Optymalizacja ciągła

Antoni Kowalski, nr indeksu: 122579, zajęcia: czwartek: 16:50

6 września 2018

1 Opis problemu

Zadaniem postawionym przez prowadzącego zajęcia było wizualizowanie danych wielowymiarowych w przestrzeni dwuwymiarowej.

2 Opis rozwiązania problemu

Ponieważ w przestrzeni o dowolnej liczbie wymiarów zawsze możemy policzyć odległość pomiędzy dwoma punktami rozwiązaniem może być przedstawienie danych na przestrzeni dwuwymiarowej z odległościami między punktami takimi samymi jak w oryginalnym wymiarze. Będziemy więc starać się minimalizować wyrażenie:

$$f(x_1, ..., x_n) = \sum_{i}^{n} \sum_{j=i+1}^{n} (||x_i - x_j|| - d_{ij})^2$$

gdzie x_i i x_j to punkty w nowej przestrzeni dwuwymiarowej, a d_{ij} to odległość między nimi w starej przestrzeni $2n$ wymiarowej.

Funkcje f będziemy minimalizować za pomocą algorytmu spadku wzdłuż gradientu oraz jego ulepszonej wersji która maksymalizuje spadek funkcji w każdej iteracji (algorytm Cauchy'ego). Spadek maksymalizować będziemy za pomocą algorytmu przeszukiwania dychotomicznego.

3 Obliczenie gradientu

Do obliczenia gradientu użyłem internetowego narzędzia *Wolfram Alpha*.

Dla pojedynczego składnika naszej funkcji:

$$(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij})^2$$

pochodna po x_i/y_i wynosi:

$$\frac{2(x_i/y_i - x_j/y_j)(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij})^2}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$$

natomiast po x_j/y_j wynosi:

$$-\frac{2(x_i/y_i - x_j/y_j)(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij})^2}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$$

a pochodna cząstkowa sumy (całej funkcji) będzie równać się sumie pochodnych składników po wybranej zmiennej. Gradient natomiast będzie wektorem poszczególnych pochodnych cząstkowych.

Warto zwrócić uwagę, że pochodna nie ma ciągłej dziedziny, może więc okazać się, że przyjęte początkowo punkty dla których pochodna nie jest określona spowodują błąd działania algorytmu.

4 Sprawdzenie obliczeń gradientu numerycznie

Obliczenie gradientu zostało sprawdzone numerycznie wzorem:

$$f_{x_p}(x_1, ..., x_p, ..., x_n)] \approx \frac{f(x_1, ..., x_p + \delta, ..., x_n) - f(x_1, ..., x_p - \delta, ..., x_n)}{2\delta}$$

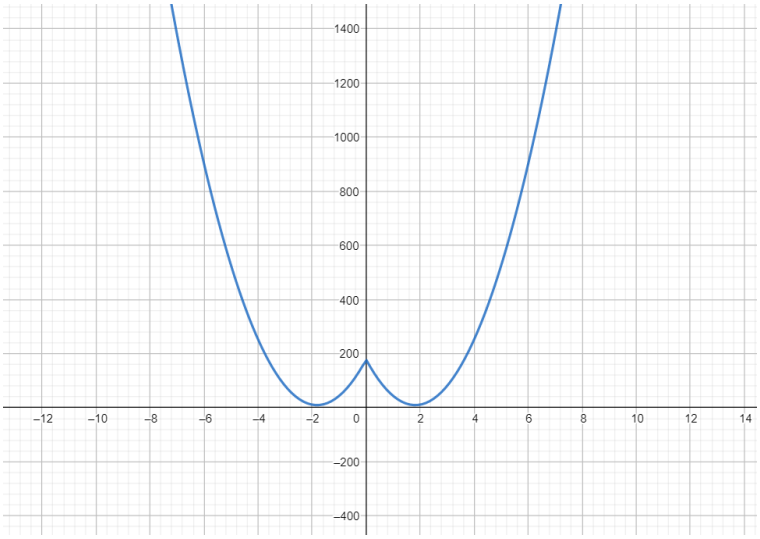
Dla odpowiednio małego δ. Wyniki potwierdziły poprawność obliczenia gradientu metodą analityczną.

5 Algorytm przeszukiwania dychotomicznego

Aby algorytm zadziałał poprawnie musimy wiedzieć czy minimalizowana funkcja:

$$g(\alpha) = f(x_k - \alpha \bigtriangledown f(x_k))$$

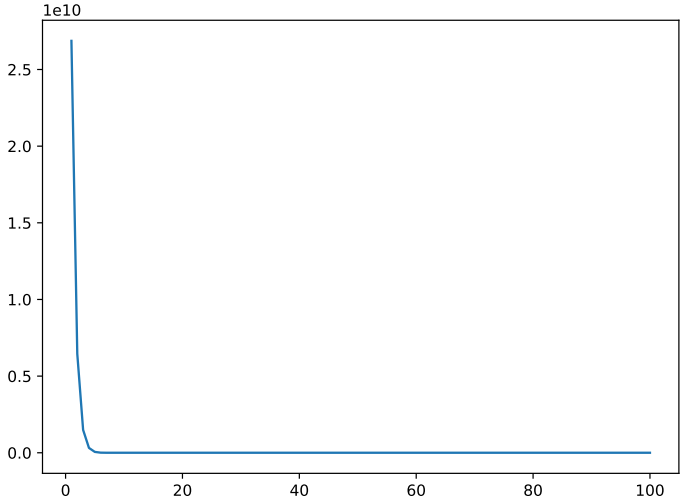
jest jednomodalana.



Rysunek 1: Wykres funkcji $g(\alpha) = f(x_k - \alpha \bigtriangledown f(x_k))$ dla przykładowego x_k i $\bigtriangledown f(x_k)$

Po narysowaniu wykresu funkcji (Rysunek 1) możemy zaobserwować że funkcja jest symetryczna, oraz, że jest jednomodalna w przedziałach $[\infty, 0)$ oraz $(0, \infty]$. Do minimalizowania funkcji dobrany został ręcznie przedział $[10^{-19}, 20]$ który w praktyce pozwala efektywnie minimalizować naszą funkcję.

Jak możemy zaobserwować na wykresie pokazującym przebieg minimalizacji funkcji przez algorytm przeszukiwania dychotmicznego duży spadek wartości możemy zaobserwować już po 10 iteracjach. Ta liczba iteracji została użyta w dalszej części do minimalizowania spadku.

