**1. Wstęp**

Zadaniem napisanego przeze mnie programu było obliczenie globalnego minimum funkcji Rosenbrocka. Funkcja Rosenbrocka wyrażona jest wzorem:

Dla prostoty oznaczeń zmienne funkcji f przyjmuję jako , zamiast standardowego x i y. Tak więc jakiś punkt należący do funkcji f będzie wektorem oznaczonym po prostu jako x

x=[]

Funkcja Rosenbrocka to funkcja dwóch zmiennych. Do obliczenia minimum globalnego tej funkcji wybrałam metodę Powella. Wiedzę o tym, jak działa ta metoda oraz wszystkie inne metody potrzebna przy metodzie Powella czerpałam z książki „: The Art of Scientific Computing. Numerical Recipes 3rd Edition” autorstwa W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, oraz z materiału wideo dostępnego pod linkiem: <https://www.youtube.com/watch?v=1Z_4sBNoZj4>

**2. Funkcja „powell”.**

Metoda Powella, którą w programie realizuje funkcja „powell”, poszukuje minimum globalnego funkcji dwóch zmiennych w postaci ciągu minimalizacji jednowymiarowych. Jednowymiarowa funkcja, która będzie minimalizowana to pomocnicza funkcji g o jednej zmiennej alfa. Funkcja g wyrażona jest następująco:

gk(alfa) = f(xk + alfa \* pk), gdzie:

xk – punkt, w którym znajdujemy aktualne przybliżenie minimum funkcji Rosenbrocka.

Punkt xk to wektor składający się z dwóch wartości: wartości z osi x i wartości z osi y

alfa- minimum funkcji gk będące też minimum kierunkowym funkcji Rosenbrocka

pk – aktualny kierunek poszukiwań minimum funkcji Rosenbrocka

Funkcja g: geometrycznie jest to “ślad” trójwymiarowego wykresu funkcji f(x) przeciętego płaszczyzną zawierającą punkt xk i wektor pk.

Algorytm funkcji „powell’:

***0****.* Dopóki odległość między punktem x znalezionym w ostatnim cyklu a punktem x znalezionym w przedostatnim cyklu jest większa od zadanej tolerancji (tol = 1.0e-9), wykonuj punkty 1-6. Punkty 1-6 stanowią jeden cykl obliczania minimum globalnego funkcji Rosenbrocka. Cykle liczone są za pomocą licznika „k”.

***1.*** Oznacz znalezione w poprzednim cyklu xn+1 jako x0 (w kodzie x0 to xOld), czyli

xOld = x.copy(), ponieważ od niego będzie startował następny cykl.

Ten punkt x0 (czyli xOld) jest aktualnym przybliżeniem minimum funkcji f.

***2.*** W pętli kontrolowanej przez licznik „i” (i= 0, …, n-1) dokonuj ciągu minimalizacji funkcji g w kierunkach pi, obliczając kolejne punkty xi, czyli przybliżenia minimum funkcji f. Te zadania realizowane są w poniższy sposób:

***2a.*** Wybieramy pewien kierunek poszukiwań pi .

***2b.*** Konstruujemy funkcję jednej zmiennej gi :R → R gi(α) = f(xi + αpi) i

metodą Brenta znajdujemy α takie, że funkcja gi osiąga minimum.

To znalezione alfa, czyli minimum funkcji g jest też minimum kierunkowym funkcji f.

***2c.*** Oblicz kolejne przybliżenie minimum funkcji f, czyli xi+1 = xi + αpi .

***2d****.* Wychodzimy z pętli „i” mając obliczone xn.

***3.*** Za pomocą xn (obliczonego jako ostatnie w pętli „i”) oblicz nowy kierunek poszukiwań pn - jest to sprzężony kierunek poszukiwań (to znaczy że przechodzi przez dwa poprzednio wyliczone punkty x). pn = xn – x0 . Ten kierunek zapamiętujemy, ponieważ jest to kierunek sprzężony (poruszając się wzdłuż niego poruszamy się w stronę prawdziwego minimum globalnego funkcji f).

***4.*** Konstruujemy funkcję jednej zmiennej gn :R → R gn(α) = f(xn + αpn) i metodą Brenta znajdujemy α takie, że funkcja gn osiąga minimum.

To znalezione alfa, czyli minimum funkcji g jest też minimum kierunkowym funkcji f.

***5.*** Oblicz kolejne przybliżenie minimum funkcji f wzdłuż znalezionego kierunku sprzężonego pn, czyli xn+1 = xn + αpn . To przybliżenie minimum zapamiętujemy, ponieważ zostało ono obliczone poruszając się wzdłuż sprzężonego kierunku pn (czyli w stronę prawdziwego minimum).

