Akademia Górniczo-Hutnicza w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Metody Numeryczne

Laboratorium 10: Minimalizacja wartości funkcji metodą interpolacji kwadratowej Powella

Andrzej Świętek

23.04.2024

Contents

1 Wstęp teoretyczny **2**

2 Problem **3**

3 Implementacja **3**

4 Wyniki **5**

5 Analiza wyników **11**

6 Wnioski **12**
6.1 Złożoność obliczeniowa: 12
6.2 Użyteczność: 12
6.3 Błędy i ograniczenia: 12
6.4 Optymalizacje i ulepszenia: 12

1. Wstęp teoretyczny

Metoda Powella to jedna z popularnych technik optymalizacji nieliniowej, używana do znajdowania minimum funkcji. Jest to algorytm iteracyjny, który wykorzystuje kombinację kierunków do zbliżania się do minimum. Technika ta została nazwana na cześć Michaela J. D. Powella, który opisał ją w 1964 roku.

Jest to metoda bezgradientowa. Jej podstawowym założeniem jest to, że funkcja celu może być lokalnie przybliżona za pomocą funkcji kwadratowej. Polega na znalezieniu minimum funkcji przy użyciu interpolacji kwadratowej między trzema punktami, a następnie przesunięciu tych punktów, aby znaleźć kolejne minimum.
Algorytm rozpoczyna od wyboru zestawu początkowych kierunków. Następnie, iteracyjnie porusza się wzdłuż wybranych kierunków, przemieszczając punkt w przestrzeni parametrów, aż do znalezienia minimum lub osiągnięcia z góry zadanej liczby iteracji.

Ważne jest również zauważenie, że metoda Powella może być skuteczna zarówno w przypadku funkcji jednowymiarowych, jak i wielowymiarowych, co sprawia, że jest uniwersalnym narzędziem do optymalizacji w różnych dziedzinach, takich jak nauka, inżynieria czy ekonomia.

Krok 1: Inicjalizacja:

- Ustal warunki początkowe: trzy punkty początkowe (x_1, x_2, x_3), odległe o małą wartość h od siebie.
- Określ liczbę iteracji N.
- Ustal dokładność ϵ (jeśli wymagane).

Krok 2: Iteracja:

- Dla każdej iteracji $i = 1, 2, \ldots, N$, wykonaj:
  - Oblicz punkt x_m za pomocą interpolacji Powella: $x_m = \frac{x_1+x_2}{2} - \frac{F(x_1,x_2)}{F2(x_1,x_2,x_3)}$.
  - Zapisz wartości $x_1, x_2, x_3, x_m, F[x_1, x_2], F[x_1, x_2, x_3]$ do pliku wynikowego.
  - Znajdź największą różnicę między x_m a x_1, x_2, x_3.
  - Zaktualizuj x_1, x_2, x_3 na podstawie największej różnicy:
    - Jeśli $|x_2 - x_m| > |x_1 - x_m|$ i $|x_2 - x_m| > |x_3 - x_m|$, to $x_2 = x_m$.
    - Jeśli $|x_3 - x_m| > |x_1 - x_m|$ i $|x_3 - x_m| > |x_2 - x_m|$, to $x_3 = x_m$.
    - W przeciwnym przypadku, $x_1 = x_m$.
- **Warunek zakończenia:**
  - Sprawdź, czy osiągnięto warunek zakończenia (np. wykonano określoną liczbę iteracji lub różnica między kolejnymi x jest mniejsza niż ϵ). Jeśli tak, zakończ algorytm.
- **Zakończenie:**
  - Zwróć ostatnie znalezione minimum oraz odpowiadający mu punkt (x_m).

