Raport z Ćwiczenia 1 - Bartłomiej Rasztabiga 304117

Zaimplementuj algorytm gradientu prostego oraz algorytm Newtona. Algorytm Newtona powinien móc działać w dwóch trybach:

- ze stałym parametrem kroku
- z adaptacją parametru kroku przy użyciu metody z nawrotami.

Opis implementowanych algorytmów

Dla każdego z implementowanych algorytmów podany został pseudokod oraz opcjonalnie krótki opis.

Warunkiem stopu w każdym algorytmie, jest osiągnięcie wektora zmiany punktu poniżej podanej tolerancji (w tym przypadku jest to 10^-4).

Dodatkowym warunkiem stopu jest przekroczenie maksymalnej liczby iteracji, podanej z góry. Dzięki temu można kończyć minimalizacje, które nie przybliżają nas do minimum, bo np. występują oscylacje.

Gradient, jak i hesjan są obliczane numerycznie przy pomocy pakietu numdifftools.

Algorytm gradientu prostego

Zaimplementowano prosty algorytm adaptacji kroku. Jego pseudokod przedstawia się następująco:

```
beta = 0.8
dopóki f(punkt - step * zmiana) >= f(punkt):
    krok = krok * beta
zwróć krok

gdzie zmiana = gradient w punkcie
```

Powyższy algorytm zwraca nowy krok dla danej iteracji.

```
punkt = punkt_startowy
liczba_iteracji = 0
dopóki liczba_iteracji < maksymalna_liczba_iteracji:
    gradient_w_punkcie = gradient(f, punkt)

    zmiana = gradient_w_punkcie
    krok = znajdź_nowy_krok(punkt, zmiana, krok)
    punkt = punkt - krok * zmiana

    jeżeli zmiana <= tolerancja:
        zwróć, że znaleziono minimum w punkcie

    liczba_iteracji++

zwróć, że nie znaleziono minimum
```

Algorytm Newtona ze stałym parametrem kroku

```
punkt = punkt_startowy
liczba_iteracji = 0
dopóki liczba_iteracji < maksymalna_liczba_iteracji:
    gradient_w_punkcie = gradient(f, punkt)
    odwrócony_hesjan_w_punkcie = hesjan(f, punkt)

    zmiana = gradient_w_punkcie * odwrócony_hesjan_w_punkcie
    punkt = punkt - krok * zmiana

    jeżeli zmiana <= tolerancja:
        zwróć, że znaleziono minimum w punkcie

    liczba_iteracji++

zwróć, że nie znaleziono minimum
```

Algorytm Newtona z adaptacją parametru kroku

Zaimplementowano algorytm adaptacji kroku z nawrotami, oparty na warunku Armijo – Goldsteina. Jego pseudokod przedstawia się następująco:

```
t = 1.0
alpha = 0.2
beta = 0.8
dopóki f(punkt - t * krok * zmiana) > f(punkt) + alpha * t * krok *
gradient_w_punkcie^T * (-zmiana):
    t = t * beta
zwróć t * krok

gdzie zmiana = odwrócony hesjan w punkcie * gradient w punkcie
```

Powyższy algorytm zwraca nowy krok dla danej iteracji.

```
punkt = punkt_startowy
liczba_iteracji = 0
dopóki liczba_iteracji < maksymalna_liczba_iteracji:
    gradient_w_punkcie = gradient(f, punkt)
    odwrócony_hesjan_w_punkcie = hesjan(f, punkt)

    zmiana = gradient_w_punkcie * odwrócony_hesjan_w_punkcie
    krok = znajdź_nowy_krok(punkt, gradient_w_punkcie, zmiana, krok_poczatkowy)
    punkt = punkt - krok * zmiana

    jeżeli zmiana_punktu <= tolerancja:
        zwróć, że znaleziono minimum w punkcie

    liczba_iteracji++

zwróć, że nie znaleziono minimum
```

Opis planowanych eksperymentów numerycznych

Podczas porównywania algorytmów funkcją do minimalizacji będzie:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} x_i^2, \ \mathbf{x} \in [-100, 100]^n \subset \mathbb{R}^n.$$

Porównanie algorytmów

Pierwszym planowanym eksperymentem jest porównanie trzech algorytmów pod względem liczby iteracji potrzebnych do osiągnięcia minimum.

W tym eksperymencie wykorzystam dwie wartości n, trzy wartości alpha oraz krok początkowy równy 0.01. Razem powstanie 6 przypadków par n i alpha.

```
n = [10, 20]
alpha = [1, 10, 100]
krok = 0.01

tolerancja = 0.0001
maksymalna liczba iteracji = 5000
```

Wpływ początkowego kroku

Drugim planowanym eksperymentem jest sprawdzenie wpływu początkowego kroku na liczbę iteracji potrzebnych do osiągnięcia minimum.

W tym eksperymencie wykorzystam wartość n równą 10 oraz alpha równe 1.

```
n = 10
alpha = 1
krok = [0.01, 0.1, 0.5, 1.0, 1.5, 1.6, 1.7, 1.8, 1.9]

tolerancja = 0.0001
maksymalna liczba iteracji = 5000
```

Porównanie wariantów metody Newtona

Trzecim planowanym eksperymentem jest sprawdzenie wpływu algorytmu adaptacji kroku na liczbę iteracji potrzebnych do osiągnięcia minimum.

W tym eksperymencie wykorzystam wartość n równą 10 oraz alpha równe 1. Ponadto użyję jedynie poniższych początkowych kroków, ponieważ aż do kroku 1.5 oba warianty algorytmu Newtona zachowują się jednakowo. Powód takiego zachowania opiszę we wnioskach.

