Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Algorytmy ewolucyjne

Sprawozdanie z projektu nr 1

Karol Borowski

Warszawa, 2019

Spis treści

1. Wstęp

Projekt ma na celu znalezienie minimum funkcji Rosenbrock'a („bananowej") bez ograniczeń:

$$f(x) = [1 - x + a]^2 + 100[y - b - (x - a)^2]^2 \tag{1.1}$$

Stałe a oraz b, a także punkty startowe zostały wylosowane z załączonej tablicy za pomocą funkcji `randi(30)`, po zainicjowaniu generatora funkcją `rng(399)`. Wylosowany został poniższy wiersz tabeli:

Tablica 1.1. Wylosowane stałe a i b oraz punkty startowe do zadania optymalizacji

Lp.	a	b	$\boldsymbol{X_1}$	$\boldsymbol{Y_1}$	$\boldsymbol{X_2}$	$\boldsymbol{Y_2}$	$\boldsymbol{X_3}$	$\boldsymbol{Y_3}$	$\boldsymbol{X_4}$	$\boldsymbol{Y_4}$
12	1	−1	3	0	2	−2	0	−2	0	0

W celu optymalizacji funkcji użyłem czterech dostępnych w MATLAB'ie algorytmów: `fminsearch`, `fminunc(quasi-newton)`, `fminunc(trust-region)`, `fminunc(steepdesc)` oraz porównałem ich działanie.

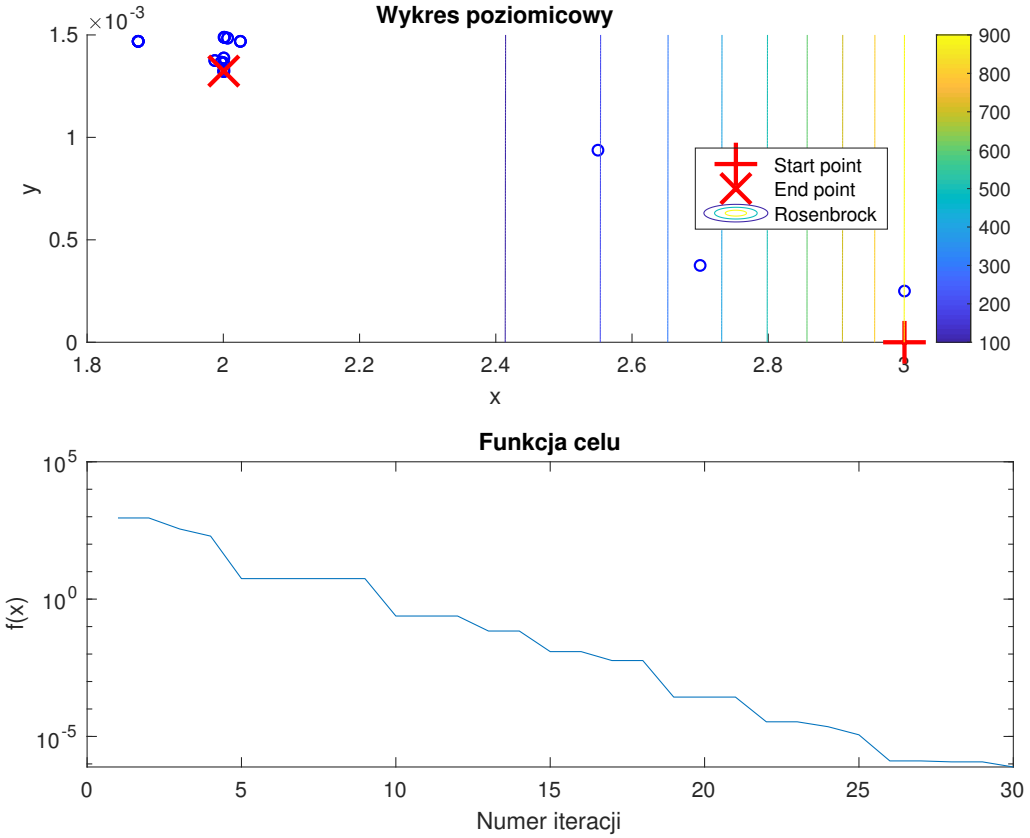
2. Optymalizacja bez pochodnych

Pierwszą wykorzystaną metodą jest algorytm `fminsearch`, czyli optymalizacja bez użycia pochodnych. Metoda daje wyniki rzędu 10^{-7}, co jest jest wynikiem bliskim 0, aczkolwiek jest możliwe osiągnięcie lepszego rezultatu. Główną wadą algorytmu jest stosunkowo duża ilość iteracji, potrzebnych do wykonania zadania. Dla jednego punktu startowego liczba ta wynosi 117, co jest dosyć sporym wynikiem.

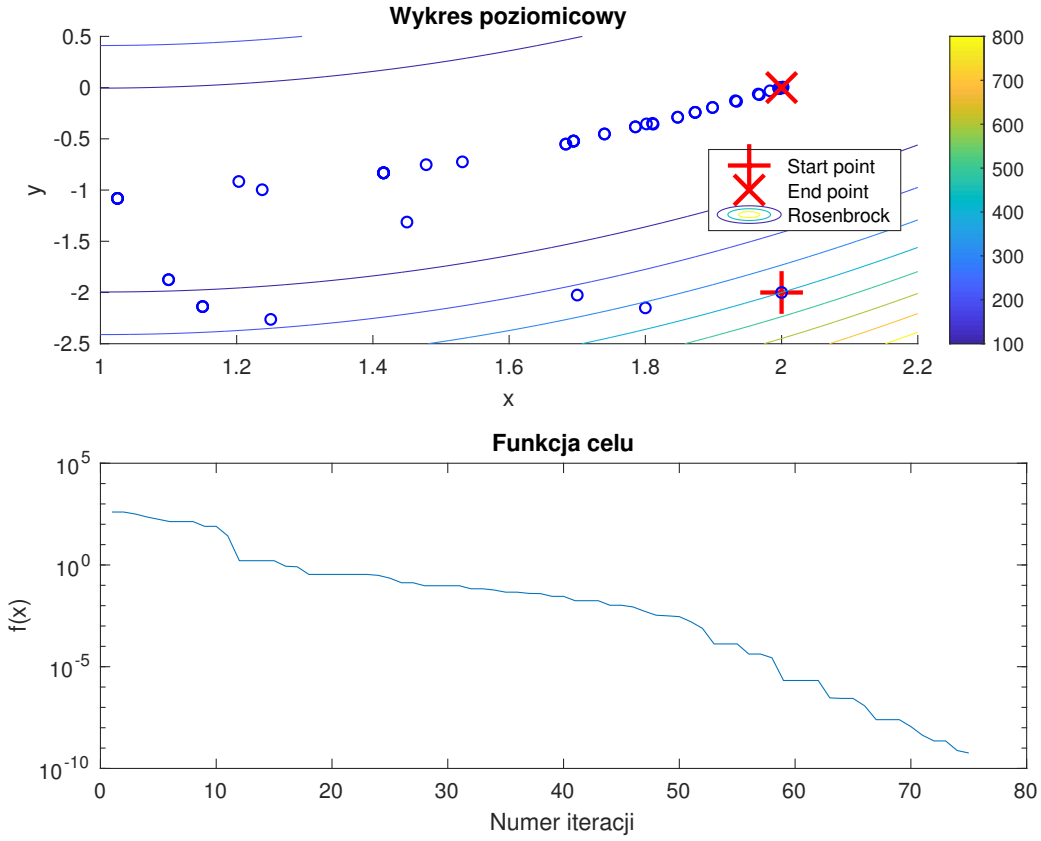
Oznaczenia:
X_0 - startowy punkt X
Y_0 - startowy punkt Y
X_{inf} - końcowy punkt X
Y_{inf} - końcowy punkt Y
dX - błąd bezwzględny punktu X
dY - błąd bezwzględny punktu Y
$f(x)$ - wartość funkcji celu
Iterations - liczba iteracji

