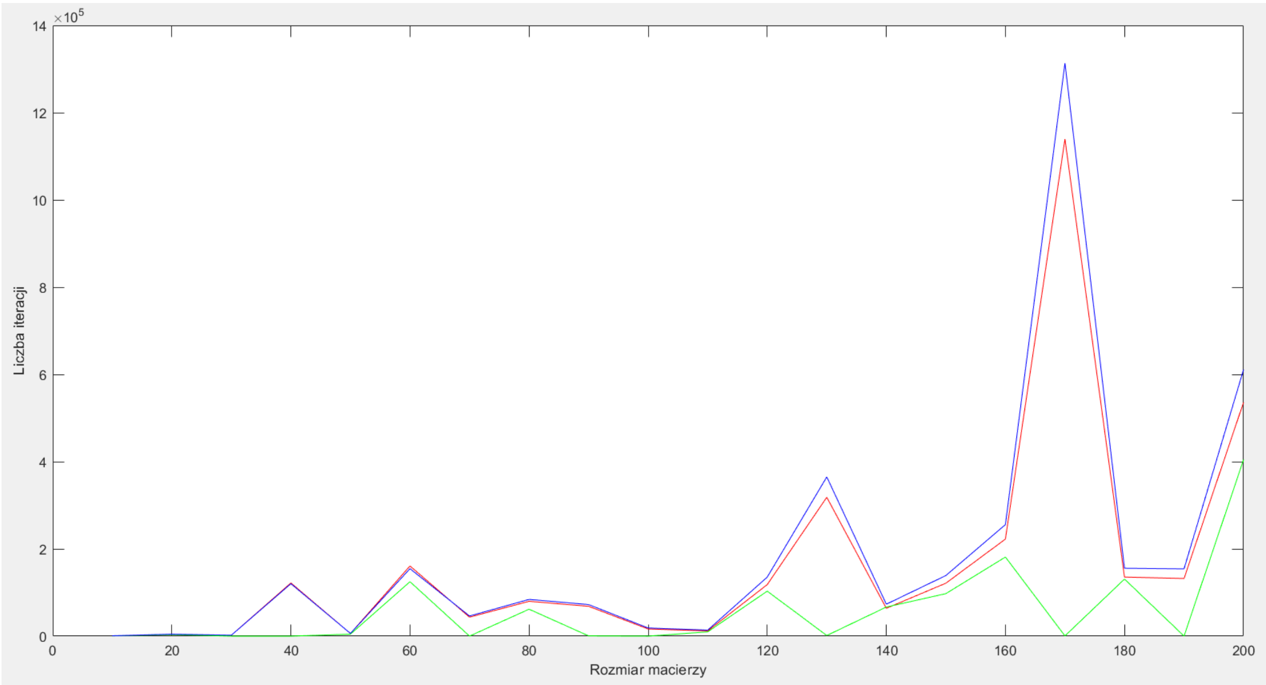
Dodatkowymi jego parametrami są κ, ρ i ϵ. Służą one do sterowania dokładnością obliczeń.

3.4 Algorytm gradientu sprzężonego

Zaimplementowany został algorytm gradientów sprzężonych, którego szkic jest na stronie 30. wykładu nr 7. Do wyznaczenia kolejnych wartości β została wykorzystana metoda Fletchera-Reeves'a. Testowane były różne warunki stopu.

3.5 Funkcje MATLABa

Wykorzystane zostały również dwie funkcje MATLABA: fminbnd oraz fminunc. Opcją, która została dodatkowo ustawiona jest maksymalna liczba iteracji funkcji fminunc, ponieważ domyślna (400) była zbyt mała.



