Lab 7c,
Programowanie matematyczne

Piotr Onyszczuk, gr. C

16 I 2022

1 Treść zadania

Celem zadania jest znalezienia promienia oraz środka najmniejszej kuli zawierającej wszystkie wylosowane punkty z przestrzeni R^3

$$r \in R, s \in R^3 \tag{1}$$

$$\begin{cases} \min\limits_{x=(s,r)} r \\ ||p^i - s|| \leq r, i = 1, ..., m \end{cases} \tag{2}$$

1.1 Postać zadania

Problem można rozwiązać sprowadzając go do rozwiązania zadania programowania kwadratowego

$$\max_{y \in \Omega_y}(-y^T P^T P y + \Sigma_{i=1}^m p_i^T p_i y_i) \tag{3}$$

$$\Omega_y : \begin{cases} \Sigma_{i=1}^m y_i = 1 \\ y \geq 0 \end{cases} \tag{4}$$

Dla $\overline{y}$ będącego RO, $s = P\overline{y}$, $r = \sqrt{fval}$, gdzie $fval$ jest wartością funkcji.

Zadanie rozwiązane zostanie z wykorzystaniem funkcji *quadprog* oraz własnej implementacji algorytmu **IPM**

1.2 Dane testowe

Do testów należy użyte zostały różne liczby losowych punktów o wartościach z przedziału $[-200, 200]$.

1.3 Uzasadnienie postaci ZD

Znajduje się na załączonych odręcznych notatkach.

2 Implementacja

Najpierw liczone są warunki początkowe do funkcji na podstawie punktu początkowego będącego "środkiem ciężkości" wylosowanych punktów. Punkt taki na pewno znajdzie się wewnątrz finalnego rozwiązania.

Dane wejściowe do *quadprog*:

- $H = -2P^TP$ - zanegowana i podwojona macierz z funkcji "max"; musimy zanegować, ponieważ algorytmy rozwiązują funkcję "min"; podwajamy, aby macierz pasowała do schematu $\frac{1}{2}x^Tx$
- $f = -\Sigma_{i=1}^{m} p_i^T p_i$ - zanegowana wartość z funkcji "max"; j.w.
- Aeq - wektor jedynek, $beq = 1$; zapewniamy, aby suma wartości rozwiązania była $= 1$
- lb - wektor zer; zapewniamy aby wartości były ≥ 0

2.1 IPM

Algorytm **IPM** składa się z następujących kroków. Pętla ograniczona przez liczbę iteracji:

1. Sprawdzenie warunków STOPu dla nieograniczonych zadań
2. Obliczenie kroku r (δ-modyfikowana dynamicznie)
3. Obliczenie kierunku kroku d
4. Obliczenie optymalnej długości kroku *alpha*
5. Modyfikacja wektorów
6. Sprawdzenie warunków STOPu w oparciu o *eps*
7. Powrót do 1.

Dodatkowo we wzorach, pozbywamy się Y^{-1} oraz W, ponieważ od razu mamy ograniczenia równościowe.