Tablica 2.1: Porównanie punktów startowych

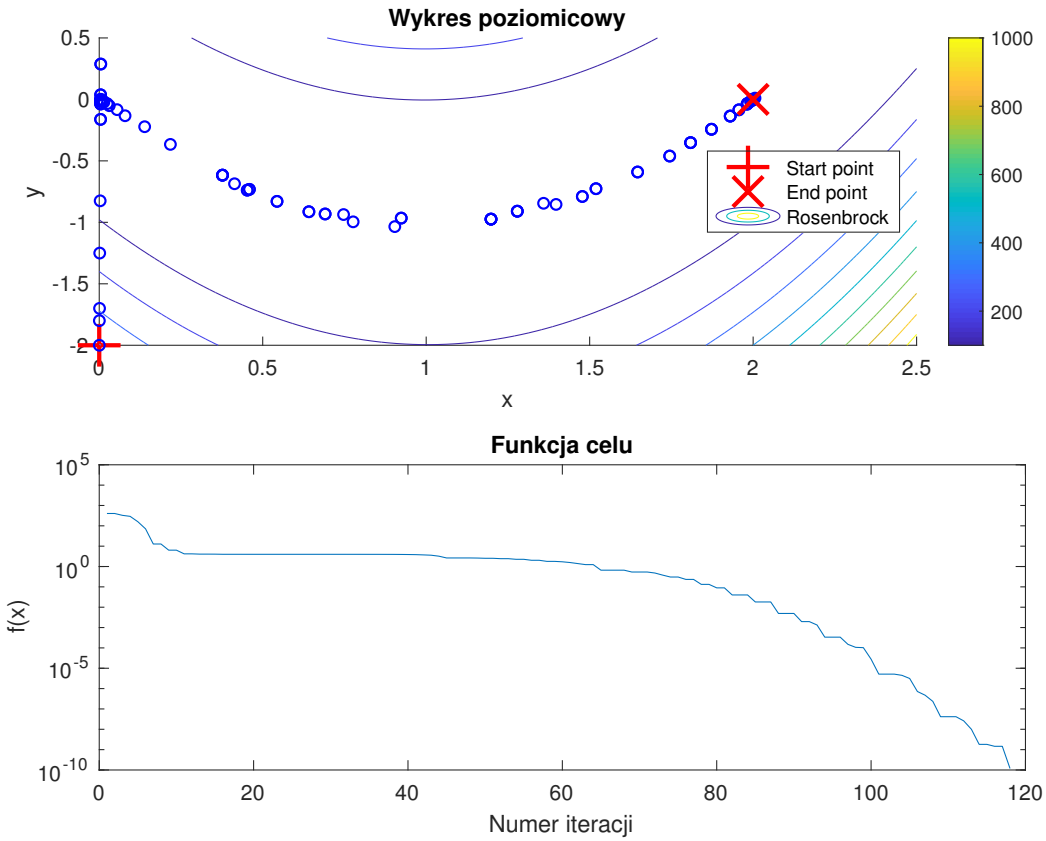
X_0	Y_0	X_{inf}	Y_{inf}	dX	dY	$f(x)$	Iterations
3	0	2	$1{,}32 \cdot 10^{-3}$	$-6{,}31 \cdot 10^{-4}$	$-1{,}32 \cdot 10^{-3}$	$7{,}73 \cdot 10^{-7}$	29
2	-2	2	$8{,}29 \cdot 10^{-6}$	$-2{,}96 \cdot 10^{-6}$	$-8{,}29 \cdot 10^{-6}$	$5{,}73 \cdot 10^{-10}$	74
0	-2	2	$-2{,}17 \cdot 10^{-5}$	$1{,}08 \cdot 10^{-5}$	$2{,}17 \cdot 10^{-5}$	$1{,}19 \cdot 10^{-10}$	117
0	0	2	$-9{,}7 \cdot 10^{-6}$	$3{,}67 \cdot 10^{-6}$	$9{,}7 \cdot 10^{-6}$	$5{,}73 \cdot 10^{-10}$	106



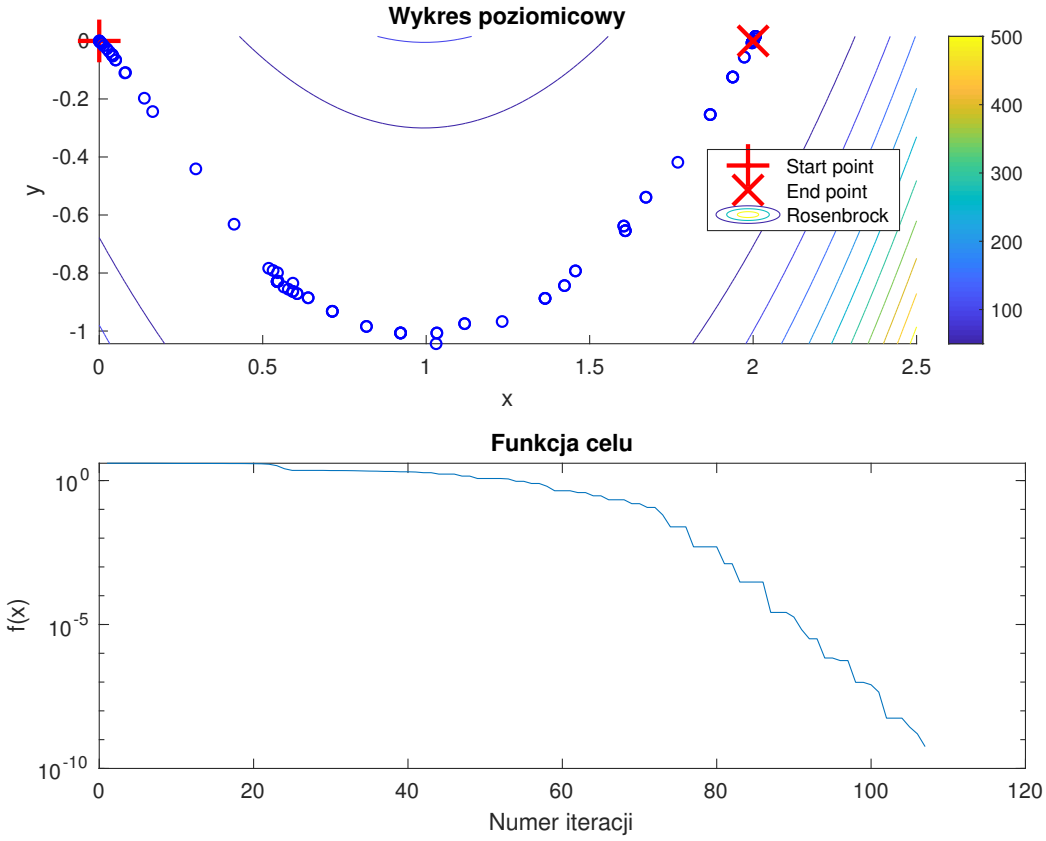
Rysunek 2.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