Wycinek 1: Algorytmy

4 Wyniki i wnioski

4.1 Różne algorytmy

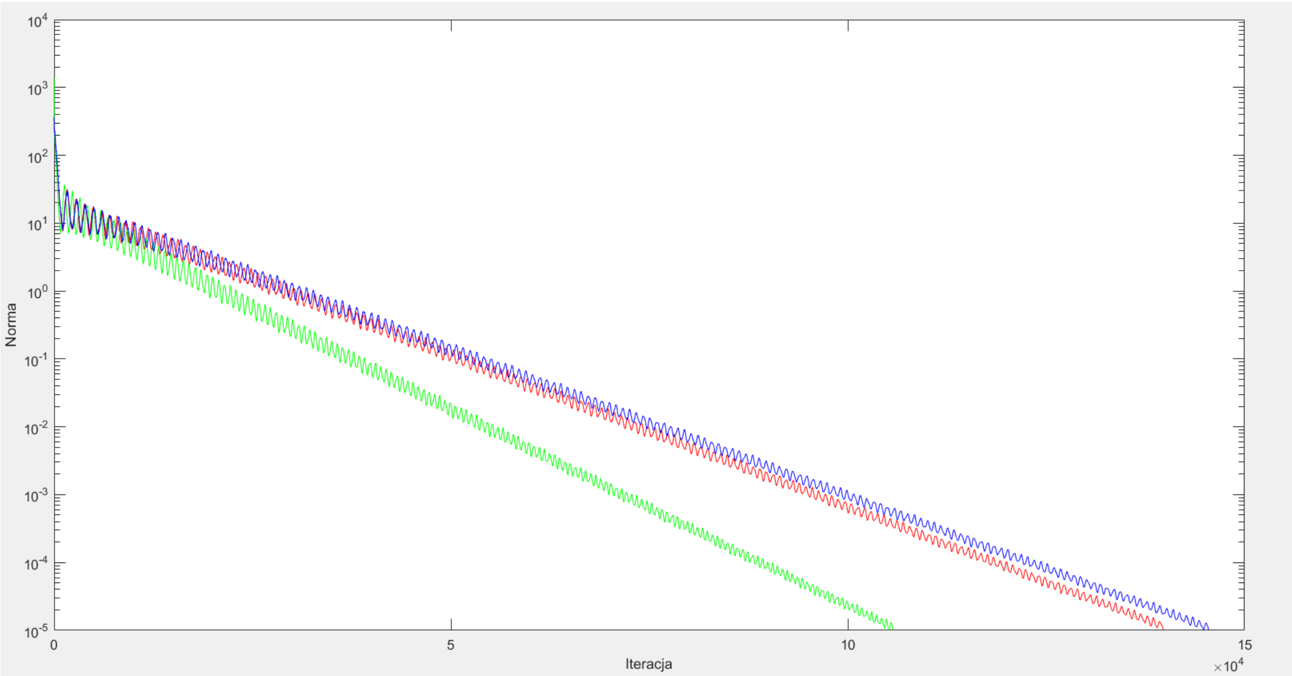
Porównane zostały różne algorytmy do minimalizacji funkcji wykorzystane w algorytmie FR (Wycinek 1):

- gold (czerwony)
- Armijo (zielony)
- fminbnd (niebieski)

, $\epsilon = 10^{-5}$, $p1 = 0$, $p2 = 3$

Sytuacje, gdy liczba iteracji dla algorytmu Armijo spada blisko zera są, gdy algorytm zwraca NaN w rozwiązaniu. Jest to spowodowane najprawdopodobniej niską odpornością tego rozwiązania na błędy numeryczne. W pozostałych przypadkach wszystkie trzy podejścia zwracają podobne rozwiązania.

Można zatem zauważyć że jeśli z wykorzystaniem Armijo rozwiązanie jest poprawne, wówczas algorytm FR potrzebuje mniej iteracji niż w innych przypadkach (za to Armijo wykonuje ich sporo). Dwa pozostałe podejścia dają bardzo podobne liczby iteracji.



Wycinek 2: Algorytmy - norma

Zauważalny jest również wzrost liczby iteracji wraz z wzrostem rozmiaru macierzy.

Ponadto dla $n = 100$ oraz przypadku gdzie wszystkie podejścia zwracają rozwiązanie został wykonany wykres normy gradientu od liczby iteracji (Wycinek 2).

Widoczny jest najszybszy spadek dla Armijo oraz podobne zachowanie dla dwóch pozostałych.

