Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Algorytmy ewolucyjne

Sprawozdanie z projektu nr 1

Karol Borowski

Warszawa, 2019

Spis treści

1. Wstęp 2
2. Optymalizacja bez pochodnych 3
3. Optymalizacja za pomocą szacowanych pochodnych 6
4. Optymalizacja z użyciem gradientu 9
5. Metoda najmniejszych kwadratów 12
6. Wnioski 15

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

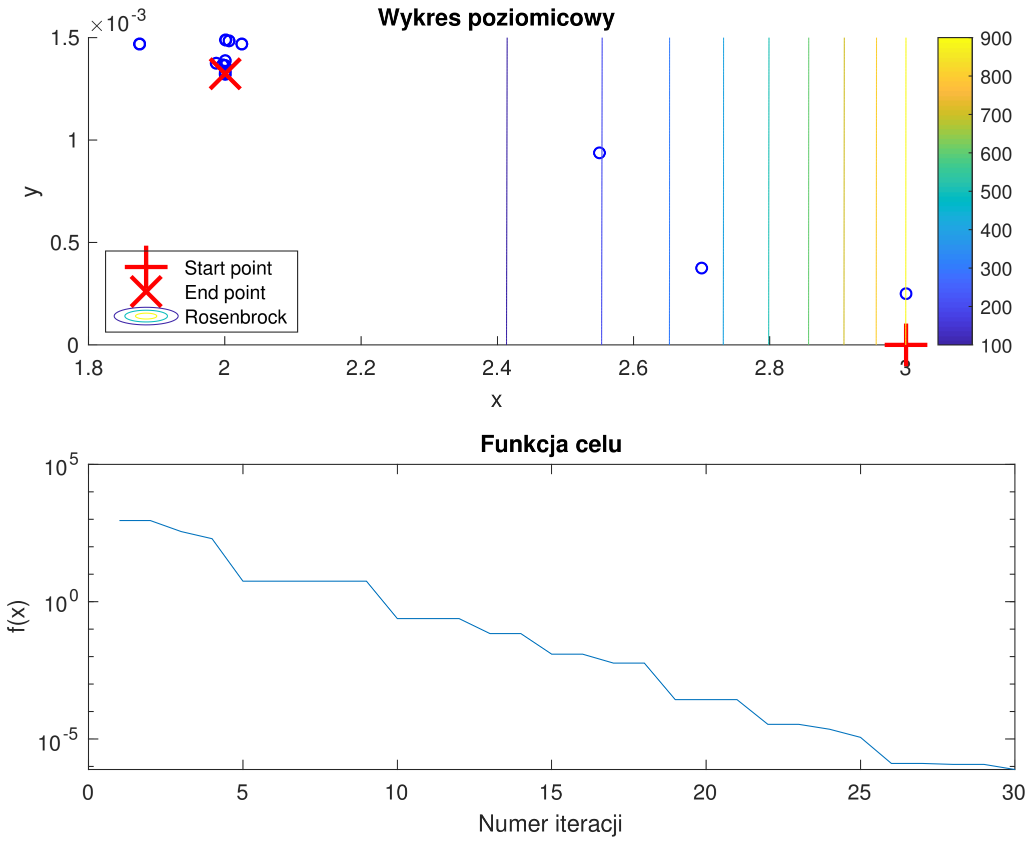
W celu optymalizacji funkcji użyłem czterech dostępnych w MATLAB'ie algorytmów: `fminsearch`, `fminunc(quasi-newton)`, `fminunc(trust-region)`, `lsqnonlin` oraz porównałem ich działanie.

2. Optymalizacja bez pochodnych

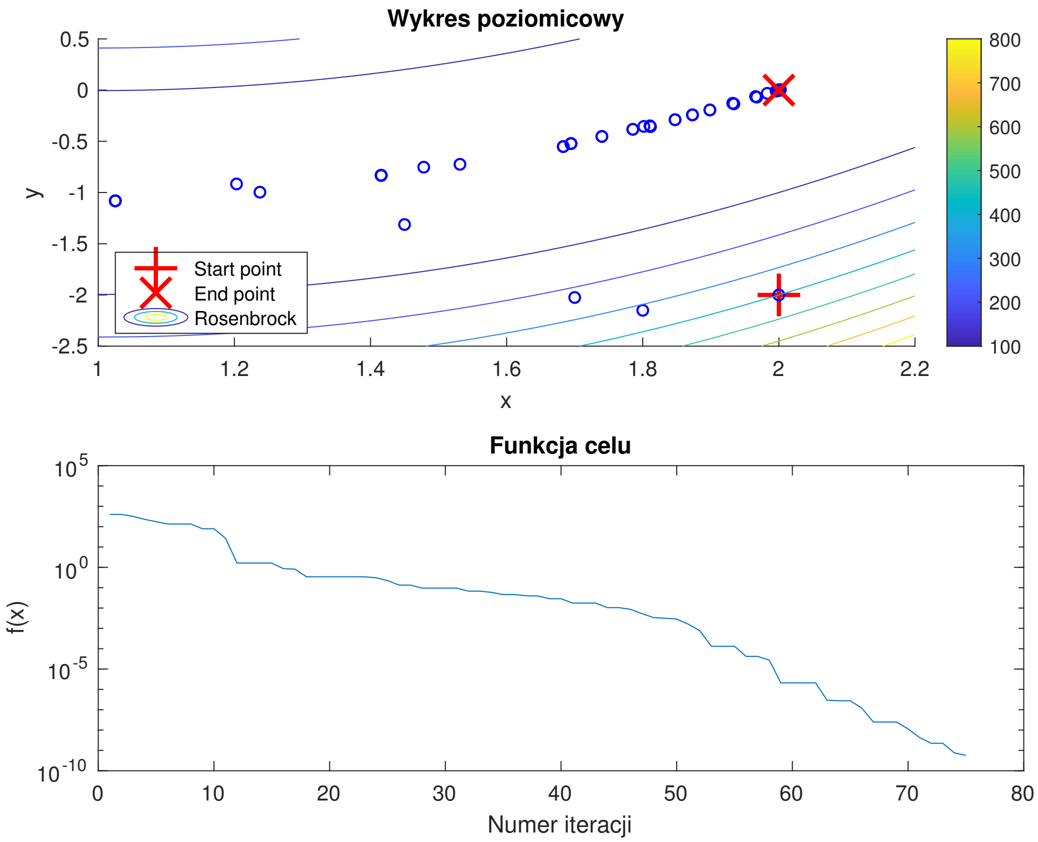
Pierwszą wykorzystaną metodą jest algorytm `fminsearch`, czyli optymalizacja bez użycia pochodnych. Metoda daje wyniki rzędu 10^{-7}, co jest jest wynikiem bliskim 0, aczkolwiek jest możliwe osiągnięcie lepszego rezultatu. Główną wadą algorytmu jest stosunkowo duża ilość iteracji, potrzebnych do wykonania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