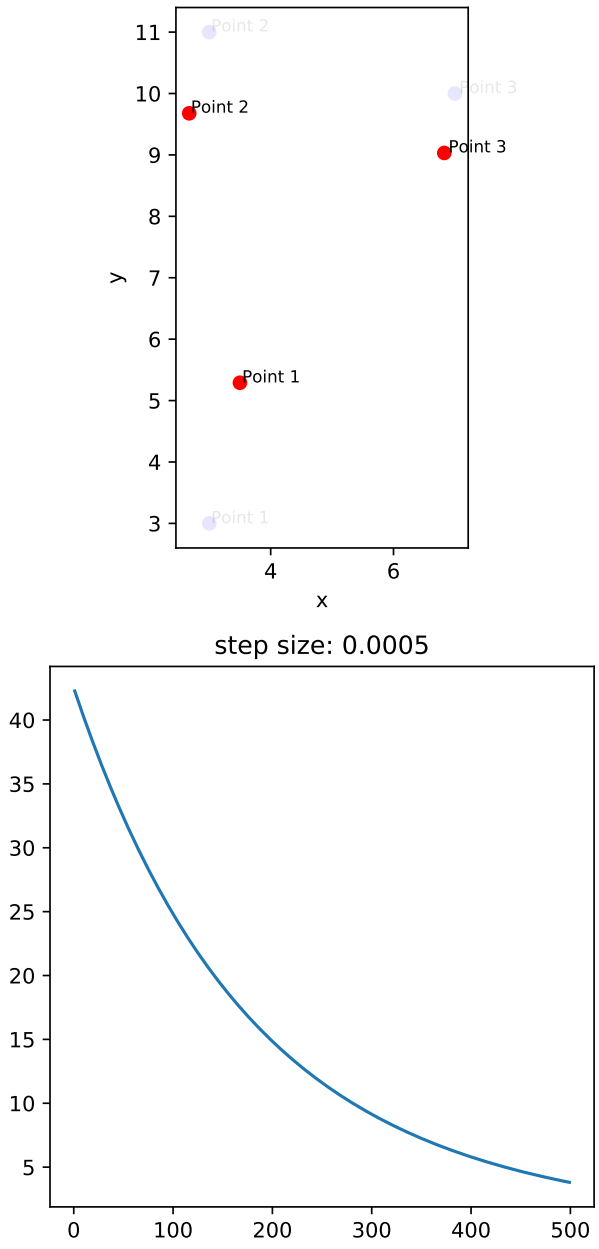


Rysunek 2: Wykres przedstawiający spadek wartości funkcji celu podczas dobierania optymalnej długości kroku

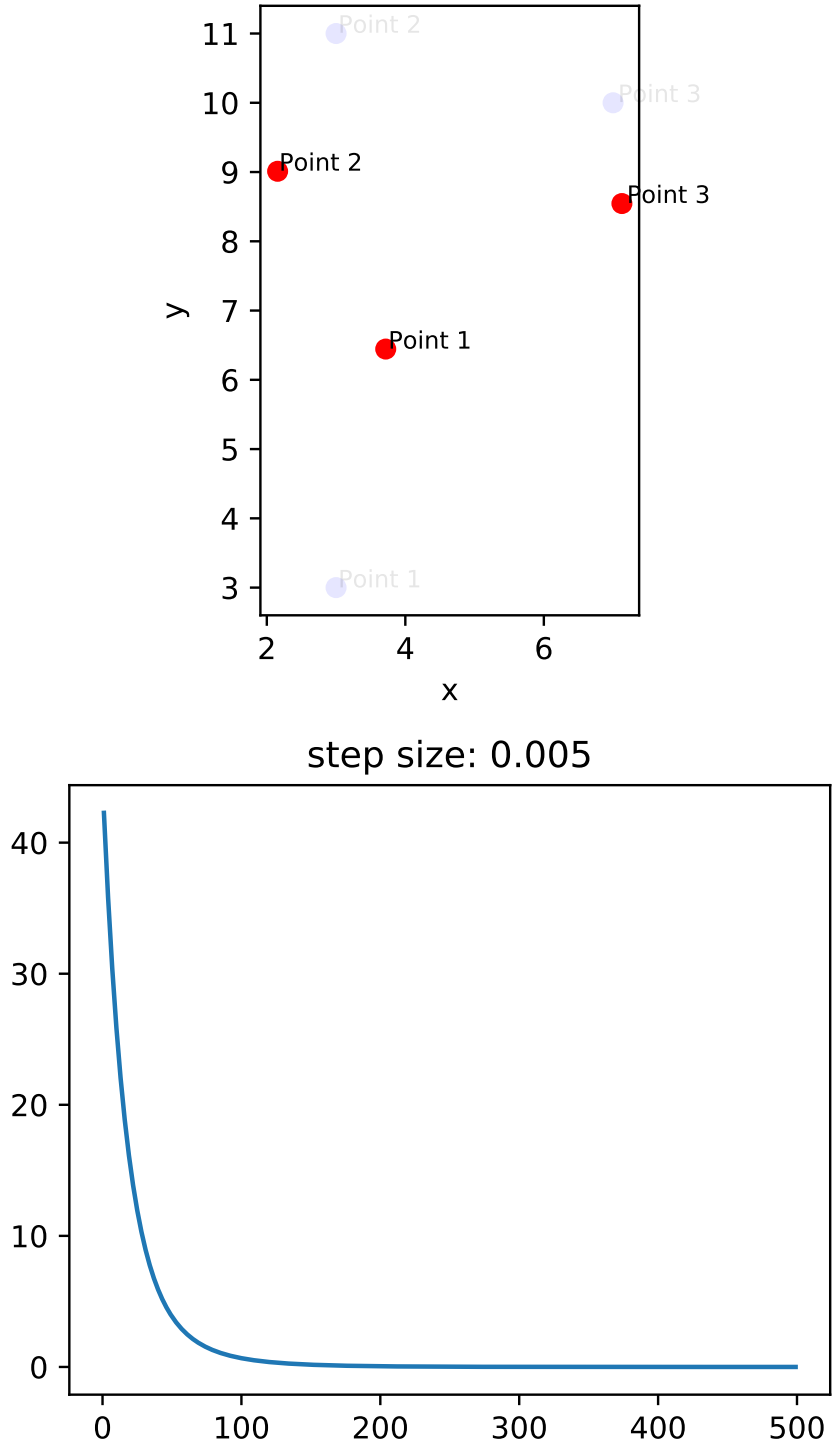
6 Wersja algorytmu z długością kroku zależną od czasu

Postanowiłem również spróbować wersji dobierania kroku zależnej od czasu przedstawionej na laboratoriach.W tej wersji długość kroku zależna była od ilości iteracji i wyliczana ze wzoru: $\alpha = 1/\sqrt{n}$ gdzie n to numer iteracji.

