Programowanie matematyczne - zadanie 5

Paulina Przybyłek

Grupa laboratoryjna C
9 grudnia 2022

Oświadczam, że niniejszy raport wraz z załączonymi m-plikami zostały wykonane przez mnie samodzielnie.

1 Treść zadania

Zadanie polega na znajdowaniu rozwiązania optymalnego (RO) dla nieosobliwego kwadratowego układu równań liniowych $Ax = b$, gdzie $A \in \mathcal{R}^{nxn}$, $b \in \mathcal{R}^n$, przy czym A jest symetryczna i dodatnio określona o losowych współczynnikach całkowitych z przedziału $[p_1, p_2]$. W rozwiązaniu zadania przyjęto przedział $[-10, 10]$ i parametr $n = 10$. Rozwiązanie to może być znajdowane wprost bądź poprzez minimalizację wypukłej funkcji kwadratowej odpowiadającej temu układowi. Rozważa się w zadaniu poniższe metody uzyskania RO:

- dokładne rozwiązanie - rozwiązanie układu równań $x = A\backslash b$;
- wbudowana funkcja *fminunc* w Matlabie;
- metoda najszybszego spadku wykorzystująca analityczny wzór dla funkcji kwadratowej do obliczania kroku minimalizacji;
- algorytm quasi-newtonowski DFP wykorzystujący analityczny wzór dla funkcji kwadratowej do obliczania kroku minimalizacji;
- algorytm quasi-newtonowski DFP stosujący algorytm złotego podziału do obliczania kroku minimalizacji;

Za punkt początkowy potrzebny do niektórych metod przyjmowano wektor złożony z samych zer.

Pierwsze trzy punkty z powyższej listy metod zastosowano podczas zajęć, dlatego raport nie zawiera dokładnych opisów wykonanych kroków podczas ich implementacji. Podsumowując to działanie - udało się otrzymać normy różnic RO między dokładnym rozwiązaniem a zaimplementowanymi metodami na bardzo małym poziomie, co dowodzi poprawności działania implementacji. Liczba iteracji w metodzie najszybszego spadku okazała się dość duża.

1.1 Algorytm złotego podziału

Algorytm złotego podziału jest iteracyjny, oparty o wybór dwóch punktów próbnych x_k^1 i x_k^2. Na wstępnie otrzymywany jest cały przedział niepewności $[0, \alpha_{max}]$, gdzie wartość α_{max} uzyskuje się za pomocą algorytmu w podsekcji 1.1.1. Jeden z punktów próbnych położony jest wewnątrz zredukowanego nowego przedziału, wówczas druga część przedziału - zawierająca punkt z większą wartością rozważanej funkcji:

- jeśli $f(x_k^1) < f(x_k^2)$ to eliminowany jest odcinek $[x_k^2, \alpha_{max}]$,
- natomiast jeśli $f(x_k^1) >= f(x_k^2)$ to eliminowany jest odcinek $[0, x_k^1]$.

W każdej iteracji, wyznaczany podprzedział obejmujący poszukiwane minimum jest zmniejszany o stały współczynnik $c \approx 0.61804$. Algorytm wyznacza kolejne wartości x_k^1 i x_k^2 oraz zmniejsza przedział przeszukiwań minimum aż do spełnienia warunku stopu będący normą różnicy punktów granicznych przedziału mniejszą od przyjętej tolerancji. Wówczas szukany punkt to średnia z tych punktów granicznych.

1.1.1 Wyznaczanie α_{max}

Algorytm wyznacza przedział niepewności $[0, \alpha_{max}]$, czyli wylicza dla podanej funkcji wartość prawego końca przedziału α_{max}. Polega na wyznaczaniu trzech punktów poczynając od 0 i przesuwając się prawo o pewną wartość Δ (przy czym Δ się zmienia w trakcie), aż do spełnienia warunku $3P$, gdzie to wybrane trzy punkty $a_1 < a_2 < a_3$ spełniają warunki $f(a_1) > f(a_2)$ i $f(a_2) < f(a_3)$, gdzie $f()$ to funkcja, którą rozważamy. Wówczas, gdy warunki zachodzą, to α_{max} równa się wartości a_3.