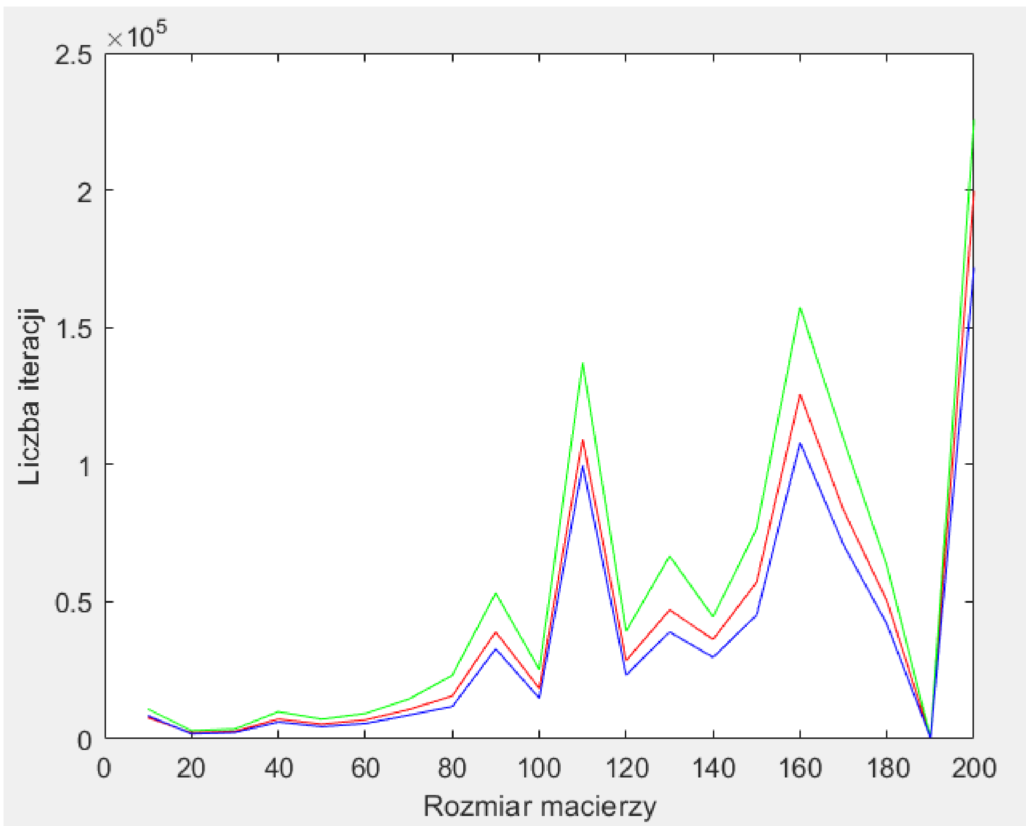
4.2 Warunki stopu dla FR

Porównana została liczba iteracji dla różnych warunków stopu algorytmu FR (Wycinek 3):

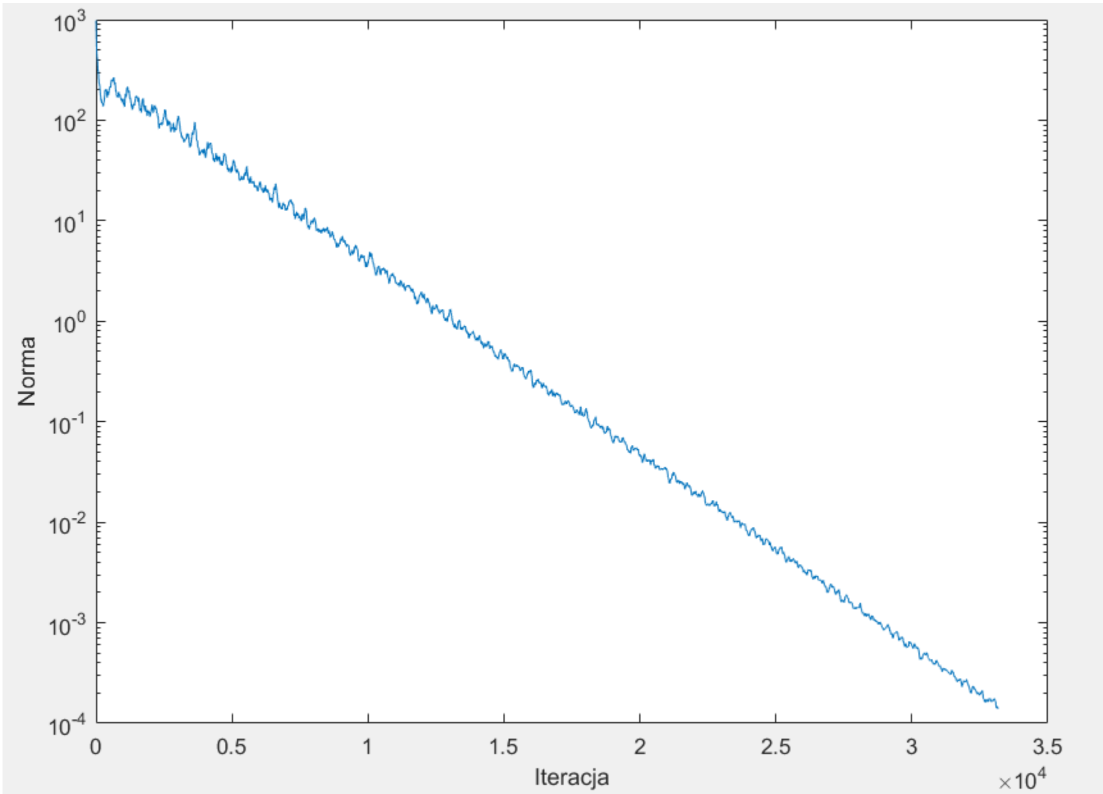
- $||\nabla f(x_k)|| < \epsilon$ (czerwony)
- $||d_k|| < \epsilon$ (zielony)
- $||x_{k+1} - x_k|| < \epsilon$ (niebieski)

, $\epsilon = 10^{-5}$, $p1 = 0$, $p2 = 3$, algorytm - gold

Można zauważyć, że te warunki są sobie prawie równoważne, tj. wykresy mają taką samą charakterystykę. Najszybciej kończącym jest warunek na różnicę



Wycinek 3: Warunki stopu



Wycinek 4: Warunki stopu - norma

wartości w kolejnych krokach. Liczba iteracji ma tendencję rosnącą z rozmiarem macierzy.