2. Problem

1. Zaprogramować metodę interpolacji Powella do znalezienia minimum wartości funkcji.
2. Dla funkcji $f_1(x) = \ln(x^5 + 3x^2 + x + 9)$:
   - Sporządzić wykres funkcji w zakresie $x \in [-1.5, 1]$.
   - Przyjąć jako punkty startowe: $x_1 = -0.5$, $x_2 = x_1 + h$, $x_3 = x_2 + h$, $h = 0.01$ i znaleźć kolejne 10 przybliżeń położenia minimum. W każdej iteracji do pliku proszę zapisać: x_1, x_2, x_3, x_m, $F[x_1, x_2]$, $F[x_1, x_2, x_3]$.
   - Powtórzyć rachunki z poprzedniego podpunktu dla: $x_1 = -0.9$, $x_2 = x_1 + h$, $x_3 = x_2 + h$, $h = 0.01$. Dane zapisać do pliku.
3. Dla funkcji $f_2(x) = x^6$:
   - Przyjąć jako punkty startowe: $x_1 = 1.5$, $x_2 = x_1 + h$, $x_3 = x_2 + h$, $h = 0.01$ i znaleźć kolejne 100 przybliżeń położenia minimum. W każdej iteracji do pliku proszę zapisać jak poprzednio: x_1, x_2, x_3, x_m, $F[x_1, x_2]$, $F[x_1, x_2, x_3]$.

3. Implementacja

Pierwszym krokiem było zdefiniowanie rozpatrywanych funkcji dla których znajdujemy minima.

```
double f_1(double x) {
    return log( pow(x,5) + 3* pow(x,2) + x + 9 );
}

double f_2(double x) {
    return pow(x,6);
}
```

Listing 1: Dwie rozpatrywane funkcje

Następnie postępując zgodnie z algorytmem interpolacji Powella, tworzymy funckje do wyznaczania ilorazów:

$$F[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

oraz

$$F[x_1, x_2, x_3] = \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_1}$$

```
double F(double x1, double x2, double (*f)(double)) {
    return (f(x2) - f(x1)) / (x2 - x1);
}

double F2(double x1, double x2, double x3, double (*f)(double)) {
    return  (
        (f(x3) - f(x2)) / (x3 - x2) - (f(x2) - f(x1)) / (x2 - x1)
     ) / (x3 - x1);
}
```

Ostatecznie wykorzystując przygotowane metody tworzymy funkcje odpowiedzialną za interpolację Powell'a, która w daleszej części kodu będzie nam wyznaczać xm. Funkcja przyjmuje 3 punkty x_1, x_2, x_3 oraz wskaźnik do funkcji $f(x)$ (aby kod był bardziej generyczny)

```
double powell_interpolation(double x1, double x2, double x3, double (*f)(double))
     {
    return (x1 + x2)/2 - 0.5 * F(x1, x2, f) / F2(x1, x2, x3, f);
}
```

Listing 2: Element interpolacji Powell'a

Mając funkcje intepolującą tworzymy funkcję odpowiedzialną za wykonanie danej liczby iteracji celem znalezienia satysfakcjonującego minimum. Funkcja jako parametr otrzymuje wskaźnik do rozpatrywanej funkcji, początkowe punkty x_1, x_2, x_3, liczbę iteracji oraz nazwę pliku do którego zapisze uzyskane wyniki.

```
void find_minimum(double (*f)(double), double x1, double x2, double x3, int
    iterations, double h, const std::string& filename) {
    std::ofstream file(filename);
    file << "x1\tx2\tx3\txm\tF[x1,x2]\tF[x1,x2,x3]\n";
    for (int i = 0; i < iterations; ++i) {
        double xm = powell_interpolation(x1, x2, x3, f);
        file << x1 << "\t" << x2 << "\t" << x3 << "\t" << xm << "\t" << F(x1, x2,
     f) << "\t" << F2(x1, x2, x3, f) << "\n";

        double max = fabs(x1-xm);
        if(fabs(x2-xm) > max){ // Moment "sortowania"
            max = fabs(x2-xm);
            x2 = xm;
        }
        else if(fabs(x3-xm) > max){
            max = fabs(x3-xm);
            x3 = xm;
        }
        else{
            x1 = xm;
        }
    }
}
```

