METODY OBLICZENIOWE W NAUCE I TECHNICE

Laboratorium 5 | Zagadnienie aproksymacji średniokwadratowej wielomianami algebraicznymi

Jakub Kaliński | Informatyka | rok II
Grupa numer 5 | Piątek | Godzina 15:00 - 16:30

3 maja 2025

1 Treść i przebieg zadania

1.1 Cel ćwiczenia

Celem ćwiczenia było wyznaczenie przybliżenia zadanej funkcji, wykorzystując **aproksymację średniokwadratową wielomianami algebraicznymi** oraz analizę dokładności tego przybliżenia w zależności od liczby punktów dyskretyzacji (n) i stopnia wielomianu aproksymującego (m).

1.2 Zadana funkcja i przedział

Analizowaną funkcją jest:

$$f(x) = \sin\left(\frac{kx}{\pi}\right) e^{\frac{-mx}{\pi}} \quad \text{gdzie } k = 4.0, \text{ oraz } m = 0.4$$

Funkcja ta była badana w przedziale $[a, b]$, gdzie $a = -2\pi^2$ oraz $b = \pi^2$. Wartości stałych k, m, a, b oraz π są zdefiniowane w pliku `function.c` i dostępne globalnie w projekcie poprzez `common.h`.

1.3 Przebieg ćwiczenia

Ćwiczenie obejmowało następujące kroki zrealizowane w środowisku C, z wykorzystaniem skryptu Pythona i Gnuplot do wizualizacji:

1. **Implementacja podstawowych modułów (C):** Stworzono funkcje do obliczania wartości funkcji $f(x)$ (`function.c`), generowania węzłów (`nodes.c`), obliczania błędów (`error.c`), operacji wejścia/wyjścia (`fileio.c`) oraz rozwiązywania układów równań liniowych (`linear_algebra.c`).

2. **Dyskretyzacja funkcji (C):** Dla różnych wartości n (liczby punktów dyskretyzacji), od $n = 2$ do zadanej przez użytkownika wartości N_{max}, wyznaczono n punktów (x_i, y_i) w przedziale $[a, b]$, gdzie $y_i = f(x_i)$. Zgodnie z implementacją (`nodes.c`, `main.c`), zastosowano **równomierne rozmieszczenie** punktów x_i (generowane przez funkcję `uniformNodes`). Wygenerowane punkty (x_i, y_i) dla każdego n zapisano do osobnych plików `.dat` w katalogu `data/`.

3. **Aproksymacja średniokwadratowa (C):** Dla każdej liczby punktów n oraz dla różnych stopni wielomianu m (gdzie $0 \leq m < n$ i $m \leq M_{max}$ zadanego przez użytkownika), wyznaczono wielomian algebraiczny $P_m(x) = a_0 + a_1 x + \cdots + a_m x^m$, który minimalizuje sumę kwadratów błędów w punktach dyskretyzacji:

$$S(a_0, \ldots, a_m) = \sum_{i=0}^{n-1} (P_m(x_i) - y_i)^2 \to \min$$

Zadanie to sprowadza się do znalezienia wektora współczynników $\mathbf{a} = [a_0, a_1, \ldots, a_m]^T$, które minimalizują powyższą sumę S.

Podejście teoretyczne (Równania normalne): Minimalizacja S względem każdego współczynnika a_j (poprzez przyrównanie pochodnych cząstkowych $\partial S/\partial a_j$ do zera) prowadzi do układu $m+1$ równań liniowych, znanego jako **układ równań normalnych**:

$$\mathbf{Ga} = \mathbf{B}$$

gdzie:

- $\mathbf{G}$ jest symetryczną, dodatnio półokreśloną macierzą Grama o wymiarach $(m+1)\times(m+1)$, której elementy są dane wzorem:

$$G_{jk} = \sum_{i=0}^{n-1} x_i^{j+k} \quad (\text{dla } j,k = 0,1,\ldots,m)$$

- $\mathbf{a} = [a_0, a_1, \ldots, a_m]^T$ jest wektorem szukanych współczynników wielomianu.
- $\mathbf{B}$ jest wektorem prawych stron o wymiarach $(m+1)\times 1$, którego elementy są dane wzorem:

$$B_j = \sum_{i=0}^{n-1} y_i x_i^j \quad (\text{dla } j = 0,1,\ldots,m)$$

Rozwiązanie tego układu równań daje szukane współczynniki $\mathbf{a}$.

