Jakub Kowalczyk Ćwiczenie 1. – Algorytm gradientu prostego

Celem eksperymentu było zbadanie wpływu wartości parametrów kroku oraz punktu startowego dla algorytmu gradientu prostego, skutkiem znalezienia minimum określonej funkcji n-zmiennych w stosunkowo niskiej liczbie iteracji.

$$f(x) = \frac{1}{4}x^4$$

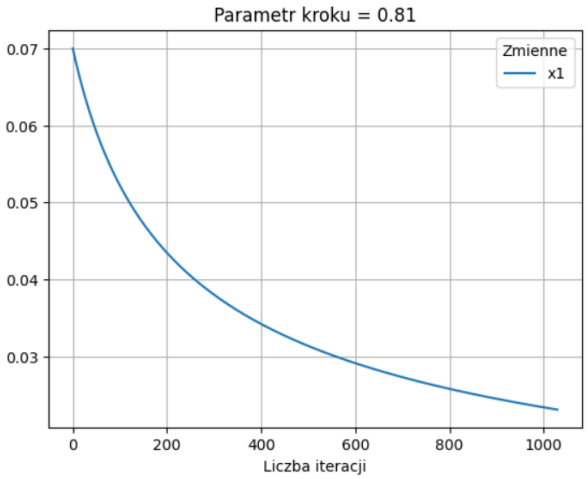
W przypadku funkcji f(x), która jest funkcją wielomianową jednej zmiennej przyjmującą tylko nieujemne wartości, spodziewałem się minimum f(0) = 0

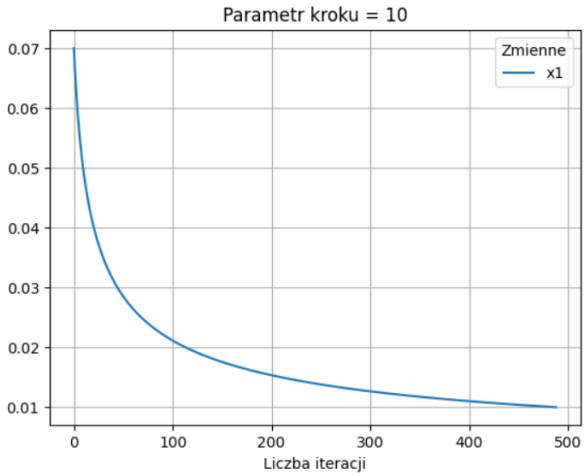
Z 40 losowo wygenerowanych punktów przeanalizuję przypadki krańcowe oraz ciekawe zjawiska. Punkty w zestawieniu tabelarycznym posortowałem względem odległości od spodziewanego minimum, czyli początku układu współrzędnych.

Punkt (x1, ..., xn)	Krok 0.01 (Iteracje, wynik)	Krok 0.21	Krok 0.41	Krok 0.61	Krok 0.81
0.07	(1, 0.07)	(1327, 0.0362)	(1201, 0.029)	(1103, 0.0254)	(1029, 0.0231)
0.25	(4200, 0.1)	(1773, 0.0362)	(1429, 0.029)	(1255, 0.0254)	(1144, 0.0231)
-0.3	(4444, -0.1)	(1784, -0.0362)	(1434, -0.029)	(1259, -0.0254)	(1146, -0.0231)
-0.42	(4716, -0.1)	(1797, -0.0362)	(1441, -0.029)	(1263, -0.0254)	(1149, -0.0231)
-0.58	(4850, -0.1)	(1803, -0.0362)	(1443, -0.029)	(1264, -0.0254)	(1150, -0.0231)
0.6	(4860, 0.1)	(1803, 0.0362)	(1443, 0.029)	(1264, 0.0254)	(1150, 0.0231)
-0.61	(4864, -0.1)	(1803, -0.0362)	(1444, -0.029)	(1264, -0.0254)	(1150, -0.0231)
-0.64	(4877, -0.1)	(1804, -0.0362)	(1444, -0.029)	(1265, -0.0254)	(1150, -0.0231)
0.73	(4905, 0.1)	(1805, 0.0362)	(1444, 0.029)	(1265, 0.0254)	(1150, 0.0231)
-0.87	(4932, -0.1)	(1806, -0.0362)	(1444, -0.029)	(1264, -0.0254)	(1149, -0.0231)
1.1	(4957, 0.1)	(1806, 0.0362)	(1444, 0.029)	(1259, 0.0254)	(2, 0.0219)
-1.15	(4960, -0.1)	(1807, -0.0362)	(1444, -0.029)	(1253, -0.0254)	(1064, 0.0231)
1.16	(4961, 0.1)	(1807, 0.0362)	(1444, 0.029)	(1251, 0.0254)	(1099, -0.0231)
1.21	(4964, 0.1)	(1807, 0.0362)	(1443, 0.029)	(1221, 0.0254)	(1142, -0.0231)
1.49	(4975, 0.1)	(1806, 0.0362)	(1382, 0.029)	(1265, -0.0254)	(1136, 0.0231)
1.63	(4978, 0.1)	(1806, 0.0362)	(1393, -0.029)	(1264, -0.0254)	(10000, nan)
1.63	(4978, 0.1)	(1806, 0.0362)	(1393, -0.029)	(1264, -0.0254)	(10000, nan)
1.65	(4979, 0.1)	(1806, 0.0362)	(1417, -0.029)	(1261, -0.0254)	(10000, nan)
-1.75	(4981, -0.1)	(1804, -0.0362)	(1442, 0.029)	(1267, -0.0254)	(10000, nan)
-1.88	(4983, -0.1)	(1801, -0.0362)	(1445, 0.029)	(10000, nan)	(10000, nan)
-1.93	(4984, -0.1)	(1798, -0.0362)	(1445, 0.029)	(10000, nan)	(10000, nan)
1.95	(4984, 0.1)	(1796, 0.0362)	(1445, -0.029)	(10000, nan)	(10000, nan)
-1.95	(4984, -0.1)	(1796, -0.0362)	(1445, 0.029)	(10000, nan)	(10000, nan)
-2.02	(4985, -0.1)	(1783, -0.0362)	(1439, 0.029)	(10000, nan)	(10000, nan)
-2.05	(4985, -0.1)	(1771, -0.0362)	(1395, 0.029)	(10000, nan)	(10000, nan)
2.13	(4986, 0.1)	(1578, 0.0362)	(1446, 0.029)	(10000, nan)	(10000, nan)
-2.23	(4987, -0.1)	(1569, 0.0362)	(10000, nan)	(10000, nan)	(10000, nan)
-2.71	(4990, -0.1)	(1807, 0.0362)	(10000, nan)	(10000, nan)	(10000, nan)
-2.72	(4990, -0.1)	(1807, 0.0362)	(10000, nan)	(10000, nan)	(10000, nan)
-2.74	(4990, -0.1)	(1807, 0.0362)	(10000, nan)	(10000, nan)	(10000, nan)
2.8	(4990, 0.1)	(1804, -0.0362)	(10000, nan)	(10000, nan)	(10000, nan)
-3.12	(4991, -0.1)	(10000, nan)	(10000, nan)	(10000, nan)	(10000, nan)
3.35	(4992, 0.1)	(10000, nan)	(10000, nan)	(10000, nan)	(10000, nan)