Listing 3: Funkcja odpowiedzialna za wykonanie n iteracji i znalezienia minimum

```
int main() {
    double h = 0.01;

    double x_1 = -0.5;
    double x_2 = x_1 + h;
    double x_3 = x_2 + h;

    find_minimum(f_1, x_1, x_2, x_3, 10, h, "data/f1_results.csv");

    //------------------------------------------------------------------------

    x_1 = -0.9;
    find_minimum(f_1, x_1, x_1 + h, x_1 + 2 * h, 10, h, "data/f1_results_2.csv");

    //------------------------------------------------------------------------

```

```
    double x_1_f2 = 1.5;
    double x_2_f2 = x_1 + h;
    double x_3_f2 = x_2 + h;

    find_minimum(f_2, x_1_f2, x_2_f2, x_3_f2, 100, h, "data/f2_results.csv");

    return 0;
}
```

4. Wyniki

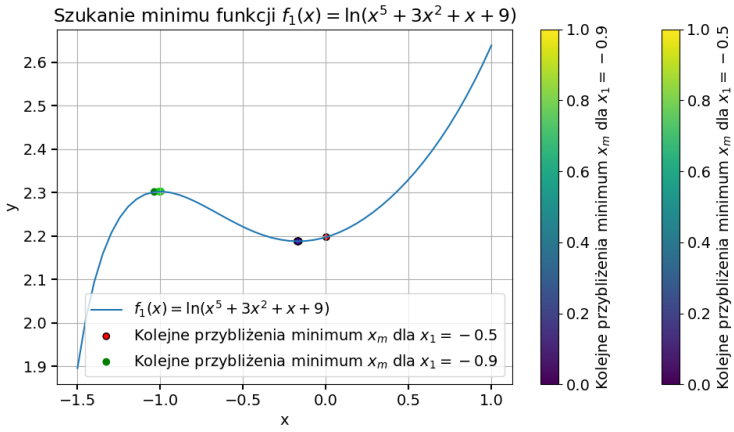

Figure 1: Wykres reprezentujący kolejne przybliżenia szukanego minimu dla funkcji $f_1(x)$ dla dwóch róznych punktów startowych x_1 w zadanym przedziale

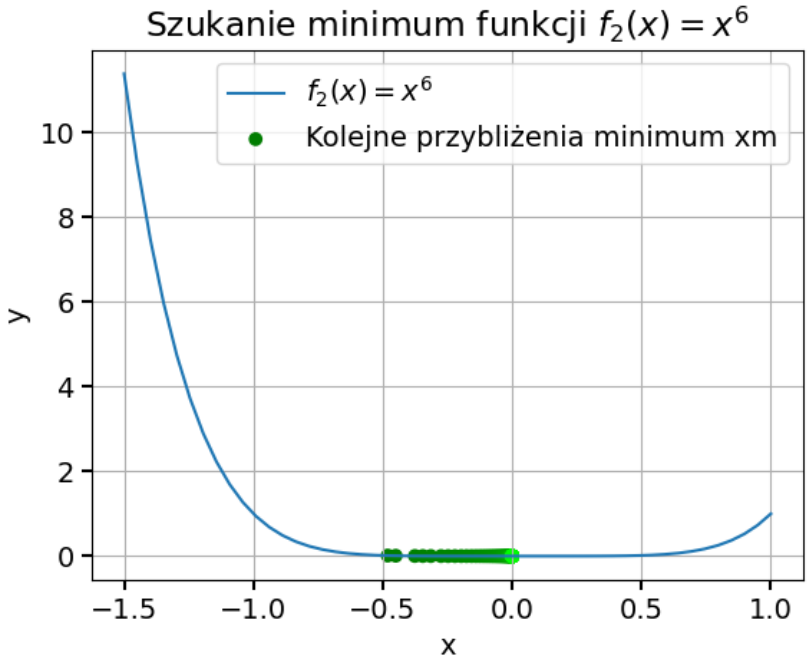

Figure 2: Wykres reprezentujący kolejne przybliżenia szukanego minimu dla funkcji $f_2(x) = x^6$ w zadanym przedziale