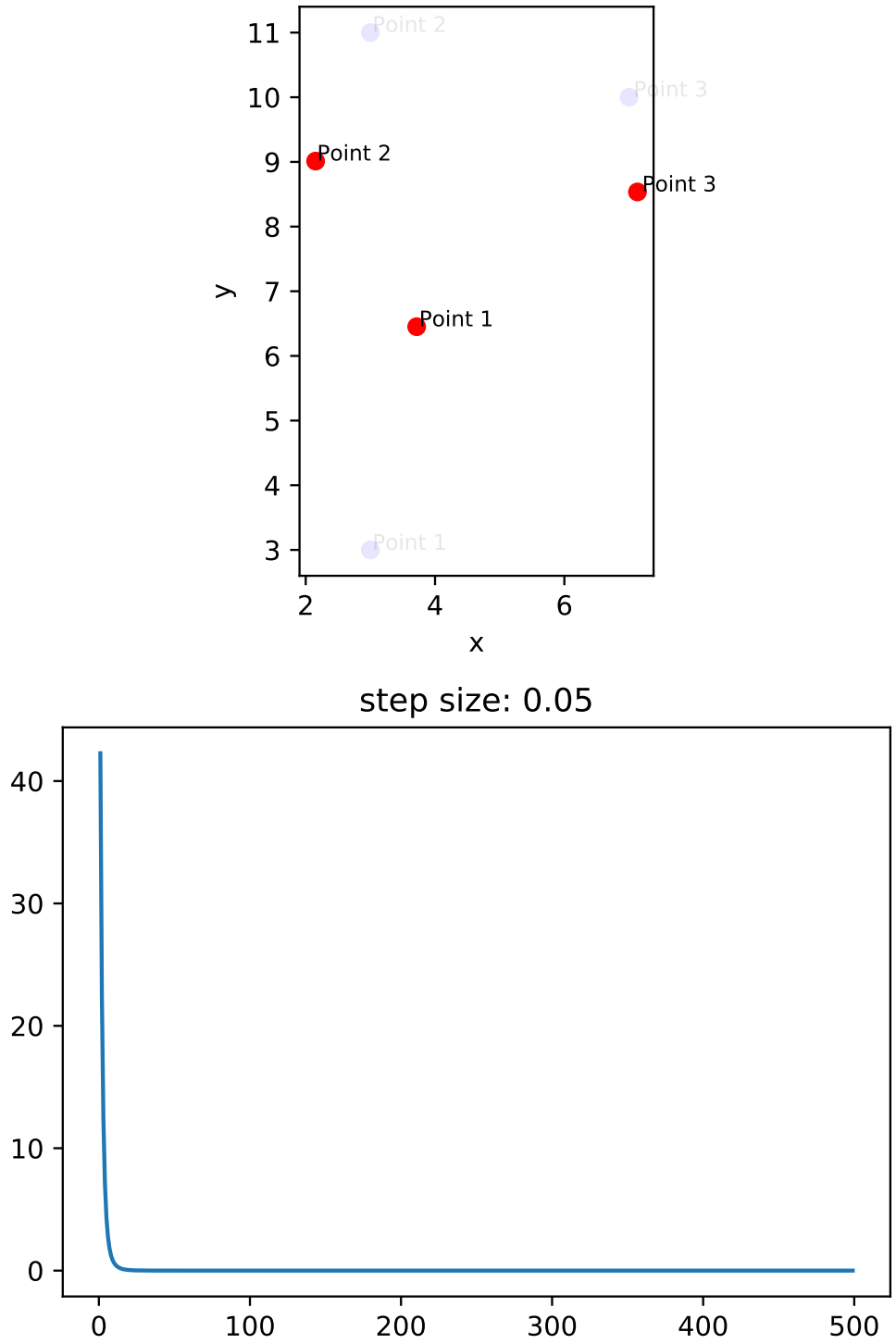
7 Wyniki dla problemu trójkąta, wersja ze stałą długością kroku



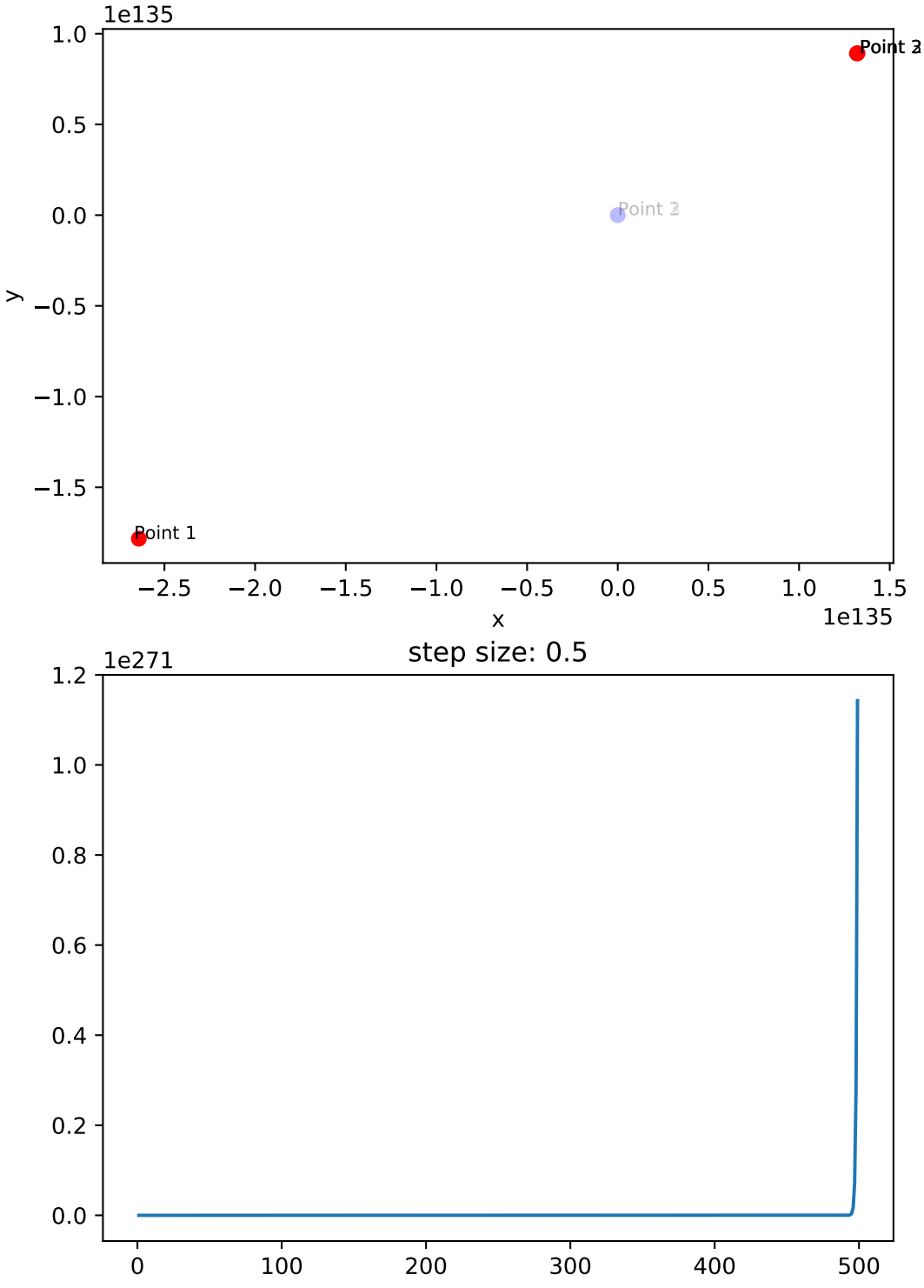
Rysunek 3: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta, Długość kroku : 0.0005, liczba iteracji: 500. Końcowa wartość funkcji: $3,795$.



Rysunek 4: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta, Długość kroku : 0.005, liczba iteracji: 500. Końcowa wartość funkcji: $9,81 \cdot 10^{-5}$.