Dla punktu 0.07:

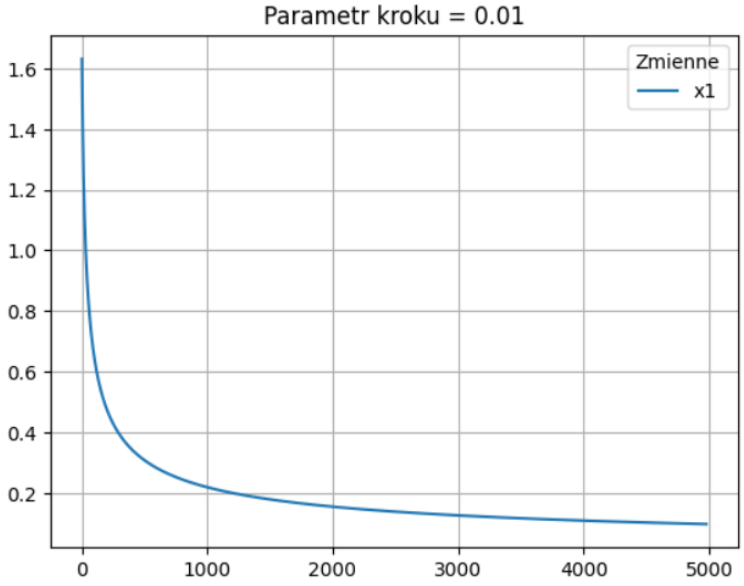
W przypadku punktu znajdującego się bardzo blisko szukanego minimum parametr kroku o wartości 0.01 nie wykonuje dalszych iteracji, gdyż różnica między x_0 a x_1 (diff = 0.01*$\nabla q(0.07)$) jest bardzo niska i mieści się w dozwolonej tolerancji równej 0.00001. Niskie wartości kroku nie mają zatem sensu. Dla skrajnie małego punktu początkowego algorytm radzi sobie dla dużej wartości kroku (np. ß = 10), osiągając przy tym liczbę iteracji poniżej 500.

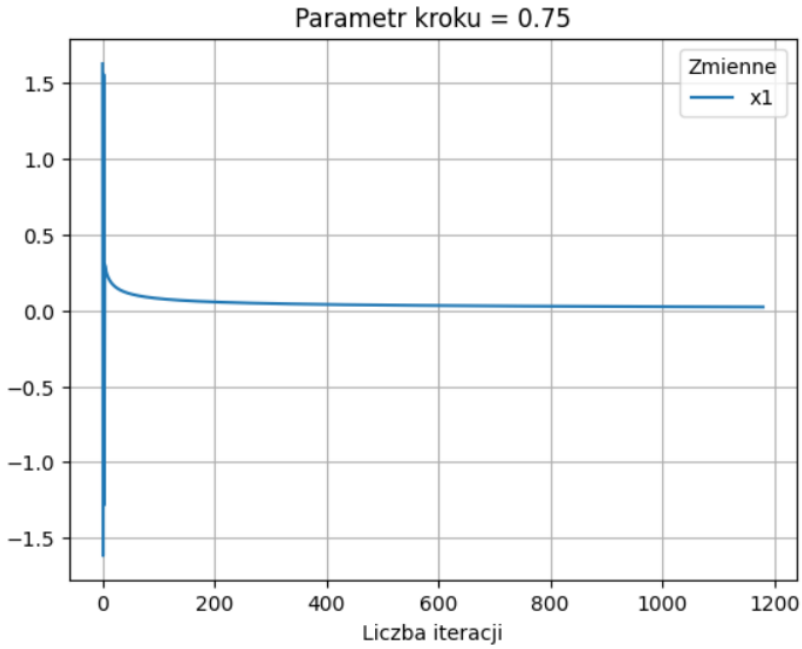




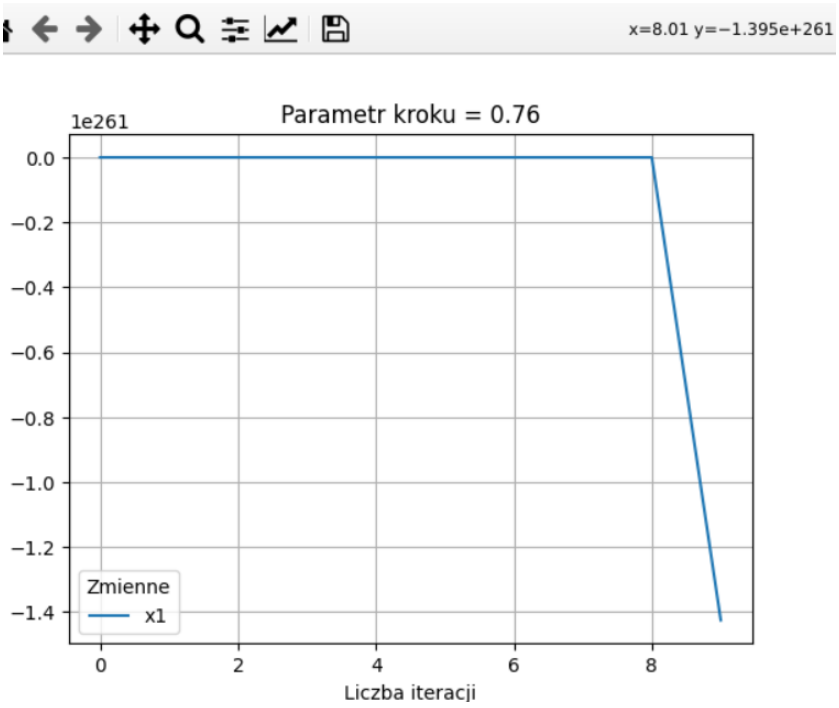
Dla punktu 1.63:

Dla bardziej oddalonego punktu niska wartość kroku znajduje minimum, lecz robi to w stałej liczbie około 5000 iteracji (dla ß = 0.01). Małym krokiem bardzo powoli zbliża się minimum z czasem gdy iksy gwałtownie spadną poniżej wartości x = 1, dlatego też dokładna odległość punktu nie ma prawie żadnego znaczenia. Najefektywniejszą znalezioną ß-tą jest liczba 0.75, która z odpowiednią prędkością znajduje minimum w liczbie niespełna 1200 iteracji.





Z pozornie pożądaną szybkością kroku nie można przesadzić, gdyż wówczas funkcja rozbiega się do nieskończoności i nie uzyskuje wyniku, jak to się dzieje w przypadku ß = 0.76



Dla punktu 1.1:

Interesujący jest punkt 1.1 i krok 0.81, dla których liczba wykonanych iteracji wynosi jedynie 2.

Policzmy x1 = 1.1 – 0.81 * ∇q(1.1) = 1.1 – 1,07811 = 0,02189

diff = 0.81 * ∇q(0,02189) = 0.0000084… , co jest zgodne z oczekiwaną precyzją, więc algorytm kończy działanie z wynikiem

x2 = 0,02189 – 0.81 * ∇q(0,02189) = 0.218815…

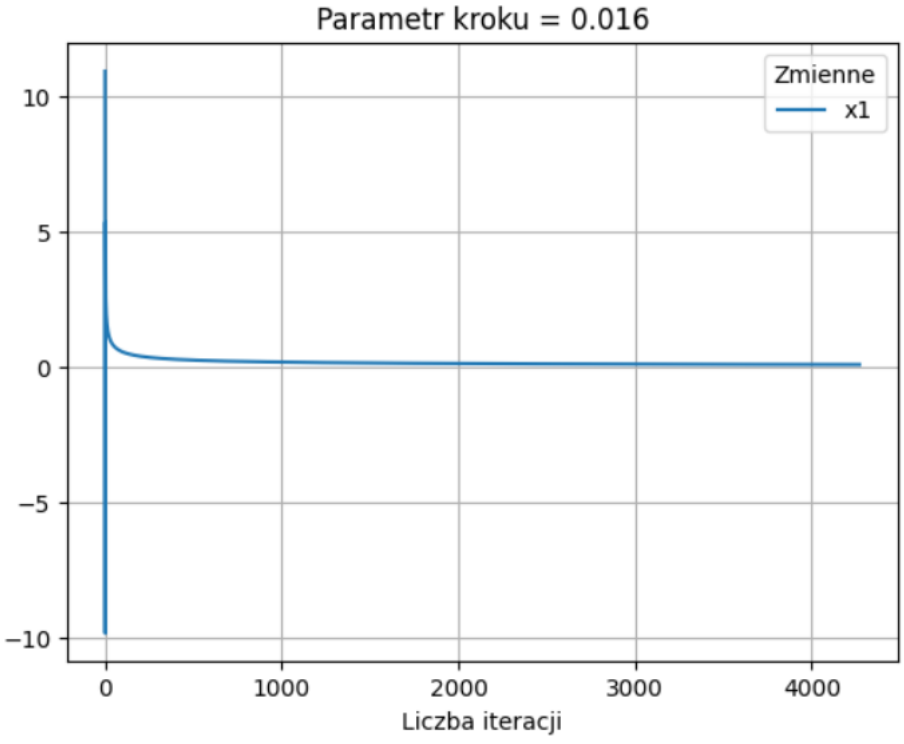
co po zaokrągleniu daje 0.219 – wszystko się zgadza

Można nazwać go zatem nazwać złotym strzałem dla zadanej precyzji – uzyskuje ją zanim wpadnie w pętle bardzo małych kroków zbliżających do wyniku.

W tabeli nie ująłem punktów o odległości bezwzględnej większej niż 3, gdyż dają te same przewidywalne wyniki, tzn. dla kroku 0.01 algorytm do pewnego momentu znajduje minimum w około 5000 iteracji, a dla większych kroków jest rozbieżny do nieskończoności i nie potrafi przynieść rezultatu.

Dla punktu x = 10.9:

Dla dalszych punktów efektywne są tylko bardzo małe wartości kroku, aby uniknąć rozbieżności funkcji do nieskończoności. Najefektywniej dla x=10.9:

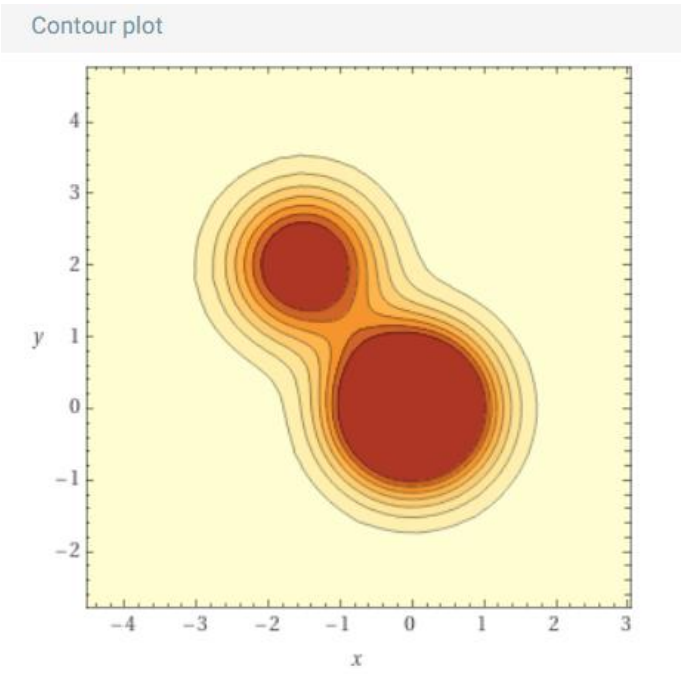


Teraz przeanalizuję wybrane punkty funkcji dwóch zmiennych.