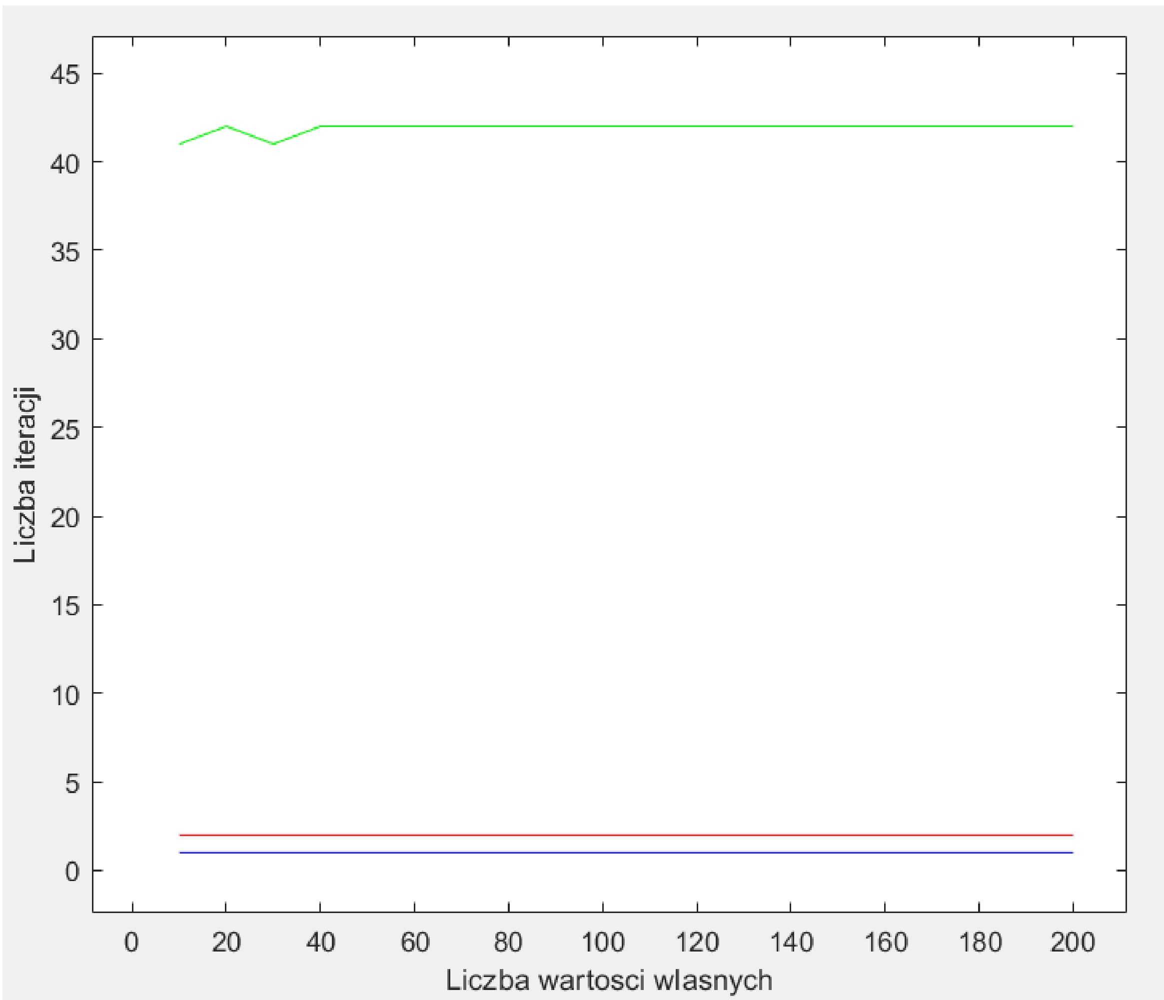
Ponadto dla $n = 100$ oraz warunku na gradient został wykonany wykres normy gradientu od liczby iteracji (Wycinek 4). Warunki stopu nie mają wpływu na liczenie normy gradientu, więc ich porównywanie nic nie wniesie.