Tablica 2.1: Porównanie punktów startowych

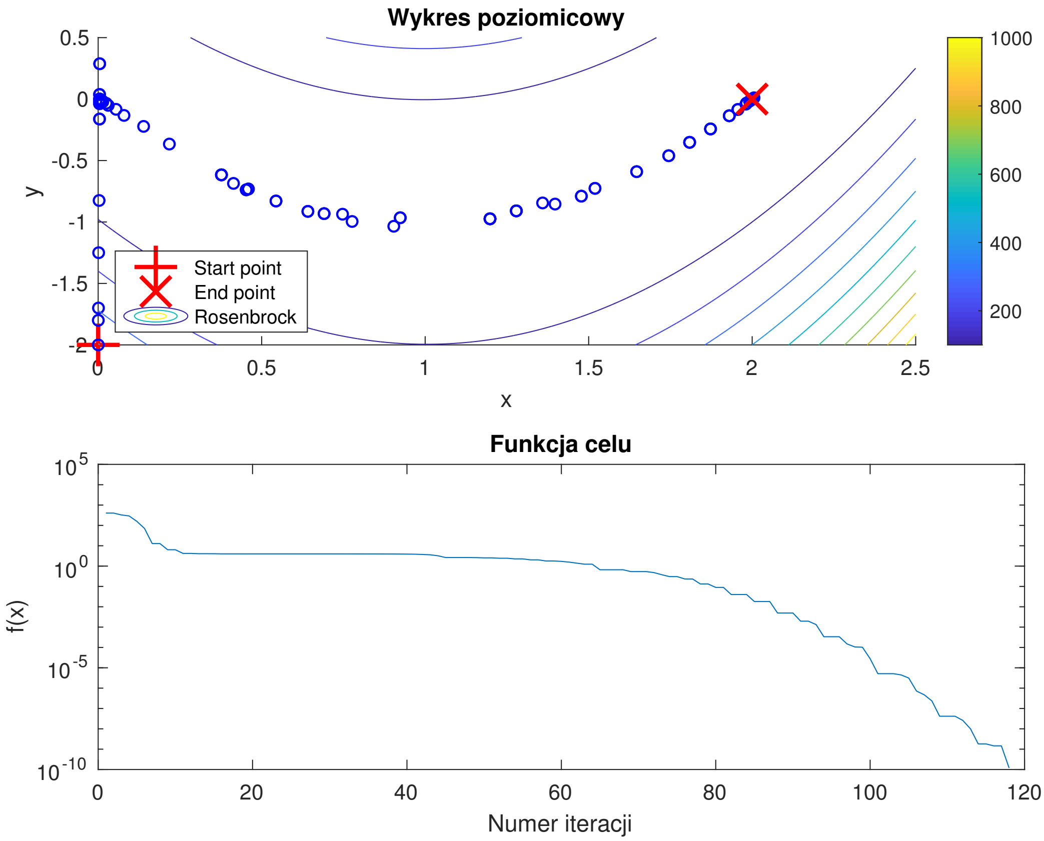
X_0	Y_0	X_{inf}	Y_{inf}	$f(x)$	Iterations
3	0	2	$1{,}32 \cdot 10^{-3}$	$7{,}73 \cdot 10^{-7}$	29
2	-2	2	$8{,}29 \cdot 10^{-6}$	$5{,}73 \cdot 10^{-10}$	74
0	-2	2	$-2{,}17 \cdot 10^{-5}$	$1{,}19 \cdot 10^{-10}$	117
0	0	2	$-9{,}7 \cdot 10^{-6}$	$5{,}73 \cdot 10^{-10}$	106



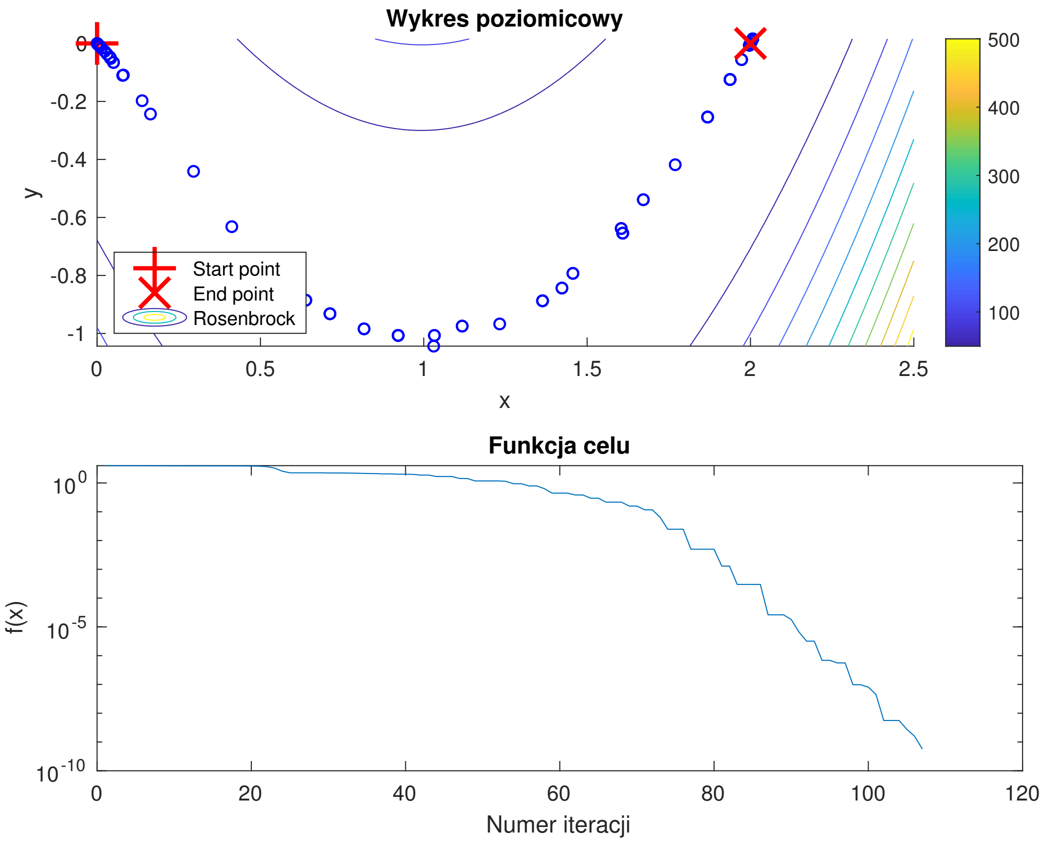
Rysunek 2.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



