Analiza wypukła, Matematyka, 2. st.
semestr zimowy 2019/2020
Problemy do zaprogramowania (do zrobienia po jednym w grupach)

1. Napisz implementację metody iteracyjnej Newtona szukania minimum funkcji (z poprawką zamieniającą zdefiniowaną w metodzie Newtona macierz D_k na macierz jednostkową, gdy ta pierwsza nie jest dodatnio określona) z długością kroku metody wyznaczoną przy pomocy metody *Golden-section search* (patrz [AnLu07], rozdz. 4.4). Pochodne mają być liczone symbolicznie. Następnie porównaj szybkość jej działania z metodą Fletchera-Reevesa (patrz [AnLu07], rozdz. 6.6). Testy wykonaj dla funkcji Rosenbrocka
$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$
oraz jakichś niekwadratowych funkcji wypukłych dużej liczby zmiennych.

2. Napisz implementację metody iteracyjnej Newtona szukania minimum funkcji (z poprawką zamieniającą zdefiniowaną w metodzie Newtona macierz D_k na macierz jednostkową, gdy ta pierwsza nie jest dodatnio określona) z długością kroku metody wyznaczoną przy pomocy metody *Golden-section search* (patrz [AnLu07], rozdz. 4.4). Pochodne mają być liczone symbolicznie. Następnie porównaj szybkość jej działania z metodą Davidona-Fletchera-Powella (patrz [AnLu07], rozdz. 7.5). Testy wykonaj dla funkcji Rosenbrocka (1) oraz jakichś niekwadratowych funkcji wypukłych dużej liczby zmiennych.

3. Napisz implementację metody iteracyjnej Newtona szukania minimum funkcji (z poprawką zamieniającą zdefiniowaną w metodzie Newtona macierz D_k na macierz jednostkową, gdy ta pierwsza nie jest dodatnio określona) z długością kroku metody wyznaczoną przy pomocy metody *Golden-section search* (patrz [AnLu07], rozdz. 4.4). Pochodne mają być liczone symbolicznie. Następnie porównaj szybkość jej działania z metodą Broydena-Fletchera-Goldfarba-Shanno (patrz [AnLu07], rozdz. 7.6). Testy wykonaj dla funkcji Rosenbrocka (1) oraz jakichś niekwadratowych funkcji wypukłych dużej liczby zmiennych.

4. Napisz implementację metody subgradientowej szukania minimum funkcji wypukłej z $\mathbb{R}^n$ w $\mathbb{R}$ (czyli bez rzutowania) ze zmniejszającą się długością kroku (w sposób gwarantujący zbieżność). Program ma działać dla funkcji będących maksimami kilku wielomianów n zmiennych. Ma sprawdzać, czy te wielomiany są funkcjami wypukłymi. Przetestuj jej działanie dla kilku funkcji więcej niż dwóch zmiennych.

5. Napisz implementację metody Franka-Wolfe'a szukania minimum funkcji przy liniowych ograniczeniach na argumenty, z długością kroku metody wyznaczoną przy pomocy reguły Armijo. Program powinien umożliwiać użytkownikowi podanie funkcji, dla której będzie szukane minimum, punktu początkowego oraz ograniczeń. Pochodne mają być liczone symbolicznie. Programy liniowe potrzebne do wyznaczenia kierunku dającego największy spadek mają być rozwiązywane przy pomocy wbudowanych lub bibliotecznych procedur do rozwiązywania programów liniowych. Przetestuj działanie (zbieżność i jej szybkość) swojego programu na przykładzie minimalizacji jakiejś funkcji wypukłej wielu zmiennych z rosnącą liczbą liniowych ograniczeń.

6. Napisz implementację metody rzutowania gradientu dla szukania minimum funkcji przy liniowych ograniczeniach na argumenty, z długością kroku metody wyznaczoną przy pomocy reguły Armijo. Program powinien umożliwiać użytkownikowi podanie funkcji, dla której będzie szukane minimum, punktu początkowego oraz ograniczeń. Pochodne mają być liczone symbolicznie. Programy kwadratowe potrzebne do rzutowania mają być rozwiązywane przy pomocy wbudowanych lub bibliotecznych procedur do rozwiązywania programów kwadratowych. Przetestuj działanie (zbieżność i jej szybkość) swojego programu na przykładzie minimalizacji jakiejś funkcji wypukłej wielu zmiennych z rosnącą liczbą liniowych ograniczeń.

7. Problem minimalizacji funkcji f przy założeniu, że jakieś inne funkcje $h_1, \ldots, h_k$ są równe zero można próbować rozwiązać przy pomocy metody mnożników Lagrange'a. Można też rozwiązywać go numerycznie przy pomocy opartej na niej tzw. metody mnożników, której jedną z wersji podaję poniżej ($h(x)$ należy rozumieć jako wektor wartości kolejnych funkcji h_i w punkcie x). Niech
$$L_c(x, \lambda) := f(x) - \lambda^T h(x) + \frac{c}{2}\|h(x)\|_2^2.$$

Startujemy od $\lambda := 0$, $c := 10$, $\varepsilon := 1$.
Dla $k = 1, 2, \ldots$ powtarzane są następujące kroki:

1. Szukamy przybliżonego minimum x^* funkcji $L_c(\cdot, \lambda)$ metodą najszybszego spadku (z krokiem wybieranym przy pomocy reguły Armijo), startując z punktu x_0. Pętlę metody najszybszego spadku przerywamy w momencie, gdy $\|\nabla_x L_c(x_k, \lambda)\|_2 < \min\{\varepsilon, \varepsilon\|h(x_k)\|_2\}$.
2. $\lambda := \lambda + ch(x^*)$, $c := 4c$, $\varepsilon := \frac{\varepsilon}{2}$, $x_0 := x^*$.

Końcowa wartość x^* powinna być dobrym przybliżeniem warunkowego minimum funkcji f.

Napisz implementację metody mnożników. Program powinien umożliwiać użytkownikowi podanie funkcji f, dla której będzie szukane minimum, oraz funkcji h_i, definiujących ograniczenia. Pochodne mają być liczone symbolicznie. Przetestuj jej działanie dla jakichś (niekoniecznie wypukłych) problemów minimalizacji.

Literatura:

[AnLu07] A. Antoniou, W.-S. Lu, Practical Optimization, Springer Science+Business Media, LLC, 2007