$$g(x) = 2 - \exp\left\{-x_1^2 - x_2^2\right\} - 0.5\exp\left\{-(x_1 + 1.5)^2 - (x_2 - 2)^2\right\}$$

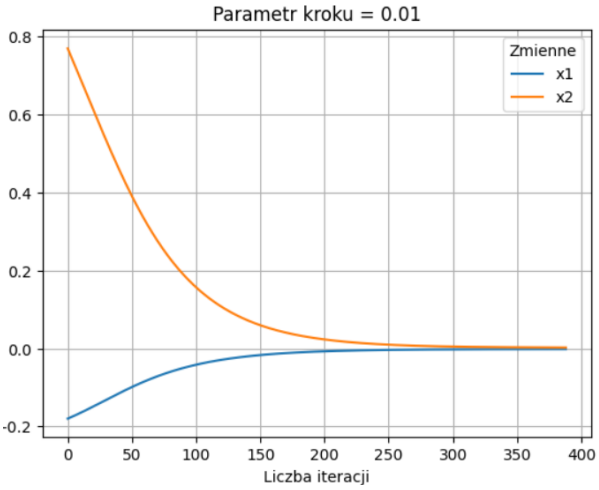
Punkt (x1, ..., xn)	Krok 0.01 (Iteracje, wynik)	Krok 0.21	Krok 0.41	Krok 0.61	Krok 0.81
[-0.18 0.77]	(388, (-0.0016, 0.0024))	(22, (-0.0015, 0.002))	(9, (-0.0015, 0.002))	(9, (-0.0015, 0.002))	(22, (-0.0015, 0.0019))
[-0.75 0.6]	(401, (-0.002, 0.0024))	(23, (-0.0015, 0.002))	(10, (-0.0015, 0.002))	(9, (-0.0015, 0.002))	(23, (-0.0015, 0.002))
[0.64 0.75]	(396, (-0.0011, 0.0024))	(23, (-0.0015, 0.002))	(10, (-0.0015, 0.002))	(9, (-0.0015, 0.002))	(24, (-0.0015, 0.0019))
[-0.93 -0.43]	(410, (-0.0019, 0.0017))	(23, (-0.0015, 0.0019))	(10, (-0.0015, 0.002))	(9, (-0.0015, 0.002))	(24, (-0.0015, 0.002))
[-0.49 -1.24]	(466, (-0.0016, 0.0015))	(26, (-0.0015, 0.0019))	(11, (-0.0015, 0.002))	(11, (-0.0015, 0.002))	(25, (-0.0015, 0.002))
[-1.44 -0.03]	(497, (-0.002, 0.002))	(28, (-0.0015, 0.002))	(12, (-0.0015, 0.002))	(10, (-0.0015, 0.002))	(22, (-0.0015, 0.002))
[-1.03 -1.23]	(538, (-0.0019, 0.0015))	(30, (-0.0015, 0.0019))	(14, (-0.0015, 0.002))	(11, (-0.0015, 0.002))	(25, (-0.0015, 0.0019))
[1.63 -0.11]	(564, (-0.001, 0.0019))	(31, (-0.0015, 0.002))	(14, (-0.0015, 0.002))	(12, (-0.0015, 0.002))	(22, (-0.0015, 0.002))
[-1.37 1.66]	(611, (-1.4934, 1.9909))	(41, (-1.4939, 1.9919))	(21, (-1.4939, 1.9919))	(13, (-1.4939, 1.9919))	(9, (-1.4939, 1.9919))
[-1.65 -1.43]	(1197, (-0.002, 0.0015))	(62, (-0.0015, 0.0019))	(30, (-0.0015, 0.002))	(24, (-0.0015, 0.002))	(34, (-0.0015, 0.0019))
[1.62 1.49]	(1244, (-0.001, 0.0024))	(64, (-0.0015, 0.002))	(32, (-0.0015, 0.002))	(25, (-0.0015, 0.002))	(36, (-0.0015, 0.0019))
[2.19 1.44]	(4660, (-0.001, 0.0023))	(228, (-0.0015, 0.002))	(116, (-0.0015, 0.002))	(82, (-0.0015, 0.002))	(79, (-0.0015, 0.002))
[-2.61 0.58]	(1225, (-1.4945, 1.9909))	(71, (-1.4939, 1.9919))	(36, (-1.4939, 1.9919))	(24, (-1.4939, 1.9919))	(17, (-1.4939, 1.9919))
[2.55 1.59]	(1, (2.55, 1.59))	(1293, (-0.0015, 0.002))	(662, (-0.0015, 0.002))	(449, (-0.0015, 0.002))	(355, (-0.0015, 0.002))
[-2.8 -1.17]	(1, (-2.8, -1.17))	(1469, (-0.0015, 0.0019))	(752, (-0.0015, 0.002))	(510, (-0.0015, 0.002))	(394, (-0.0015, 0.0019))
[3.23 -0.52]	(1, (3.23, -0.52))	(5568, (-0.0015, 0.0019))	(2852, (-0.0015, 0.002))	(1921, (-0.0015, 0.002))	(1462, (-0.0015, 0.002))
[-1.18 -3.07]	(1, (-1.18, -3.07))	(6162, (-0.0015, 0.0019))	(3156, (-0.0015, 0.002))	(2126, (-0.0015, 0.002))	(1618, (-0.0015, 0.0019))
[1.59 -2.97]	(1, (1.59, -2.97))	(9915, (-0.0015, 0.0019))	(5079, (-0.0015, 0.002))	(3418, (-0.0015, 0.002))	(2591, (-0.0015, 0.0019))