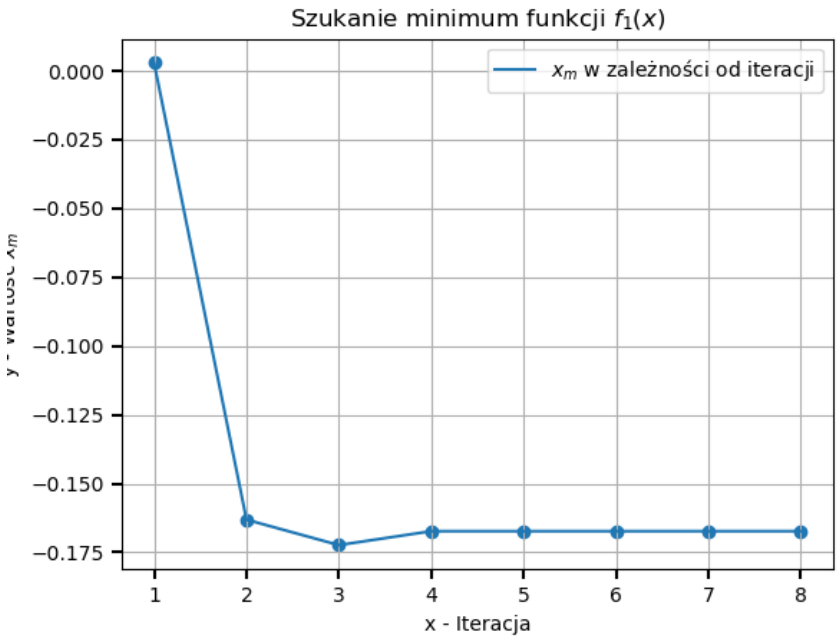

Figure 3: Zmiany x_m w funkcji iteracji dla $f_1(x)$ dla punktu startowego $x_1 = -0.5$

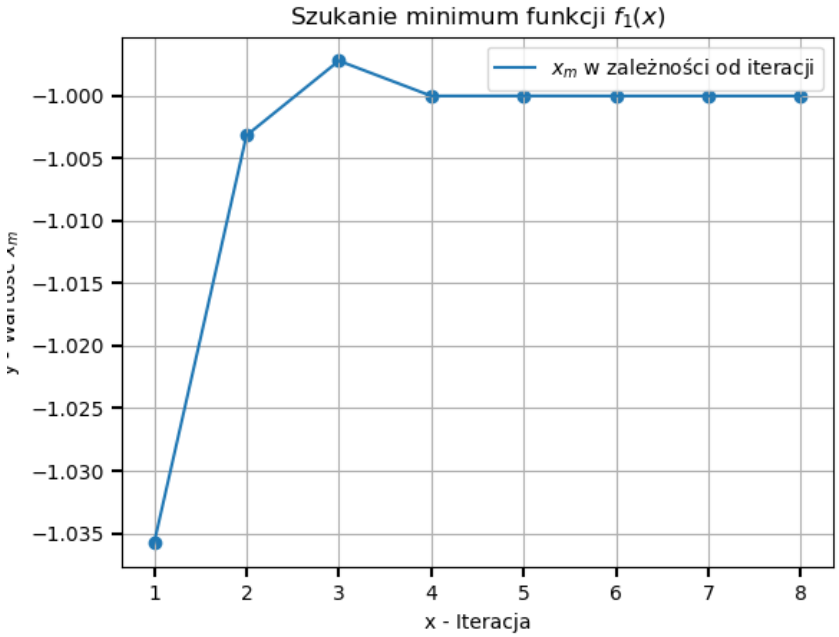

Figure 4: Zmiany x_m w funkcji iteracji dla $f_1(x)$ dla punktu startowego $x_1 = -0.9$