4.3 Takie same wartości własne

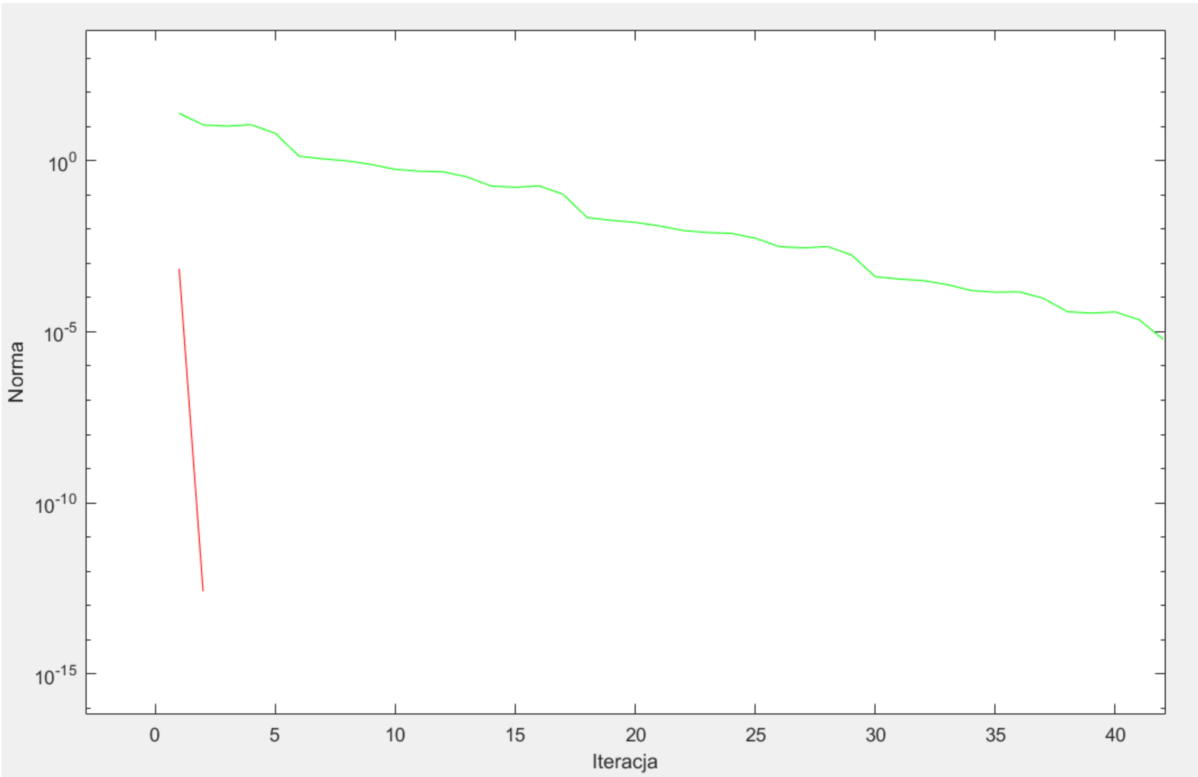
Porównane zostały różne algorytmy dla macierzy z takimi samymi wartościami własnymi (Wycinek 5):

- gold (czerwony)
- Armijo (zielony)
- fminbnd (niebieski)

, $\epsilon = 10^{-5}$, wartości własne = 6



Wycinek 5: Takie same wartości własne



Wycinek 6: Takie same wartości własne - norma

Widać że liczba iteracji jest taka sama dla wszystkich rozmiarów. Gold zbiega w 2 iteracje, fminbnd w 1, Armijo w ok. 42. Jest to związane z tym że macierz zawsze wygląda tak samo, różni się tylko rozmiarem. Macierz ta jest zawsze diagonalna. Macierz posiadająca wszystkie takie same wartości własne będzie zawsze diagonalna. W związku z tym gradient prowadzi wprost do rozwiązanie, stąd tak szybkie zbieganie algorytmów.