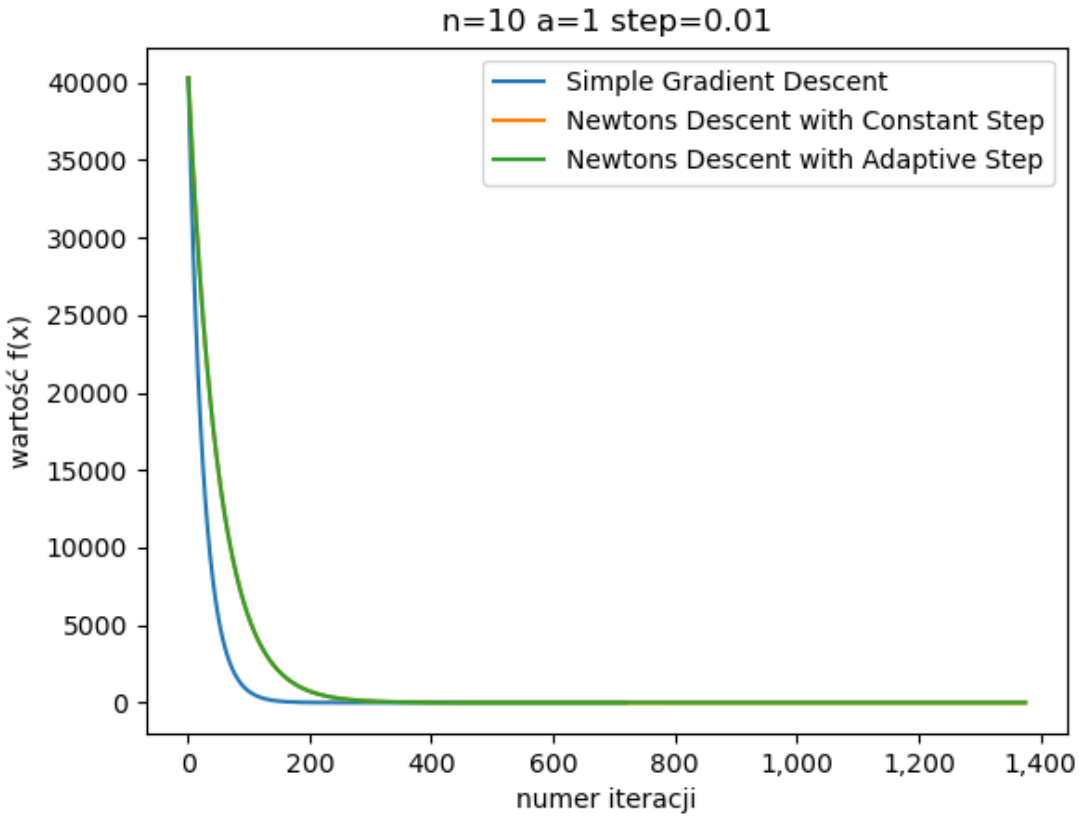
```
n = 10
alpha = 1
krok = [1.5, 1.6, 1.7, 1.8, 1.9]

tolerancja = 0.0001
maksymalna liczba iteracji = 5000
```

