Programowanie matematyczne
Laboratorium 4

Piotr Widomski, grupa D

21.11.2022

1 Postać ZPN

Rozwiązywane zadanie programowania nieliniowego ma następującą postać:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T A x - b' x$$
$$x, b \in \mathbb{R}^n,\ A \in \mathbb{R}^{n \times n}$$

Macierz A jest macierzą symetryczną dodatnio określoną o współczynnikach całkowitych. Dodatkowo $Ax = b$ jest nieosobliwym układem.

Do generowania macierzy A wykorzystana została właściwość, że jeżeli macierz D jest odwracalna, to macierz $D^T D$ jest odwracalna, symetryczna i dodatnio określona.

Funkcja $f(x) = \frac{1}{2} x^T A x - b' x$ z ZPN ma następujące właściwości:

$$f'(x) = Ax - b$$
$$f''(x) = A$$

Wynika z tego, że macierz Hessego dla funkcji f jest dodatnio określony.

2 Zastosowane algorytmy

Ogólny schemat iteracyjny rozwiązywania ZPN wygląda następująco:

$$x_0 - \text{punkt startowy}$$
$$x_{k+1} = x_k + \alpha_k d_k,\ (k = 0, 1, \dots),$$

gdzie d_k jest kierunkiem poprawy funkcji, a $\alpha_k > 0$ długością kroku w kierunku d_k.

W każdej iteracji wyróżnione są dwa etapy:

- znalezienie kierunku poprawy d_k,
- znalezienie długości kroku α_k minimalizującego wartość funkcji dla kroku x_{k+1}, czyli $\alpha_k = \min_\alpha F(\alpha) = f(x_k + \alpha d_k)$

2.1 Wyznaczanie przedziału niepewności

Do wyznaczenia α_k użyta została metoda eliminacji przedziałów. Poszukiwany jest przedział $[a_1, a_3]$, taki że istnieje $a_2 \in [a_1, a_3]$ i $F(a_1) > F(a_2), F(a_2) < F(a_3)$ (warunek trzech punktów). Wtedy w przedziale $[a_1, a_3]$ znajduje się minimum funkcji f.

Algorytm wyznaczania przedziału niepewności wygląda następująco: $a_1 = 0$, $\overline{a} = a_1 + \Delta$

- jeżeli $F(a_1) \leqslant F(\overline{a})$, to $a_3 = \overline{a}$.
- jeżeli $F(a_1) > F(\overline{a})$, to $a_2 = \overline{a}$ i $a_3 = a_2 + \Delta$. Jeżeli punkty a_1, a_2, a_3 nie spełniają warunku trzech punktów, to wykonujemy $\Delta = 2\Delta$ oraz $a_3 = a_2 + \Delta$, aż do jego spełnienia.

Wynikowy przedział niepewności to $[a_1 = 0, a_3]$. Jako iż w wyniku nie używane jest a_2, pominięte zostało obliczanie jego wartości w pierwszym przypadku. W dalszych obliczeniach Δ zostało ustalone na 0,05.

2.2 Złoty podział

W celu znalezienia wartości α_k posiadając przedział niepewności wyznaczony z wcześniejszej metody użyta została metoda złotego podziału.

Algorytm oparty jest o wybór dwóch punktów próbnych i iteracyjną eliminację części rozważanego przedziału. W każdej iteracji rozważamy przedział $[a_k, b_k]$. Wyznaczane są punkty x_1, x_2, takie stosunek odległości x_1 od a_k do długości przedziału jest równy współczynnikowi c złotego podziału ($c = \frac{\sqrt{5}-1}{2}$) (analogicznie z punktem x_2 i b_k).

Następnie na podstawie znaku nierówności między $f(x_1)$ i $f(x_2)$ eliminowany jest jeden z przedziałów, w którym nie może znajdować się minimum. Dla $f(x_1) < f(x_2)$ eliminowany jest przedział $[x_2, b_k]$. W przeciwnym przypadku eliminowany jest przedział $[a_k, x_1$. W każdej iteracji przedział zawierający poszukiwane minimum zmniejsza się o stały współczynnik c.