Rysunek 2.2. Zadanie optymalizacji dla punktu startowego $x = 2$, $y = -2$



Rysunek 2.3. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = -2$



Rysunek 2.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

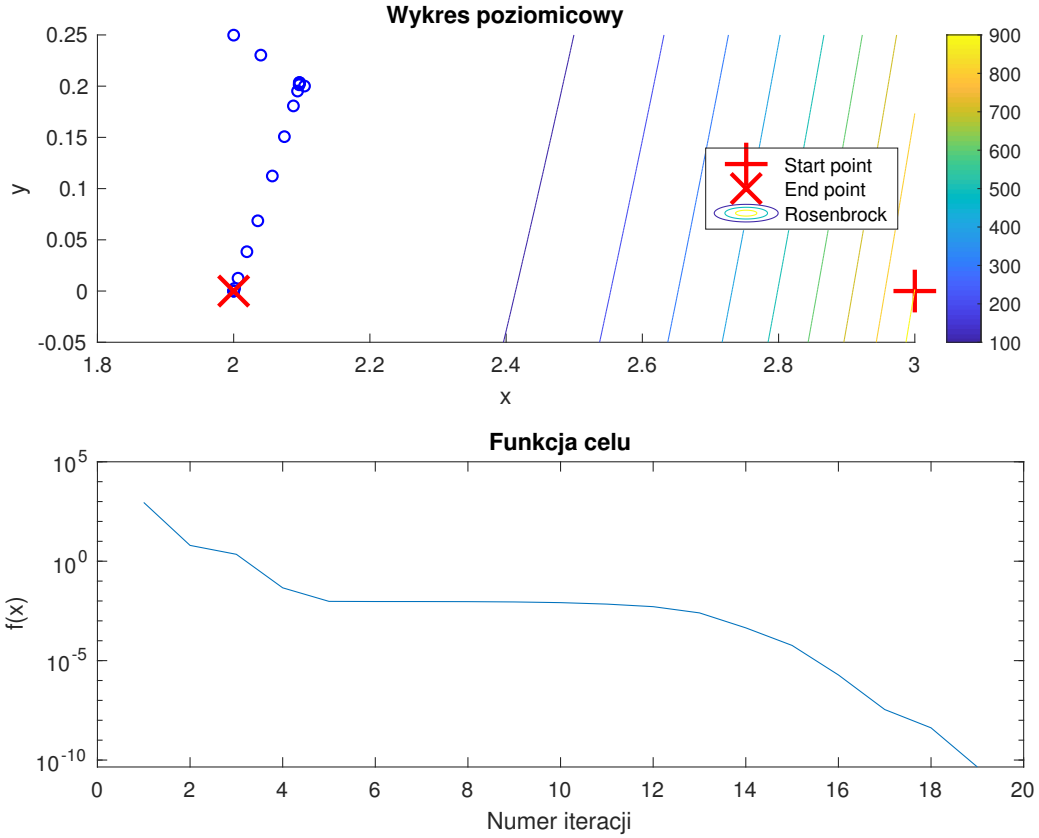
3. Optymalizacja za pomocą szacowanych pochodnych

Drugą użytą metodą jest algorytm `quasi-newton`, zawarty w funkcji `fminunc`. Jeśli chodzi o dokładność wyniku, otrzymujemy minimalnie dokładniejsze rozwiązania, niż w przypadku poprzedniego algorytmu. Wartość funkcji celu jest rzędu 10^{-11}. Widać jednak znaczną różnicę w szybkości działania algorytmów. W tym przypadku najdłuższe poszukiwanie rozwiązania trwało 29 iteracji.

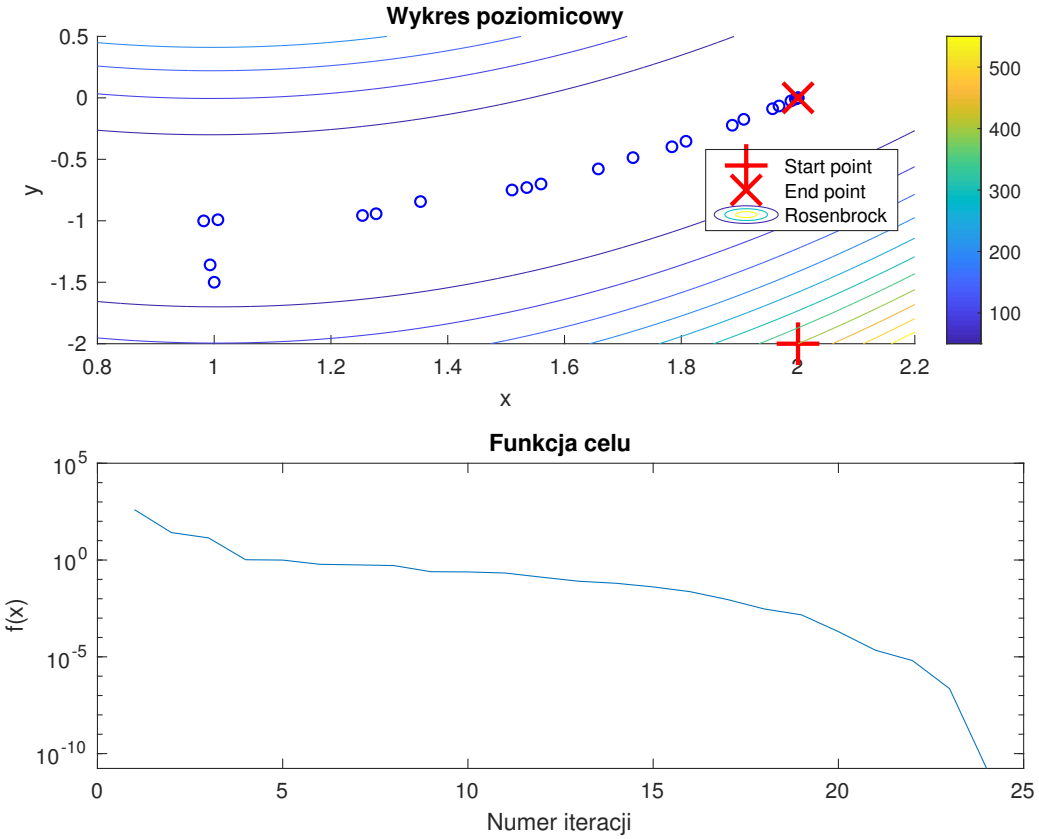
Oznaczenia:
X_0 - startowy punkt X
Y_0 - startowy punkt Y
X_{inf} - końcowy punkt X
Y_{inf} - końcowy punkt Y
dX - błąd bezwzględny punktu X
dY - błąd bezwzględny punktu Y
$f(x)$ - wartość funkcji celu
Iterations - liczba iteracji