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



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



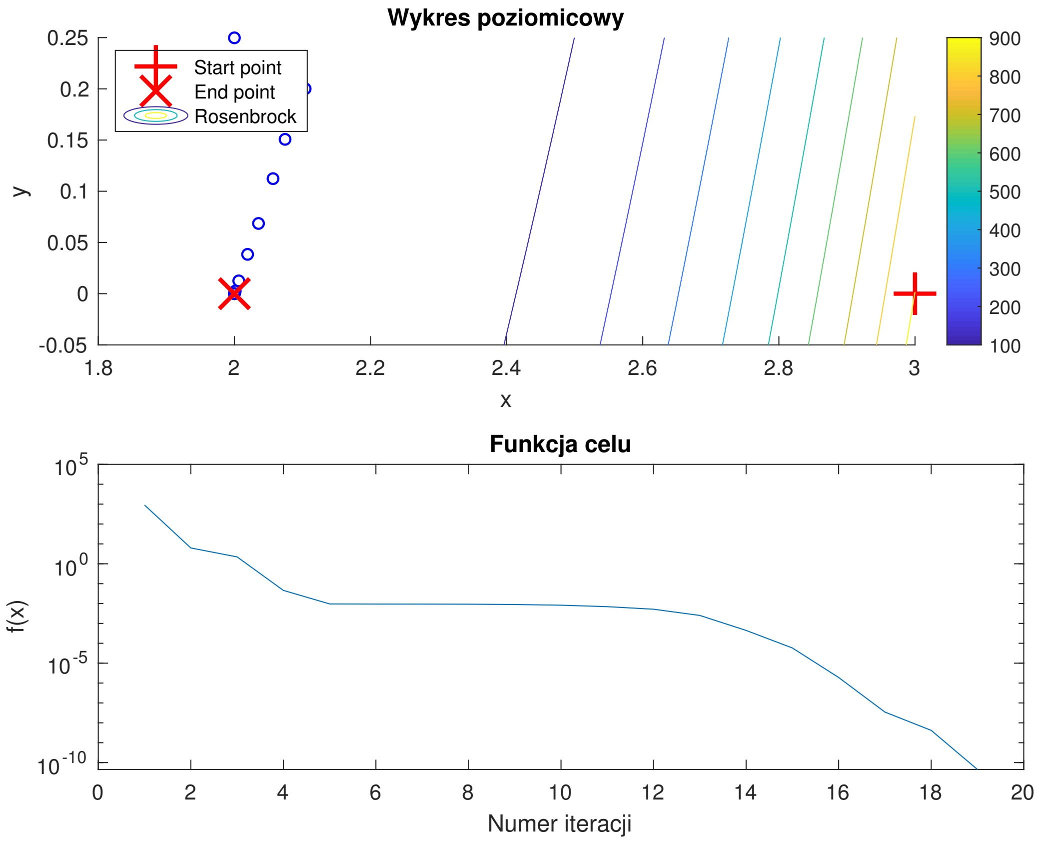
Rysunek 2.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

3. Optymalizacja za pomocą szacowanych pochodnych

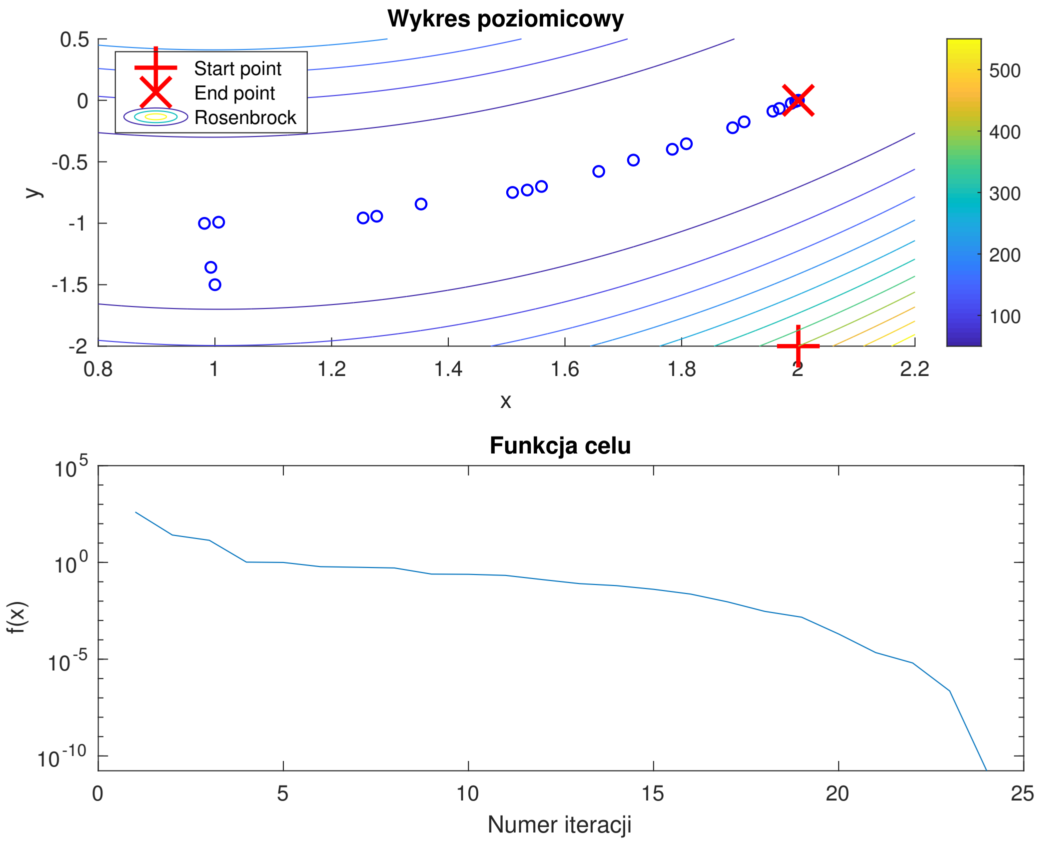
Drugą użytą metodą jest algorytm `quasi-newton`, zawarty w funkcji `fminunc`. Jeśli chodzi o dokładność wyniku, otrzymujemy minimalnie dokładniejsze rozwiązania, niż w przypadku poprzedniego algorytmu. Wartość funkcji celu jest rzędu 10^{-11}. Widać jednak znaczną różnicę w szybkości działania algorytmów.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