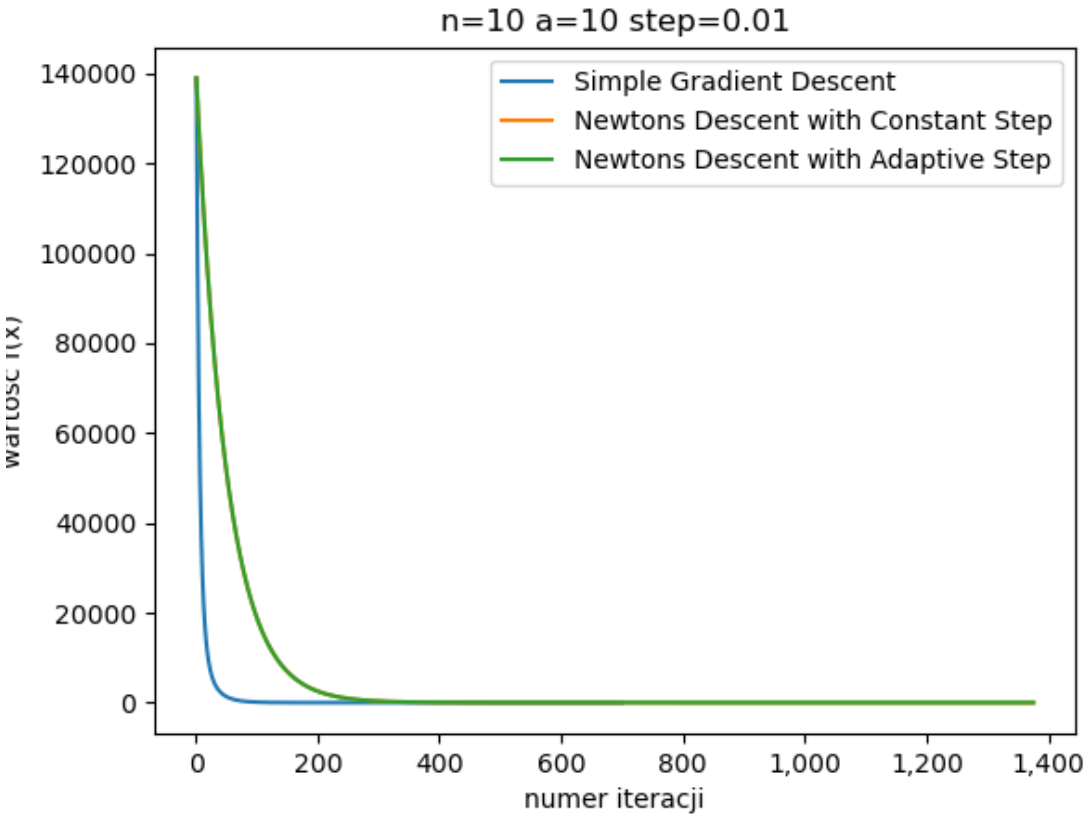
Opis uzyskanych wyników

Porównanie algorytmów

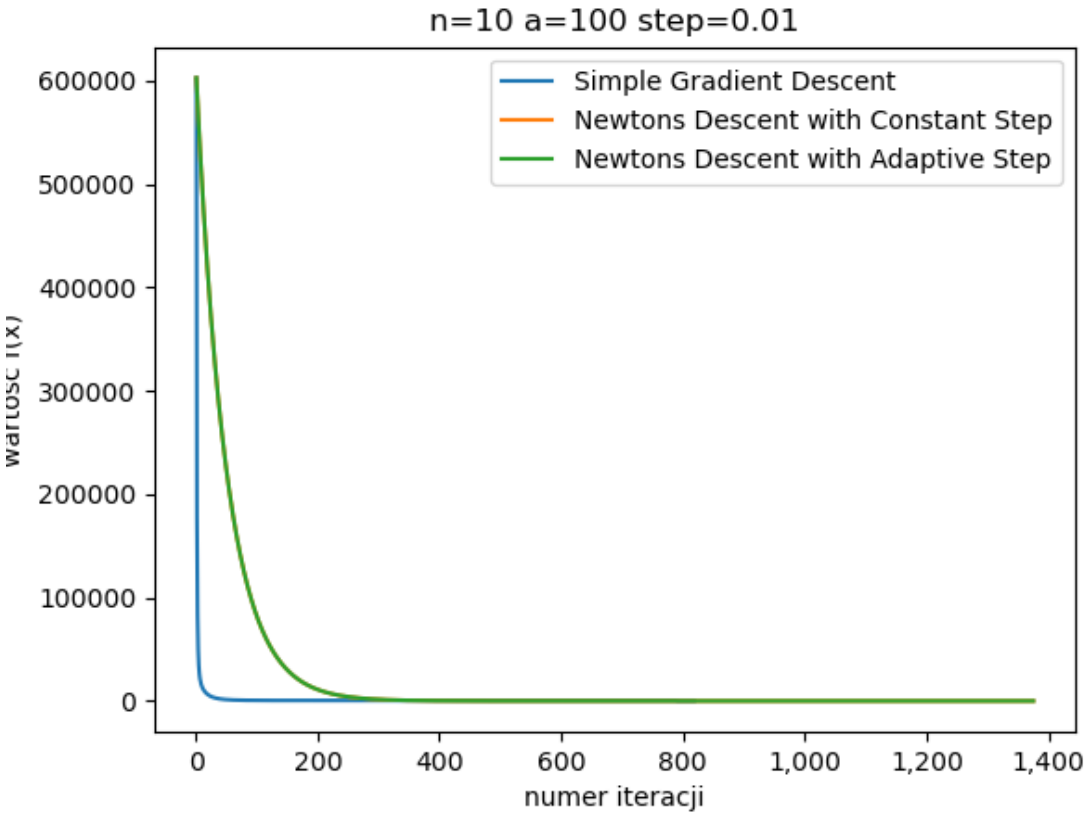
Badana jest zależność algorytmu od iteracji potrzebnych do znalezienia minimum. Poniżej znajduje się 6 wykresów dla różnych przypadków n i alpha. Krok w każdym przypadku jest równy 0.01.



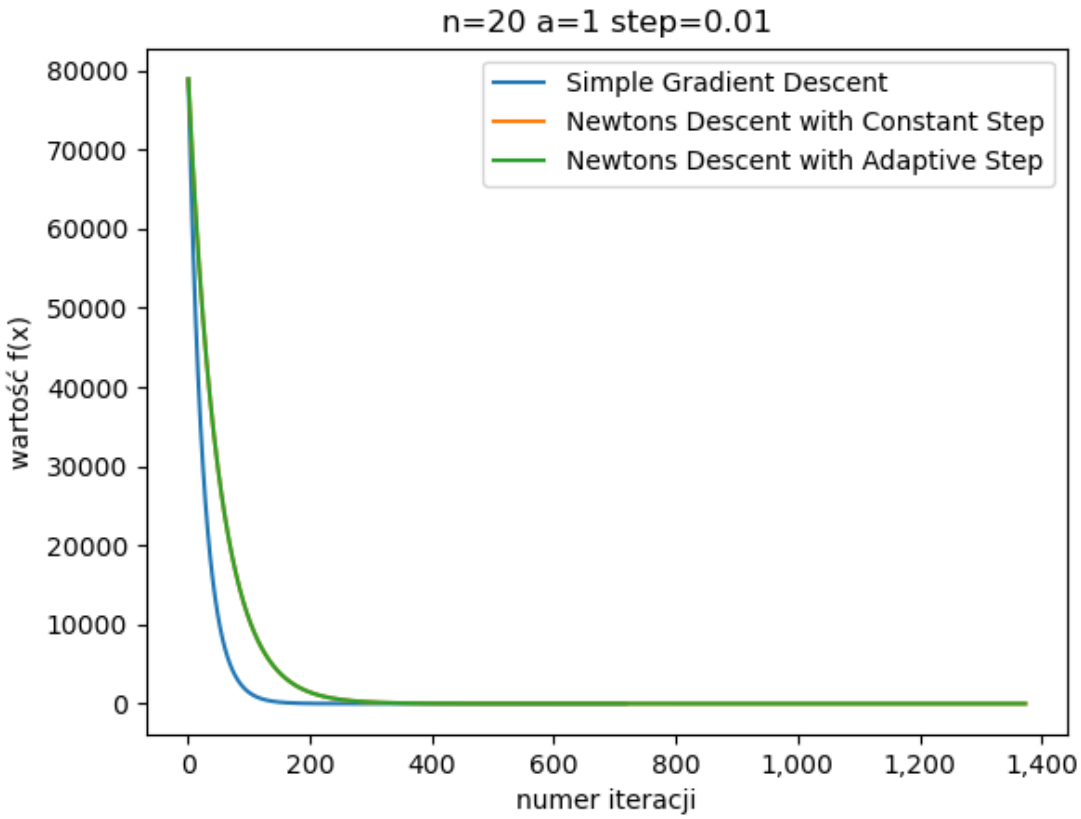
Dla n=10 a=1:
Algorytm gradientu prostego znalazł minimum w 719 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1375 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1375 iteracji.