1.2 Algorytm quasi-newtonowski DFP

Algorytm szukający rozwiązania optymalnego minimalizacji funkcji przedstawionej w zadaniu. Polega na aproksymacji macierzy odwrotności hesjanu H. Algorytm startuje dla H będącej macierzą jednostkową. Warunkiem stopu algorytmu jest norma gradientu funkcji dla danego punktu x_k uzyskanego w k-tej iteracji mniejsza od zadanej tolerancji. Wówczas x_k jest RO funkcji. Zanim jednak dojdzie się do rozwiązania optymalnego obliczane w każdej iteracji są kolejne wartości x_{k+1}, gdzie

$$x_{k+1} = x_k + \alpha(-H_k \nabla f(x_k))$$

gdzie α uzyskiwane jest w sposób analityczny bądź za pomocą algorytmu złotego podziału. Poprawiana jest macierz H o poprawkę ΔH daną wzorem:

$$H_{k+1} = H_k + \Delta H_k = H_k + \frac{s_k s_k^T}{s_k^T r_k} - \frac{H_k r_k r_k^T H_k}{r_k^T H_k r_k}$$

gdzie $s_k = x_{k+1} - x_k$, $r_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Następnie przechodzi się do kolejnej iteracji. Poza wspomnianym warunkiem stopu dodano jeszcze jeden warunek będący normą $s_k = x_{k+1} - x_k$ mniejszą od zadanej tolerancji, czyli jak różnica między kolejnymi punktami jest już niewielka to algorytm ma zakończyć szukanie, bo nie poprawia się.

2 Rozwiązanie Matlab

Rozwiązanie zadania zawiera się siedmiu plikach .m. Plik *fun.m* zawiera funkcje implementującą funkcję kwadratową od x dla danego A i b - zwraca ona wartość funkcji, jej gradient oraz hesjan. Plik *alfa_max.m* zawiera procedurę wyznaczania przedziału niepewności dla minimalizacji jednokierunkowej, a plik *gold.m* zaimplementowany algorytm złotego podziału. W pliku *DFP.m* zaimplementowano algorytm quasi-newtonowski DFP. W pliku *script.m* są przykładowe wywołania dla każdego etapu zadania (różnych wykorzystywanych metod). Komentarze krótko przedstawiają czego dotyczy dana część. Natomiast pliki *testy1.m* i *testy2.m* zostały wykorzystane do realizacji części testowej opisanej w kolejnej sekcji.

2.1 Algorytm złotego podziału

Sprawdzono algorytm dla różnych wartości tolerancji stosowanej do warunku stopu. Rysunek 1 przedstawia wartości tolerancji, liczbę iteracji oraz uzyskaną wartość kroku dla takiej samej rozważanej funkcji i tego samego przedziału. Można zauważyć, że wartość kroku zmienia się nieznacznie już od tolerancji równej 1e-5. Można by przyjąć dokładność obliczeń jako właśnie 1e-5 bądź nawet 1e-4 jeśli chcemy mniejszą dokładność wyniku a mniejszą liczbę iteracji. Oczywiście jest to przykład na tym jednym przypadku, w testach przyjęta wartość tolerancji to 1e-6.

2.2 Algorytm quasi-newtonowski DFP

Sprawdzono algorytm dla różnych wartości tolerancji stosowanej do warunków stopu. Rysunek 2 przedstawia wartości tolerancji, liczbę iteracji dla sposobu uzyskiwania wartości α w sposób analityczny i za pomocą algorytmu złotego podziału oraz wartość normy różnicy RO dla x uzyskanego przy pomocy wzoru analitycznego i algorytmu złotego podziału. Porównując oba podejścia to nie ma zmian w iteracjach podejścia ze wzorem analitycznym, natomiast dla algorytmu złotego podziału liczba iteracji