Tablica 3.1: Porównanie punktów startowych

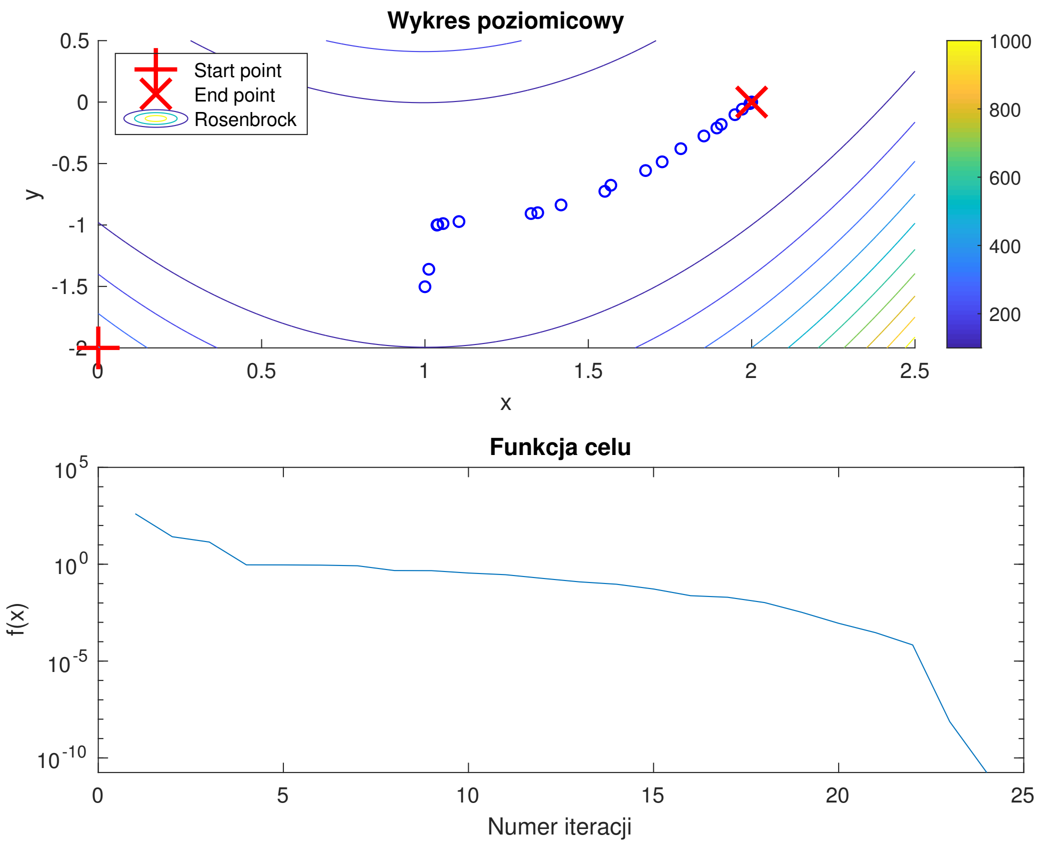
X_0	Y_0	X_{inf}	Y_{inf}	$f(x)$	Iterations
3	0	2	$-1{,}34 \cdot 10^{-5}$	$4{,}48 \cdot 10^{-11}$	18
2	-2	2	$8{,}24 \cdot 10^{-6}$	$1{,}72 \cdot 10^{-11}$	23
0	-2	2	$-6{,}07 \cdot 10^{-6}$	$1{,}77 \cdot 10^{-11}$	23
0	0	2	$-8{,}65 \cdot 10^{-6}$	$1{,}87 \cdot 10^{-11}$	29



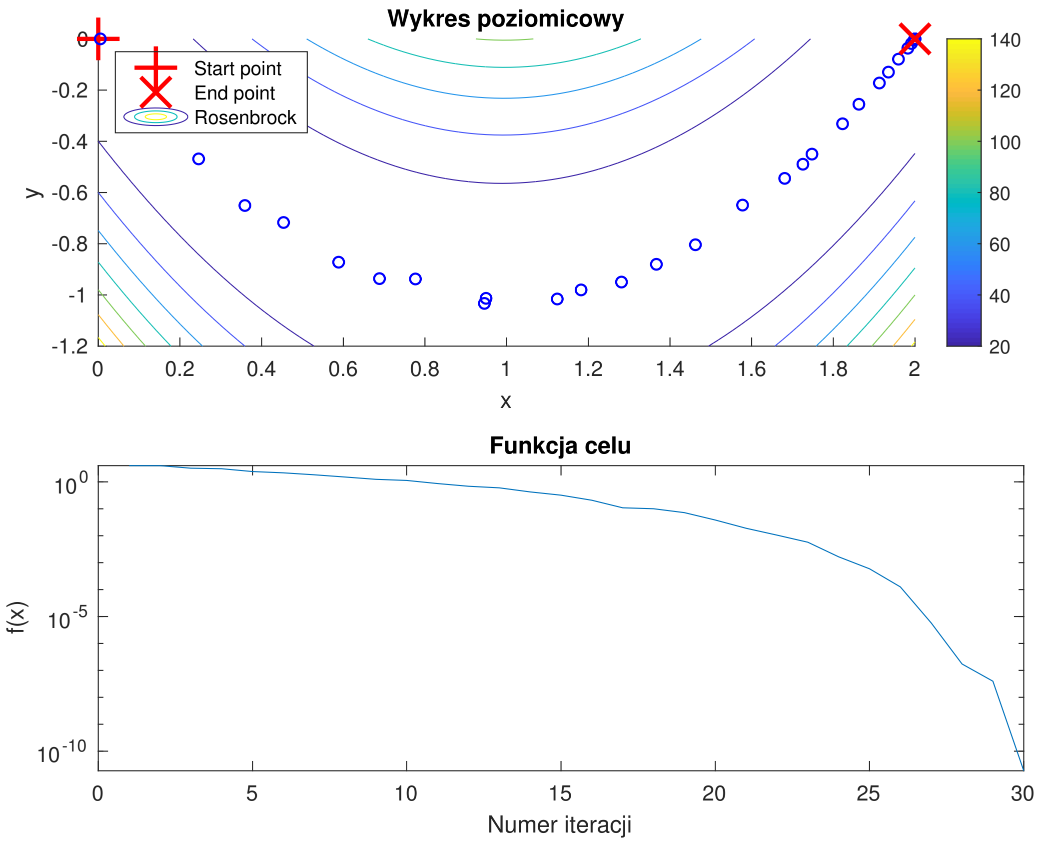
Rysunek 3.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