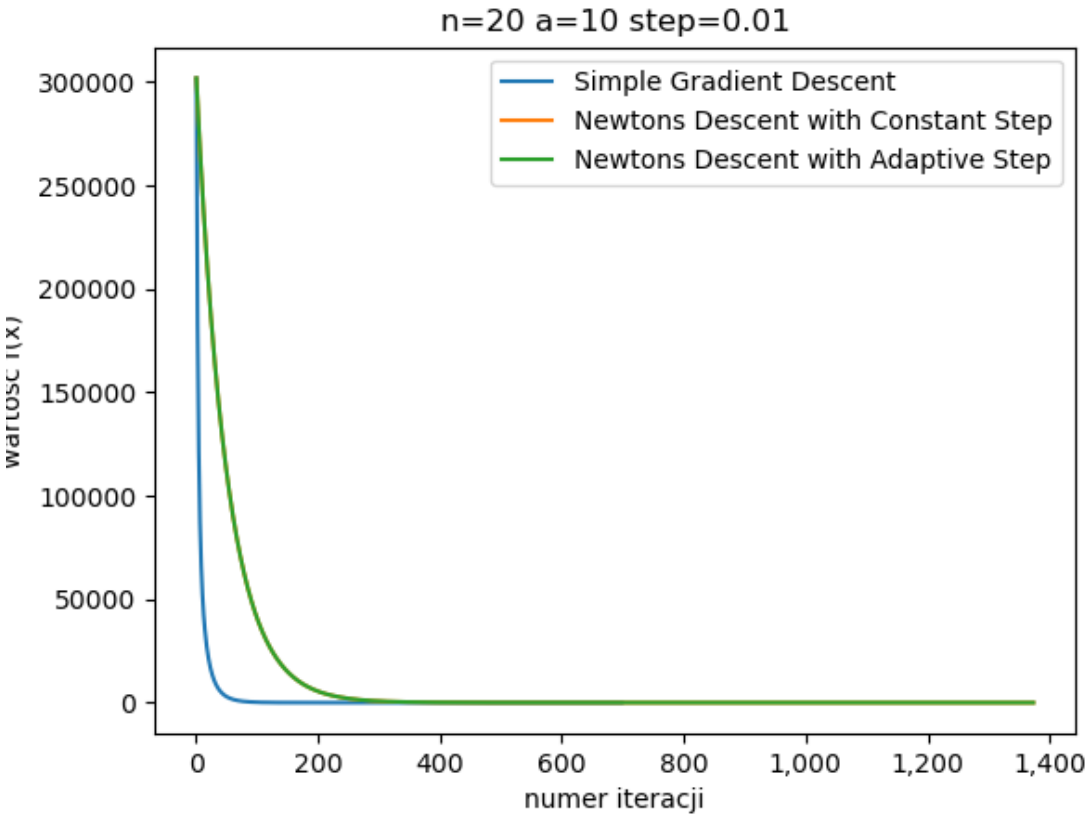
Dla n=10 a=10:
Algorytm gradientu prostego znalazł minimum w 700 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1375 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1375 iteracji.



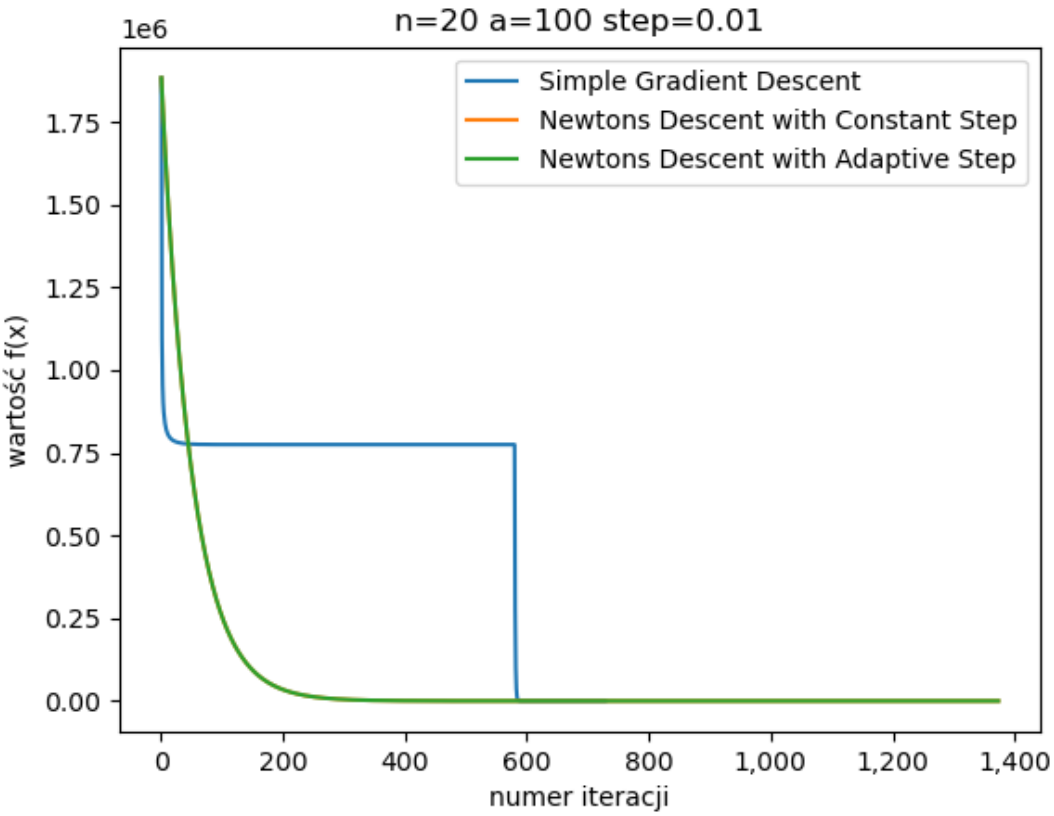
Dla n=10 a=100:
Algorytm gradientu prostego znalazł minimum w 818 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1375 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1375 iteracji.