Podejście praktyczne (Implementacja w C): W dostarczonej implementacji C (funkcja `leastSquaresApprox` w `approximation.c`):

- **Jawnie konstruowana jest macierz Grama $\mathbf{G}$ oraz wektor prawych stron $\mathbf{B}$** zgodnie z powyższymi wzorami. Zastosowano drobne optymalizacje dla potęg x^0 i x^1.
- **Układ równań $\mathbf{Ga} = \mathbf{B}$ jest rozwiązywany przy użyciu funkcji `gaussianElimination` (zaimplementowanej w `linear_algebra.c`)**. Funkcja ta stosuje metodę eliminacji Gaussa z częściowym wyborem elementu głównego (pivotingiem) w celu poprawy stabilności numerycznej.
- W przypadku, gdy eliminacja Gaussa napotka problem (np. macierz $\mathbf{G}$ okaże się osobliwa lub bliska osobliwej, co jest sygnalizowane przez wartość zwracaną -1), funkcja `leastSquaresApprox` również zwraca błąd.

Należy podkreślić, że bezpośrednie tworzenie i rozwiązywanie układu równań normalnych, nawet z pivotingiem, może być **niestabilne numerycznie**, szczególnie dla wielomianów wysokiego stopnia m. Macierz $\mathbf{G}$ staje się wtedy ***źle uwarunkowana***.

Definicja: Liczba uwarunkowania macierzy

Liczba uwarunkowania macierzy $\mathbf{A}$, oznaczana jako $\kappa(\mathbf{A})$, jest miarą określającą, jak bardzo wrażliwe jest rozwiązanie układu równań liniowych $\mathbf{Ax} = \mathbf{b}$ na małe względne zmiany (błędy) w danych wejściowych, czyli w macierzy $\mathbf{A}$ lub wektorze $\mathbf{b}$. Formalnie, jest ona często definiowana jako $\kappa(\mathbf{A}) = ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}||$, gdzie $||\cdot||$ oznacza wybraną normę macierzową.
Interpretacja:

- $\kappa(\mathbf{A}) \approx 1$: Macierz jest **dobrze uwarunkowana**. Małe błędy wejściowe prowadzą do małych błędów w rozwiązaniu. Problem jest stabilny numerycznie.
- $\kappa(\mathbf{A}) \gg 1$: Macierz jest ***źle uwarunkowana***. Małe błędy wejściowe mogą zostać znacznie wzmocnione, prowadząc do dużych błędów w rozwiązaniu. Problem jest niestabilny numerycznie, a obliczenia mogą prowadzić do utraty precyzji (orientacyjnie, można stracić $\log_{10}(\kappa(\mathbf{A}))$ cyfr znaczących). Macierze Grama dla wielomianów wysokiego stopnia są typowo źle uwarunkowane.

Po uzyskaniu współczynników $\mathbf{a}$, wartości wielomianu $P_m(x)$ dla dowolnych x (np. dla punktów do rysowania wykresu) są obliczane za pomocą wydajnego schematu Hornera (zaimplementowanego w funkcji `evaluatePolynomial` w `approximation.c`). Obliczone wartości $P_m(x)$ na gęstej siatce również zapisano do plików `.dat`.

4. **Eksperymenty numeryczne (C):** Program główny (`main.c`) przeprowadził serię obliczeń, iterując przez liczbę punktów n (od $n = 2$ do N_{max}) oraz stopień wielomianu m (od $m = 0$ do $\min(n-1, M_{max})$).

5. **Ocena błędów (C):** Dla każdej pary (n, m), dla której udało się obliczyć współczynniki, program obliczył błędy aproksymacji, porównując wartości wielomianu $P_m(x)$ z wartościami oryginalnej funkcji $f(x)$ na gęstej siatce $N_{plot} = 1000$ punktów w przedziale $[a, b]$ (funkcja `calculateError` w `error.c`). Użyto dwóch metryk:
   - Błąd maksymalny bezwzględny ($E_m = \max |P_m(x) - f(x)|$).
   - Błąd średniokwadratowy (MSE = $\frac{1}{N_{plot}} \sum (P_m(x_j) - f(x_j))^2$).

   Wyniki błędów (wraz z n i m) zostały zapisane w formacie CSV do pliku `data/approximation_heatmap_errors.csv` (funkcja `appendErrorToHeatmapFile`). W przypadku niepowodzenia obliczeń dla danej pary (n, m), zapisywano wartości 'NAN'.
6. **Generowanie skryptów wizualizacyjnych (C):** Program C wygenerował również skrypt Gnuplot (`scripts/plot_all_approximations.gp`) zawierający pętle 'do for', który umożliwia automatyczne wygenerowanie indywidualnych wykresów porównawczych dla wszystkich par (n, m).
7. **Wizualizacja wyników (Python + Gnuplot):** Wykorzystano dwa narzędzia:
   - **Python (skrypt `plot_heatmaps.py`):** Skrypt ten wczytuje dane błędów z pliku CSV i generuje **heatmapy** pokazujące zależność błędów E_m i MSE od obu parametrów (n i m). Skrypt zapisuje heatmapy (w tym wersje z adnotacjami dla mniejszego zakresu) do katalogu `plots/`.
   - **Gnuplot:** Uruchomienie skryptu `scripts/plot_all_approximations.gp` (wygenerowanego przez C) tworzy **indywidualne wykresy** porównujące funkcję $f(x)$, wielomian $P_m(x)$ oraz punkty dyskretyzacji (x_i, y_i) dla każdej pary (n, m). Wykresy te są zapisywane jako pliki `.png` w katalogu `plots/`.