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



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



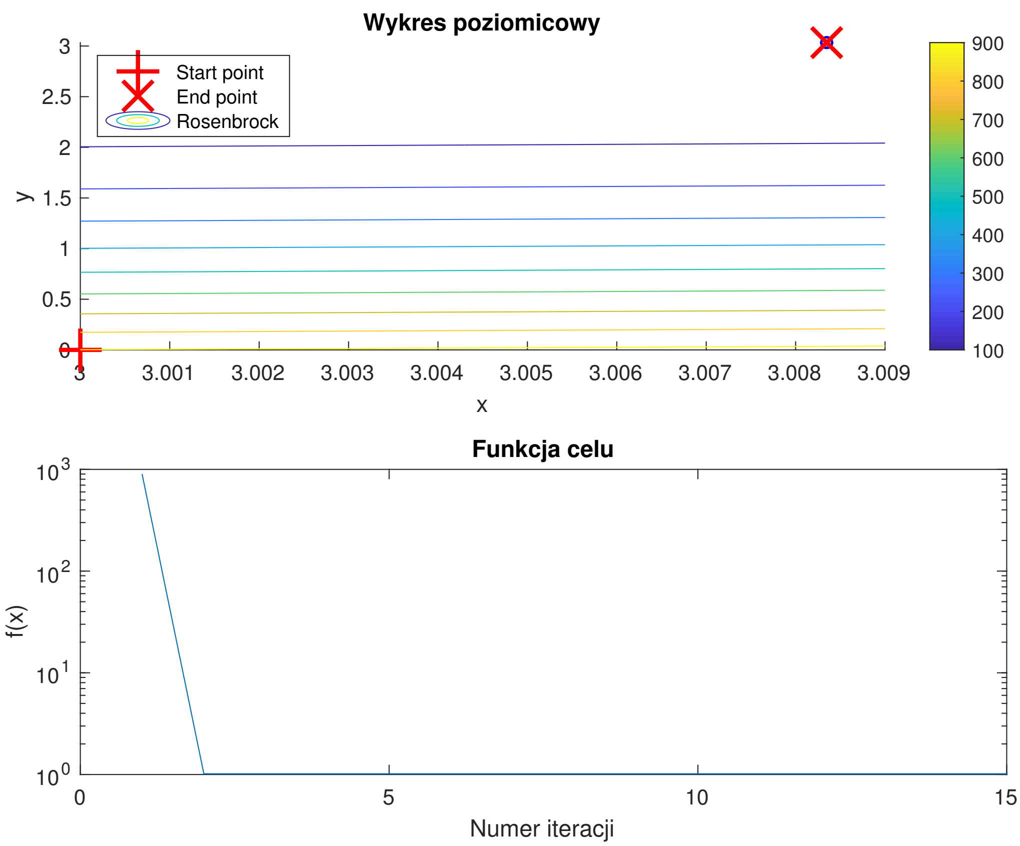
Rysunek 3.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

4. Optymalizacja z użyciem gradientu

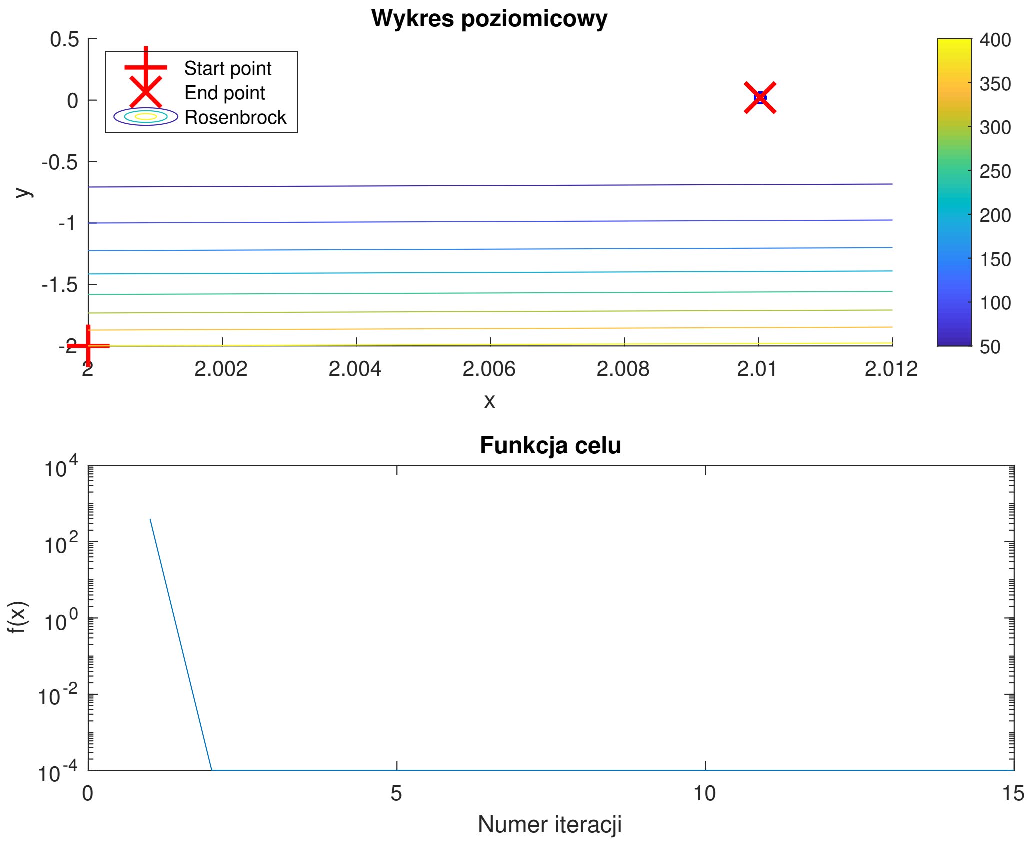
Funkcja `fminunc` udostępnia również możliwość zmiany podstawowego algorytmu, opierającego się na szacowaniu pochodnych, na algorytm `trust-region` wykorzystujący gradient funkcji. Algorytm w pewnych przypadkach jest bardzo szybki. Dla dwóch pierwszych punktów startowych, znalezienie rozwiązania zajęło jedynie 2 iteracje. Znaczą wadą jest jednak bardzo mała dokładność rozwiązań. Otrzymane rezultaty dosyć mocno odbiegają od tych oczekiwanych.