Tablica 3.1: Porównanie punktów startowych

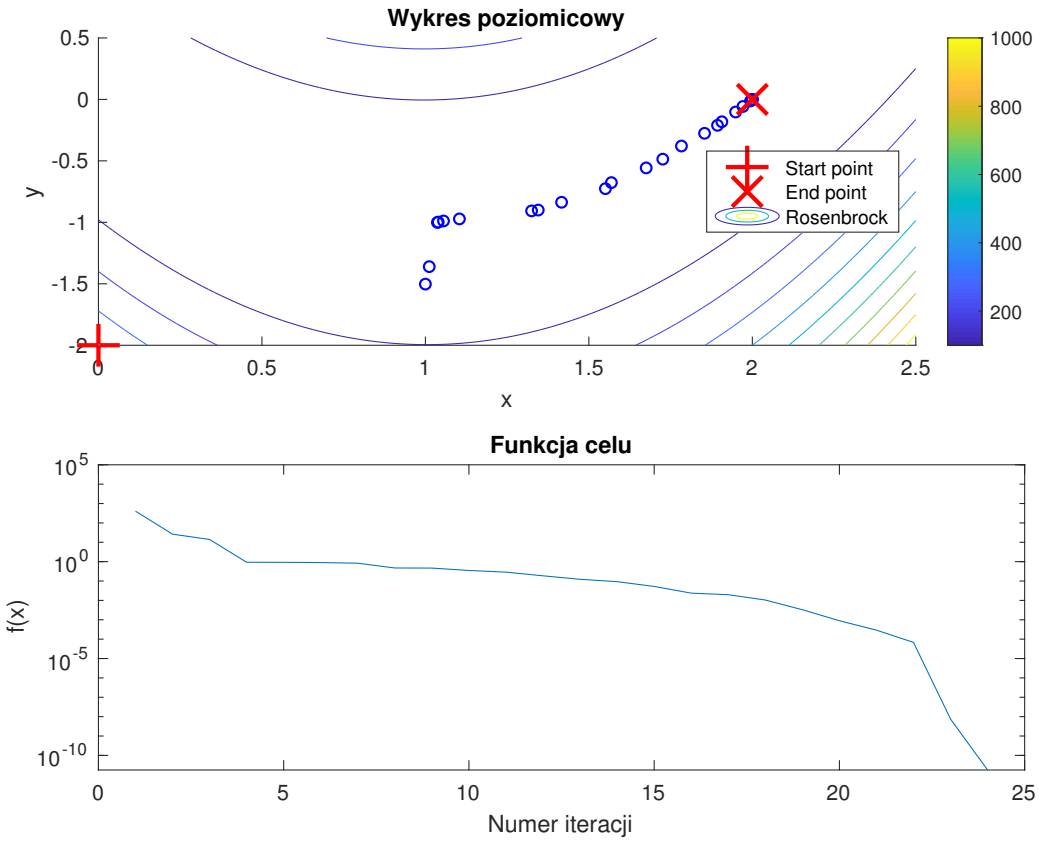
X_0	Y_0	X_{inf}	Y_{inf}	dX	dY	$f(x)$	Iterations
3	0	2	$-1{,}34 \cdot 10^{-5}$	$6{,}69 \cdot 10^{-6}$	$1{,}34 \cdot 10^{-5}$	$4{,}48 \cdot 10^{-11}$	18
2	-2	2	$8{,}24 \cdot 10^{-6}$	$-4{,}09 \cdot 10^{-6}$	$-8{,}24 \cdot 10^{-6}$	$1{,}72 \cdot 10^{-11}$	23
0	-2	2	$-6{,}07 \cdot 10^{-6}$	$3{,}17 \cdot 10^{-6}$	$6{,}07 \cdot 10^{-6}$	$1{,}77 \cdot 10^{-11}$	23
0	0	2	$-8{,}65 \cdot 10^{-6}$	$4{,}33 \cdot 10^{-6}$	$8{,}65 \cdot 10^{-6}$	$1{,}87 \cdot 10^{-11}$	29



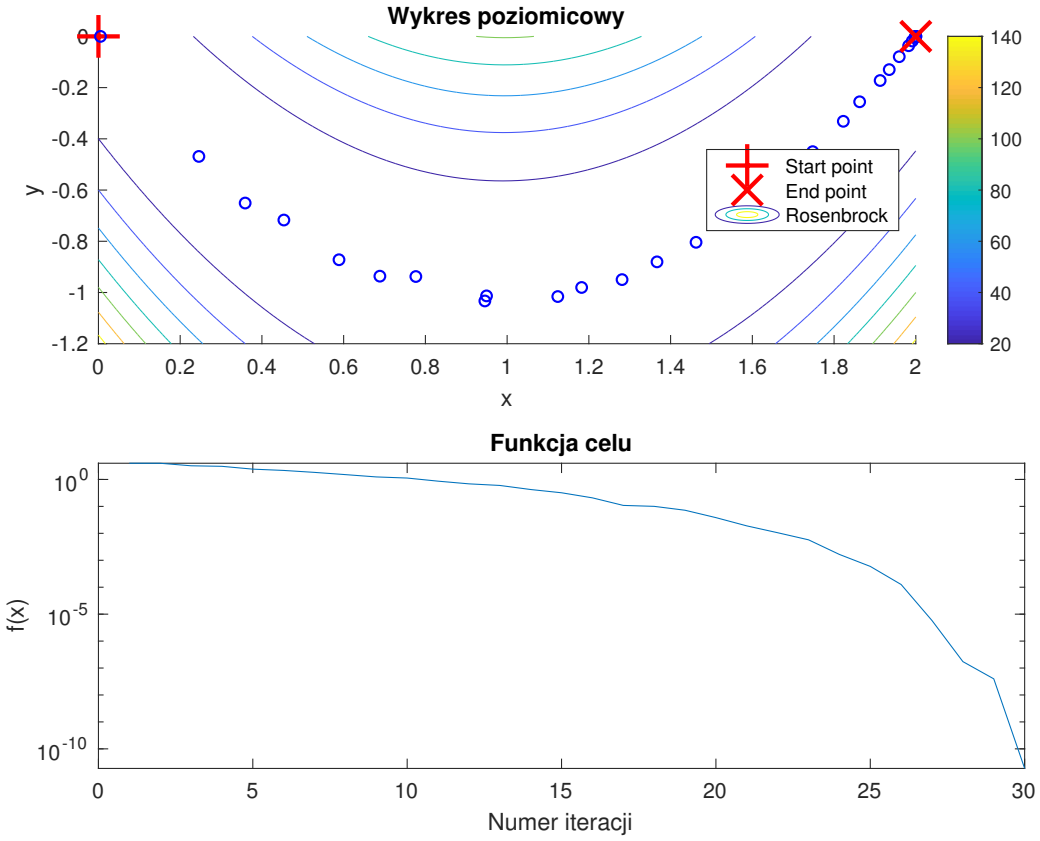
Rysunek 3.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