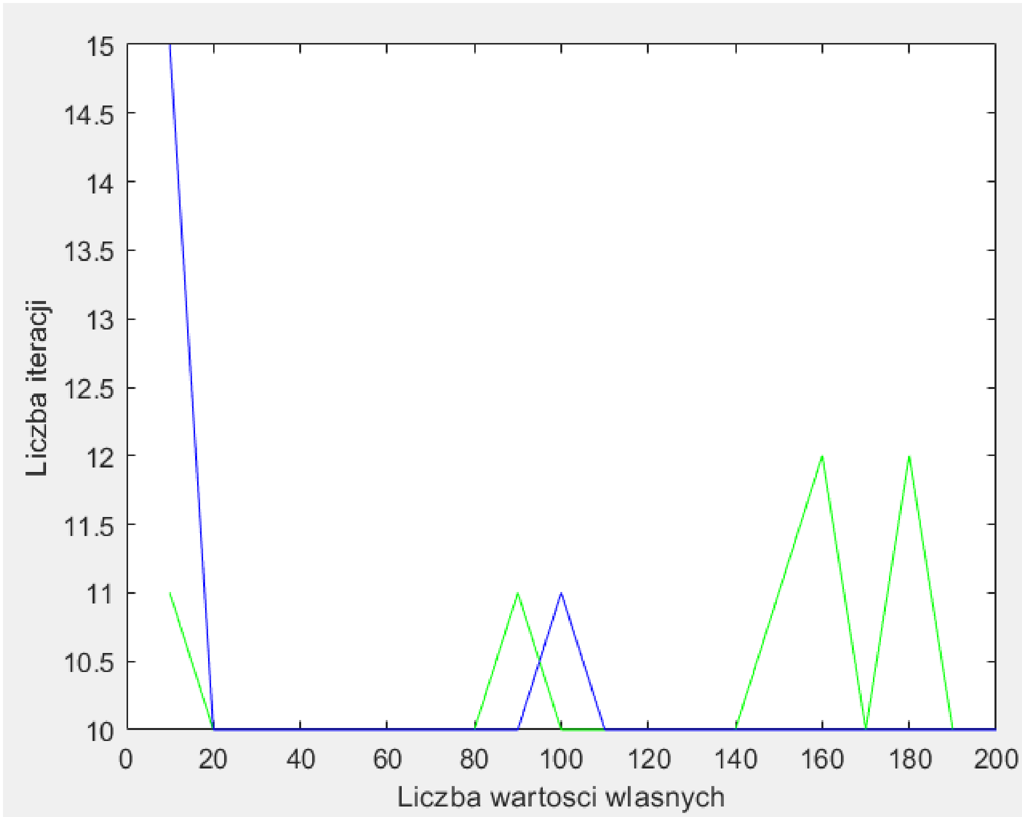
Ponadto dla $n = 100$ oraz warunku na gradient został wykonany wykres normy gradientu od liczby iteracji (Wycinek 6).

Widać że algorytmy zbiegają szybko. Wykresu dla fminbnd nie widać, bo składa się z jednego punktu. Po jednej iteracji norma ma wartość rzędu 10^{-14}.

4.4 Takie same wartości własne (oprócz jednej)

Porównane zostały różne algorytmy dla macierzy z prawie wszystkimi takimi samymi wartościami własnymi, różniła się jedna z wartości własnych (Wycinek 7):

- gold (czerwony)



Wycinek 7: Takie same wartości własne (oprócz jednej)

- Armijo (zielony)
- fminbnd (niebieski)

, $\epsilon = 10^{-5}$, te same wartości własne $= 3$, inna wartość własna $= 30$

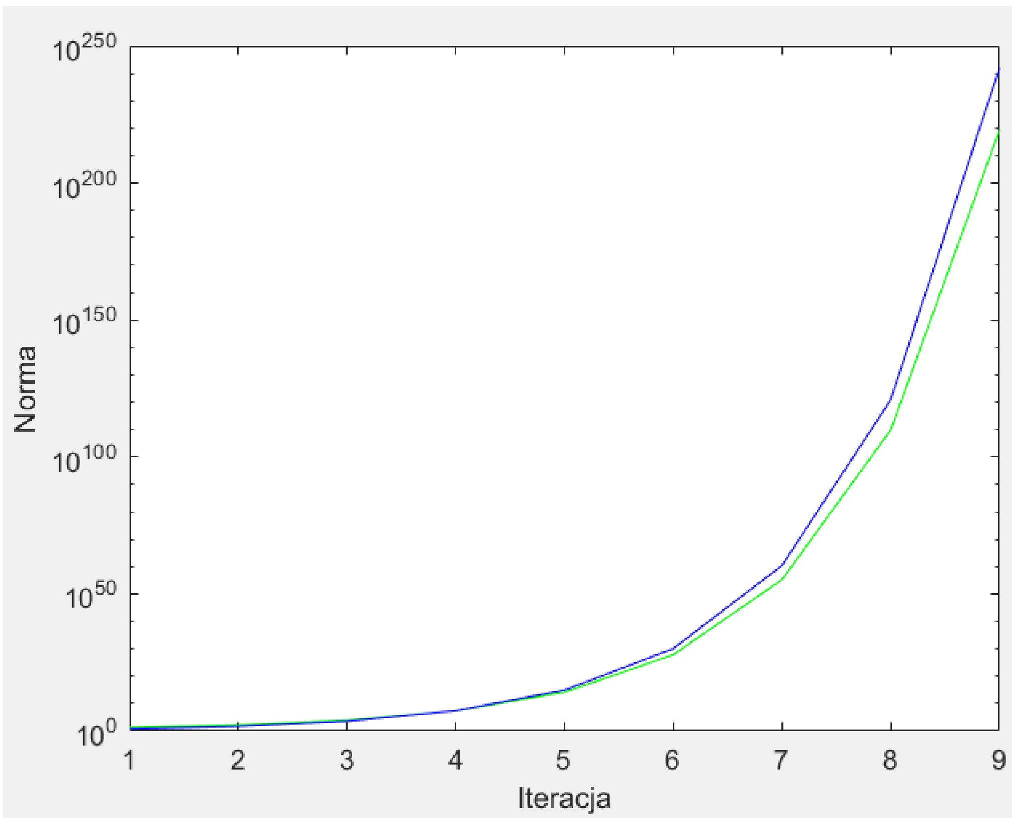
Wykres czerwony pokrywa się z wykresem niebieskim. Jak widać, inna wartość własna wpływa istotnie na zbieżność układu. Rozwiązanie odddala się od minimum. Nie zbiega nigdy.