rośnie, mimo że na początku była dość stała i podobna do drugiej metody. Cały czas są też różnice w obliczeniach RO. Duża różnica w iteracjach (dla złotego podziału) i sporo mniejsza w wartościach normy pojawia się dopiero po tolerancji równej 1e-7, stąd można by wziąć to za dobrą wartość. Oczywiście jest to przykład na tym jednym przypadku, w testach przyjęto różne wartości tolerancji.

```
Tolerancja: 1e-2, Liczba iteracji: 10, Wartość kroku: 0.27786420
Tolerancja: 1e-3, Liczba iteracji: 14, Wartość kroku: 0.27912036
Tolerancja: 1e-4, Liczba iteracji: 19, Wartość kroku: 0.27936357
Tolerancja: 1e-5, Liczba iteracji: 24, Wartość kroku: 0.27934944
Tolerancja: 1e-6, Liczba iteracji: 29, Wartość kroku: 0.27934778
Tolerancja: 1e-7, Liczba iteracji: 33, Wartość kroku: 0.27934796
Tolerancja: 1e-8, Liczba iteracji: 35, Wartość kroku: 0.27934799
```

Rysunek 1: Wartość kroku i liczba iteracji uzyskana przez algorytm złotego podziału dla ustalonej tolerancji.

```
Tolerancja: 1e-2, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 11, Wartość normy RO: 0.07487790
Tolerancja: 1e-3, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 12, Wartość normy RO: 0.00001527
Tolerancja: 1e-4, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 12, Wartość normy RO: 0.00001527
Tolerancja: 1e-5, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 12, Wartość normy RO: 0.00001527
Tolerancja: 1e-6, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 13, Wartość normy RO: 0.00001300
Tolerancja: 1e-7, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 28, Wartość normy RO: 0.00000035
Tolerancja: 1e-8, Liczba iteracji wzór analityczny: 11, Liczba iteracji gold: 31, Wartość normy RO: 0.00000028
```

Rysunek 2: Wartość normy różnicy RO dla obu metod i ich liczba iteracji uzyskana przez algorytm quasi-newtonowski DFP dla ustalonej tolerancji.

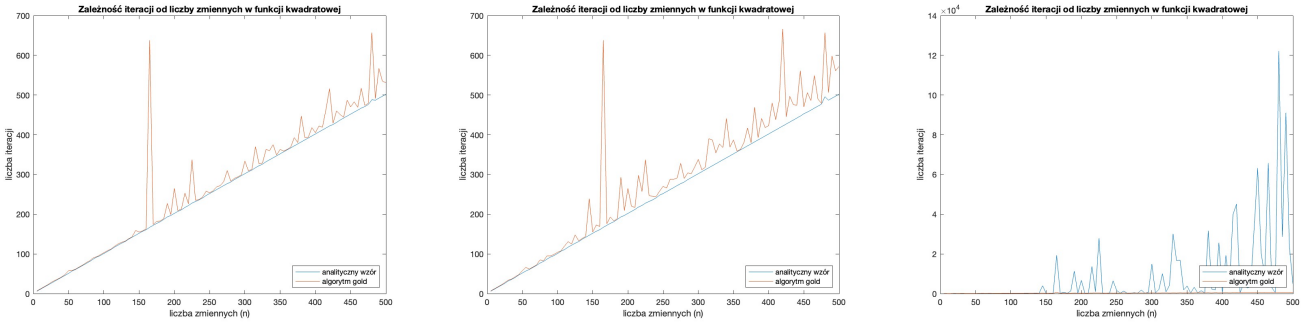
3 Testy

W ramach zadania przeprowadzono dwa większe testy, których analizy przedstawiono w tej sekcji.