Dla n=20 a=1:
Algorytm gradientu prostego znalazł minimum w 718 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1373 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1373 iteracji.



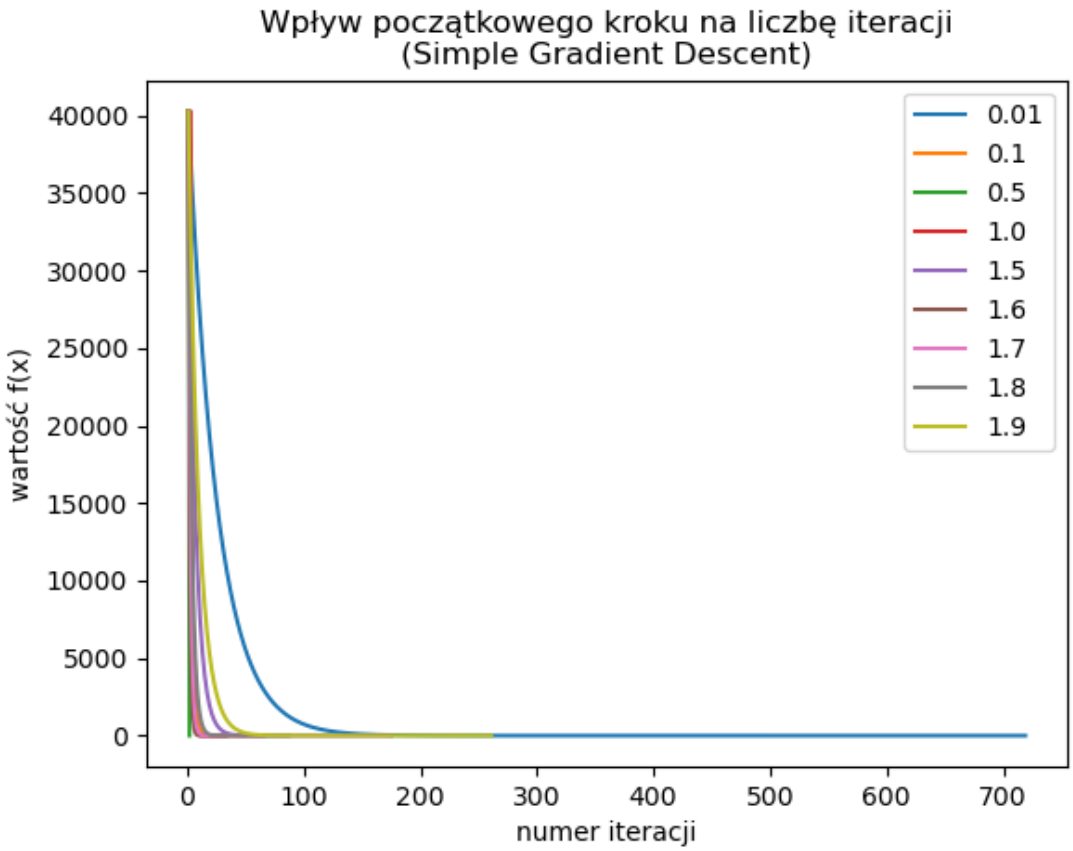
Dla n=20 a=10:
Algorytm gradientu prostego znalazł minimum w 698 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1373 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1373 iteracji.



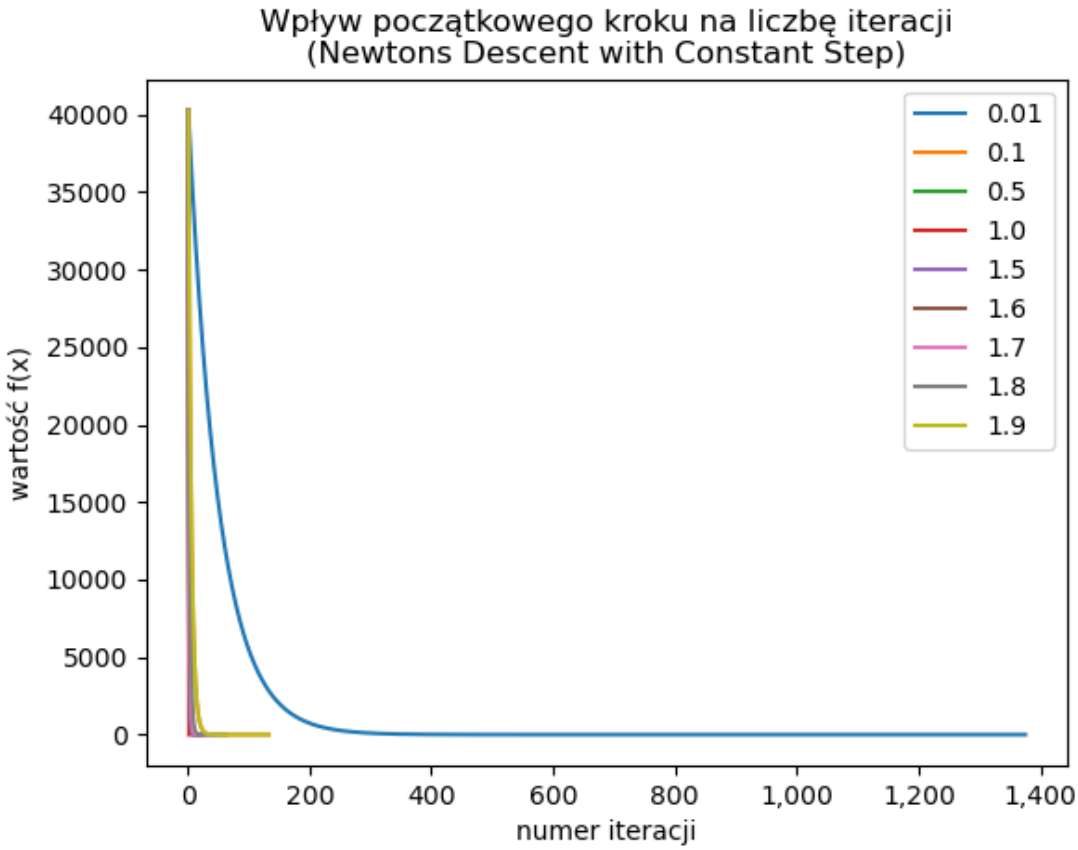
Dla n=20 a=100:
Algorytm gradientu prostego znalazł minimum w 728 iteracji.
Algorytm Newtona ze stałym krokiem znalazł minimum w 1373 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 1373 iteracji.