Celem analizy było zrozumienie, jak liczba punktów dyskretyzacji i stopień wielomianu wpływają na jakość i stabilność aproksymacji średniokwadratowej, szczególnie w kontekście użycia metody równań normalnych.

2 Dane techniczne sprzętu

Do wykonania zadania wykorzystany został komputer o poniższej specyfikacji:

- System operacyjny: *Windows 10 x64* (lub inny, na którym uruchomiono kod)
- Procesor: *Intel Core i7-11370H* o taktowaniu *3.30GHz* (lub inny)
- Pamięć RAM: *16GB* (lub inna)

3 Dane techniczne oprogramowania

3.1 System operacyjny

- Windows 10 (lub Linux/macOS, jeśli używano)

3.2 Narzędzia programistyczne i wykonawcze

- **Język implementacji obliczeń:** *C*
- **Kompilator C:** *GCC* (zgodnie z Makefile, np. wersja MinGW lub systemowa)
- **System budowania:** *Make* (do zarządzania kompilacją i uruchamianiem)
- **Narzędzie do wizualizacji (heatmapy):** *Python 3* (np. wersja 3.x)
- **Narzędzie do wizualizacji (wykresy indywidualne):** *Gnuplot* (np. wersja 5.x)

3.3 Użyte biblioteki i moduły

- **C:** Standardowe biblioteki C (`stdio.h`, `stdlib.h`, `math.h`, `string.h`). Brak zewnętrznych bibliotek C.
- **Python (dla skryptu `plot_heatmaps.py`):**
  - `pandas` – do wczytywania i manipulacji danymi z pliku CSV.
  - `numpy` – do operacji numerycznych (np. obsługa NaN/Inf).
  - `matplotlib.pyplot` – do tworzenia i zarządzania figurami wykresów.
  - `seaborn` – do generowania estetycznych heatmap.
  - `os`, `sys` – do operacji na ścieżkach plików i systemie.

3.4 Wizualizacja

Wizualizację wyników zrealizowano dwutorowo:

- **Heatmapy błędów:** Generowane przez skrypt Python (`plot_heatmaps.py`) przy użyciu bibliotek `matplotlib` i `seaborn`, na podstawie danych z pliku CSV.
- **Indywidualne wykresy aproksymacji:** Generowane przez `Gnuplot` na podstawie skryptu (`plot_all_approximat` automatycznie wygenerowanego przez program C. Skrypt ten odczytuje dane funkcji, aproksymacji i węzłów z plików `.dat`.

4 Wyznaczenie dokładności aproksymacji

W celu wyznaczenia dokładności, z jaką wielomian aproksymujący $P_m(x)$ przybliża zadaną funkcję $f(x)$, wykorzystano poniższe metryki błędów, obliczane przez program C (`error.c`) na gęstej siatce $N_{plot} = 1000$ punktów w przedziale $[a, b]$.

4.1 Błąd maksymalny (norma supremum)

Błąd maksymalny bezwzględny, zwany też błędem w normie supremum (L_∞), jest zdefiniowany jako:

$$E_m = \max_{x\in[a,b]} |P_m(x) - f(x)| \approx \max_{j=0,\dots,N_{plot}-1} |P_m(x_j^{plot}) - f(x_j^{plot})| \tag{1}$$

gdzie x_j^{plot} to punkty z gęstej siatki. Wskazuje on największe odchylenie aproksymacji od funkcji oryginalnej w całym przedziale.

4.2 Błąd średniokwadratowy (MSE)

Błąd średniokwadratowy (Mean Squared Error) mierzy średnią wartość kwadratu różnicy między aproksymacją a funkcją oryginalną na gęstej siatce:

$$MSE = \frac{1}{N_{plot}} \sum_{j=0}^{N_{plot}-1} (P_m(x_j^{plot}) - f(x_j^{plot}))^2 \tag{2}$$

Jest to miara średniego błędu aproksymacji. Czasami używa się też pierwiastka z MSE (RMSE).