Rysunek 3.2. Zadanie optymalizacji dla punktu startowego $x = 2$, $y = -2$



Rysunek 3.3. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = -2$



Rysunek 3.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

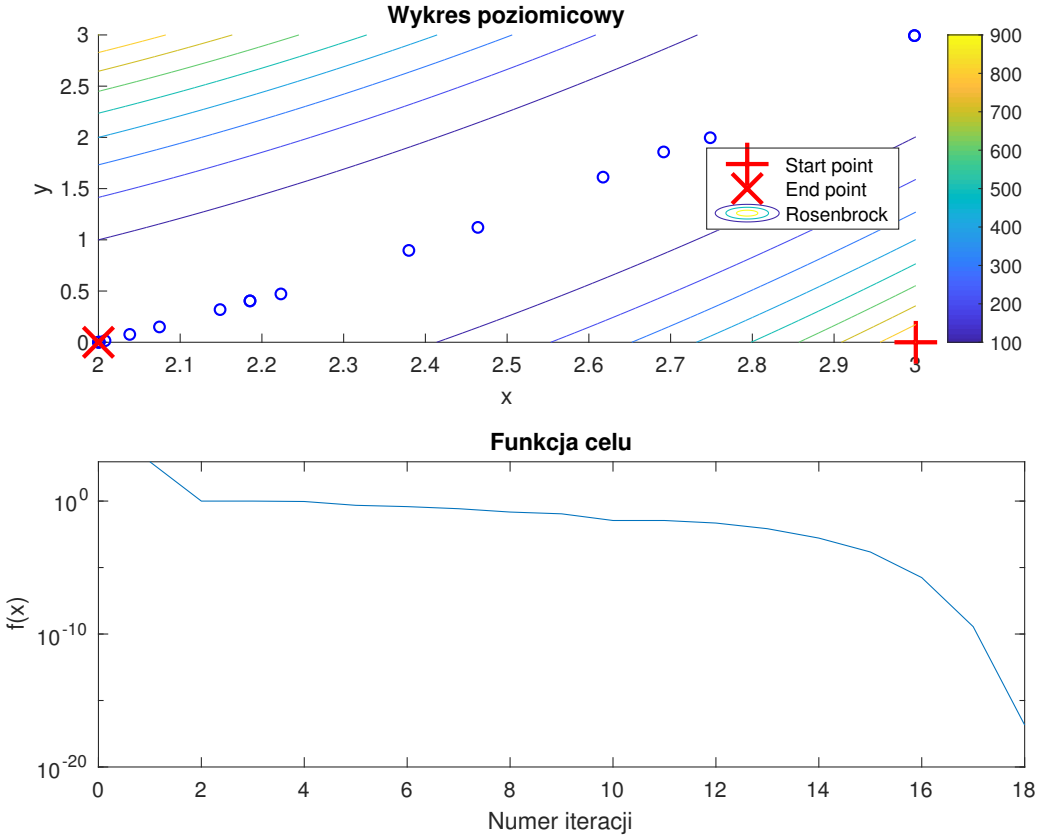
4. Optymalizacja z użyciem gradientu

Funkcja `fminunc` udostępnia również możliwość zmiany podstawowego algorytmu, opierającego się na szacowaniu pochodnych, na algorytm `trust-region` wykorzystujący gradient funkcji. Algorytm w pewnych przypadkach jest bardzo szybki. Dla jednego punktu startowego, algorytm otrzymał poprawny wynik w jednej iteracji. W pozostałych punktach startowych liczba iteracji jest podobna jak w przypadku poprzedniego algorytmu `quasi_newton`, jednak otrzymujemy wyniki bardziej zbliżone do oczekiwanych.

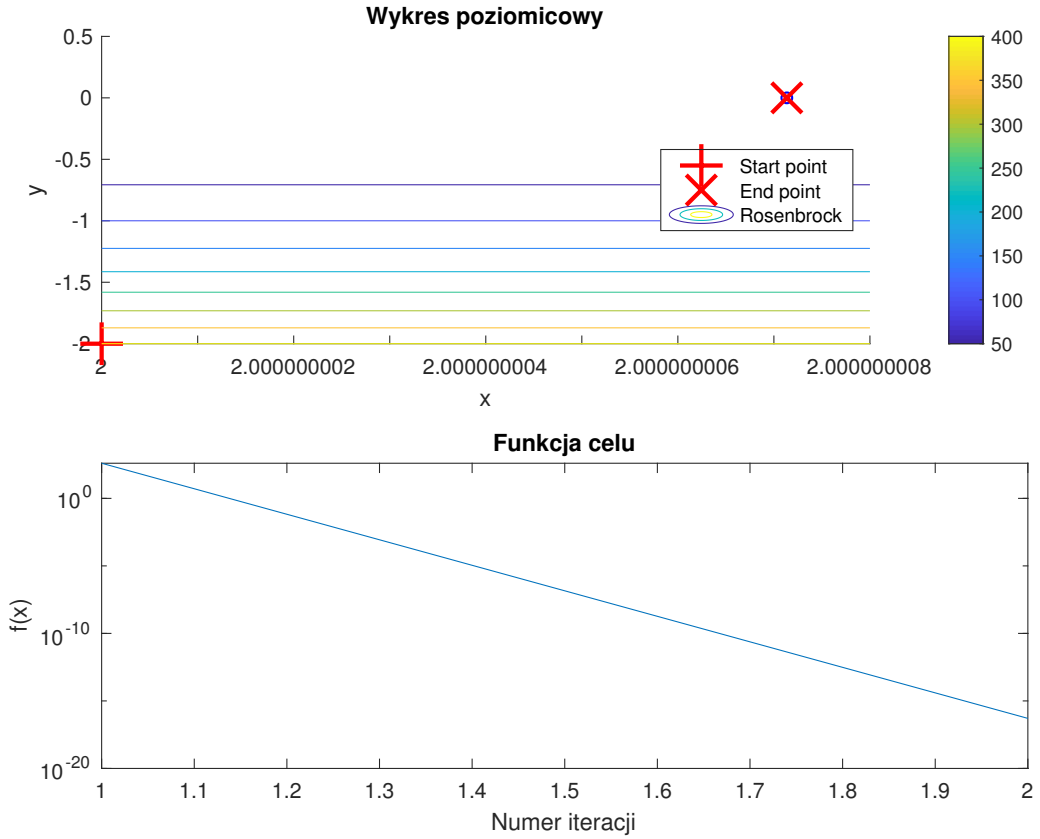
Oznaczenia:
X_0 - startowy punkt X
Y_0 - startowy punkt Y
X_{inf} - końcowy punkt X
Y_{inf} - końcowy punkt Y
dX - błąd bezwzględny punktu X
dY - błąd bezwzględny punktu Y
$f(x)$ - wartość funkcji celu
Iterations - liczba iteracji