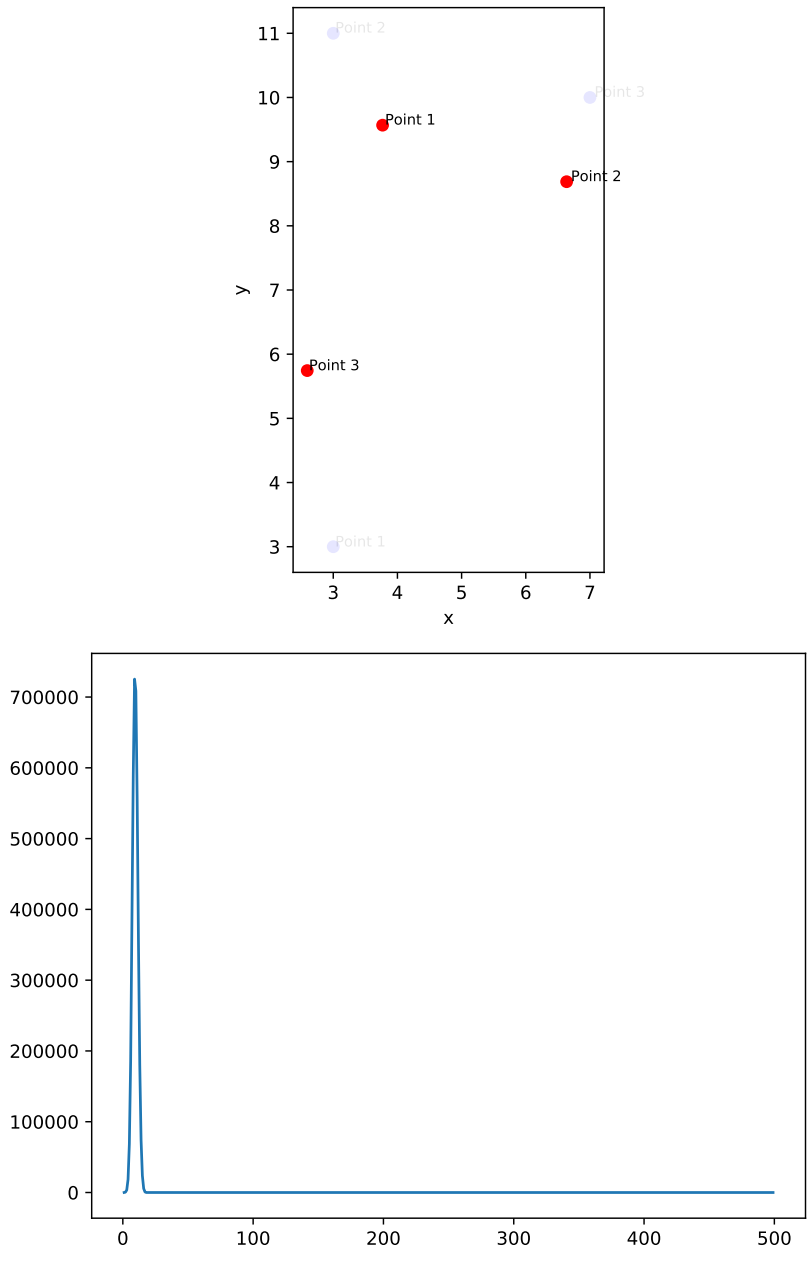


Rysunek 5: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta, Długość kroku : 0.05, liczba iteracji: 500. Końcowa wartość funkcji: $2,939 \cdot 10^{-29}$



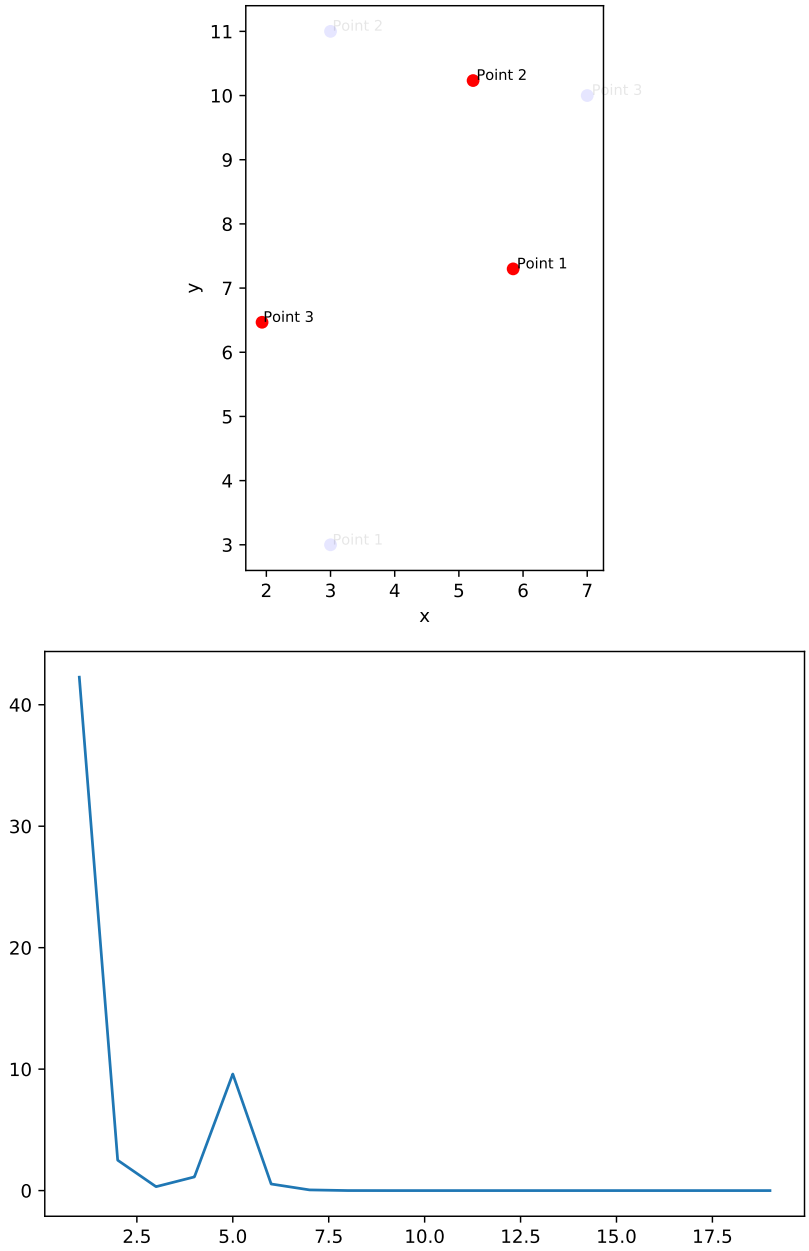
Rysunek 6: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta, Długość kroku : 0.5, liczba iteracji: 500. Końcowa wartość funkcji: $1,143 \cdot 10^{271}$

8 Wyniki dla problemu trójkąta, wersja z długością kroku zależną od czasu



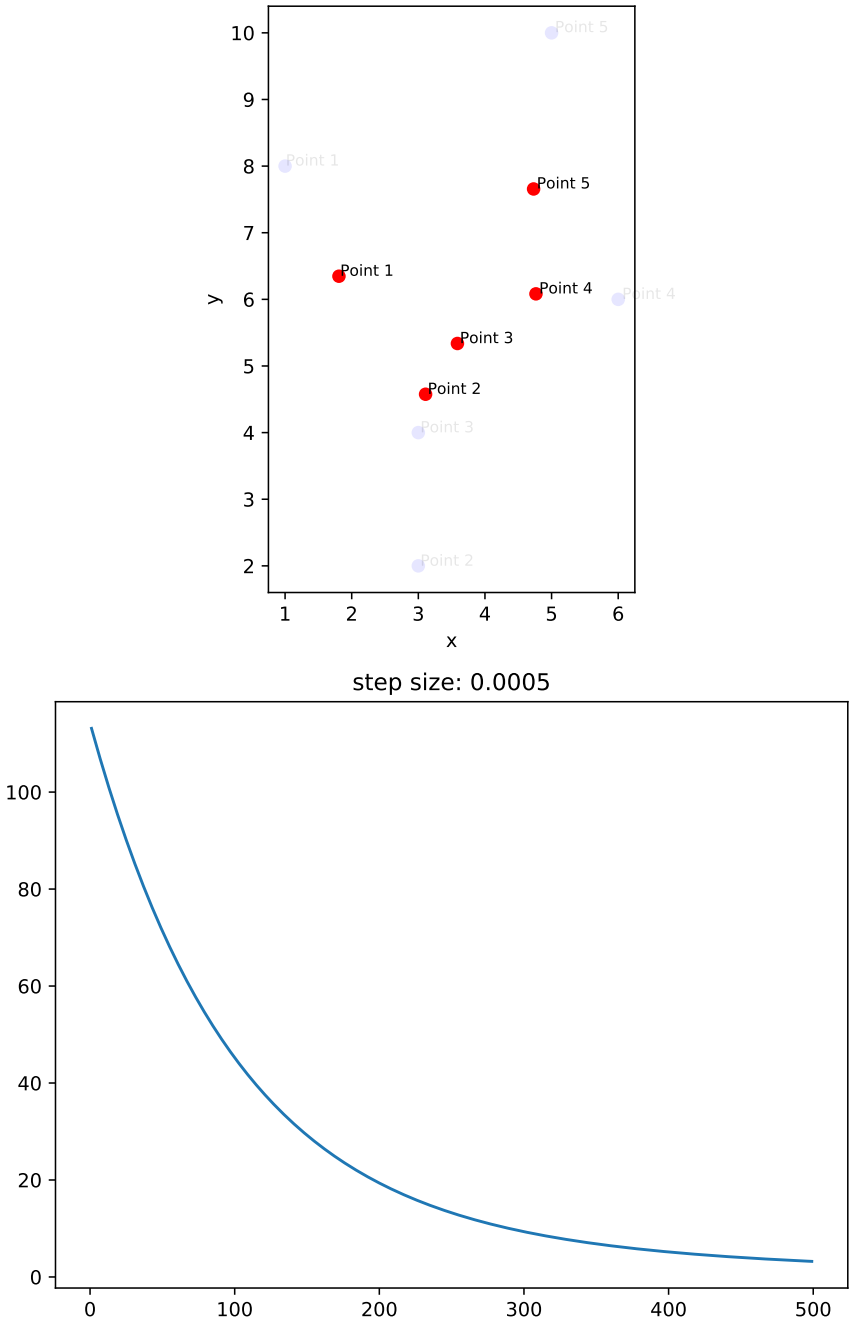
Rysunek 7: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta. Długość kroku zależna od czasu. Liczba iteracji: 500. Końcowa wartość funkcji: $2,465 \cdot 10^{-29}$