4.3 Interpretacja wyników

Analiza obu wskaźników błędu w zależności od liczby punktów dyskretyzacji n oraz stopnia wielomianu m pozwala ocenić jakość i stabilność procesu aproksymacji:

- Spadek błędów wraz ze wzrostem m (dla ustalonego n) wskazuje na poprawę dopasowania wielomianu do danych.
- Zbyt duży stopień m (bliski $n-1$) może prowadzić do problemów numerycznych związanych ze złym uwarunkowaniem macierzy Grama $\mathbf{G}$. Nawet jeśli uda się rozwiązać układ równań, wynikowy wielomian może wykazywać duże oscylacje między punktami próbkowania, co prowadzi do wzrostu błędów E_m i MSE na gęstej siatce.

- Wzrost błędów dla bardzo dużych m jest sygnałem problemów numerycznych (złe uwarunkowanie macierzy Grama) i/lub zjawiska przeuczenia (overfittingu).
- Wpływ liczby punktów n: Zwiększanie n generalnie powinno poprawiać jakość aproksymacji (dla odpowiednio dobranego m), dostarczając więcej informacji o funkcji i potencjalnie poprawiając uwarunkowanie problemu dla danego m.

Heatmapy błędów pozwalają na szybką wizualizację tych zależności i identyfikację obszarów (n, m), gdzie aproksymacja jest najlepsza lub gdzie pojawiają się problemy numeryczne.

5 Złe uwarunkowanie i stabilność numeryczna

Podobnie jak w interpolacji wielomianowej, również w aproksymacji średniokwadratowej wielomianami algebraicznymi wysokiego stopnia pojawia się problem złego uwarunkowania, szczególnie przy użyciu standardowej bazy potęgowej $(1, x, x^2, \ldots, x^m)$.

5.1 Przyczyna problemu

Problem wynika z faktu, że kolumny macierzy Grama $\mathbf{G}$, której elementy to sumy x_i^{j+k}, stają się "prawie"liniowo zależne dla dużych m. Oznacza to, że macierz $\mathbf{G}$ staje się **źle uwarunkowana** (jej liczba uwarunkowania $\kappa(\mathbf{G})$ jest bardzo duża). Jest to szczególnie widoczne przy równomiernym rozmieszczeniu punktów x_i. Złe uwarunkowanie oznacza, że małe błędy numeryczne (wynikające z ograniczonej precyzji arytmetyki zmiennoprzecinkowej) lub małe zmiany w danych wejściowych (x_i, y_i) mogą prowadzić do dużych zmian w obliczonych współczynnikach wielomianu a podczas rozwiązywania układu $\mathbf{Ga} = \mathbf{B}$.

5.2 Skutki

Skutkiem złego uwarunkowania macierzy $\mathbf{G}$ mogą być:

- **Niepowodzenie rozwiązania układu:** Metoda eliminacji Gaussa (nawet z pivotingiem) może napotkać dzielenie przez bardzo małą liczbę (bliską zeru numerycznemu) i zwrócić błąd osobliwości (jak zaimplementowano w `gaussianElimination`). W takim przypadku program C zapisuje 'NAN' jako wartości błędów.
- **Niedokładne współczynniki:** Nawet jeśli uda się uzyskać rozwiązanie, obliczone współczynniki a_k mogą mieć duże błędy numeryczne.
- **Duże błędy aproksymacji:** Wielomian z niedokładnymi współczynnikami może wykazywać duże oscylacje między punktami próbkowania, co prowadzi do dużego błędu maksymalnego E_m i MSE na gęstej siatce, mimo że formalnie minimalizuje on sumę kwadratów w punktach x_i.
- **Wrażliwość na dane:** Niewielka zmiana jednego punktu (x_i, y_i) może drastycznie zmienić wielomian aproksymujący wysokiego stopnia.

Zjawisko to jest związane z efektem Runge'go znanym z interpolacji, choć mechanizm (minimalizacja sumy kwadratów vs. przejście przez punkty) i manifestacja (problemy numeryczne i globalne dopasowanie vs. oscylacje głównie na krańcach) są nieco inne.

5.3 Rozwiązania (niezaimplementowane w tym kodzie C)

Problem złego uwarunkowania w aproksymacji można łagodzić przez:

- **Ograniczenie stopnia wielomianu m:** Wybór stopnia znacznie mniejszego niż liczba punktów n. Jest to podstawowa strategia stosowana w praktyce.
- **Użycie baz ortogonalnych:** Zastosowanie wielomianów ortogonalnych (np. Legendre'a, Czebyszewa) jako funkcji bazowych zamiast $1, x, x^2, \ldots$ prowadzi do znacznie lepiej uwarunkowanych (często diagonalnych lub bliskich diagonalnym) macierzy układu równań.