[-0.65 -3.31]	(1, (-0.65, -3.31))	(10000, (-0.4934, -2.5126))	(5215, (-0.0015, 0.002))	(3510, (-0.0015, 0.002))	(2661, (-0.0015, 0.002))
[0.62 3.33]	(5889, (-1.4929, 1.9925))	(294, (-1.4939, 1.9919))	(151, (-1.4939, 1.9919))	(101, (-1.4939, 1.9919))	(76, (-1.4939, 1.9919))
[3.22 -1.15]	(1, (3.22, -1.15))	(10000, (3.004, -1.0728))	(6909, (-0.0015, 0.002))	(4648, (-0.0015, 0.002))	(3517, (-0.0015, 0.002))
[-2.36 2.66]	(772, (-1.495, 1.9928))	(49, (-1.494, 1.9919))	(25, (-1.4939, 1.9919))	(16, (-1.4939, 1.9919))	(11, (-1.4939, 1.9919))
[-0.83 3.46]	(1036, (-1.4936, 1.9929))	(62, (-1.4939, 1.9919))	(31, (-1.4939, 1.9919))	(20, (-1.4939, 1.9919))	(14, (-1.4939, 1.9919))
[-1.3 3.34]	(892, (-1.4939, 1.9929))	(55, (-1.4939, 1.9919))	(28, (-1.4939, 1.9919))	(18, (-1.4939, 1.9919))	(12, (-1.4939, 1.9919))
[-2.84 2.2]	(888, (-1.4949, 1.9922))	(54, (-1.494, 1.9919))	(28, (-1.4939, 1.9919))	(18, (-1.4939, 1.9919))	(12, (-1.4939, 1.9919))
[2.97 2.32]	(1, (2.97, 2.32))	(1, (2.97, 2.32))	(1, (2.97, 2.32))	(1, (2.97, 2.32))	(1, (2.97, 2.32))
[-3.77 0.41]	(10000, (-3.6006, 0.5281))	(844, (-1.494, 1.9919))	(433, (-1.4939, 1.9919))	(291, (-1.4939, 1.9919))	(219, (-1.4939, 1.9919))
[3.7 0.98]	(1, (3.7, 0.98))	(1, (3.7, 0.98))	(1, (3.7, 0.98))	(1, (3.7, 0.98))	(1, (3.7, 0.98))
[1.12 -3.78]	(1, (1.12, -3.78))	(1, (1.12, -3.78))	(1, (1.12, -3.78))	(1, (1.12, -3.78))	(1, (1.12, -3.78))
[-1.19 -3.83]	(1, (-1.19, -3.83))	(1, (-1.19, -3.83))	(1, (-1.19, -3.83))	(1, (-1.19, -3.83))	(1, (-1.19, -3.83))
[-1.85 3.56]	(1043, (-1.4943, 1.9929))	(62, (-1.4939, 1.9919))	(32, (-1.4939, 1.9919))	(21, (-1.4939, 1.9919))	(14, (-1.4939, 1.9919))
[3.93 -0.85]	(1, (3.93, -0.85))	(1, (3.93, -0.85))	(1, (3.93, -0.85))	(1, (3.93, -0.85))	(1, (3.93, -0.85))
[3.62 1.87]	(1, (3.62, 1.87))	(1, (3.62, 1.87))	(1, (3.62, 1.87))	(1, (3.62, 1.87))	(1, (3.62, 1.87))
[3.98 1.16]	(1, (3.98, 1.16))	(1, (3.98, 1.16))	(1, (3.98, 1.16))	(1, (3.98, 1.16))	(1, (3.98, 1.16))
[-2.14 3.63]	(1192, (-1.4944, 1.9929))	(70, (-1.4939, 1.9919))	(36, (-1.4939, 1.9919))	(23, (-1.4939, 1.9919))	(16, (-1.4939, 1.9919))
[2.72 3.22]	(1, (2.72, 3.22))	(1, (2.72, 3.22))	(1, (2.72, 3.22))	(1, (2.72, 3.22))	(1, (2.72, 3.22))
[-2.09 3.84]	(1491, (-1.4944, 1.9929))	(84, (-1.4939, 1.9919))	(43, (-1.4939, 1.9919))	(28, (-1.4939, 1.9919))	(20, (-1.4939, 1.9919))
[-3.74 2.41]	(2977, (-1.4949, 1.9923))	(155, (-1.494, 1.9919))	(80, (-1.4939, 1.9919))	(53, (-1.4939, 1.9919))	(39, (-1.4939, 1.9919))
[3.54 3.5]	(1, (3.54, 3.5))	(1, (3.54, 3.5))	(1, (3.54, 3.5))	(1, (3.54, 3.5))	(1, (3.54, 3.5))
[3.85 -3.35]	(1, (3.85, -3.35))	(1, (3.85, -3.35))	(1, (3.85, -3.35))	(1, (3.85, -3.35))	(1, (3.85, -3.35))