Tablica 4.1: Porównanie punktów startowych

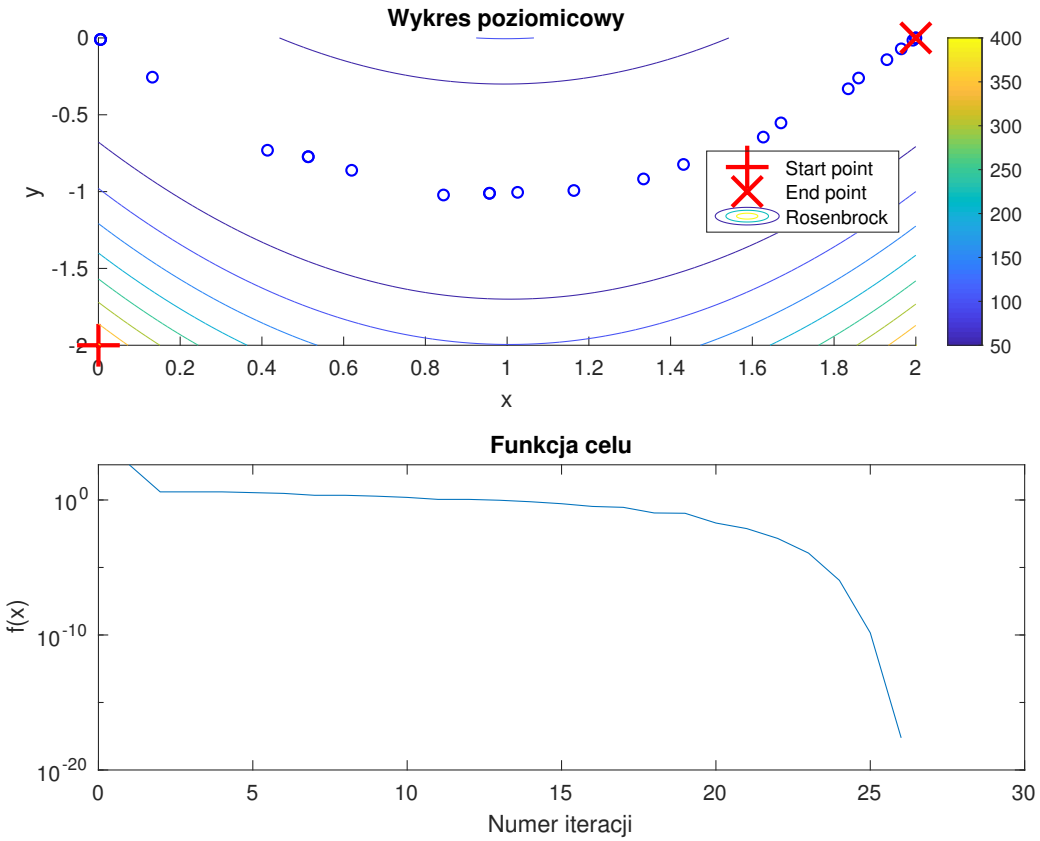
X_0	Y_0	X_{inf}	Y_{inf}	dX	dY	$f(x)$	Iterations
3	0	2	$7{,}01 \cdot 10^{-9}$	$-3{,}57 \cdot 10^{-9}$	$-7{,}01 \cdot 10^{-9}$	$1{,}44 \cdot 10^{-17}$	17
2	-2	2	$1{,}43 \cdot 10^{-8}$	$-7{,}13 \cdot 10^{-9}$	$-1{,}43 \cdot 10^{-8}$	$5{,}09 \cdot 10^{-17}$	1
0	-2	2	$-2{,}98 \cdot 10^{-9}$	$1{,}47 \cdot 10^{-9}$	$2{,}98 \cdot 10^{-9}$	$2{,}42 \cdot 10^{-18}$	25
0	0	2	$-1{,}99 \cdot 10^{-7}$	$9{,}43 \cdot 10^{-8}$	$1{,}99 \cdot 10^{-7}$	$2{,}06 \cdot 10^{-14}$	26



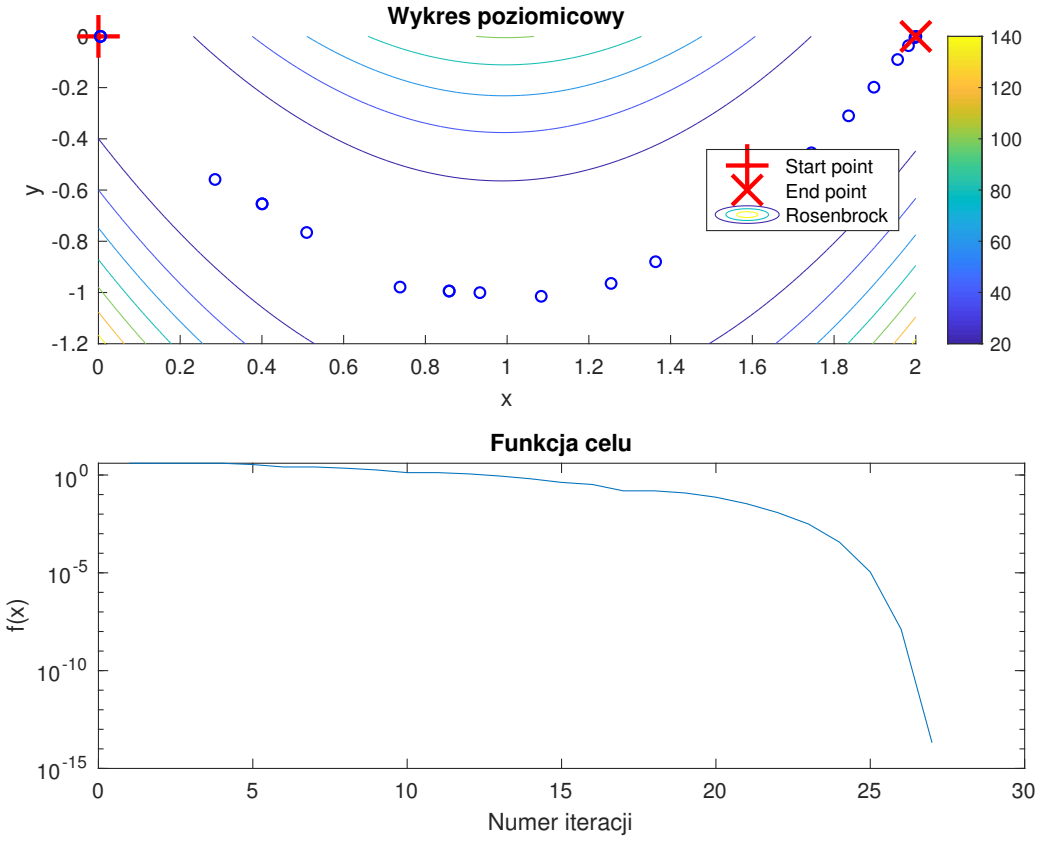
Rysunek 4.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