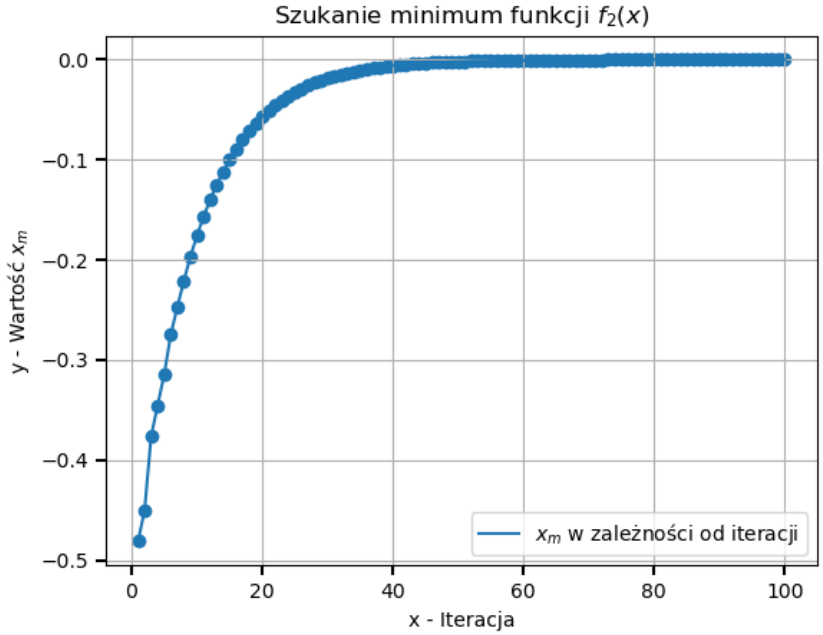

Figure 5: Zmiany x_m w funkcji iteracji dla $f_2(x)$

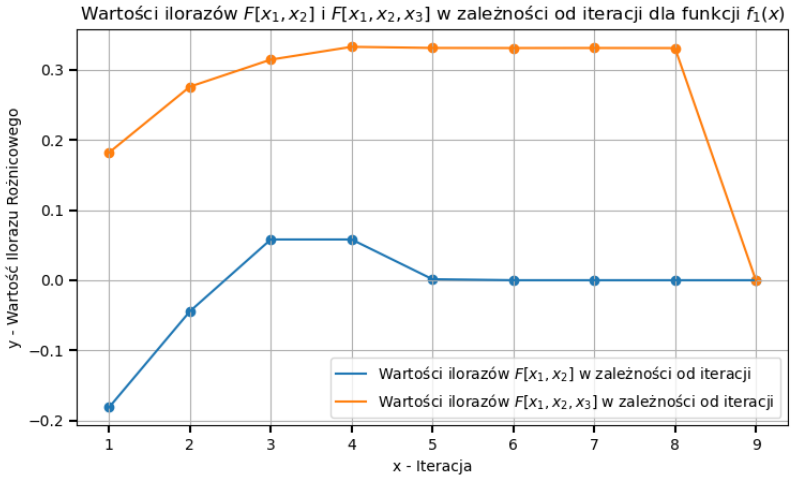

Figure 6: Zmiany ilorazu 1 i 2 rzędu w funkcji iteracji dla $f_1(x)$ dla punktu startowego $x_1 = -0.5$