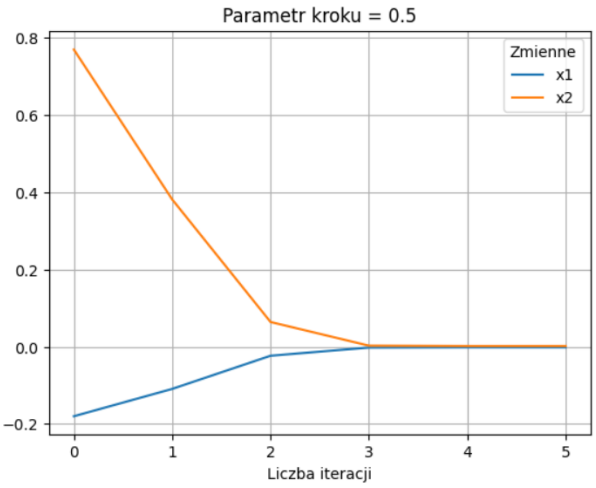
Spodziewane minima znalazłem za pomocą wykresów na Wolfram Alpha. Są to odpowiednio punkty (0,0) oraz około (-1.5, 2)

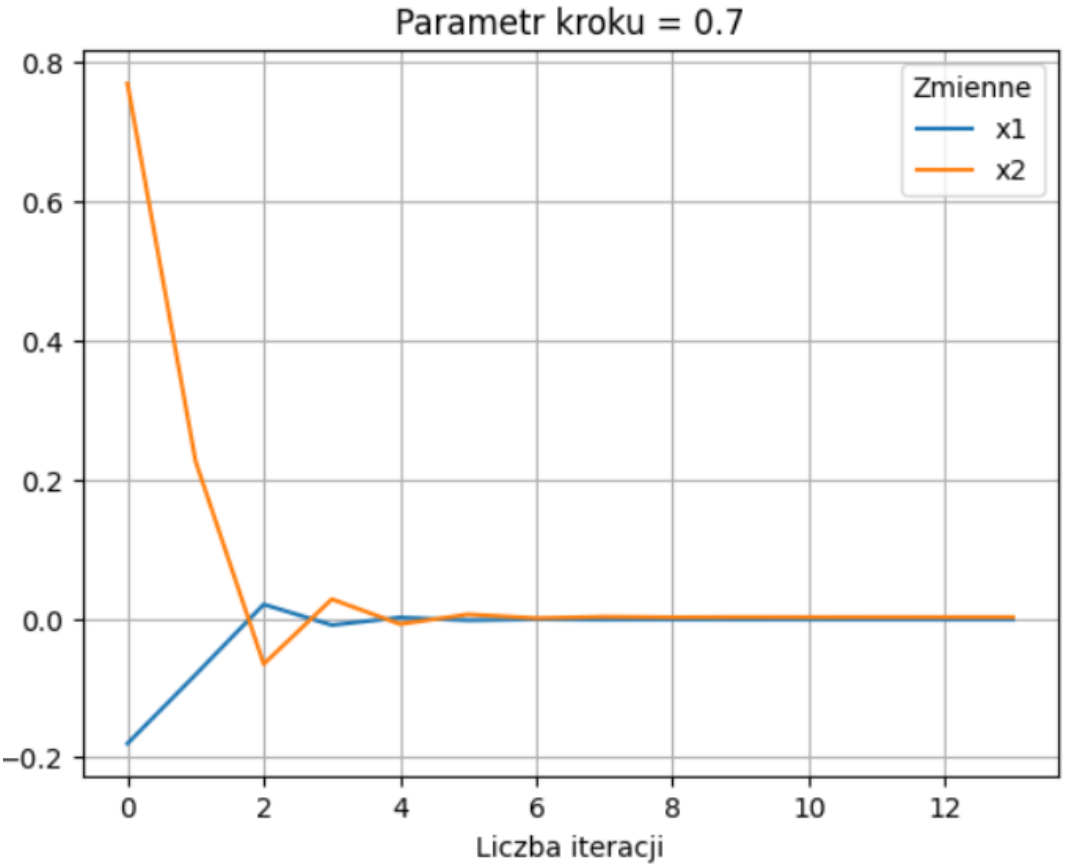


Dla punktu (-0.18, 0.77):

Tym razem punkt nie jest na tyle blisko, aby funkcja nie wykonała iteracji dla ß = 0.01 , lecz zajmuje jej to dużo powtórzeń. Najoptymalniejszym krokiem jest wartość 0.5, dzięki czemu algorytm potrzebuje jedynie 5 iteracji. Należy jednak uważać ze zwiększaniem wartości, gdyż dla ß=0.7 możemy łatwo zaobserwować na wykresie oscylację wokół szukanego minimum, przez co algorytm dłużej go szuka.