Wpływ początkowego kroku

Badana jest zależność wartości początkowego kroku od iteracji potrzebnych do znalezienia minimum. Poniżej znajdują się 3 wykresy, po jednym dla każdego algorytmu. Kroki dla wszystkich algorytmów to [0.01, 0.1, 0.5, 1.0, 1.5, 1.6, 1.7, 1.8, 1.9]

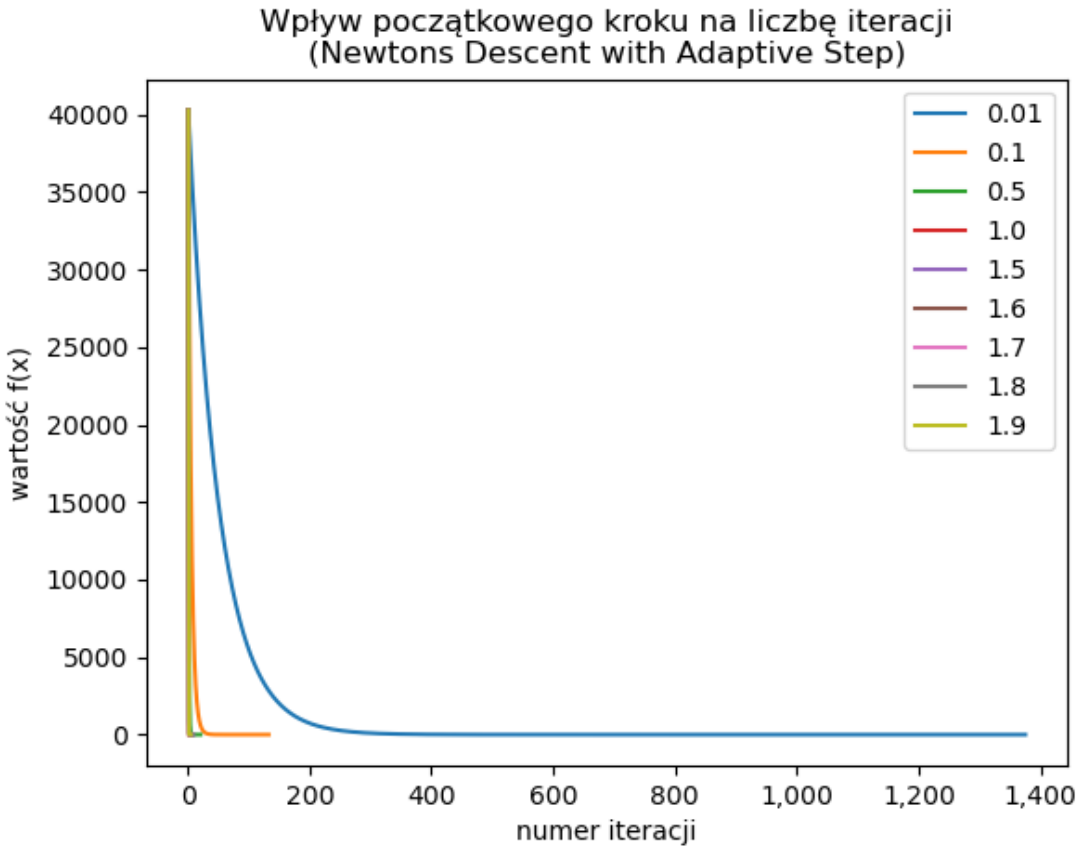


Dla algorytmu gradientu prostego:
Dla kroku 0.01 algorytm znalazł minimum w 719 iteracji.
Dla kroku 0.1 algorytm znalazł minimum w 66 iteracji.
Dla kroku 0.5 algorytm znalazł minimum w 2 iteracji.
Dla kroku 1.0 algorytm znalazł minimum w 32 iteracji.
Dla kroku 1.5 algorytm znalazł minimum w 175 iteracji.
Dla kroku 1.6 algorytm znalazł minimum w 34 iteracji.
Dla kroku 1.7 algorytm znalazł minimum w 50 iteracji.
Dla kroku 1.8 algorytm znalazł minimum w 87 iteracji.
Dla kroku 1.9 algorytm znalazł minimum w 261 iteracji.



Dla algorytmu Newtona ze stałym krokiem:

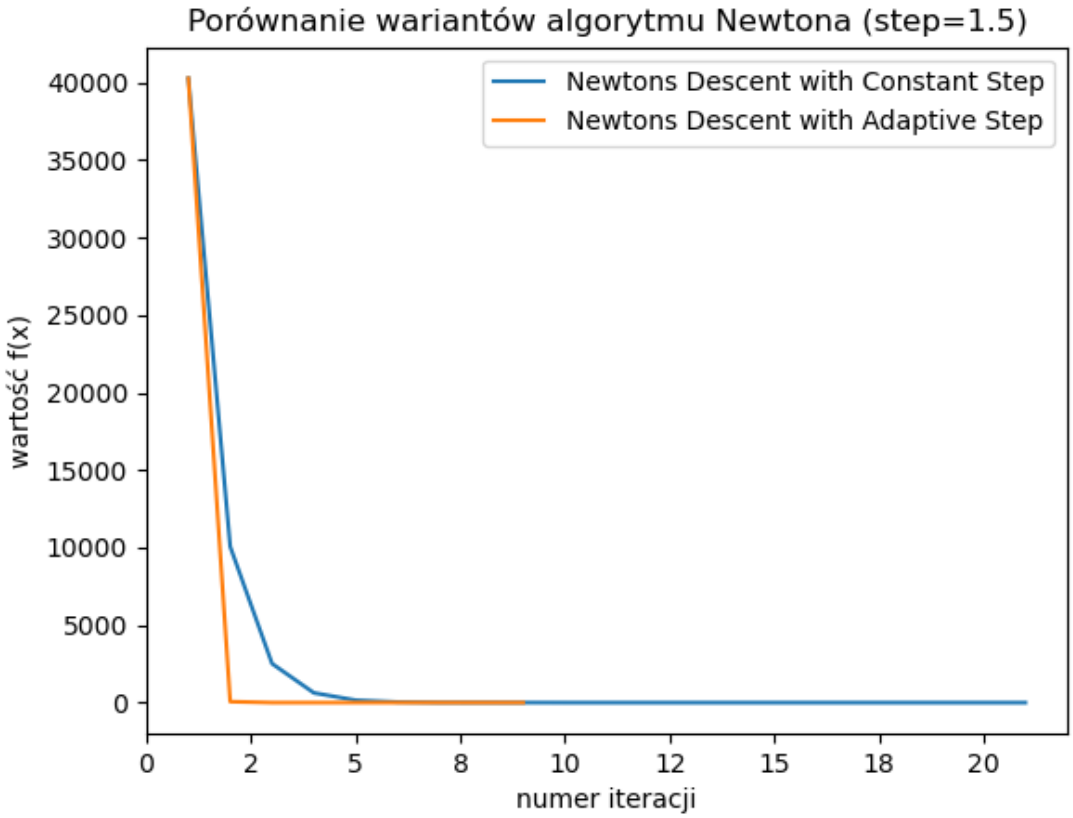
Dla kroku 0.01 algorytm znalazł minimum w 1375 iteracji.
Dla kroku 0.1 algorytm znalazł minimum w 133 iteracji.
Dla kroku 0.5 algorytm znalazł minimum w 21 iteracji.
Dla kroku 1.0 algorytm znalazł minimum w 2 iteracji.
Dla kroku 1.5 algorytm znalazł minimum w 21 iteracji.
Dla kroku 1.6 algorytm znalazł minimum w 29 iteracji.
Dla kroku 1.7 algorytm znalazł minimum w 40 iteracji.
Dla kroku 1.8 algorytm znalazł minimum w 63 iteracji.
Dla kroku 1.9 algorytm znalazł minimum w 133 iteracji.



Dla algorytmu Newtona z adaptacją kroku:
Dla kroku 0.01 algorytm znalazł minimum w 1375 iteracji.
Dla kroku 0.1 algorytm znalazł minimum w 133 iteracji.
Dla kroku 0.5 algorytm znalazł minimum w 21 iteracji.
Dla kroku 1.0 algorytm znalazł minimum w 2 iteracji.
Dla kroku 1.5 algorytm znalazł minimum w 9 iteracji.
Dla kroku 1.6 algorytm znalazł minimum w 5 iteracji.
Dla kroku 1.7 algorytm znalazł minimum w 7 iteracji.
Dla kroku 1.8 algorytm znalazł minimum w 9 iteracji.
Dla kroku 1.9 algorytm znalazł minimum w 5 iteracji.

Porównanie wariantów metody Newtona

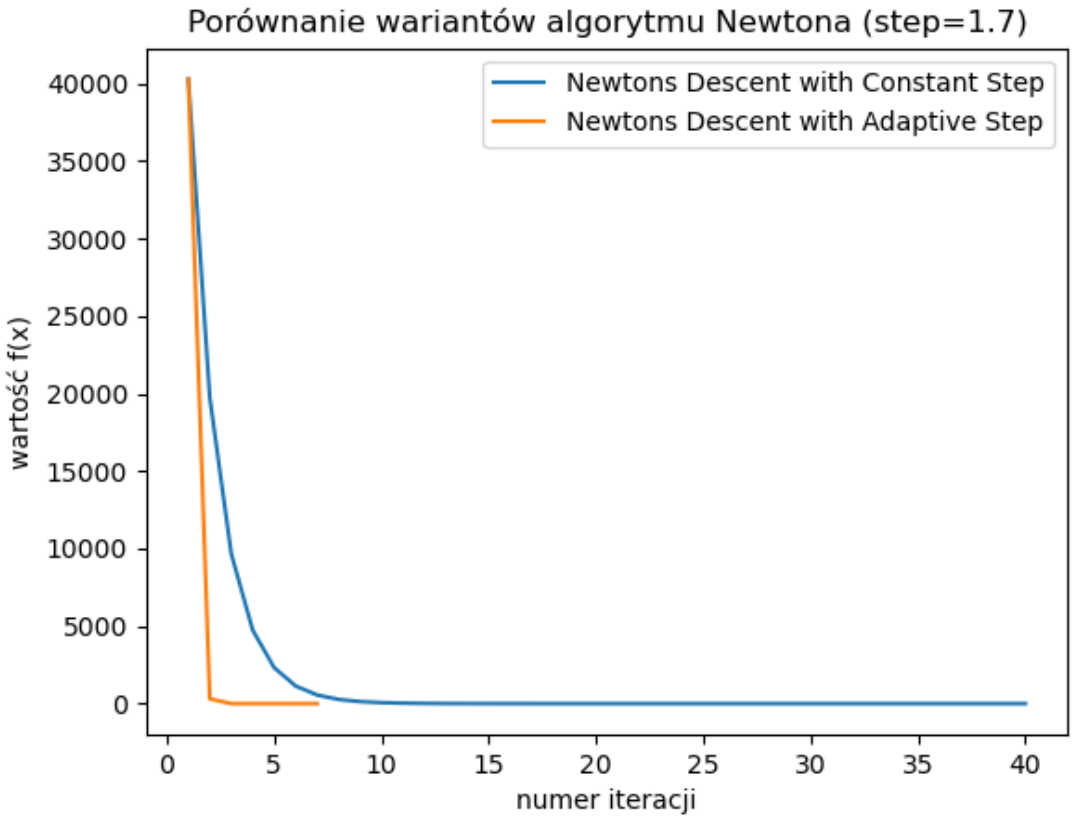
Badana jest zależność wariantu metody Newtona od iteracji potrzebnych do znalezienia minimum. Do badania zostaną wykorzystane kroki = [1.5, 1.7, 1.9]. Pomijam kroki niższe od 1.5, ponieważ dla nich oba warianty znajdują minimum w taką samą liczbę iteracji.