Tablica 4.1: Porównanie punktów startowych

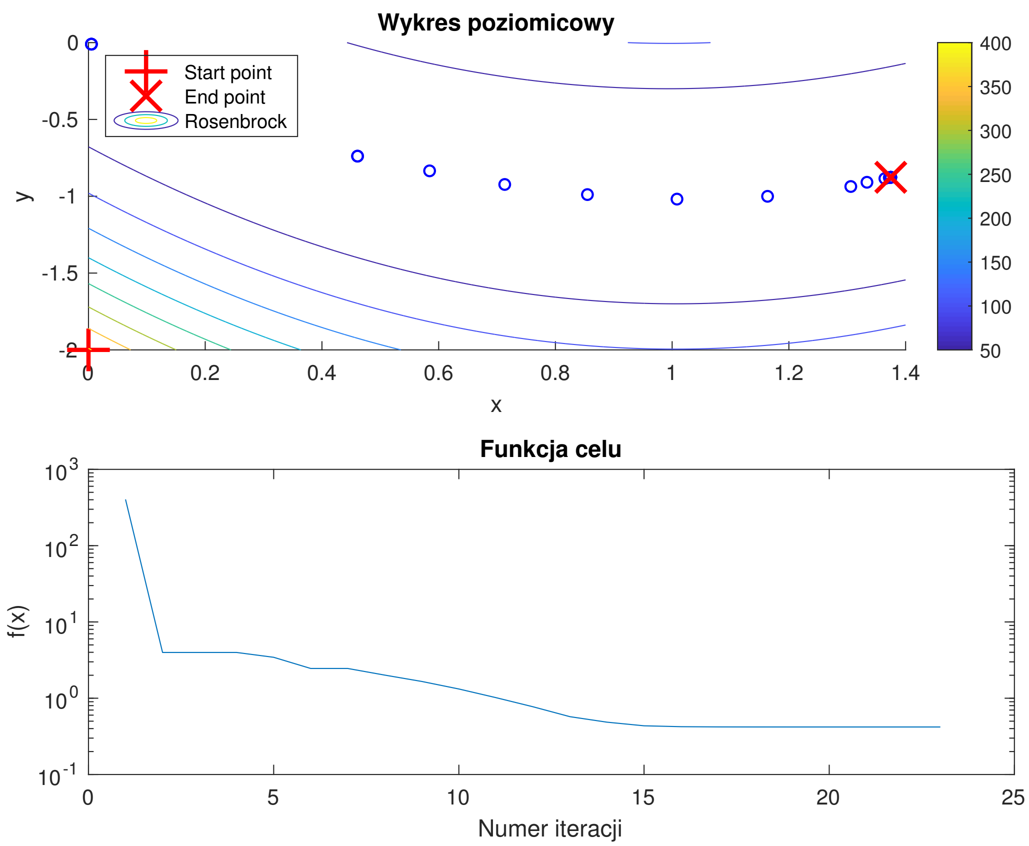
X_0	Y_0	X_{inf}	Y_{inf}	$f(x)$	Iterations
3	0	3,01	3,03	1,02	2
2	-2	2,01	$2{,}01 \cdot 10^{-2}$	$1{,}02 \cdot 10^{-4}$	2
0	-2	1,37	$-0{,}88$	0,42	19
0	0	1,37	$-0{,}88$	0,43	19



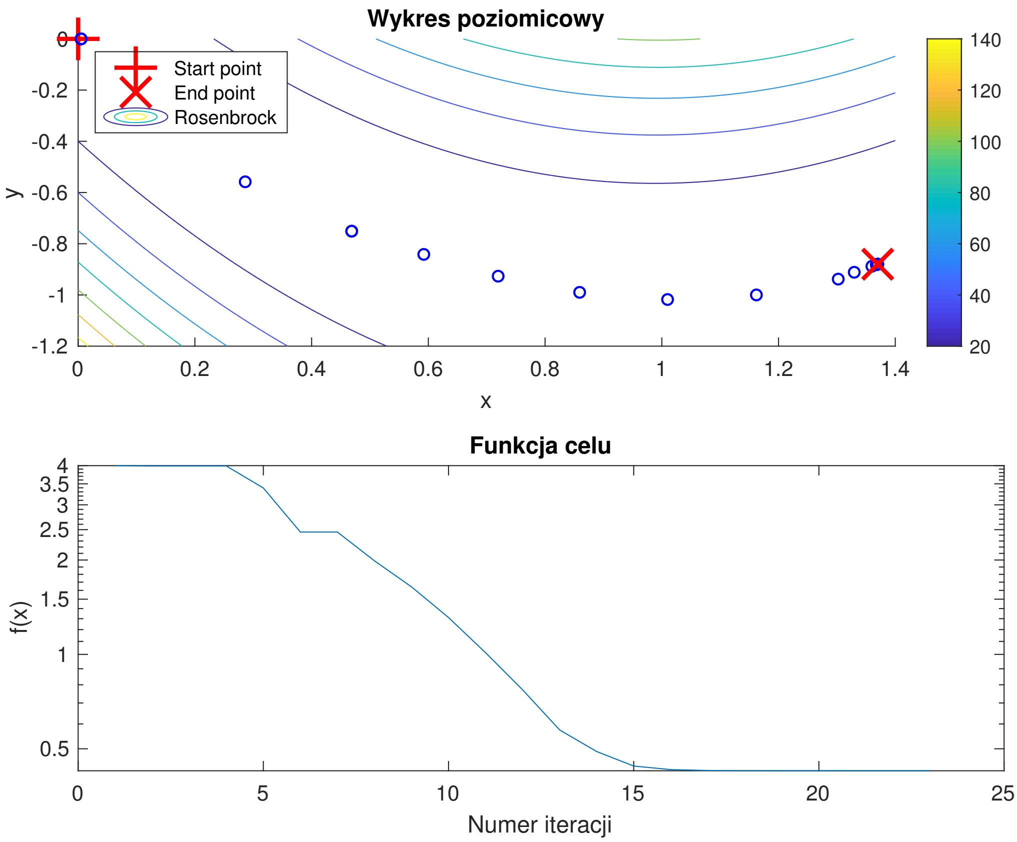
Rysunek 4.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



