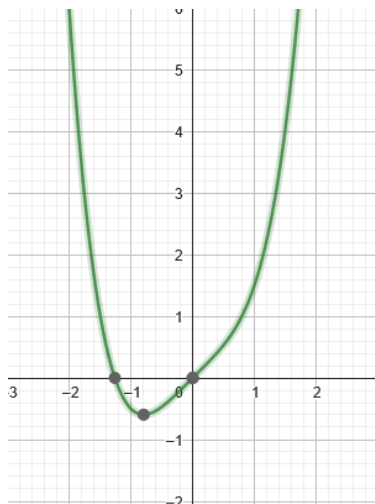


Rafał Mironko, 331405

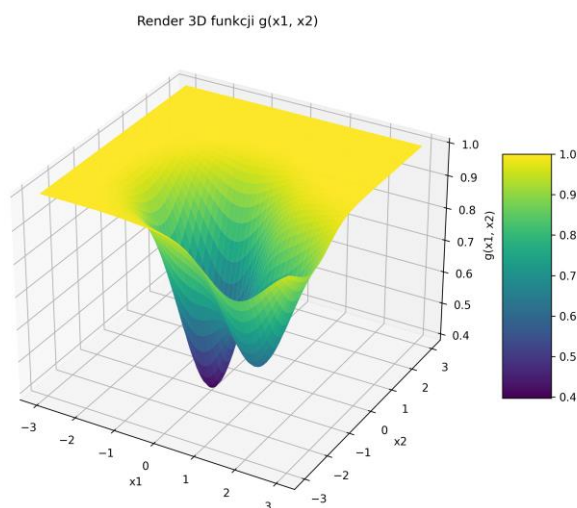
## Ćwiczenie 1 – Metoda Gradientu Prostego

Celem eksperymentów jest zbadanie wpływu rozmiaru kroku dla różnych, losowych punktów początkowych.

Są dwie funkcje.  $F(x)$  wyraźnie ma jedno minimum. Można oczekiwać, że nieważne od punktu początkowego, uda się odkryć minimum globalne.



$G(x)$  ma jedno minimum globalne i dwa lokalne. Do tego, w punktach oddalonych od tych minimów  $g(x)$  jest bardzo płaska. Moje wstępne oczekiwania są takie, że funkcja trafi w jedno z dwóch minimów lokalnych albo utknie na prawie-płaszczyźnie daleko od nich.



Badanie będzie następujące: wylosujemy kilka punktów, będących w rozsądnej odległości od naszych minimów lokalnych. Następnie, dla każdego z tych punktów sprawdzimy, jaki rozmiar kroku jest optymalny (lub najlepszy ze sprawdzonych rozwiązań) i przy jakim rozmiarze kroku nie

uda się osiągnąć żadnego minimum. Przy każdym badaniu punkty będą inne, ale rozmiar kroku taki sam.

Badanie  $f(x)$

Badania zacząłem od wybrania pewnego punktu  $x=-2$ . Następnie sprawdziłem różne rozmiary kroku i zachowanie się algorytmu. Wygląda na to, że  $\beta=0.26$  jest w tym punkcie wartością graniczną, po przekroczeniu której algorytm nie wyznaczy minimum globalnego.