Ponadto dla $n = 100$ oraz warunku na gradient został wykonany wykres normy gradientu od liczby iteracji (Wycinek 8).

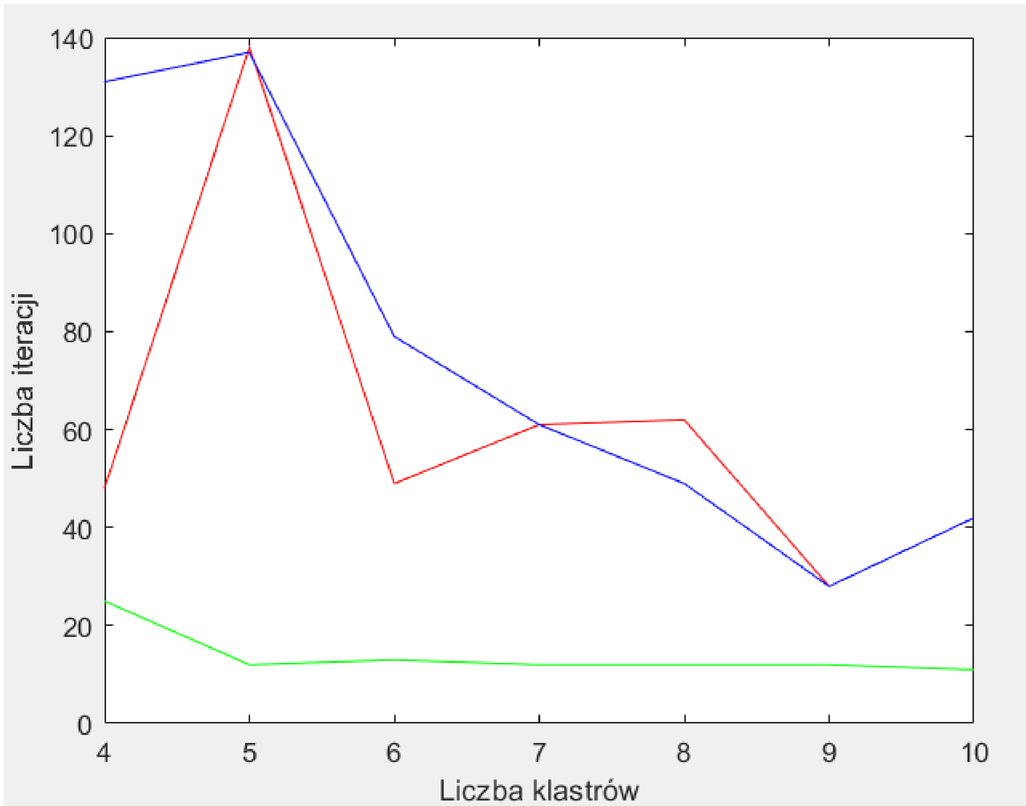
Jak wspomniano wyżej, widać że algorytmy nie zbiegają, w związku z tym ta norma rośnie. Tu również wykres czerwony pokrywa się z niebieskim.