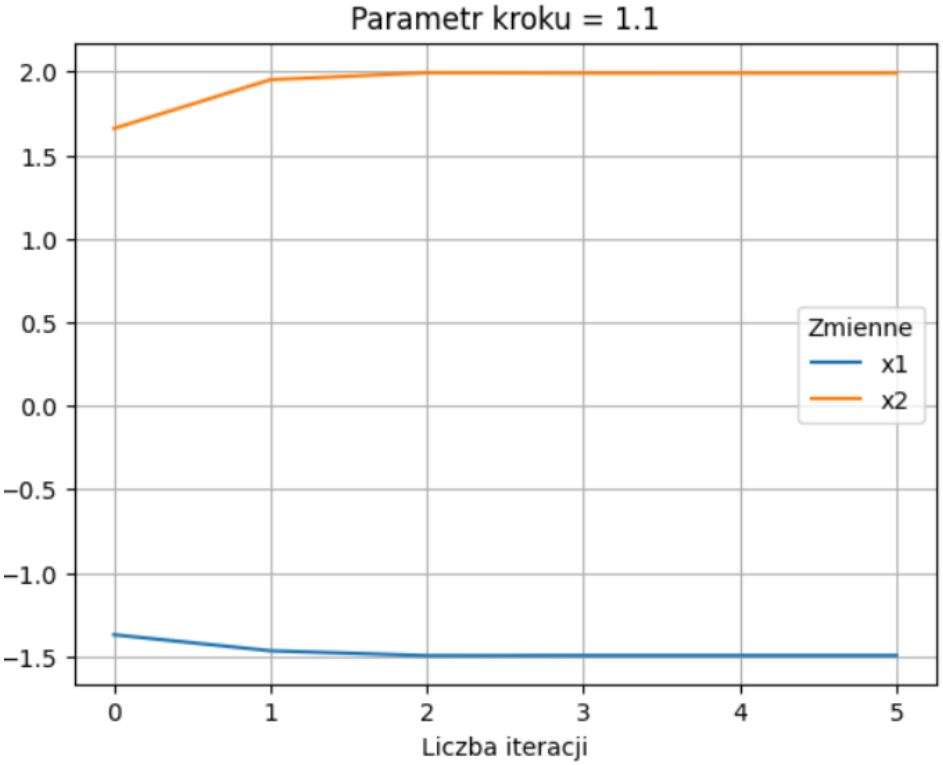


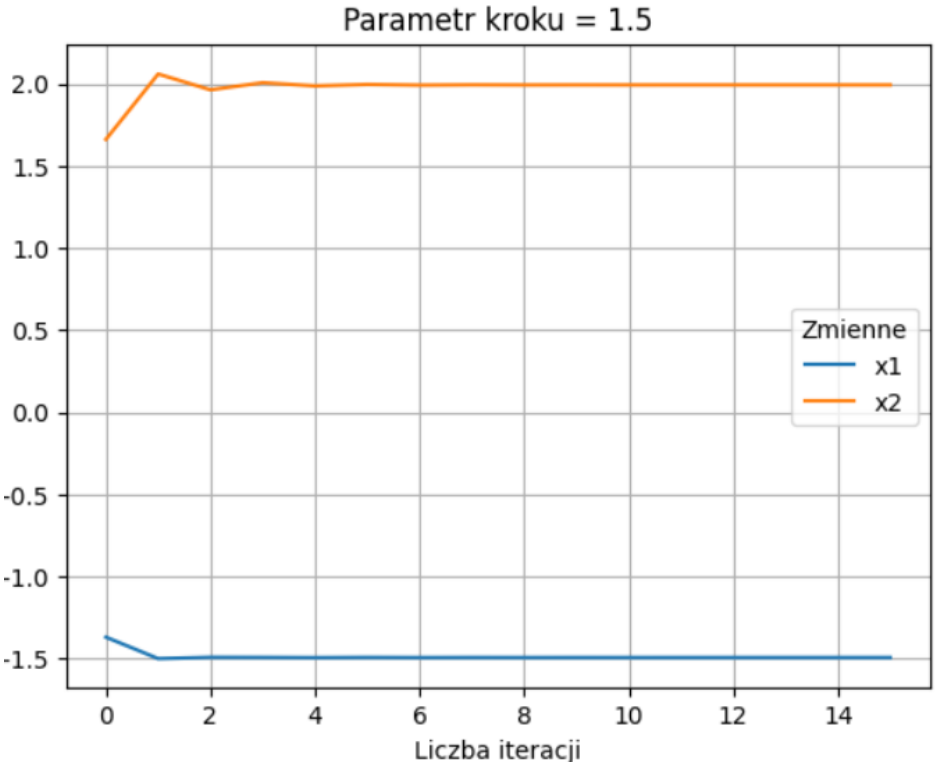




Dla punktu (-1.37, 1.66):

Jest to przypadek gdzie punkt startowy znajduje się bardzo blisko szukanego minimum (-1.5, 2), dlatego do niego dąży w niskiej liczbie kroków. Najoptymalniejszym parametrem jest ß = 1.1. Przy ß = 1.5 widzimy „przestrzelenie” kolejnych iksów.





Dla punktu (3.22, -1.15):

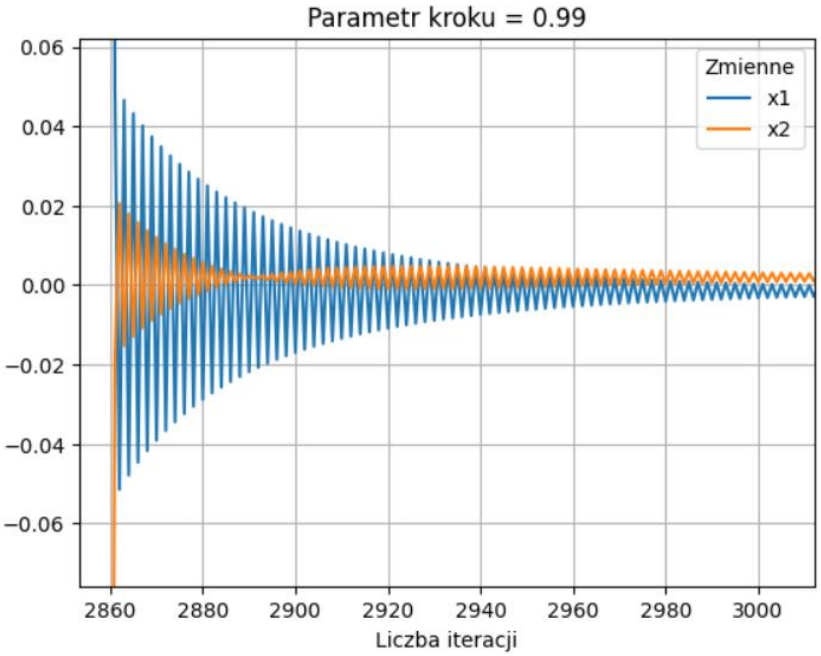
Dla kroku = 0.01 wykonuje jedną iterację i kończy działanie, gdyż ustawiona precyzja zmiennej difference mu na to nie pozwala

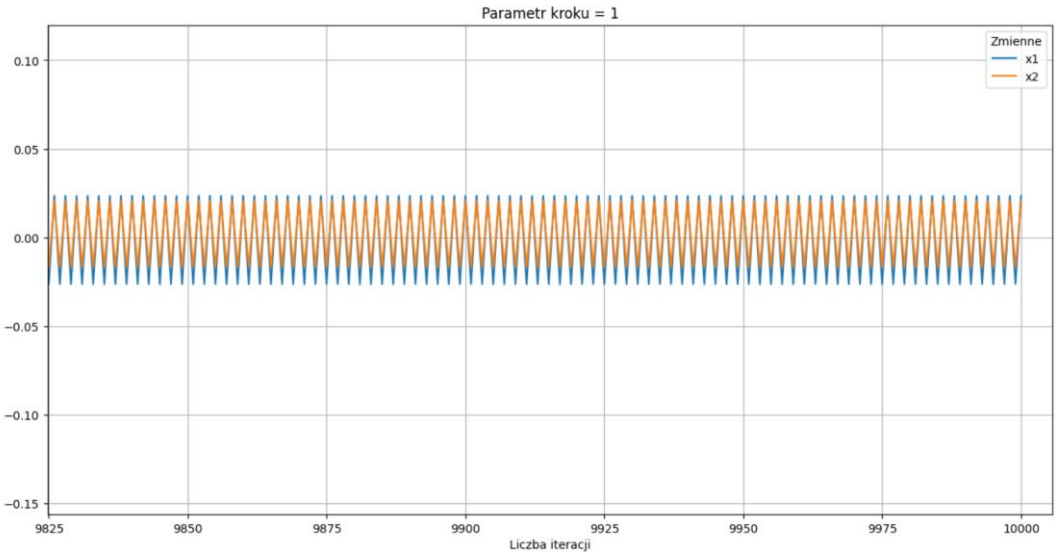
Dla kroku = 0.21 wykonuje maksymalną liczbę iteracji i nie jest w stanie dojść do minimum, przez zbyt małe przybliżanie się do ostatecznego punktu

Dla kroku >= 0.3, ale < 1 znajduje minimum

Dla kroku = 0.9 działa najefektywniej w liczbie 3190 iteracji

Dla kroku >= 1 oscyluje w okolicy minimum nie osiągając go w wyznaczonej liczbie 10000 iteracji