Iteracje wykonywane są aż do momentu osiągnięcia maksymalnej liczny iteracji, lub gdy długość rozważanego przedziału jest mniejsza niż ϵ. Znalezionym minimum jest wtedy $\overline{x} = \frac{a_k + b_k}{2}$

2.3 BFGS

Do rozwiązywania ZPN użyta została quasi-Newtonowska metoda BFGS. Metoda Newtona oblicza kierunek poprawy

$$d_k = -[\nabla^2 f(x_k)]^{-1} \cdot \nabla f(x_k)$$

Metody quasi-Newtonowskie zastępują liczenie odwrotności macierzy Hessego poprzez iteracyjną ewaluację:

$$[\nabla^2 f(x_k)]^{-1} \approx H_k$$
$$[\nabla^2 f(x_{k+1})]^{-1} \approx H_{k+1}$$
$$H_{k+1} = H_k + \Delta H_k$$

Spełniony musi być warunek quasi-newtonowski dla funkcji kwadratowej:

$$A^{-1} r_k = s_k,$$

gdzie $s_k = x_{k+1} - x_k = \alpha_k d_k$ i $r_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

W metodzie BFGS wartość H_k wyznaczana jest następująco:

$$H_{k+1} = H_k + \Delta H_k = \left(I - \frac{s_k r_k^T}{s_k^T r_k}\right) H_k \left(I - \frac{r_k s_k^T}{s_k^T r_k}\right) + \frac{s_k s_k^T}{s_k^T r_k}$$

Zaimplementowany algorytm przebiega następująco:

1. x_0 - punkt startowy, $H_0 = I$.
2. $k = 0$.
3. Sprawdzenie warunku stopu $||\nabla f(x_k)|| < \epsilon$.
4. $d_k = -H_k \nabla f(x_k)$.
5. Rozwiązanie zadania minimalizacji kierunkowej. Wywołanie algorytmu definiuje, czy użyte jest rozwiązanie analityczne ($\alpha_k = \frac{-\nabla f(x_k) d_k}{d^T A d}$), czy opisana wcześniej metoda złotego podziału z $\epsilon = 10^{-6}$.
6. Sprawdzenie warunku stopu $\alpha_k < \epsilon$ i $||\alpha_k d_k|| < \epsilon$.
7. $x_{k+1} = x_k + \alpha_k d_k$.
8. Obliczenie H_{k+1}.
9. $k = k + 1$ i przejście do kroku 3.

3 Przykład obliczeniowy

Dla przykładowej instancji problemu o wartościach:

$$A = \begin{bmatrix} 19 & 15 & 20 & 14 & 21 \\ 15 & 19 & 20 & 14 & 23 \\ 20 & 20 & 32 & 20 & 28 \\ 14 & 14 & 20 & 14 & 19 \\ 21 & 23 & 28 & 19 & 30 \end{bmatrix}$$
$$b = \begin{bmatrix} 3 & 3 & 1 & 2 & 3 \end{bmatrix}^T$$

Rozwiązanie dokładne znalezione przez rozwiązanie układu równań $Ax = b$ wynosi:

$$x = \begin{bmatrix} 0{,}291\,7 & 0{,}541\,7 & -0{,}416\,7 & 0{,}583\,3 & -0{,}5 \end{bmatrix}^T$$

Następnie rozwiązanie ZPN znalezione zostało przy użyciu algorytmu BFGS z $\epsilon = 10^{-6}$ z punktem startowym $x_0 = 0$. W pierwszej iteracji długość kroku znaleziona przy użyciu algorytmu złotego podziału wyniosła $\alpha_1 = 0{,}009\,87$, znaleziona w 24 iteracjach. Wyszukiwanie było wykonywane z $\epsilon = 10 - 6$, a różnica z wartością znalezioną analitycznie wynosi $2{,}902\,9 \cdot 10^{-8}$.