9 Wyniki dla problemu trójkąta, algorytm Cauchy'ego

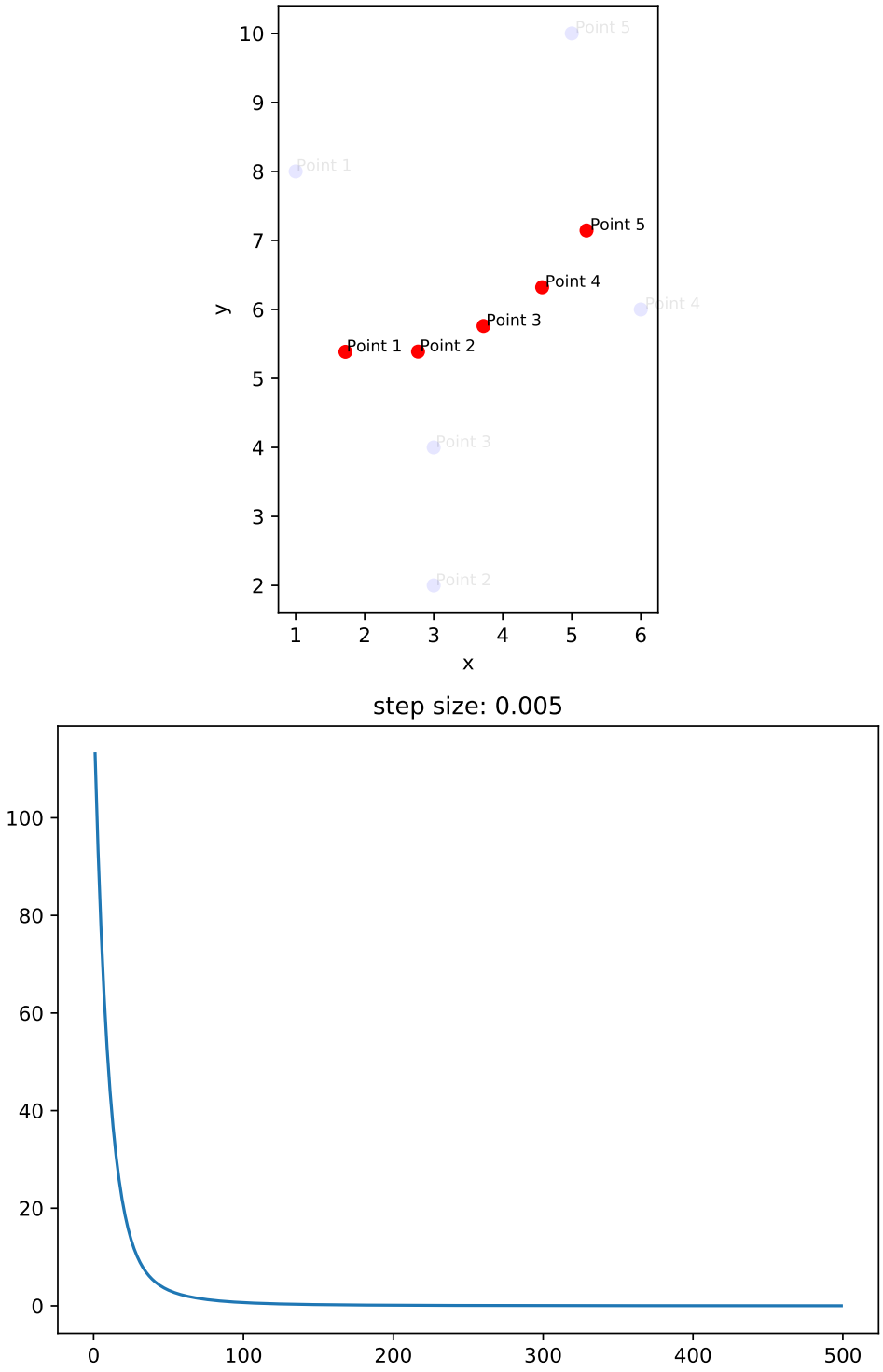


Rysunek 8: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu trójkąta. Długość kroku maksymalizująca spadek funkcji. Liczba iteracji: 20.