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



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



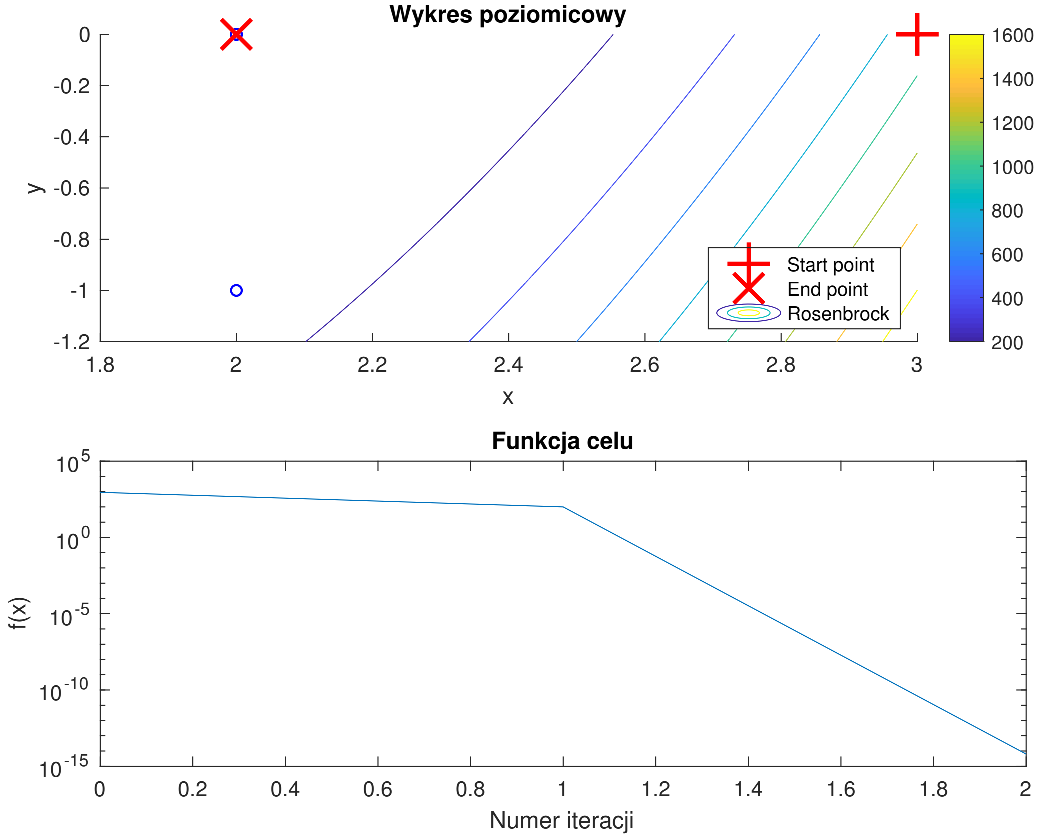
Rysunek 4.4. Zadanie optymalizacji dla punktu startowego $x = 0$, $y = 0$

5. Metoda najmniejszych kwadratów

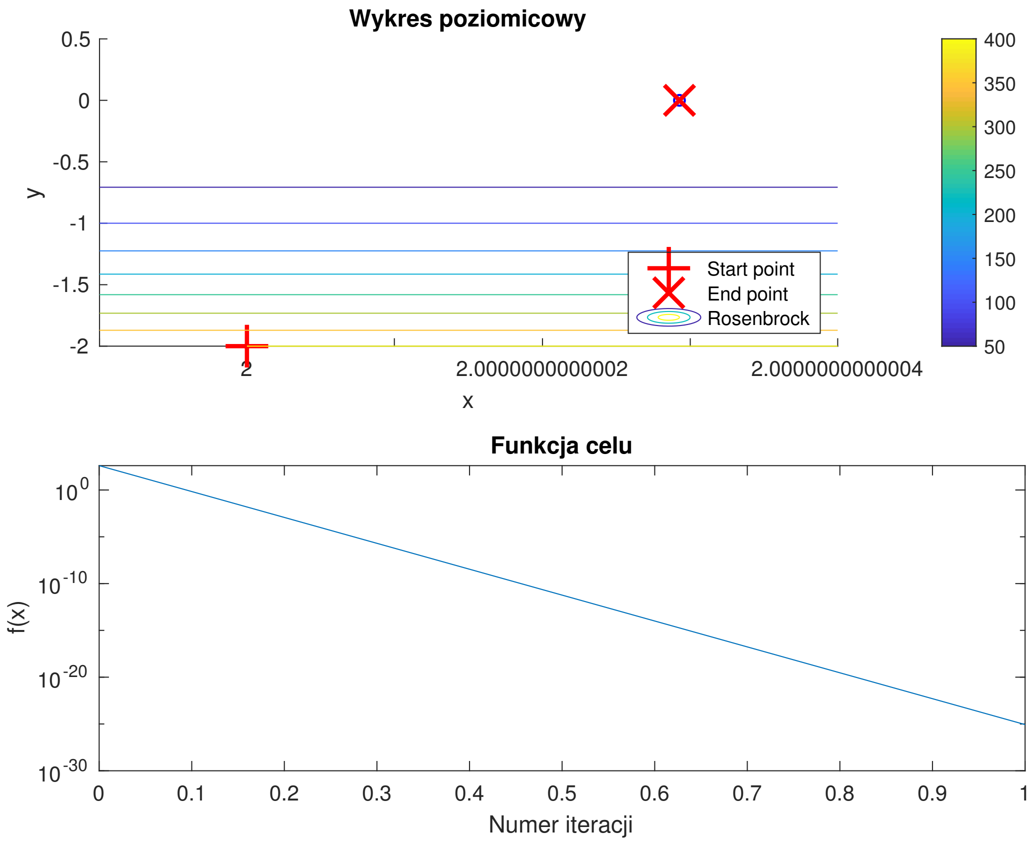
Ostatnim rozpatrzonym algorytmem jest `lsqnonlin`, czyli metoda najmniejszych kwadratów. Jest to metoda łącząca szybkość wykonania jak i dokładność rozwiązania. Otrzymujemy niemalże idealne wyniki zachowując przy tym małą liczbę iteracji od 1 do 17, w przypadku sprawdzonych punktów startowych.