3.1 Eksperyment 1

Pierwszy test (plik *testy1.m*) zawiera badanie zależności iteracji algorytmu quasi-newtonowskiego DFP od liczby zmiennych n w rozważanej funkcji kwadratowej. Dodatkowo obliczano wartość normy różnicy dokładnego rozwiązania i uzyskanego przez ten algorytm oraz wartość normy różnicy macierzy odwrotności hesjanów dla tych dwóch podejść. Powtórzono test 100 razy zmieniając parametr n od wartości 5 do 500 z krokiem równym 5. Rysunek 3 pokazuje zależność iteracji od tego parametru w zależności od ustalonej tolerancji (patrz opis pod rysunkiem). Można zauważyć, że po przekroczeniu wartości 140 dla pierwszego wykresu po lewej liczba iteracji dla algorytmu z gold robi się momentami bardzo duża, przy czym zawsze jest większa niż algorytmu a analitycznym wzorem. W przypadku zwiększenia trochę tolerancji (wykres środkowy) „pików" na wykresie robi się coraz więcej dla gold, natomiast dla analitycznego wzoru za dużo się nie zmienia. Co innego przedstawia wykres trzeci (po prawej) - wówczas liczba iteracji dla analitycznego jest dużo większa niż drugiej metody. Mimo to po przekroczeniu 140 zmiennych algorytmy zachowują się już dużo gorzej.

Rysunek 4 przedstawia średnie wyniki norm różnic dla odwrotności hesjanów i RO obu wariantów algorytmu quasi-newtonowskiego DFP dla tolerancji równej 1e-4. Różnice te są z rozwiązaniem dokładnym układu, przy czym odwrotność hesjanu jest liczona za pomocą funkcji wbudowanej *inv*. Jak widać wartości te są duże poza normą różnic RO dla rozwiązania dokładnego i analitycznego wzoru, a nawet można by rzec że bardzo duże. Jeśli rozwiązanie optymalne różni się o taką wartość od dokładnego rozwiązania to trudno nazwać je optymalnym. Jednak mediana tych norm różnicy rozwiązań między dokładnym a uzyskanym przez algorytm z gold wynosi 0.0034. Dodatkowo sprawdzono, że spora część zmiennych ma takie właśnie zbliżone wartości. Analogicznie jest z medianą dla norm odwrotności hesjanów - dla gold wartość ta wynosi 3.8367 a dla analitycznego rozwiązania 0.0014. Rysunek 5 pokazuje te same wartości, ale jedynie do $n = 100$, czyli pierwsze 20 iteracji. Jak widać na małych wartościach algorytm działa dobrze, wyniki są zbliżone. To samo widać na wykresie iteracji - do małych liczb n nie ma „pików" i wartości te liczą się dobrze.



Rysunek 3: Wykresy zależności liczby iteracji od liczby zmiennych w funkcji dla tolerancji równej (od lewej) 1e-3, 1e-4, 1e-8.

```
Średnia wartość normy różnicy RO (gold): 143051.2988
Średnia wartość normy różnicy RO (analiztyczny wzór): 0.0485
Średnia wartość normy różnicy odwrotności hesjanu (gold): 64983.5278
Średnia wartość normy różnicy odwrotności hesjanu (analiztyczny wzór): 32448.7684
```

Rysunek 4: Wartości średnie norm uzyskanych rozwiązań i odwrotności hesjanów dla rozważanych dwóch wariantów algorytmu quasi-newtonowskiego DFP dla tolerancji 1e-4

```
Średnia wartość normy różnicy RO (gold): 0.0210
Średnia wartość normy różnicy RO (analiztyczny wzór): 0.0001
Średnia wartość normy różnicy odwrotności hesjanu (gold): 0.3216
Średnia wartość normy różnicy odwrotności hesjanu (analiztyczny wzór): 0.8494
```

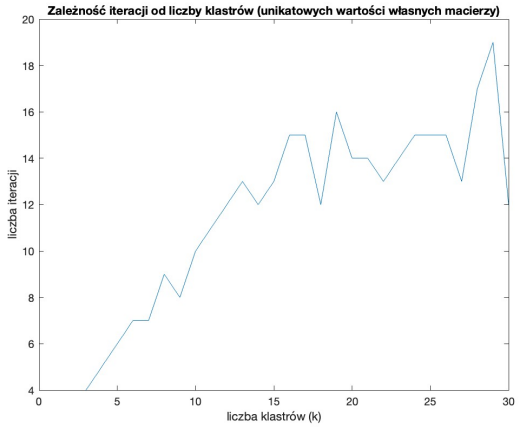
Rysunek 5: Wartości średnie norm uzyskanych rozwiązań i odwrotności hesjanów dla rozważanych dwóch wariantów algorytmu quasi-newtonowskiego DFP dla tolerancji 1e-4, jednak dla pierwszych 10 iteracji, czyli do $n = 100$