***6****.* Idź do punktu 0.

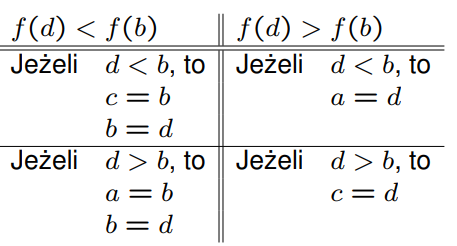
**3. Funkcja „bracket”**

Ta funkcja służy do wstępnej lokalizacji minimum funkcji jednej zmiennej. To znaczy do wyznaczenia przedziału, w którym będziemy szukać tego minimum.Mówimy, że trzy punkty (a, b, c) ograniczają (lokalizują) minimum funkcji f(x), jeżeli a < b < c : g(a) > g(b), g(c) > g(b) (\*)

Jak znaleźć takie punkty? Obliczamy wartość funkcji w dwóch punktach. Jeżeli nie zachodzą bardzo szczególne okoliczności, wyznaczają one lokalny kierunek spadku funkcji. Wyznaczamy trzeci punkt idąc w kierunku spadku, z pewnym krokiem (step). Jeśli warunek (\*) nie jest spełniony, podwajamy krok, zawsze biorąc pod uwagę dwa ostatnio obliczone punkty. Trzeba założyć maksymalną dopuszczalną liczbę kroków), aby zabezpieczyć się przed nieskończoną iteracją, gdybyśmy znaleźli się na monotonicznie malejącej gałęzi funkcji.

**4. Funkcja „new\_bracket”**

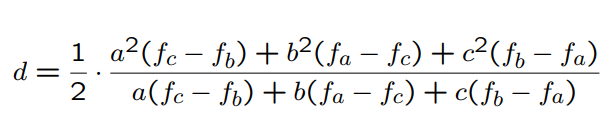
Ta funkcja robi podobną rzecz, co funkcja „bracket”, z tym że oblicza ona nowy przedział wstępnej lokalizacji minimum, gdy dostarczymy nowy punkt d, taki, że a < d < c, d ≠ b. Ten nowy punkt d jest nowym przybliżeniem minimum funkcji jednowymiarowej (czyli alfa). Wyliczanie nowego przedziału wygląda tak:



„Przesuwając” punkty a, b, c, d, analogicznie „przesuwamy” też obliczone wartości funkcji dla tych punktów (na przykład c = b ⇒ fc = fb ≡ f(b)), żeby nie trzeba było ich ponownie obliczać.

**5. Funkcja „parabolic\_interpolation”**

Przez trzy punkty (a, fa≡f(a)), (b, fb), (c, fc) przeprowadzamy parabolę, a jako punkt d bierzemy jej minimum, obliczone za pomocą wzoru:



**6. Funkcja „bisection”**

Ta funkcja oblicza punkt, będący środkiem zadanego przedziału (a, c).

**7. Funkcja „brent”**

Metoda Brenta, która w programie jest realizowana przez funkcję „brent” służy do obliczania minimum funkcji jednej zmiennej. W moim programie oblicza ona alfa, czyli minimum funkcji g, będące też minimum kierunkowym funkcji f. Metoda Brenta poszukuje alfa w zawężonym przedziale (a, c). W metodzie Brenta alfa oznaczone jest przez „d”. (Skrót myślowy – przez d (czyli alfa) rozumiem punkt, w którym znajduje się minimum funkcji g. Wartość tego minimum to g(d)).

Algorytm metody Brenta:

***0.*** Dopóki różnica długości między starym przedziałem poszukiwań minimum a nowym przedziałem jest większa od zadanej tolerancji, wykonuj:

***1a.*** Dla pierwszego cyklu: W przedziale wstępnej lokalizacji minimum (wyznaczonego przez funkcję „bracket” szukamy minimum funkcji g (to minimum oznaczone jest przez d). Robimy to za pomocą funkcji parabolic\_interpolation.

***1b.*** Dla wszystkich następnych cykli: W przedziale wstępnej lokalizacji minimum (wyznaczonego przez funkcję „new\_ bracket” szukamy minimum funkcji g- czyli alfa (to minimum oznaczone jest przez d). Robimy to za pomocą funkcji parabolic\_interpolation.

***2a.*** Akceptujemy tak obliczone minimum d, jeżeli:

- d leży wewnątrz przedziału a < d < c, oraz

- rozmiar nowego przedziału, wyznaczonego przez funkcję new\_bracket, jest mniejszy niż połowa przedziału obliczonego w przedostatnim cyklu.

Jeżeli akceptujemy tak obliczone d, to zwracamy punkt w którym znaleziono minimum (czyli d) i wartość tego minimum w tym punkcie. (return d, g(d))

***2b***. Jeżeli nie akceptujemy nowego d, dokonujemy bisekcji przedziału [a, c] metodą „bisection” i jako d bierzemy jego środek, a dalej wyznaczamy nowy przedział lokalizacji minimum, zawierający nowo obliczone d.

***3.*** Wróć do punktu 0.

**8. Wynik działania programu**

Według literatury minimum funkcji Rosenbrocka wynosi 0 i jest zlokalizowane w punkcie [1, 1].

Mój program prawidłowo zlokalizował ten punkt i obliczył w tym punkcie minimum o wartości

6.943103248178465e-24. Jest to zadowalający wynik, gdyż ta wartość jest bardzo bliska 0.

Obliczenie minimum wymagało 15 cykli metody Powella.

Program wykonuje też wizualizację działania algorytmu: rysowana jest w 3D funkcja Rosenbrocka, a czerwonymi kropkami zaznaczone są kolejne obliczane przybliżenia minimum funkcji Rosenbrocka (tylko te minima, które były wyliczane w kierunku sprzężonym).