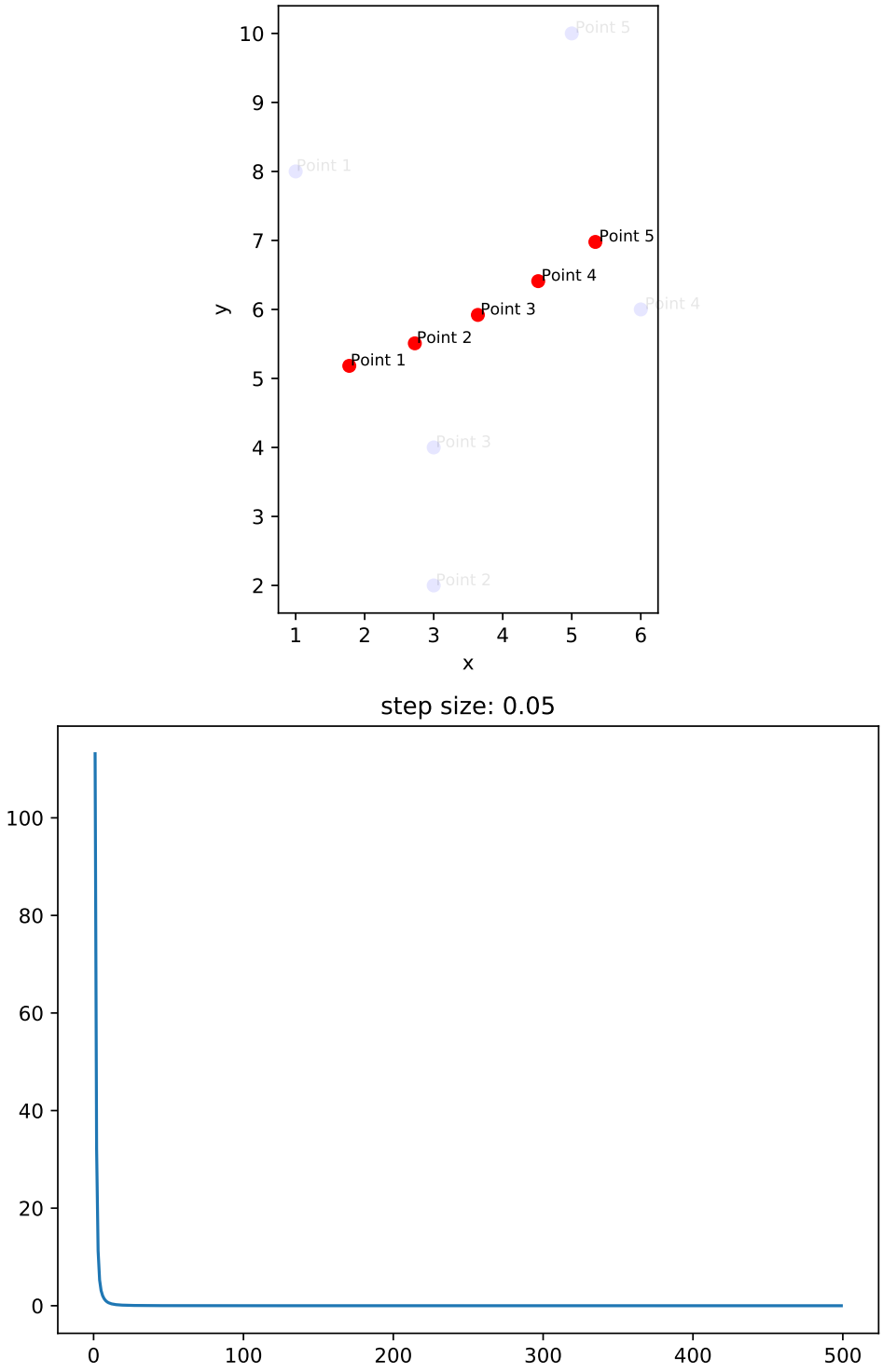
10 Wyniki dla problemu linii, wersja ze stałą długością kroku



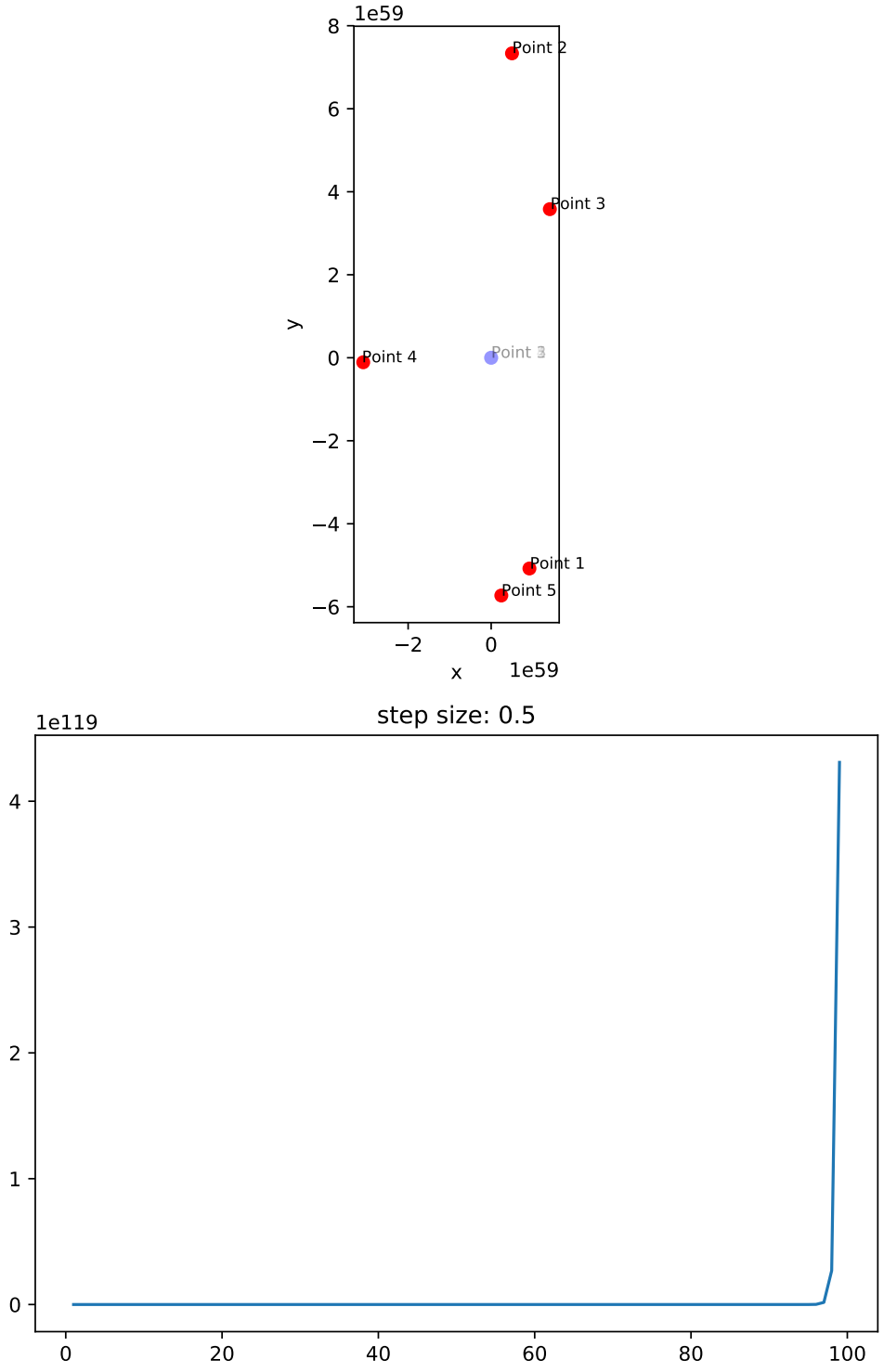
Rysunek 9: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii, Długość kroku : 0.0005, liczba iteracji: 500.



Rysunek 10: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii, Długość kroku : 0.005, liczba iteracji: 500.