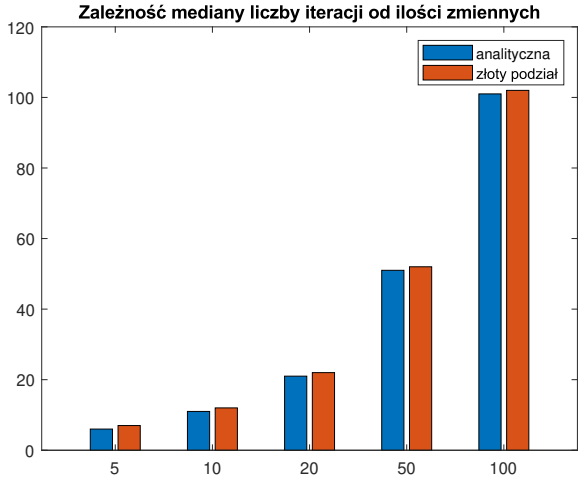
Liczba iteracji dla wariacji algorytmu wykorzystującego metodę złotego podziału oraz metodę analityczną obliczania długości kroku wynosiła 6. Błąd metody analitycznej wyniósł $7{,}655\,1 \cdot 10^{-15}$, a metody złotego podziału $2{,}670\,1 \cdot 10^{-7}$.

Norma różnicy odwrotności hesjanu obliczonego analitycznie z odwrotnością hejsanu z ostatniej iteracji algrorytmu BFGS wyniósł $9{,}879\,1 \cdot 10^{-14}$.

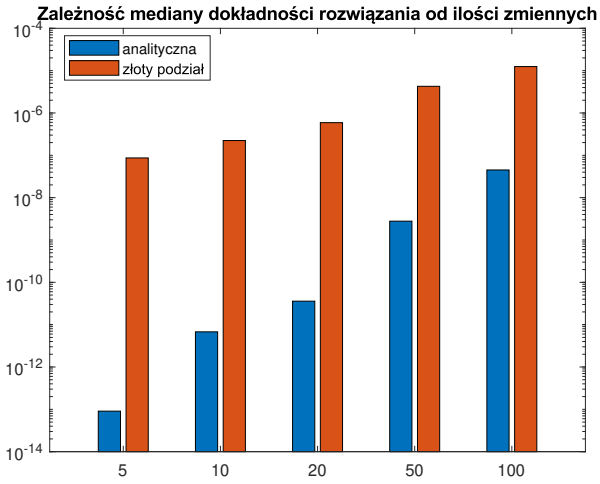
4 Analiza wyników

4.1 Testy 1.

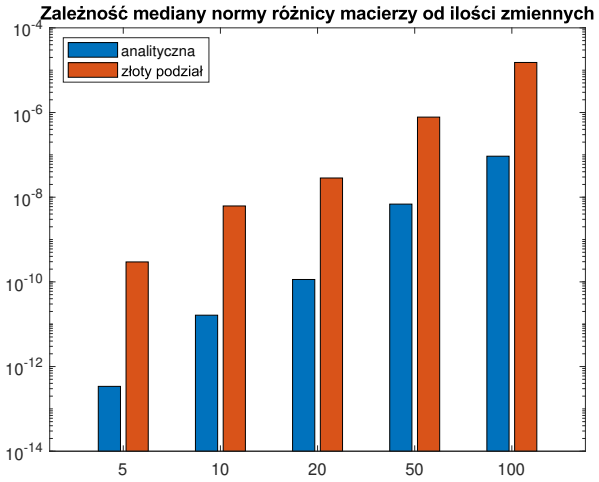
W celu sprawdzenia poprawności wyników zaimplementowanego algorytmu przeprowadzony został test dla $N = 100$ instancji problemu. Sprawdzona została mediana liczny iteracji, normy różnicy uzyskanego wyniku z rozwiązaniem dokładnym oraz normy różnicy odwrotności hesjanu uzyskanego w algorytmie BFGS z obliczonym analitycznie w zależności od rozmiaru problemu. Mediana została wybrana zamiast średniej, ze względu na występowanie złośliwych przypadków, które znacznie zaburzają obliczaną średnią.