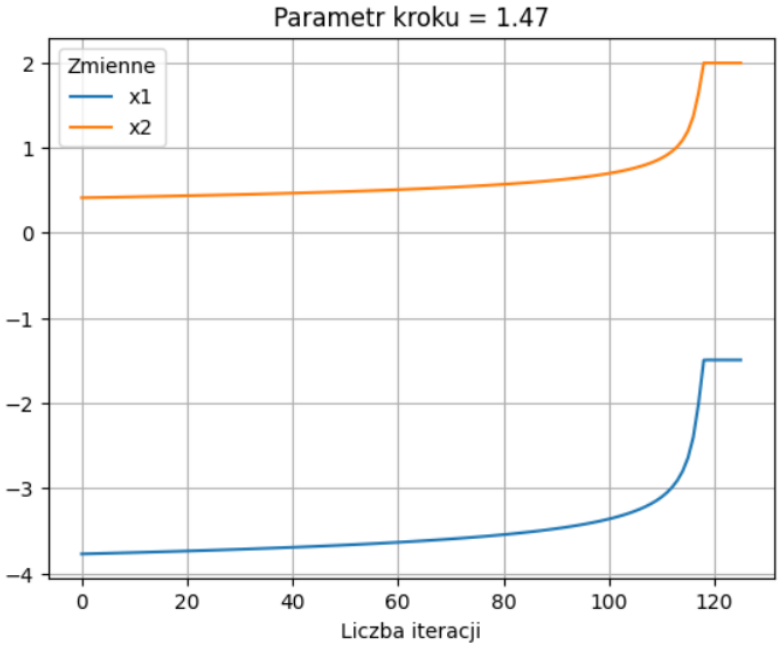


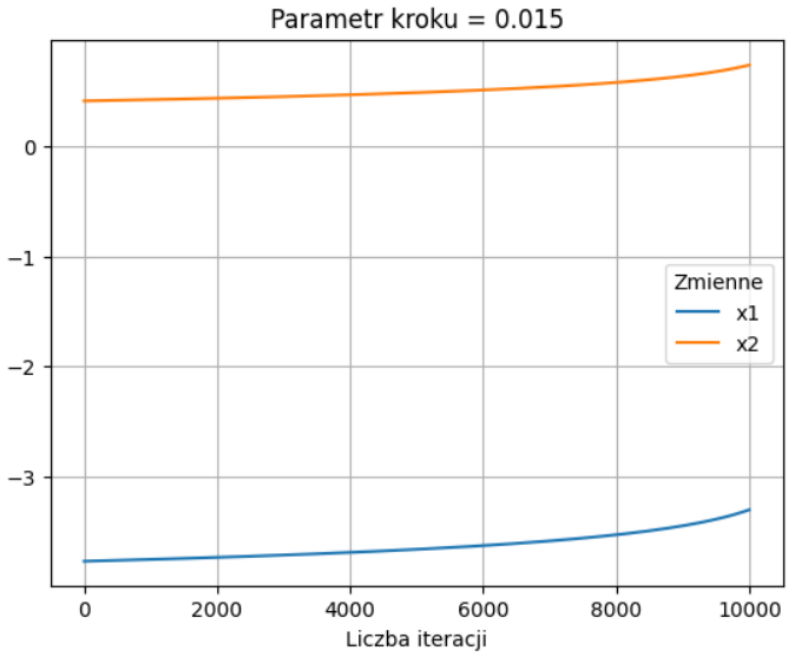


Dla punktu (-3.77, 0.41):

Najmniejszy krok ß=0.015 nie pozwala algorytmowi na dojście do minimum w 10 000 iteracjach, najoptymalniejszym parametrem jest wartość 1.47 – jest ona dosyć duża, gdyż wraz ze wzrostem argumentów x1, x2 maleją znacznie wartości gradientu (wykładnik jest sumą kwadratów pomnożonych przez -1), dlatego powinno się ją odpowiednio zwiększyć.

$$\nabla g(x) = \begin{bmatrix} 2x_1 \exp\left\{-x_1^2 - x_2^2\right\} + (x_1 + 1.5) \exp\left\{-(x_1 + 1.5)^2 - (x_2 - 2)^2\right\} \\ 2x_2 \exp\left\{-x_1^2 - x_2^2\right\} + (x_2 - 2) \exp\left\{-(x_1 + 1.5)^2 - (x_2 - 2)^2\right\} \end{bmatrix}$$





Wnioski:

1. Algorytm dla punktów startowych bardzo bliskich szukanego minimum i małej wartości kroku nie wykonuje się, dając przy tym mało dokładny wynik (Jeżeli dopuszczalna różnica między x_t a x_{t+1} w algorytmie nie jest wystarczająco mała).
2. Dla punktów startowych bliskich minimum algorytm działa najefektywniej przy stosunkowo dużej wartości kroku, zaś dla punktów znacznie oddalonych należy wybrać stosunkowo niską wartość kroku.
3. Wraz ze wzrostem wartości kroku (jeżeli nie przekroczy się jej powyżej dozwolonej powodującej rozbieżności) spada precyzja wykonywanych kroków, lecz otrzymany wynik jest bardziej precyzyjny ze względu na „większe przybliżenie się" do wartości minimalnej przed spełnieniem warunku na odpowiednio małą różnicę między x_t a x_{t+1}.
4. W przypadku funkcji, której gradient jest wielomianem, czas wykonywania algorytmu jest podobny niezależnie od punktu startowego dla tych samych wartości kroku, gdyż wartości gradientu szybko spadną do tych poniżej 1, a później będą tak samo małe ze względu na potęgowanie ułamków.
5. Parametr kroku nie może być ani zbyt duży, bo wtedy kolejne argumenty algorytmu rozbiegają się do nieskończoności, ani zbyt mały, gdyż wtedy algorytm wykonuje się bardzo długo lub przy implementacji z konkretną różnicą zmiennej difference, wcale się nie wykona.
6. Dla większych wartości kroku niż najoptymalniejsza, kolejne wartości algorytmu nie muszą rozbiegać do nieskończoności, lecz mogą „oscylować" w górę i w dół wokół szukanego minimum, przez co czas jego wykonywania się zwiększa.
7. Rodzaj badanej funkcji a tym samym jej gradient ma wpływ na optymalny dobór wartości kroku i punktu startowego. Należy zobaczyć jak gwałtownie wyliczane są kolejne wartości gradientu.
8. Idealnie dobrane wartość kroku i punkt startowy pozwalają uzyskać w szczególnych przypadkach dobry wynik nawet po 1 iteracji.