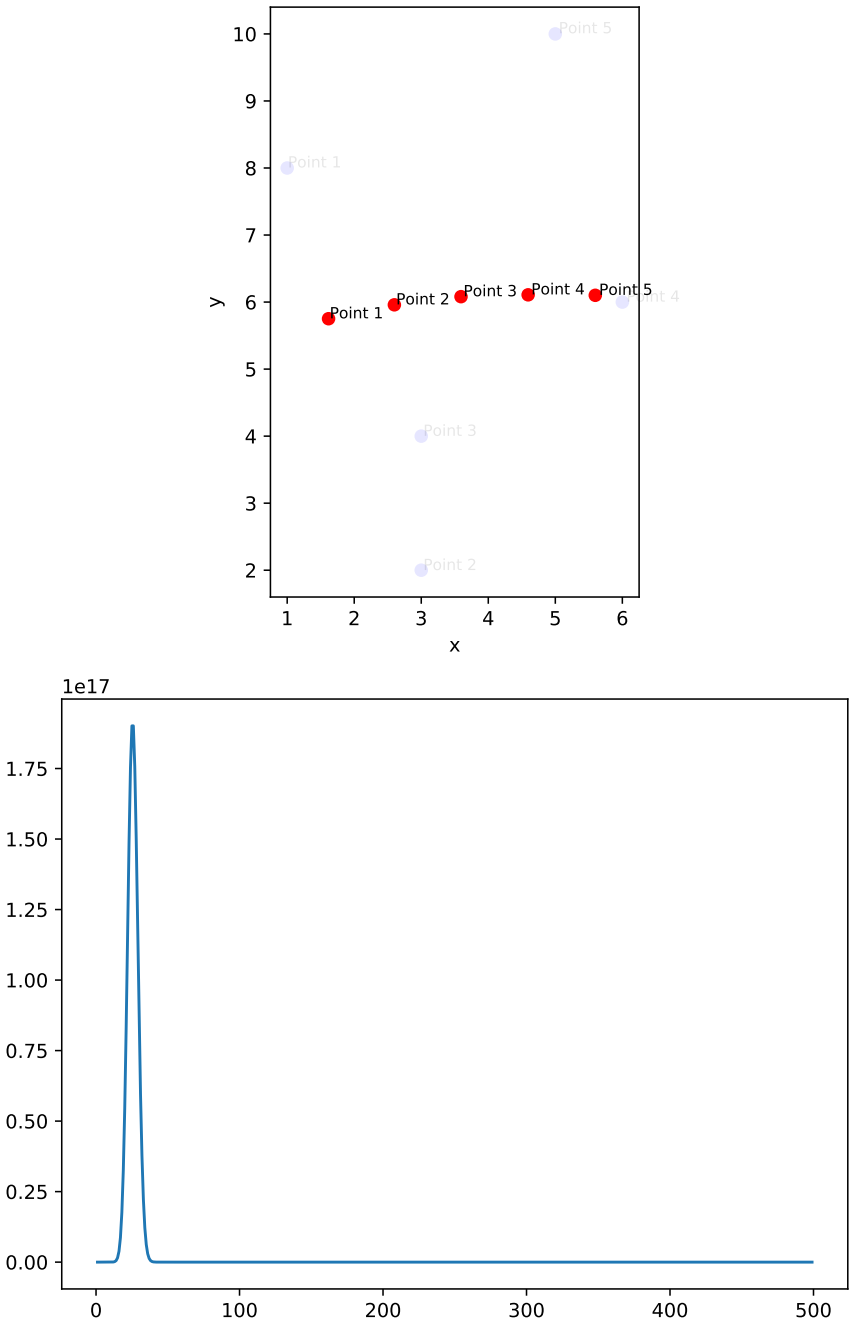


Rysunek 11: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii, Długość kroku : 0.05, liczba iteracji: 500.



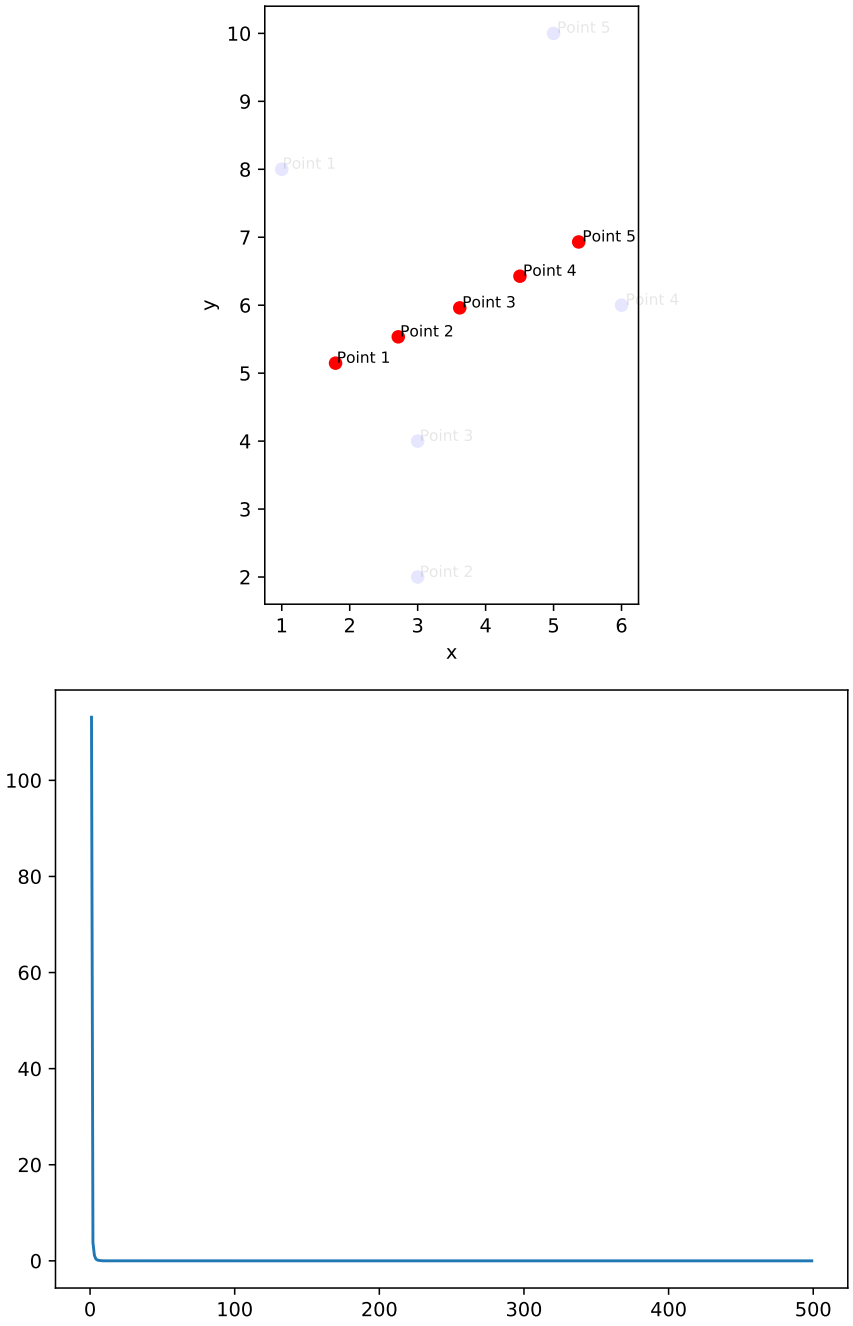
Rysunek 12: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii, Długość kroku : 0.5, liczba iteracji: 500.