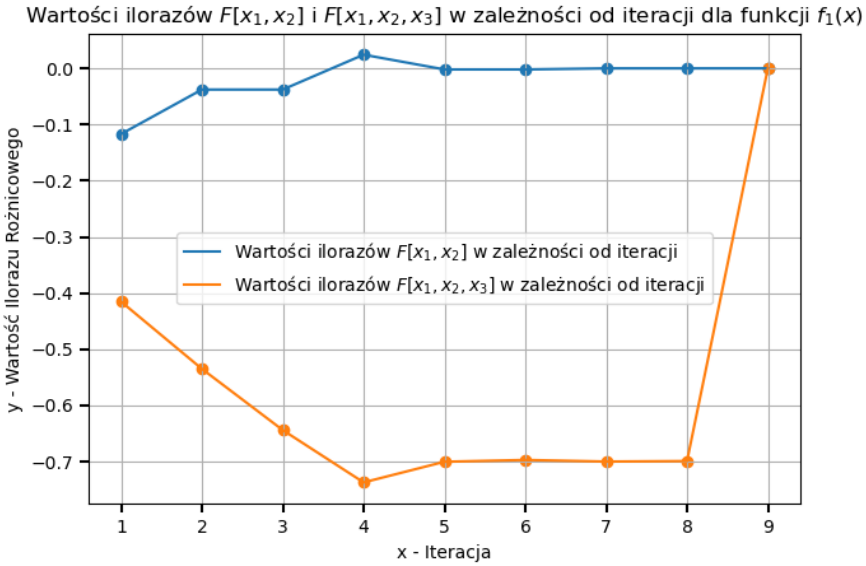

Figure 7: Zmiany ilorazu 1 i 2 rzędu w funkcji iteracji dla $f_1(x)$ dla punktu startowego $x_1 = -0.9$

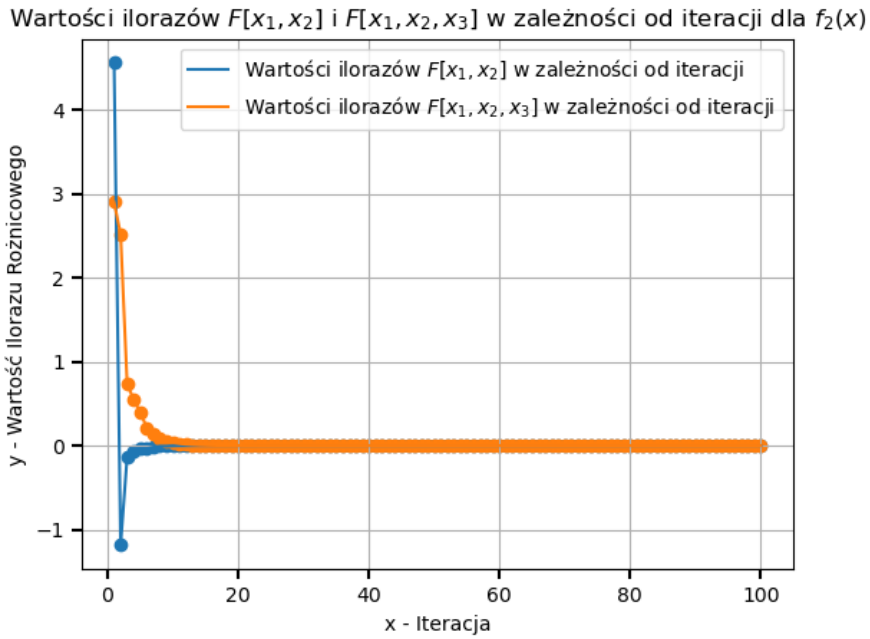

Figure 8: Zmiany ilorazu 1 i 2 rzędu w funkcji iteracji dla $f_2(x)$

5. Analiza wyników

1. **Analiza wartości ilorazów dla funkcji $f_1(x)$:**

- Przy pierwszym zestawie punktów startowych $x_1 = -0.5, x_2 = x_1 + h, x_3 = x_2 + h$, wartość $F[x_1, x_2, x_3]$ w ostatniej iteracji wynosi 0.331103, co wskazuje na to, że funkcja zmierza do minimum.

- Dla drugiego zestawu punktów startowych $x_1 = -0.9, x_2 = x_1 + h, x_3 = x_2 + h$, wartość $F[x_1, x_2, x_3]$ w ostatniej iteracji wynosi -0, co sugeruje, że metoda zbiega do maksimum zamiast minimum. Dlatego dla drugiego zestawu punktów startowych metoda nie działa poprawnie, co może wynikać z błędnej trajektorii kierującej do maksimum zamiast minimum.

- Warto zwrócić uwagę na to jak bardzo x_1 w drugim zestawie jest przesunięty do maximum lokalnego. Możliwym jest, że przeszukiwany przedział początkowy był za szeroki i za blisko innego extremum. Lepsze nakierowanie na konkretne minimu może okazać się pomocne i bezpośrednio przedładać się na skuteczność algorytmu.