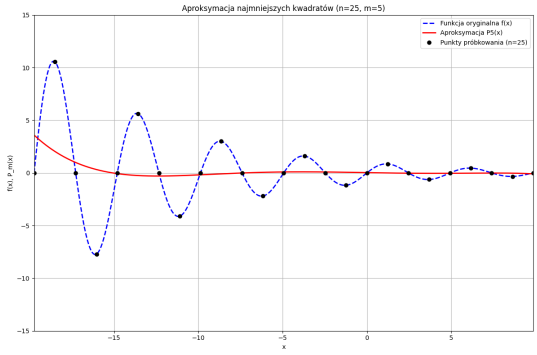
- **Stosowanie stabilnych metod numerycznych na macierzy Vandermonde'a:** Zamiast tworzyć macierz Grama $\mathbf{G}$, można sformułować problem jako minimalizację $||\mathbf{Aa} - \mathbf{y}||_2^2$, gdzie $\mathbf{A}$ jest macierzą Vandermonde'a (lub macierzą projektu). Problem ten można rozwiązać za pomocą numerycznie stabilnych metod, takich jak rozkład QR lub SVD (Singular Value Decomposition) macierzy $\mathbf{A}$. Takie podejście jest stosowane np. w funkcji `numpy.linalg.lstsq` w Pythonie. Jest ono generalnie bardziej odporne na złe uwarunkowanie niż jawne rozwiązywanie układu równań normalnych.
- **Regularyzacja:** Dodanie członów regularyzacyjnych (np. L1, L2) do minimalizowanej funkcji błędu, co ogranicza wielkość współczynników i stabilizuje rozwiązanie.

W przeprowadzonym eksperymencie C zastosowano metodę równań normalnych z eliminacją Gaussa i częściowym pivotingiem. Głównym mechanizmem radzenia sobie z niestabilnością jest ograniczenie stopnia m w stosunku do n, co jest widoczne w analizie wyników.

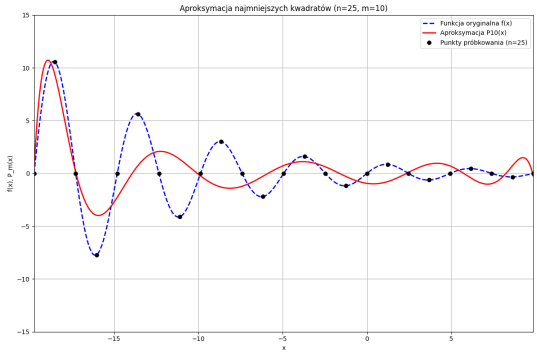
6 Przykładowe wykresy aproksymacji

Poniższe rysunki ilustrują wyniki aproksymacji dla wybranych kombinacji liczby punktów n i stopnia wielomianu m. Wykresy zostały wygenerowane przez Gnuplot na podstawie danych i skryptu z programu C. Porównują one funkcję oryginalną $f(x)$ (niebieska linia przerywana), wielomian aproksymujący $P_m(x)$ (czerwona linia ciągła) oraz punkty dyskretyzacji (x_i, y_i) (czarne punkty).