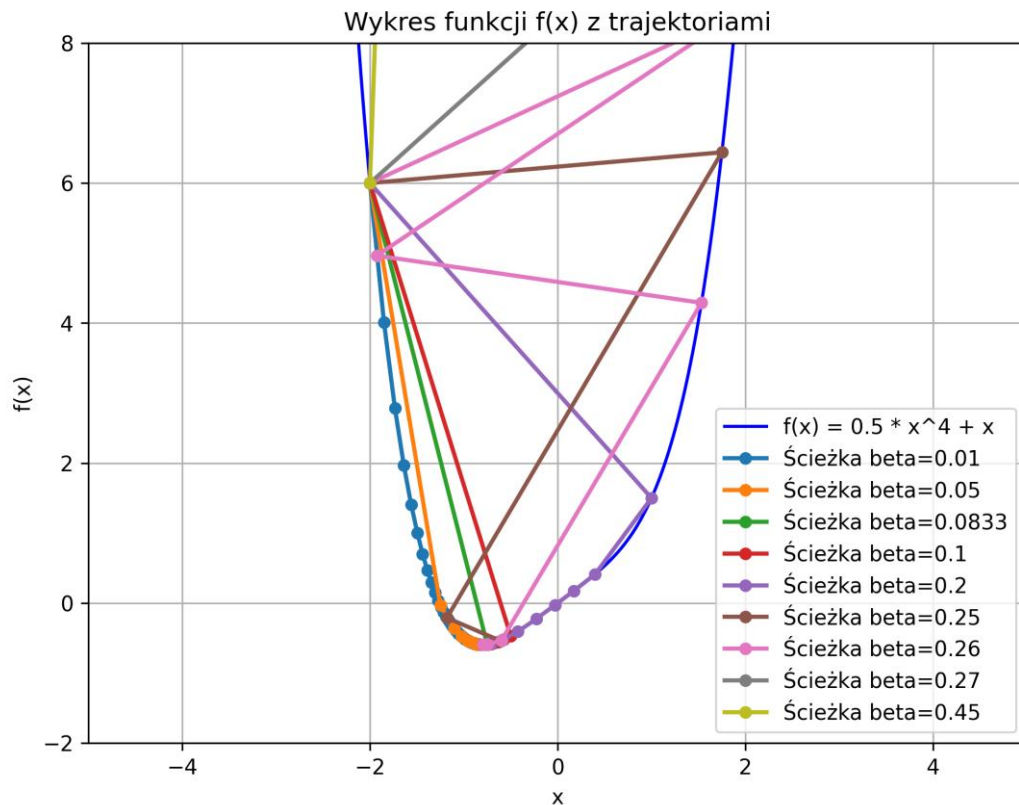


Fig. 1

Sprawdźmy, jak zachowa się algorytm przy innych punktach.

step	mean_iterations	std_iterations	mean_minimum	std_minimum
0.005	723.2	129.4797281430572	-0.5952753944750993	1.4958867187071863e-13
0.01	358.74	64.67791276780659	-0.5952753944753316	2.563018871845551e-13
0.1	29.98	6.715623574918416	-0.595275394479	2.3175813106097544e-12
0.2	12.04	2.821063629200873	-0.5952753944836977	3.022363916153003e-12
0.25	7.72	2.01037309970065	-0.5952753944862661	3.0213600177957395e-12
0.26	6.86	1.854831528737853	-0.5952753944869464	1.9756150044910167e-12
0.27	2006.0	13999.142963767461		
0.45	16025.68	36649.3979521847		

Fig. 2

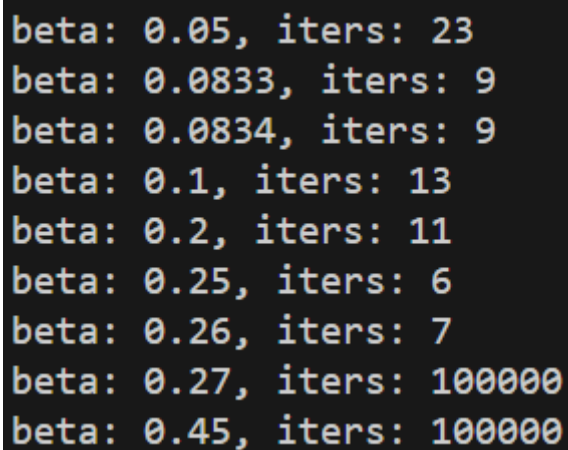
W teście było przetestowanych 50 różnych punktów. Wygląda na to, że wartość 0,25 sprawdziła się tu najlepiej – co jest dla mnie zaskakujące, bo wtedy oscylacje są na granicy ucieczki.

Sprawdziłem, że chodzi o wartości własne macierzy (mimo że tutaj macierz będzie jednowymiarowa). Oscylacje zaczynają występować, gdy beta zbliża się do dwukrotności odwrotności wartości własnej macierzy (czyli  $2/6x^2$ ).

Widzimy to na fig.1: dla punktu  $x=2$ , beta nie powinna przekroczyć  $2/6 \cdot 2^2 = 1/12$ , czyli około 0.0833 (widać na fig.1, jak przy tej wartości funkcja idealnie trafia w okolice minimum). Po przekroczeniu tej wartości, pierwszy krok będzie zbyt długi, i algorytm będzie musiał się „cofać”. Z kolei powyżej 0.2 oscylacje są naprawdę mocne, a po przekroczeniu 0.25, czyli  $6/6x^2$ , algorytm przestaje działać.