- Dla obu zestawów kolejne x_m bardzo szybko dążą do najbliższego extremum. Już po 3 iteracjach punkty zaznaczone na wykresie prawie przestają się zmieniać.

2. **Wolnozbieżność metody dla funkcji $f_2(x) = x^6$:**

- Dla funkcji $f_2(x) = x^6$, metoda interpolacji kwadratowej Powella może być wolnozbieżna ze względu na wysoki stopień wielomianu. W przypadku funkcji o wyższych stopniach, metoda może wymagać większej liczby iteracji, aby osiągnąć minimum.

3. **Najodpowiedniejszy warunek stopu:**

- Najodpowiedniejszym warunkiem stopu może być zatrzymanie iteracji po osiągnięciu określonej liczby iteracji lub po spełnieniu warunku, że różnica między kolejnymi przybliżeniami jest mniejsza niż określona wartość (np. ϵ).

4. **Rozstrzyganie o istnieniu minimum/maksimum:**

- W przypadku funkcji $f_2(x) = x^6$, aby stwierdzić, czy dany punkt jest minimum czy maksimum, możemy skorzystać z wartości drugiej pochodnej. Dla minimum druga pochodna będzie dodatnia, a dla maksimum będzie ujemna. W przypadku funkcji $f_2(x)$, druga pochodna w każdym punkcie będzie dodatnia, co oznacza, że funkcja $f_2(x)$ nie ma maksimów lokalnych ani globalnych, a jedynie minimum globalne w punkcie $x = 0$.

6. Wnioski

6.1. Złożoność obliczeniowa:

- Metoda Powella jest algorytmem iteracyjnym, który w każdej iteracji wykonuje obliczenia w oparciu o interpolację kwadratową.
- Złożoność obliczeniowa tej metody zależy głównie od liczby iteracji oraz od złożoności obliczeniowej samej funkcji celu.
- W praktyce złożoność czasowa może być trudna do oszacowania ze względu na zależność od konkretnego problemu optymalizacji.

6.2. Użyteczność:

- Metoda Powella jest szeroko stosowana w różnych dziedzinach, zarówno w naukach ścisłych, jak i w praktyce inżynierskiej czy ekonomicznej.
- Jest to algorytm bezgradientowy, co oznacza, że może być używany w przypadku funkcji, dla których gradient nie jest łatwo dostępny lub nie istnieje.
- Jednakże, ze względu na swój charakter iteracyjny, może wymagać większej liczby iteracji niż metody gradientowe w celu znalezienia optymalnego rozwiązania.

6.3. Błędy i ograniczenia:

- Metoda Powella może być podatna na błędy numeryczne, szczególnie w przypadku funkcji, które są źle uwarunkowane lub posiadają "wąskie gardła" w przestrzeni parametrów.
- Metoda Powella okazała się być wrażliwą na dobór punktów początkowych co było szczególnie widoczne w przypadku 2 zestawów dla funkcji $f_1(x)$.
- Odpowiedni dobór punktów początkowych oraz parametrów kontrolnych (takich jak liczba iteracji i tolerancja) jest kluczowy dla skuteczności algorytmu.
- Istnieje ryzyko utknięcia w minimum lokalnym, szczególnie w przypadku funkcji wielowymiarowych, co może ograniczać użyteczność metody.

6.4. Optymalizacje i ulepszenia:

- Istnieją różne warianty metody Powella oraz techniki ulepszające jej wydajność i dokładność, takie jak modyfikacje kryteriów zatrzymania czy adaptacyjne wybieranie kierunków.
- Wybór odpowiednich strategii aktualizacji punktów oraz optymalizacji parametrów może znacząco wpłynąć na efektywność algorytmu.

W przypadku praktycznego zastosowania metody Powella zaleca się przeprowadzenie eksperymentów oraz analizy wyników pod kątem efektywności, dokładności oraz stabilności algorytmu dla konkretnego problemu optymalizacji.