11 Wyniki dla problemu linii, wersja z długością kroku zależną od czasu



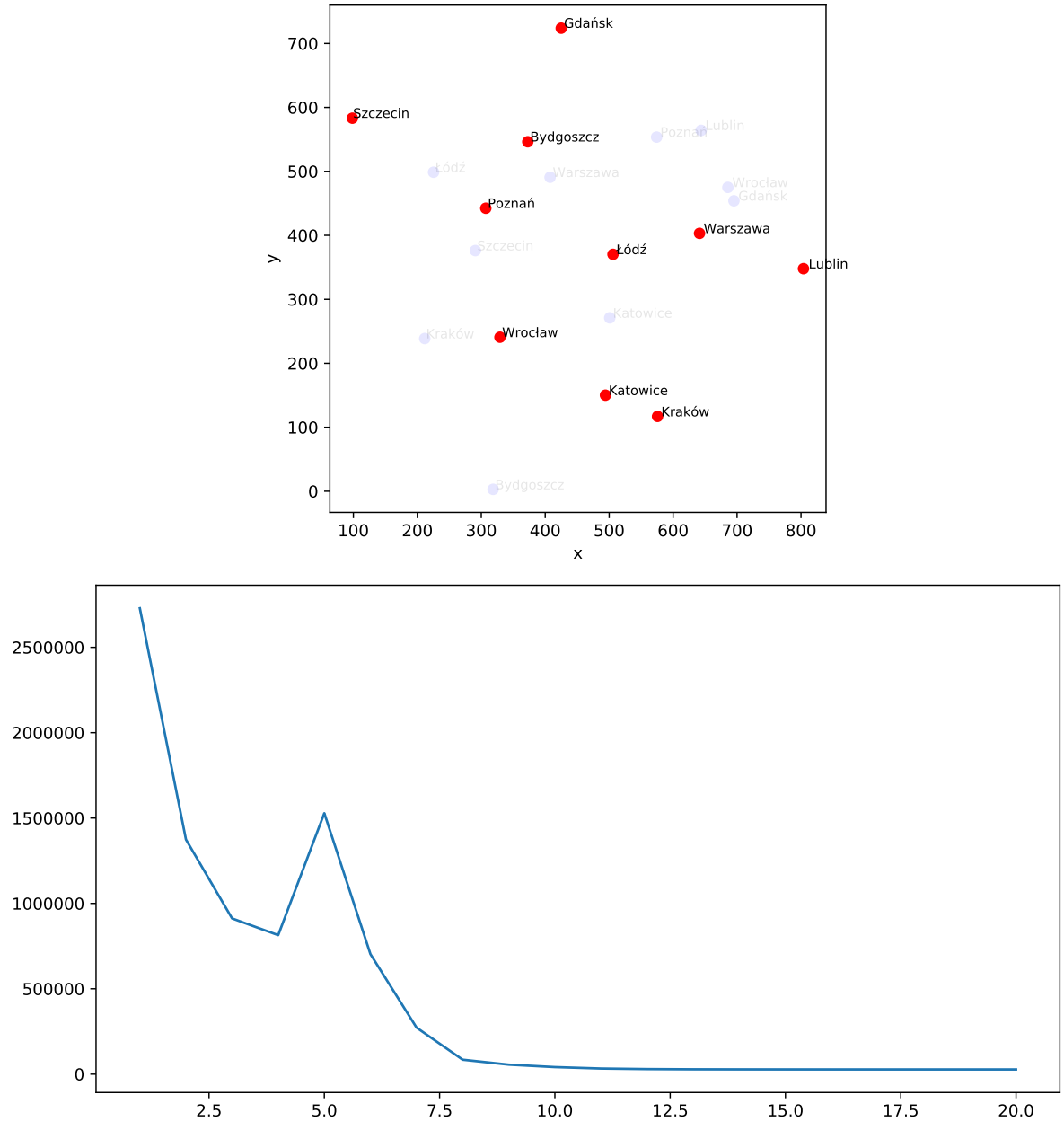
Rysunek 13: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii. Długość kroku zależna od czasu.Liczba iteracji: 500

12 Wyniki dla problemu linii, algorytm Cauchy'ego



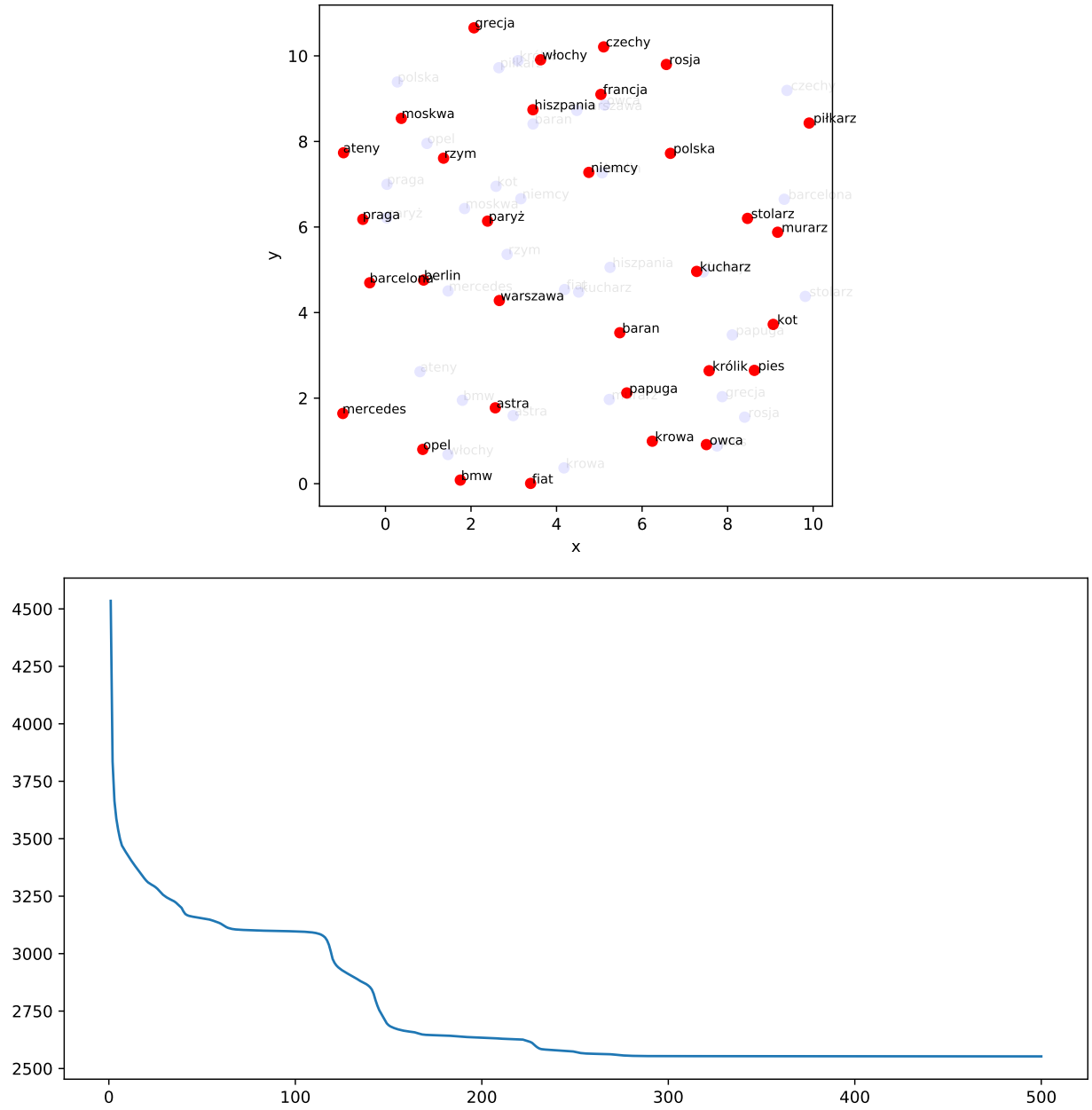
Rysunek 14: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu linii. Długość kroku maksymalizująca spadek funkcji.

13 Problem miast, algorytm Cauchy'ego



Rysunek 15: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu miast. Długość kroku maksymalizująca spadek funkcji. Nie zawsze udawało się uzyskać poprawne rozmieszczenie miast - zależało to od pozycji punktów początkowych

14 Problem słów, algorytm Cauchy'ego



Rysunek 16: Początkowe (niebieskie) i końcowe (czerwone) punkty wyliczone przez algorytm oraz błąd w czasie dla problemu słów. Długość kroku maksymalizująca spadek funkcji. Widzimy wyraźnie zbliżenie do siebie słów z tych samych kategorii