4.5 Wartości własne w klastrach

Porównane zostały różne algorytmy dla macierzy z kilkoma różnymi wartościami własnymi (w klastrach) (Wycinek 9):



Wycinek 8: Takie same wartości własne (oprócz jednej) - norma



Wycinek 9: Kilka wartości własnych)

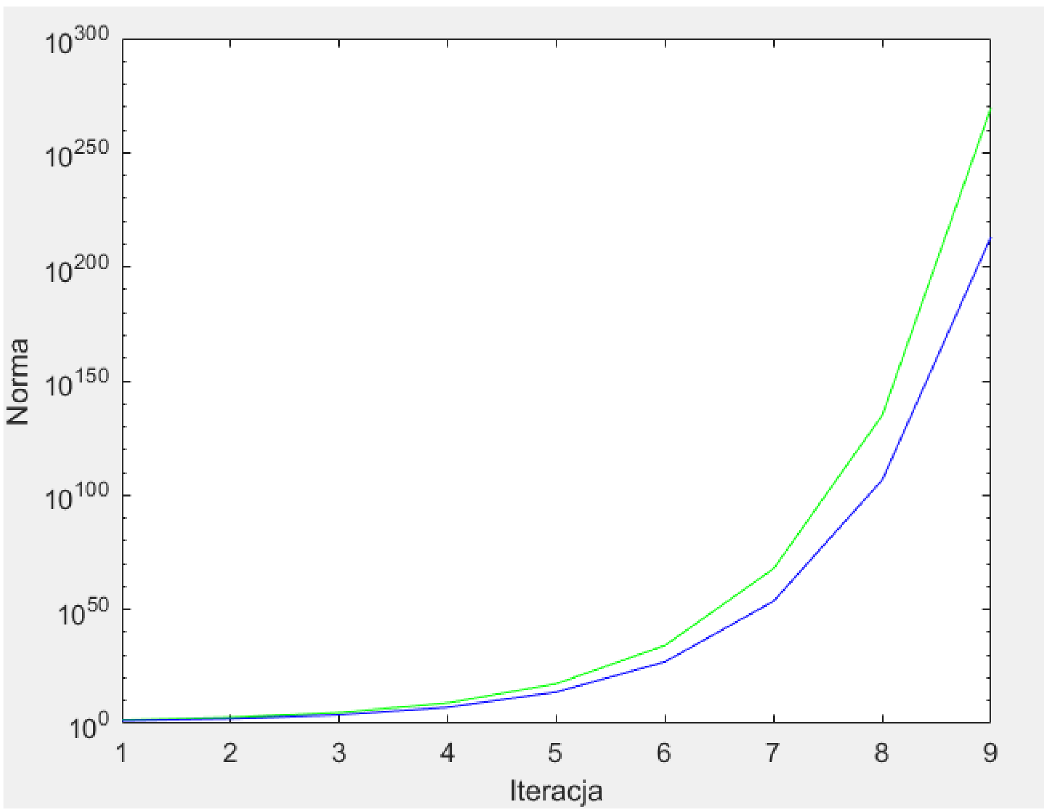
- gold (czerwony)
- Armijo (zielony)
- fminbnd (niebieski)

, $\epsilon = 10^{-5}$, kolejne wartości własne różnią się o 3, $n = 100$

Tu ponownie algorytmy nie zbiegają.

Ponadto dla $n = 100$ oraz warunku na gradient został wykonany wykres normy gradientu od liczby iteracji (Wycinek 10).

Jak wspomniano wyżej, widać że algorytmy nie zbiegają, w związku z tym ta norma rośnie.



Wycinek 10: Kilka wartości własnych - norma

5 Dodatkowe uwagi i wnioski

1. Algorytm Armijo jest bardzo zależny od parametrów. Ustalając parametry wybiera się pomiędzy czasem działania, a dokładnością rozwiązania. Parametry użyte do testów zostały wyznaczone empirycznie.

2. Część kłopotów z własnościami numerycznymi sprawiła najprawdopodobniej macierz A. Gdyby udało się wygenerować macierz o trochę bardziej zbliżonych wartościach, algorytmy mogłby działać nieco lepiej.

6 Pliki

- skrypt.m - skrypt prezentujący wywołania funkcji i realizujący podstawowe wymagania
- alfa_max - zawiera funkcję znajdującą α_{max}
- armijo.m - zawiera funkcję realizującą algorytm Armijo
- FR.m - zawiera funkcję realizującą algorytm gradientów sprzężonych
- fun.m - zawiera funkcję zwracającą funkcję do minimalizacji, jej gradient oraz hesjan
- WygenerujMacierz.m - zawiera funkcję generującą macierz spełniającą warunki zadania
- WygenerujMacierzWartosciWlasne.m - zawiera funkcję generującą macierz dla zadanych wartości własnych
- Testy/TestFR_algorytmy.m - testy porównujące gold, Armijo i fminbnd
- Testy/TestFR_warunek_stopu.m - testy porównujące różne warunki stopu
- Testy/TestFR_wartosci_wlasne_1.m - weryfikacja rozwiązań dla takich samych wartości własnych
- Testy/TestFR_wartosci_wlasne_2.m - weryfikacja rozwiązań dla takich samych wartości własnych (oprócz jednej)
- Testy/TestFR_wartosci_wlasne_3.m - weryfikacja rozwiązań dla wartości własnych w kilku klastrach