Tablica 5.1: Porównanie punktów startowych

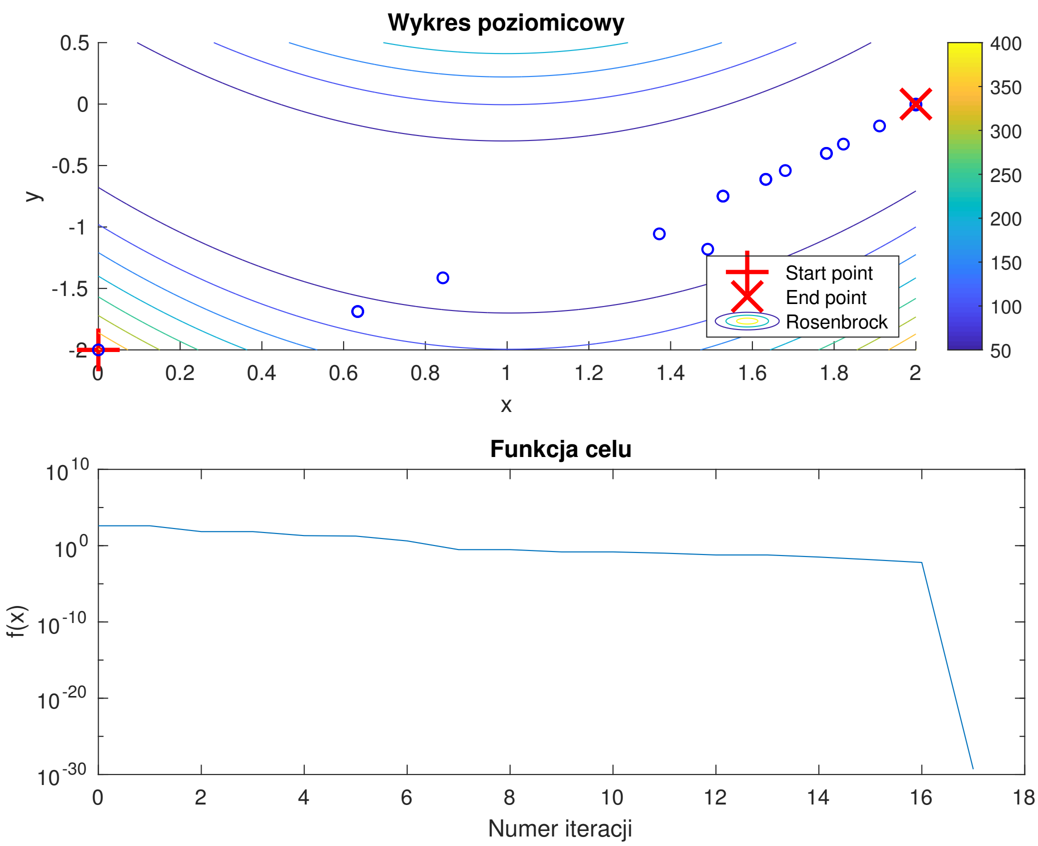
X_0	Y_0	X_{inf}	Y_{inf}	$f(x)$	Iterations
3	0	2	$-3{,}81 \cdot 10^{-22}$	0	3
2	-2	2	$5{,}88 \cdot 10^{-13}$	$8{,}61 \cdot 10^{-26}$	1
0	-2	2	$1{,}23 \cdot 10^{-16}$	$4{,}93 \cdot 10^{-30}$	17
0	0	2	$7{,}37 \cdot 10^{-18}$	0	15



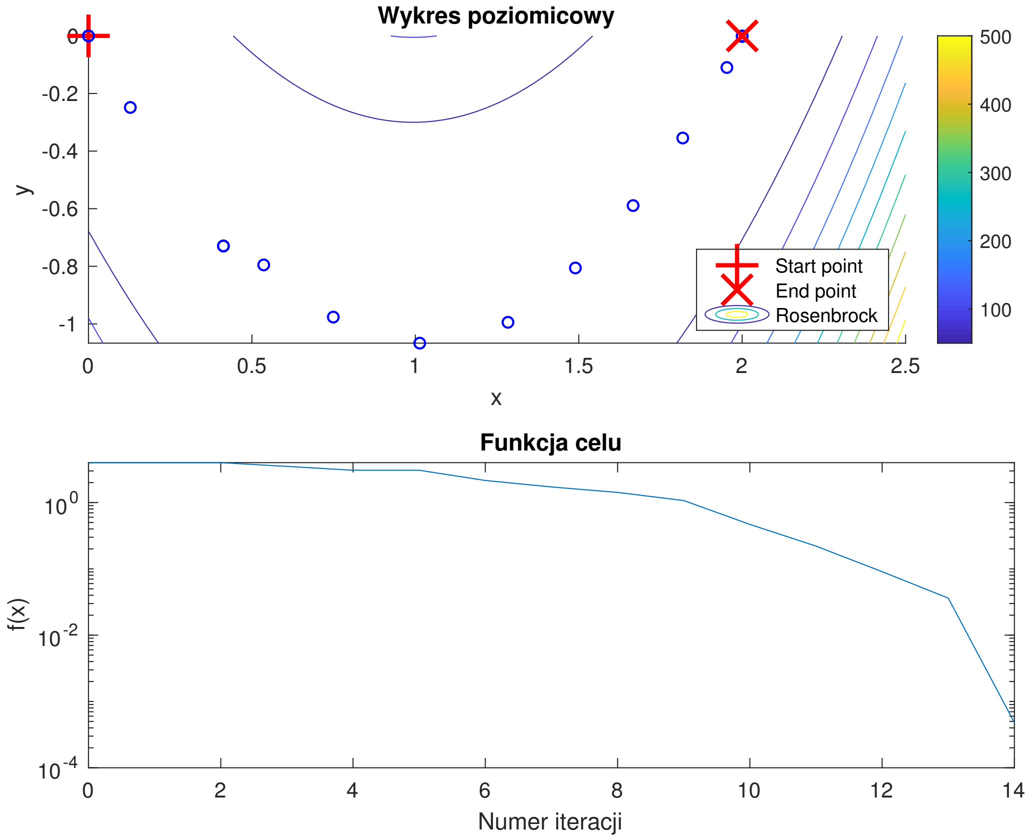
Rysunek 5.1. Zadanie optymalizacji dla punktu startowego $x = 3$, $y = 0$



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 `lsqnonlin`. Był najlepszy zarówno pod względem dokładności wyniku, jak również okazał się najszybszy. W pewnych przypadkach szybkością dorównywał mu algorytm używający gradientu funkcji, lecz dokładność tych wyników nie była zadowalająca. Pozostałe dwa algorytmy (optymalizacja bez pochodnych i z szacowaniem pochodnych) dawały bardzo zbliżone rezultaty, jednak metoda bez pochodnych była znacznie wolniejsza.