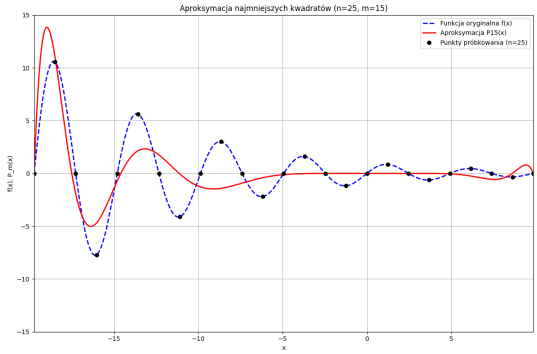
Liczba punktów n = 25



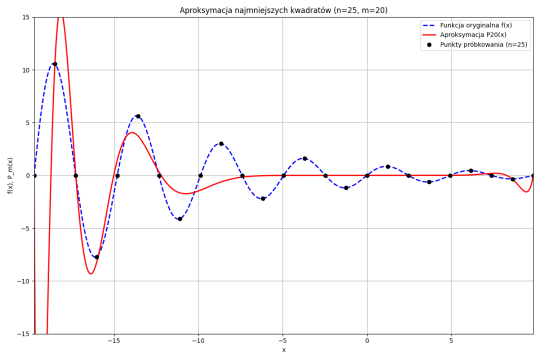
(a) Stopień m = 5



(b) Stopień m = 10

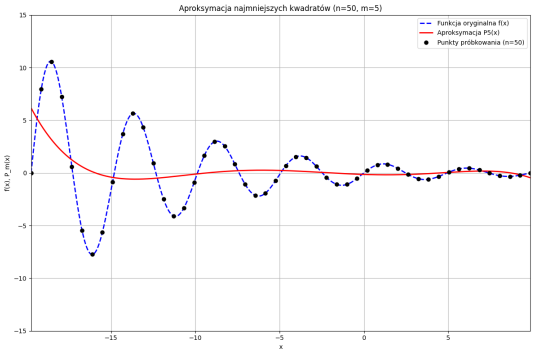


(c) Stopień m = 15

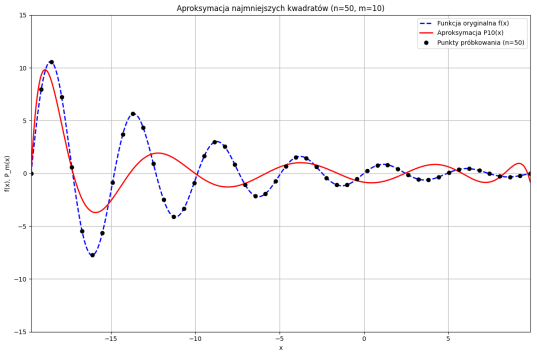


(d) Stopień m = 20

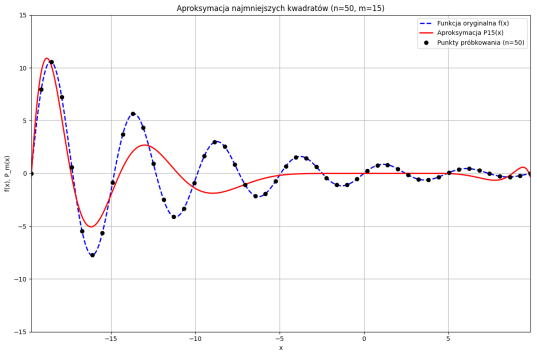
Liczba punktów n = 50



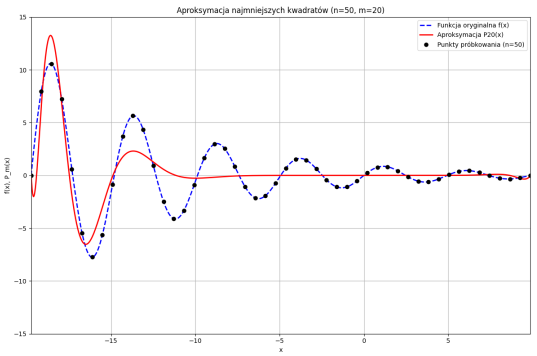
(e) Stopień m = 5



(f) Stopień m = 10



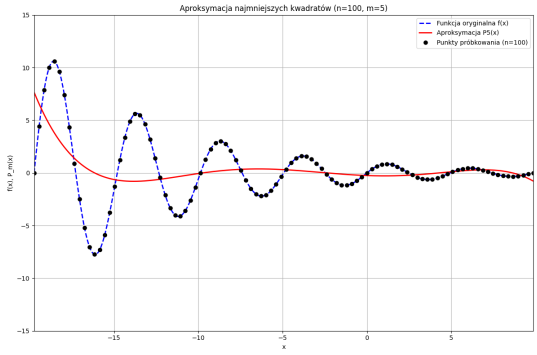
(g) Stopień m = 15



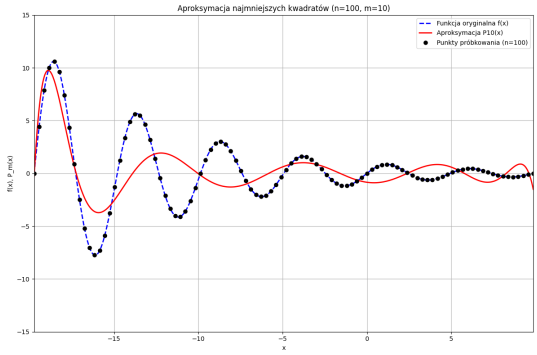
(h) Stopień m = 20

Rysunek 1: Porównanie aproksymacji dla $n = 25$ (lewa kolumna) i $n = 50$ (prawa kolumna) przy różnych stopniach m.