Hipoteza #1: niech pierwszy krok będzie dwukrotnością odwrotności wartości własnej macierzy dla danego punktu. Wtedy już po pierwszym kroku będziemy bardzo blisko minimum funkcji (w odległości mniejszej niż epsilon).

Ciekawe jest to, że jednak wykorzystanie tej wartości własnej wcale nie daje nam minimalnej liczby iteracji. Ta minimalna wartość jest osiągana tuż przed wartością ucieczki, nie wiem, z czego to wynika. Sprawdziłem to dla zmiany dokładności epsilon, z jakiegoś powodu za każdym razem te wyższe wartości są lepsze i pozwalają uzyskać mniejszą liczbę kroków:



```
beta: 0.05, iters: 23
beta: 0.0833, iters: 9
beta: 0.0834, iters: 9
beta: 0.1, iters: 13
beta: 0.2, iters: 11
beta: 0.25, iters: 6
beta: 0.26, iters: 7
beta: 0.27, iters: 100000
beta: 0.45, iters: 100000
```

Fig. 3

Widać jednak, że dla tych zbliżonych do obliczonego przez nas kroku, występuje pewna anomalia i ilość iteracji jest mniejsza. Widać też, że dla wartości trzykrotnie większej, wydaje się, że zbliżamy się do najmniejszej koniecznej ilości iteracji.

Hipoteza: gdy beta zbliża się do sześciokrotności odwrotności wartości własnej macierzy (czyli  $6/6x^2$ ), potrzeba najmniej iteracji.

Sugerowane przeszukiwanie przestrzeni: Jeśli nie chcemy wyliczać hesjana, zbadajmy wartość beta niedaleko punktu ucieczki. Tam wartość będzie najbliższa optymalnej – i to potwierdzają moje losowe testy:

```

step,mean_iterations,std_iterations,mean_minimum,std_minimum
0.01,352.6,64.95967980216652,-0.5952753944753568,2.283932613682936e-13
0.05,65.82,12.860311038229208,-0.5952753944771093,1.181991971000241e-12
0.0833,36.7,7.945438943192503,-0.595275394478771,2.1914953210087383e-12
0.1,29.74,6.269960127464927,-0.5952753944798792,2.0604020962569875e-12
0.1667,15.48,3.413151036798694,-0.595275394482653,3.0475734896807887e-12
0.2,11.58,2.9263629303283625,-0.5952753944833836,3.4022342919075614e-12
0.25,7.42,2.0502682751288916,-0.5952753944860549,3.077846908935402e-12
0.26,6.54,1.899578900704048,-0.5952753944867036,2.9108505897872483e-12
0.27,5.96,1.9074590428106184,-0.5952753944854856,2.3499558646414444e-12
0.28,2006.38,13999.088684467997,,
0.29,4006.68,19594.554469484632,,
0.3,6007.02,23746.910650010876,,
0.31,6007.84,23746.70349447266,,
0.32,8008.24,27126.890140640888,,
0.45,20023.94,39988.03005070892,,

```

Fig. 4

Niestety, nie wiem, skąd to wynika. Ciężko było mi zrozumieć badania innych profesorów nad tym. Mamy już jednak pewną intuicję szukania najlepszego rozwiązania.

$G(x_1, x_2)$

Ciężko mi policzyć wzór na wartość własną podanej funkcji wielu zmiennych. Postaram się dojść do prawie optymalnego wyniku, korzystając z metody, którą opisałem w mojej hipotezie i wnioskach z funkcji  $f$ . W końcu po to ona jest, żeby oszczędzić sobie liczenia hesjanów.