Dla kroku = 1.5:

Algorytm Newtona ze stałym krokiem znalazł minimum w 21iteracji.

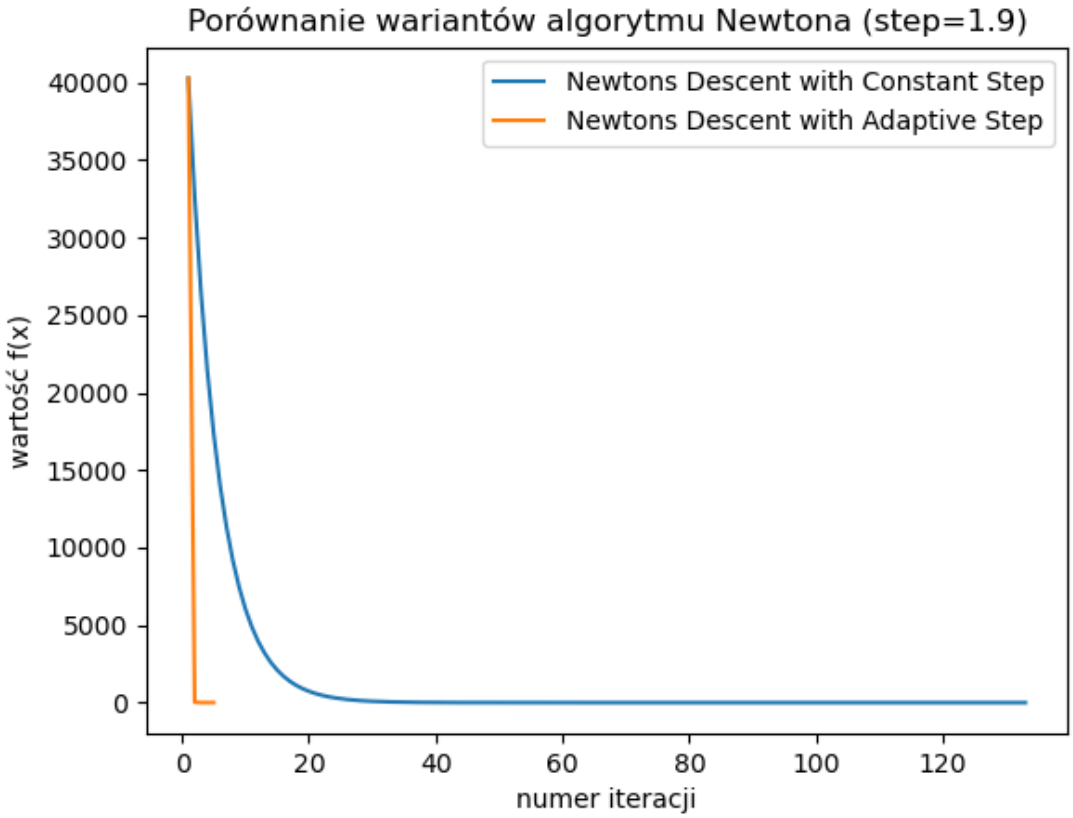
Algorytm Newtona z adaptacją kroku znalazł minimum w 9 iteracji.



Dla kroku = 1.7:

Algorytm Newtona ze stałym krokiem znalazł minimum w 40 iteracji.

Algorytm Newtona z adaptacją kroku znalazł minimum w 7 iteracji.



Dla kroku = 1.9:
Algorytm Newtona ze stałym krokiem znalazł minimum w 133 iteracji.
Algorytm Newtona z adaptacją kroku znalazł minimum w 5 iteracji.

Wnioski z przeprowadzonych badań

Porównanie algorytmów

Na podstawie otrzymanych wyników można wywnioskować, że algorytm gradientu prostego działa najszybciej dla wszystkich kroków od 0.01 do 0.5. Powyżej tego kroku oba algorytmy Newtona działają skuteczniej przez zastosowanie hesjanu.

Dla zadanego kroku (0.01) oba warianty algorytmu Newtona zbiegają w takiej samej liczbie iteracji. Wytłumaczenie tego zjawiska podane jest w punkcie trzecim sekcji wniosków.

Wpływ początkowego kroku

Algorytm gradientu prostego wraz ze wzrostem kroku do 0.5 otrzymuje znaczącą poprawę szybkości znajdowania minimum, natomiast od kroku 0.5 szybkość algorytmu spada. Jest to zapewne spowodowane "oscylacjami" algorytmu dookoła punktu minimum.

Algorytm Newtona ze stałym krokiem działa najlepiej dla kroku w okolicach 1.0, natomiast wraz ze wzrostem lub spadkiem kroku początkowego spowalnia.

Porównanie wariantów metody Newtona

Algorytm Newtona z adaptacją kroku działa tak samo, jak algorytm ze stałym krokiem dla wszystkich kroków początkowych poniżej 1.5. Z powodu specyfiki jego algorytmu adaptacji nie będzie on zmniejszać kroku i będzie zawsze zwracać krok początkowy. Przez to algorytm z adaptacją kroku będzie działać w takiej samej liczbie iteracji jak standardowy algorytm Newtona.

Natomiast powyżej tego pułapu (krok=1.5) "aktywuje" się algorytm adaptacyjny, pozwalający mu na znajdowanie minimum w mniej więcej zawsze taką samą liczbę iteracji (5-7 iteracji) niezależnie od kroku, co powoduje, że staje się on najszybszym algorytmem z wszystkich trzech porównywanych.