Rysunek 1: Mediana iteracji w zależności od rozmiaru problemu.



Rysunek 2: Mediana błędu rozwiązania w zależności od rozmiaru problemu.

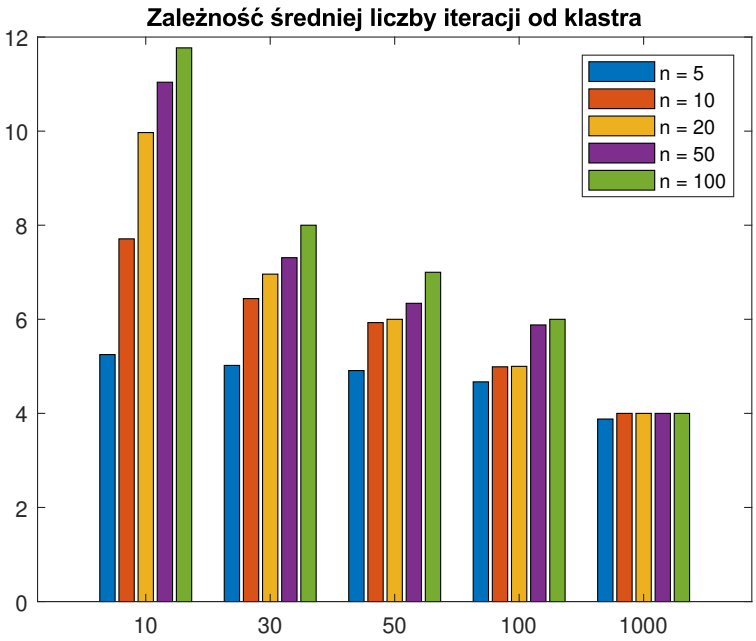


Rysunek 3: Mediana błędu odwrotności hesjanu od rozmiaru problemu.

Algorytm wykazuje liniową zależność liczby iteracji od rozmiaru problemu. Wykorzystanie metody analitycznej do obliczania długości wektora poprawy nie wykazuje dużej różnicy w porównaniu do metody złotego podziału pod względem liczny iteracji. Różnica pojawia się dla błędu rozwiązania i odwrotności hesjanu, gdzie metoda analityczna wykazuje dokładność o kilka rzędów większą. Oba błędy rosną wraz ze wzrostem rozmiaru problemu.

4.2 Testy 2.

W celu zbadania zachowania algorytmu BFGS z krokiem analitycznym w zależności od wartości własnych macierzy A wykonane zostały dodatkowe testy.



Rysunek 4: Średnia iteracji w zależności od klastra i rozmiaru problemu (wielkość klastra 10).

Do generowania macierzy o zadanych wartościach własnych wykorzystany został wzór $A = QDQ'$, gdzie D jest macierzą zawierającą zadane wartości własne na diagonali, a D jest odwracalną zortogonalizowaną macierzą.

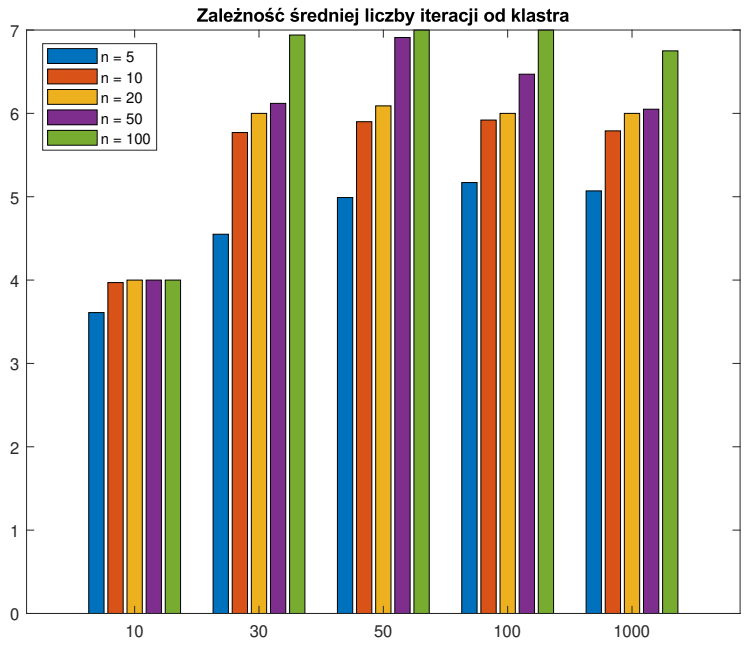
Przygotowane zostało 5 klastrów wartości własnych o wartościach środkowych $10, 30, 50, 100$ i 1000. Dla każdego klastra sprawdzona została średnia liczba iteracji dla rozmiarów problemów $n = 5, 10, 20, 50, 100$, każda dla $N = 100$ instancji problemu o wartościach własnych macierzy A leżących blisko wartości środkowej.