Rysunek 4.2. Zadanie optymalizacji dla punktu startowego $x = 2$, $y = -2$



Rysunek 4.3. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = -2$



Rysunek 4.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

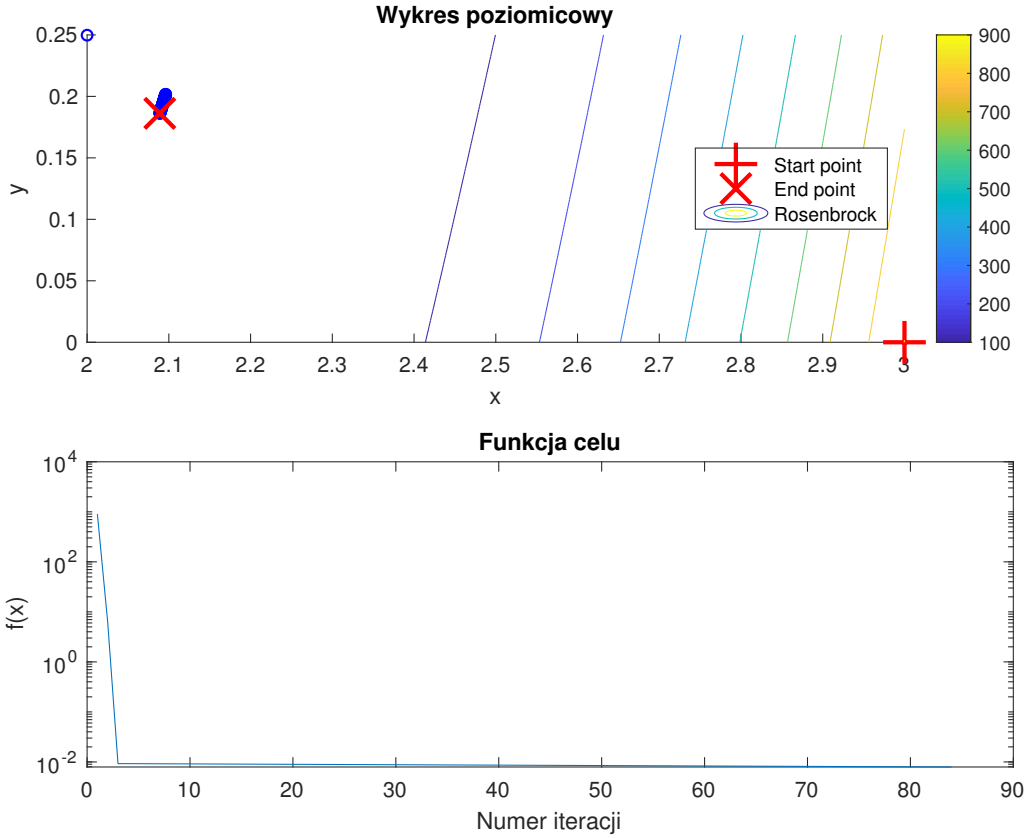
5. Metoda najszybszego spadku

Ostatni rozpatrzony algorytm również korzysta z funkcji `fminunc` oraz algorytmu `quasi_newton`, lecz tym razem ustawiamy opcję `HessUpdate` na `steepdesc`, dzięki czemu otrzymujemy algorytm opierający się na metodzie najszybszego spadku. Maksymalną liczbę obliczeń funkcji celu ustawiłem na 1000, lecz nawet wtedy algorytm nie był w stanie podać zadowalającego wyniku. Dokładność algorytmu pozostawia wiele do życzenia, nawet przy większej liczbie iteracji, algorytm bardzo wolno zbliża się do oczekiwanego wyniku. Początkowo można zaobserwować bardzo szybki spadek wartości funkcji celu, ale wraz ze wzrostem iteracji, zmiany te są coraz mniejsze.

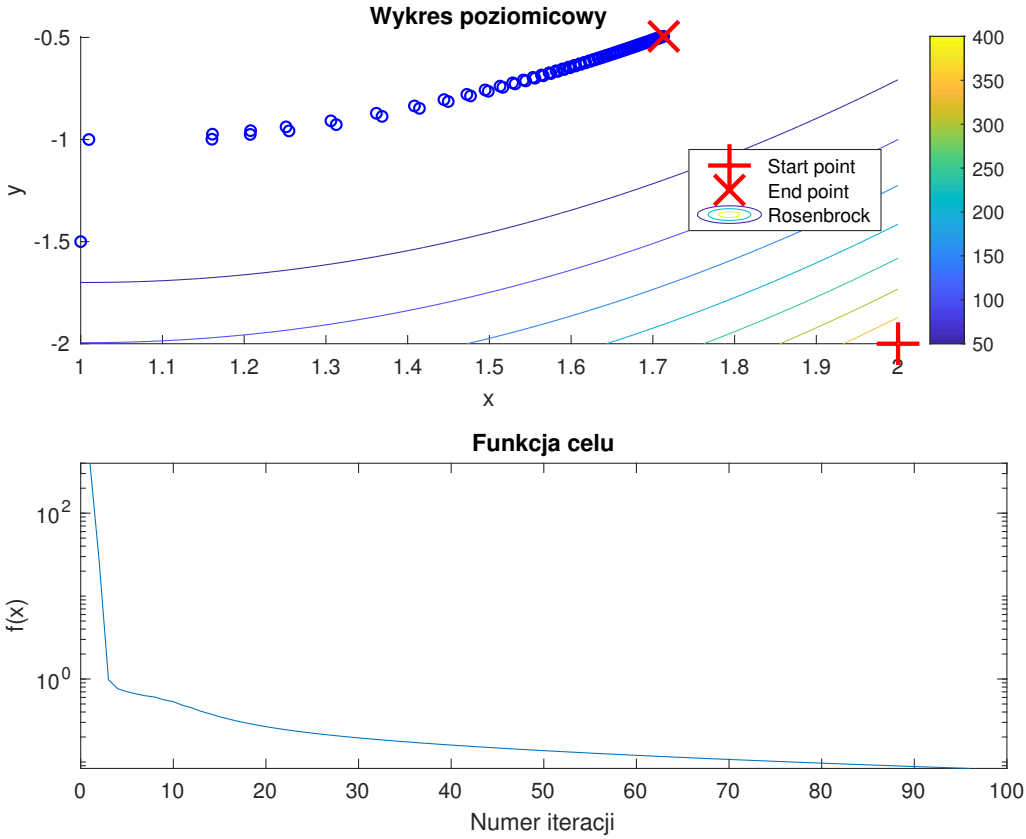
Oznaczenia:
X_0 - startowy punkt X
Y_0 - startowy punkt Y
X_{inf} - końcowy punkt X
Y_{inf} - końcowy punkt Y
dX - błąd bezwzględny punktu X
dY - błąd bezwzględny punktu Y
$f(x)$ - wartość funkcji celu
Iterations - liczba iteracji