3.2 Eksperyment 2

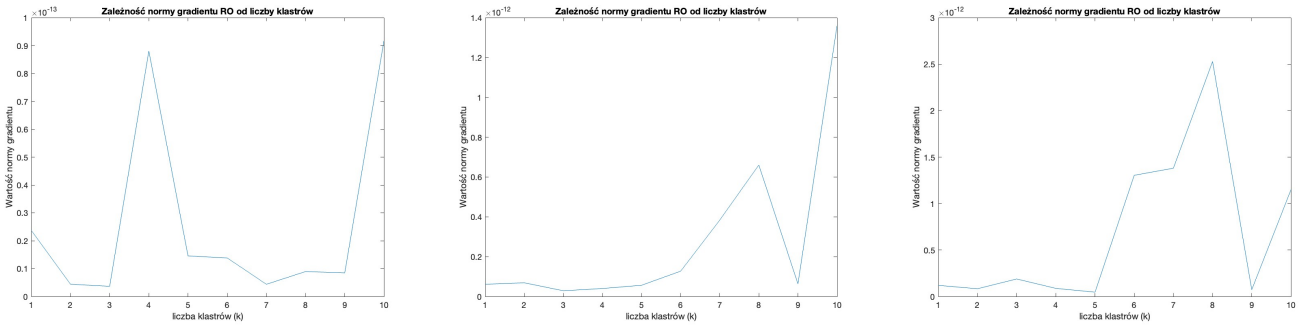
Drugi test (plik *testy2.m*) zawiera analizę dla tego samego układu (bądź funkcji) jak w reszcie zadania, jednak inna jest macierz A. Mianowicie A jest symetryczna dodatnio określona i narzucono jej wektor wartości własnych **całkowitych** z zakresu $[1, p]$. Taka macierz tworzona jest jako $A = QDQ^T$, gdzie macierz D to macierz diagonalna, gdzie wszędzie są zera a na przekątnej ma ona wartości własne A. Natomiast macierz Q to macierz ortonormalna kwadratowa.

Tak więc test polega na sprawdzeniu liczby iteracji algorytmu DFP z krokiem analitycznym w zależności od liczby k klastrów wartości własnych. To znaczy, że macierz A posiada dokładnie k unikatowych wartości własnych (czyli mogą się one powtarzać). Wartości są losowane z odrębnych przedziałów, więc każda losowana wartość jest większa niż poprzednia, dodatkowo wspomniane wcześniej $p = 10k$. Rysunek 6 pokazuje zależność iteracji tego algorytmu od liczby klastrów, czyli tych unikatowych wartości własnych dla $n = 30$ (liczba zmiennych w funkcji). Liczba iteracji jest zmienna. Na postawie tego testu nie można uznać, że zawsze tak jest, jednak widać, że iteracje rosną wraz z liczbą unikatowych wartości własnych mimo iż momentami następuje chwilowy spadek tej liczby.

Rysunek 7 pokazuje natomiast dla innych danych już, gdzie k=1,. . . ,10, zależność normy gradientu RO dla $n = 10$ (po lewej), $n = 50$ (środek) i $n = 100$. Nie można jednoznacznie powiedzieć czegoś o zachowaniu tej wartości. Ewentualnie tyle, że dla małych k (mniejszych niż 5) i dużych n, wartości te są zbliżone do siebie i bardzo małe. Poza tym warto zauważyć, że wartości tych gradientów są bardzo niskie (1e-12 czy 1e-13).



Rysunek 6: Zależność liczby iteracji od liczby klastrów wartości własnych dla algorytmu DFP z krokiem analitycznym



Rysunek 7: Zależność wartości normy gradientu od liczby klastrów wartości własnych dla algorytmu DFP z krokiem analitycznym dla $n = 10$ (po lewej), $n = 50$ (środek) i $n = 100$