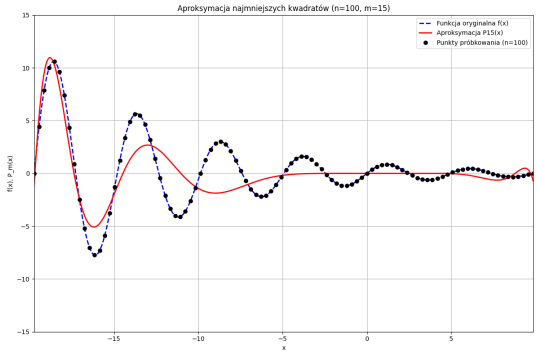
Liczba punktów n = 100



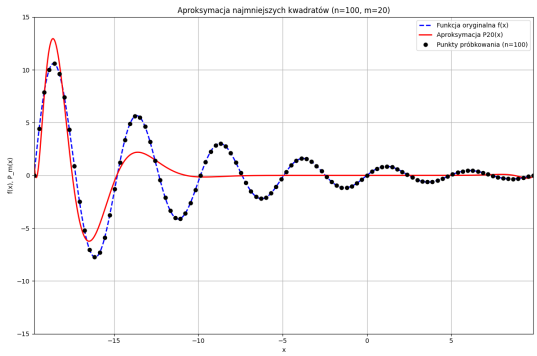
(a) Stopień m = 5



(b) Stopień m = 10

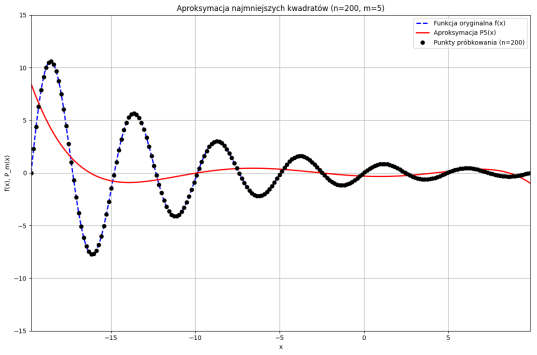


(c) Stopień m = 15

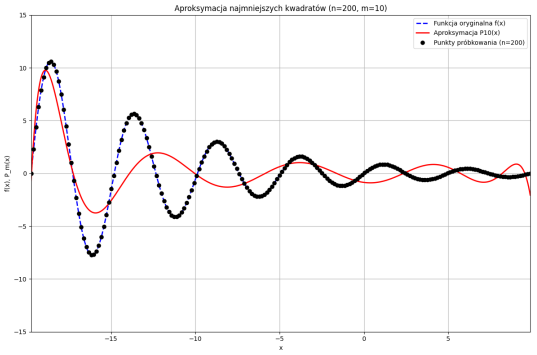


(d) Stopień m = 20

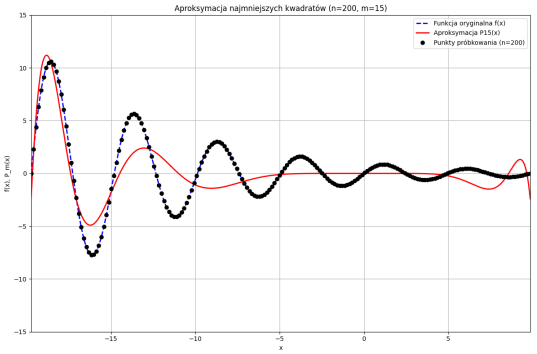
Liczba punktów n = 200



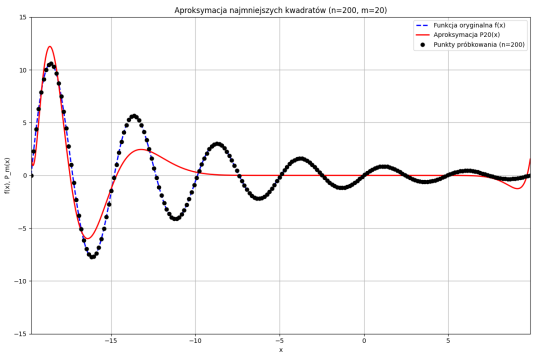
(e) Stopień m = 5



(f) Stopień m = 10



(g) Stopień m = 15

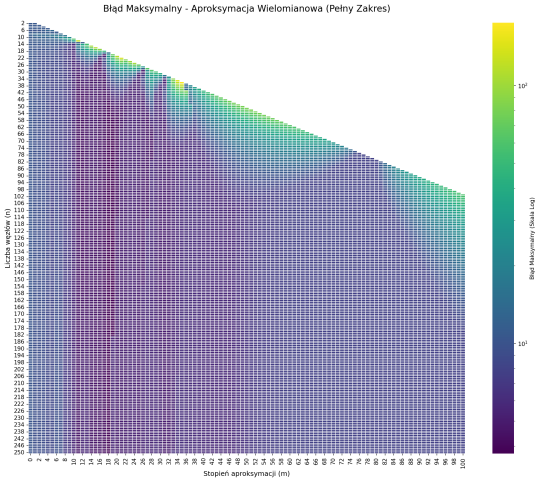


(h) Stopień m = 20

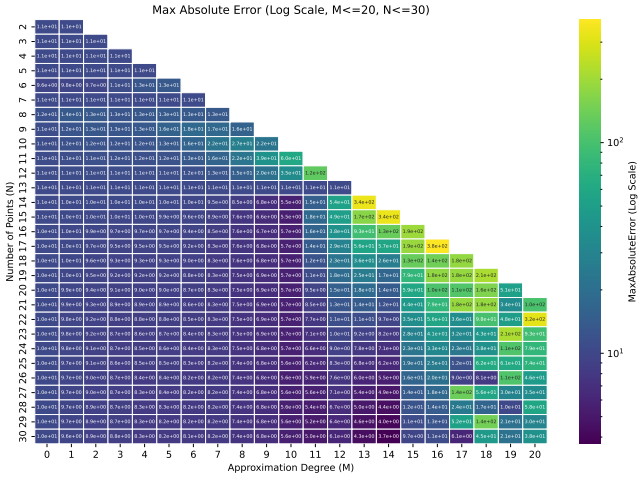
Rysunek 2: Porównanie aproksymacji dla $n = 100$ (lewa kolumna) i $n = 200$ (prawa kolumna) przy różnych stopniach m.

7 Analiza błędów aproksymacji - Heatmapy

Zbiorczą analizę wpływu obu parametrów (n i m) na jakość aproksymacji umożliwiają poniższe mapy ciepła (heatmapy), wygenerowane przez skrypt Python na podstawie danych z pliku CSV. Kolory reprezentują wartość błędu w skali logarytmicznej – jaśniejsze kolory oznaczają mniejszy błąd, ciemniejsze - większy. **Oś X reprezentuje stopień wielomianu (M), a oś Y liczbę punktów (N).**