Tablica 5.1: Porównanie punktów startowych

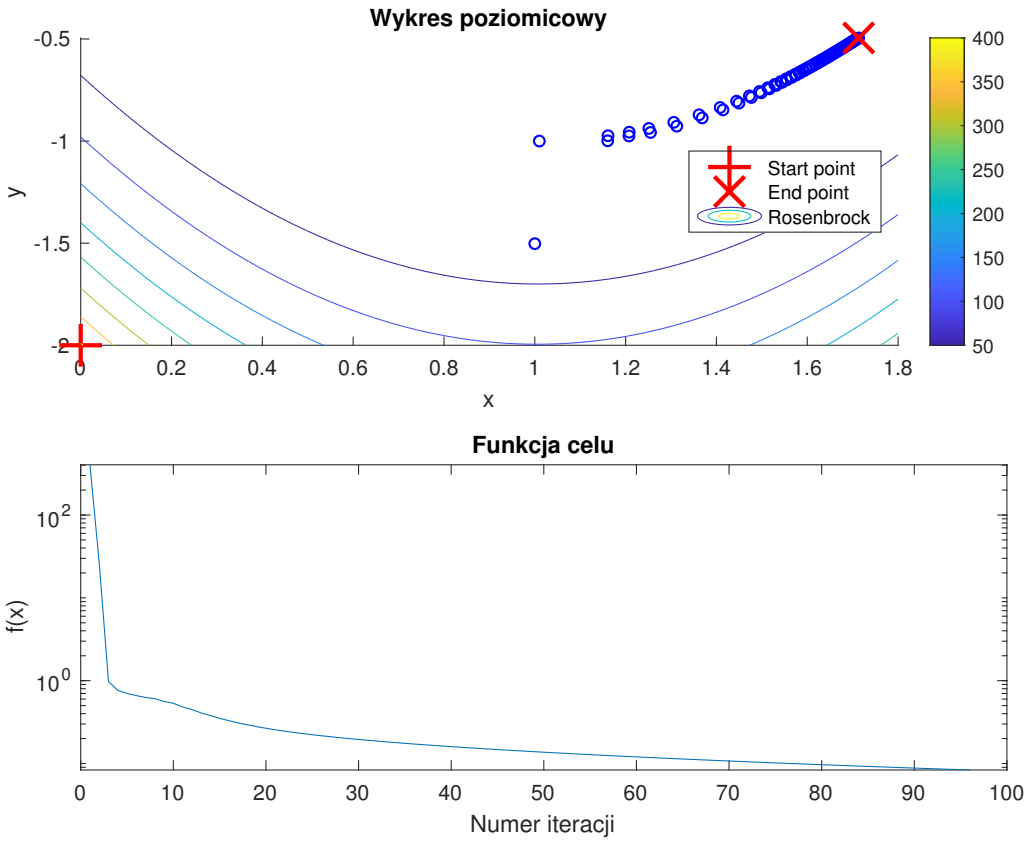
X_0	Y_0	X_{inf}	Y_{inf}	dX	dY	$f(x)$	Iterations
3	0	2,09	0,19	$-8{,}89 \cdot 10^{-2}$	$-0{,}19$	$7{,}93 \cdot 10^{-3}$	84
2	-2	1,71	$-0{,}49$	0,29	0,49	$8{,}35 \cdot 10^{-2}$	96
0	-2	1,71	$-0{,}49$	0,29	0,49	$8{,}35 \cdot 10^{-2}$	96
0	0	2	$9{,}01 \cdot 10^{-3}$	$-4{,}48 \cdot 10^{-3}$	$-9{,}01 \cdot 10^{-3}$	$2{,}02 \cdot 10^{-5}$	83



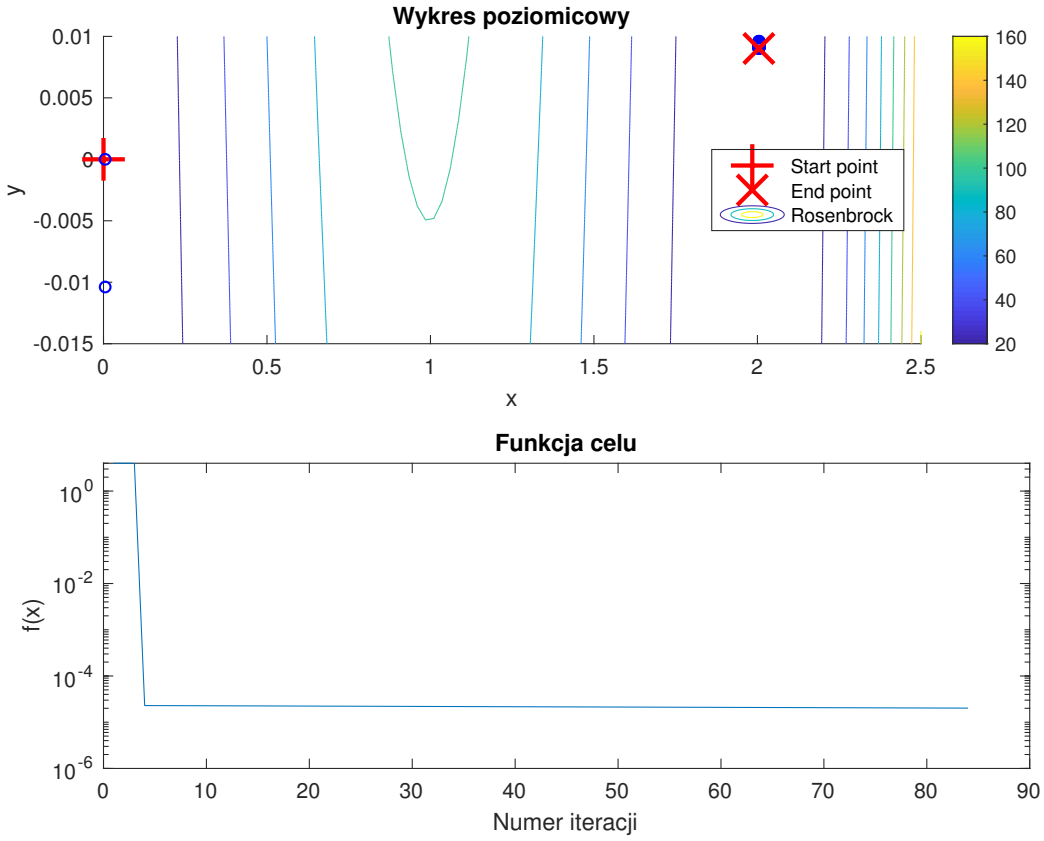
Rysunek 5.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



Rysunek 5.2. Zadanie optymalizacji dla punktu startowego $x = 2$, $y = -2$



Rysunek 5.3. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = -2$



Rysunek 5.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

6. Wnioski

Najlepszą metodą zdecydowanie okazał się algorytm `trust_region` wykorzystujący gradient funkcji celu. Był najlepszy zarówno pod względem dokładności wyniku, jak również okazał się najszybszy. W pewnych przypadkach szybkością dorównywał mu algorytm `quasi_newton`, lecz odstawał dokładnością o kilka rzędów wielkości. Algorytm `fminsearch` dawał porównywalne wyniki jeśli chodzi o dokładność, lecz wykonywał przy tym o wiele więcej iteracji. Najsłabszy okazał się algorytm najszybszego spadku, który dawał niezadowalające rezultaty zarówno jeśli o dokładność obliczeń jak i szybkość.