Wpierw sprawdzone zostało zachowanie dla stałego rozmiaru każdego z klastra równego 10. Wyniki przedstawione są na rysunku 4. Wyniki wskazują, że czym większa wartość oczekiwana klastra, tym mniejszy wpływ na liczbę iteracji ma rozmiar problemu.

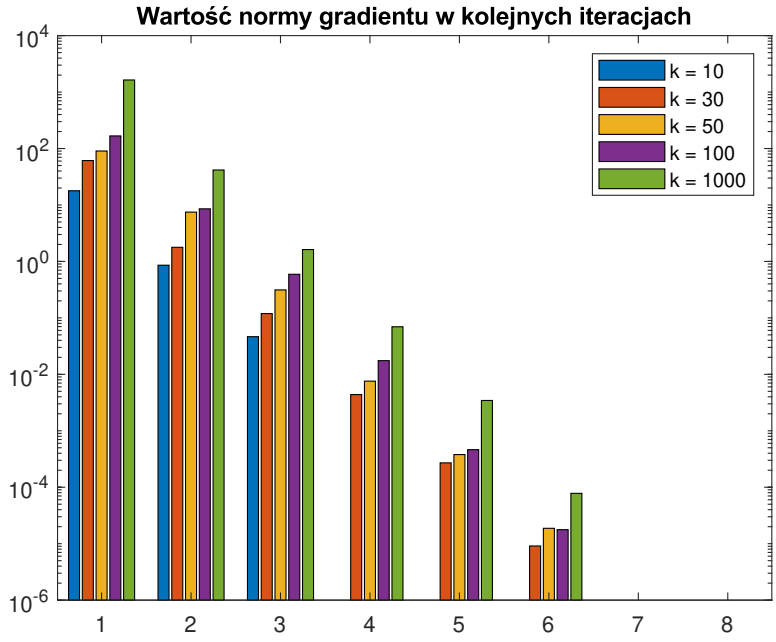
Następnie sprawdzone zostało zachowanie dla rozmiaru każdego z klastra równego 20% wartości środkowej. Wyniki przedstawione są na rysunku 5. W tym przypadku wszystkie klastry wykazały zbliżone wyniki, z najmniejszym klastrem wykazującym zbywalny wpływ rozmiaru problemu na liczbę iteracji.

Postawiona została hipoteza, że ilość iteracji zależy przede wszystkim od wariancji wartości własnych, a rozmiar problemu ma drugorzędny wpływ.

Dodatkowo zbadana została wartość normy gradientu w kolejnych iteracjach w zależności od klastra i ustalonego $n = 10$. Wyniki przedstawione na rysunku 6. wykazują spadek wykładniczy.



Rysunek 5: Średnia iteracji w zależności od klastra i rozmiaru problemu (wielkość klastra 20%).



Rysunek 6: Wartość normy gradientu w kolejnych iteracjach w zależności od klastra ($n = 10$).

5 Oświadczenie

Oświadczam, że niniejsza praca stanowiąca podstawę do uzyskania osiągnięcia efektów uczenia się z przedmiotu "Programowanie Matematyczne" została wykonana przeze mnie samodzielnie.

Piotr Widomski
298919