Przeprowadziłem badanie i oto moje wnioski. Badanie jednego punktu (1,0) daje bardzo podobne rezultaty, jeśli chodzi o najlepszą długość kroku, co badanie punktów generowanych losowo. Oto badania w punkcie (1,0):

```

beta: 0.25, iters: 68
beta: 0.5, iters: 32
beta: 0.75, iters: 19
beta: 0.8, iters: 18
beta: 0.85, iters: 16
beta: 0.9, iters: 15
beta: 0.95, iters: 16
beta: 1, iters: 17
beta: 1.5, iters: 58
beta: 2, iters: 10000

```

Fig. 5

A oto dla punktów losowych, generowanych w przedziale  $(-2;-3)$ ,  $(3,1)$ :

```

step,mean_iterations,std_iterations,mean_minimum,std_minimum
0.01,1496.8,304.8681026280053,0.4318793425299935,0.07822374815281087
0.05,296.0,61.282950320623435,0.43187934252771976,0.07822374815249737
0.0833,175.8,36.93995127230137,0.431879342527229,0.0782237481542854
0.1,145.6,30.897249068485042,0.43187934252742527,0.07822374815193751
0.1667,85.8,18.540765895722863,0.43187934252054544,0.07822374815567727
0.2,70.8,15.522886329545802,0.43187934251813676,0.07822374815474188
0.25,55.6,12.49959993599796,0.43187934251811166,0.07822374815134277
0.5,25.0,6.356099432828281,0.431879342515679,0.0782237481562377
0.75,14.8,4.44522154178574,0.43187934249291837,0.07822374814630474
0.8,13.4,4.029888335921977,0.43187934249293286,0.07822374815928204
0.85,12.0,4.09878030638384,0.4318793424975837,0.07822374814593512
0.9,11.2,4.069397989875161,0.4318793424939097,0.07822374814257271
0.95,12.2,3.867815921162743,0.4318793424925344,0.0782237481409498
1.0,12.4,3.8781438859330635,0.43187934249209425,0.07822374814064875
1.5,43.2,18.723247581549522,0.4318793425064317,0.0782237481280433
1.6,129.4,61.399022793526605,0.43187934251391724,0.07822374812842392
1.62,222.2,107.45492078076276,0.43187934251680626,0.07822374813277094
1.65,80001.6,39996.8,0.43281688291375,0.07775497794871133
1.7,80001.8,39996.399999999999,0.44707713246129577,0.07062485315788572
1.8,80001.8,39996.399999999999,0.4732209233118215,0.05755295773475387
2.0,80003.6,39992.799999999996,0.5176653677576624,0.03533073551532463
2.5,100000.0,0.0,0.6000000000000001,1.4043333874306804e-16
3.0,100000.0,0.0,0.5593522755024202,0.05365719558212322

```

Fig. 6

Wygląda na to, że krok 0.9 jest tutaj blisko rozwiązania optymalnego. Potem występują oscylacje, a po osiągnięciu mniej więcej 1,9-krotności następuje ucieczka. To jest bardzo zbliżone do obserwacji profesorów z książki „sztuczna inteligencja dla inżynierów”. Mówią oni, że ucieczka nastąpi po osiągnięciu dwukrotności punktu optymalnego i wygląda na to, że może to być prawda – osiągnąłem bardzo zbliżony efekt.

Warto też wspomnieć, co stanie się, gdy trafimy na miejsce wykresu, gdzie funkcja bardzo powoli maleje. Wtedy wszystko będzie zależało od epsilon i funkcja zadziała poprawnie, ale tylko przy bardzo niewielkiej wartości epsilon. Raczej nic ciekawego tam się nie dzieje, dlatego nie testowałem tego zbyt długo – tylko by zaburzyć ciekawe wyniki.

Końcowe wnioski:

- jest kilka sposobów na osiągnięcie prawie optymalnego rozwiązania (pod kątem ilości iteracji, zachowując dokładność). Najlepiej znaleźć odwrotność wartości własnej macierzy dla danego punktu. Potem sprawdzamy jej wielokrotności, jedna z nich będzie rozwiązaniem optymalnym. Możemy też przeszukiwać przestrzeń rozwiązań i szukać miejsca, gdzie rozpoczną się oscylacje – czyli, gdy pierwszy krok „przeskoczy” minimum funkcji. Jakaś wielokrotność tego kroku będzie prawdopodobnie rozwiązaniem optymalnym.
- różne wielokrotności wydają się działać dla różnych wymiarów.
- raczej nie warto badać wartości funkcji daleko od miejsc, gdzie są wszystkie ekstrema lokalne. Tam zwykle nic ciekawego się nie dzieje i tylko może to zaburzyć wyniki badań ekstremów.