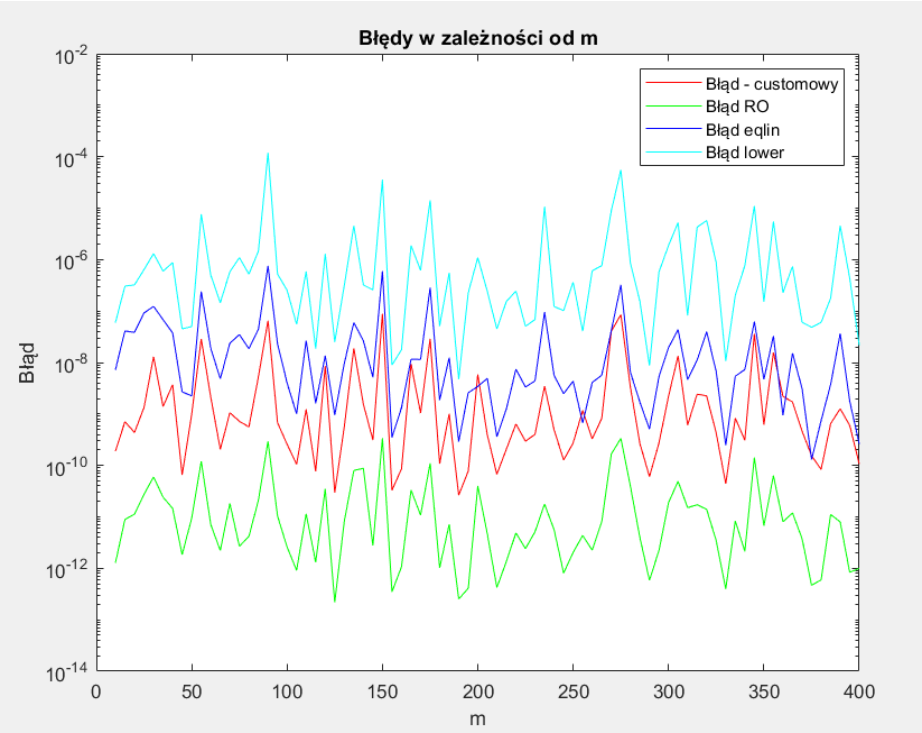
3 Wyniki i wnioski

3.1 Porównanie

Testy zostały przeprowadzone na rozmiarach danych z przedziału $[10, 400]$, z epsilonem ustawionym jako 10^{-6}.

Porównane zostały różnice pomiędzy wynikami quadprog a **IPM** na podstawie czterech wskaźników (Wycinek 1):

- Błąd RO - norma z różnicy pomiędzy zwróconymi RO



Wycinek 1: Błędy

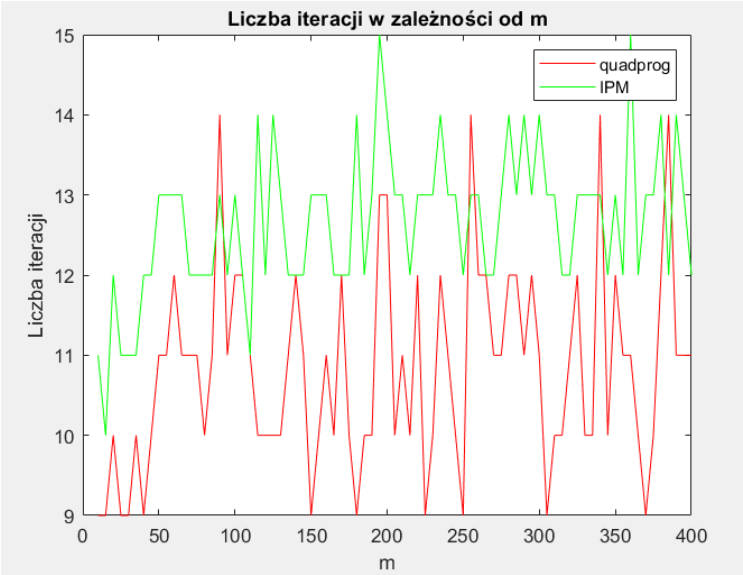
- Błąd eqlin - norma z różnicy pomiędzy zwróconymi wartościami lambdy eqlin
- Błąd lower - norma z różnicy pomiędzy zwróconymi wartościami lambdy lower
- Błąd customowy - porównanie finalnego wyniku, tj. norma z różnicy pomiędzy otrzymanymi środkami kół zsumowana z normą z różnicy promieni

Można zauważyć że podstawowa miara, tj błąd RO nie przekracza 10^{-8}, więc jest sporo mniejsza od zadanego epsilona. Pozostałe wskaźniki są nieco większe, ale ich wartości nie powinny budzić niepokoju.

Co więcej, wartym odnotowania jest że błędy nie zależą od rozmiaru zadania (mamy zawsze taki sam epsilon).

Na tym samym zbiorze danych została porównana też liczba iteracji pomiędzy *quadprog* a **IPM** (Wycinek 2).

Widać, że liczba iteracji *quadprog* mieści się w przedziale $[5, 15]$, podobnie z liczbą iteracji **IPM**. Co prawda **IPM** zazwyczaj robi minimalnie więcej iteracji, jednak są to tak małe wartości, że otrzymany efekt można uznać za bardzo dobry. Czasami nawet **IPM** robi mniej iteracji.



Wycinek 2: Iteracje

Należy zauważyć również że liczba iteracji nie rośnie z rozmiarem zadania, średnia liczba iteracji pozostaje na tym samym poziomie.

3.2 Wnioski

Podsumowanie najważniejszych obserwacji:

- Implementacja **IPM** ma wysoką skuteczność. Rozwiązanie jest zawsze zwracane i jest bardzo bliskie temu otrzymanemu z *quadprog*.
- **IPM** robi nieco więcej iteracji, ale są to liczby mniejsze niż 15, więc rezultat jest bardzo dobry.
- Liczba iteracji nie zwiększa się z rozmiarem zadania (dla testowanych rozmiarów).

4 Pliki

- rysuj3d.m - funkcja otrzymana razem z treścią zadania, służąca do rysowania rozwiązania
- skrypt.m - skrypt realizujący zadanie
- IPM.m - implementacja algorytmu IPM