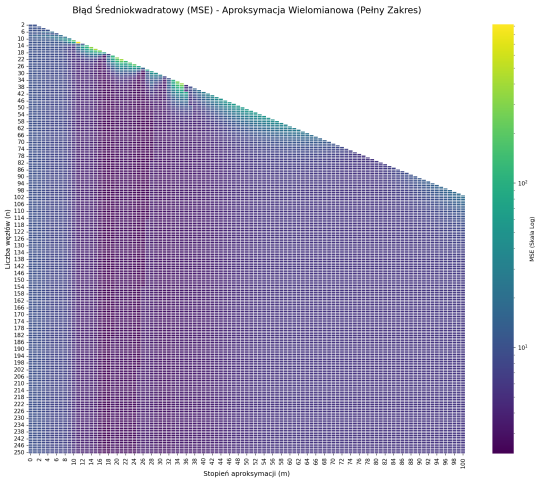


(a) Błąd Maksymalny E_m (Pełny Zakres)

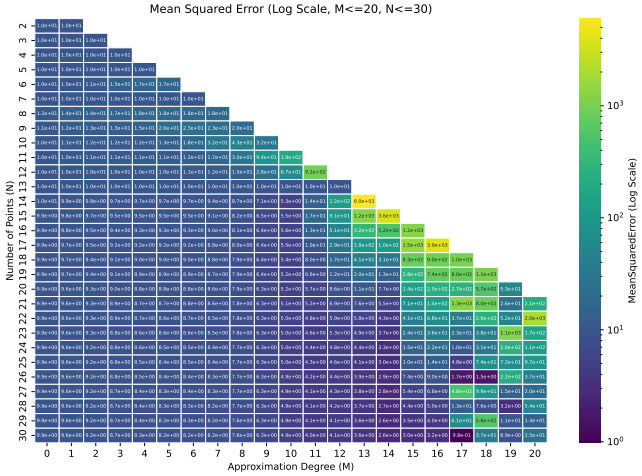


(b) Błąd Maksymalny E_m (N $\leq$ 30, M $\leq$ 20)

Rysunek 3: Heatmapy Błędu Maksymalnego E_m (skala log) w zależności od liczby punktów n (oś Y) i stopnia wielomianu m (oś X).



(a) Błąd Średniokwadratowy MSE (Pełny Zakres)



(b) Błąd Średniokwadratowy MSE (N $\leq$ 30, M $\leq$ 20)

Rysunek 4: Heatmapy Błędu Średniokwadratowego (MSE) (skala log) w zależności od liczby punktów n (oś Y) i stopnia wielomianu m (oś X).

7.1 Interpretacja wyników i obserwacje

Analiza heatmap oraz danych liczbowych z pliku CSV prowadzi do następujących wniosków:

- **Wpływ stopnia wielomianu m (dla stałego n):** Dla ustalonej liczby punktów n, zwiększanie stopnia wielomianu m (ruch w prawo na heatmapie) początkowo prowadzi do poprawy dokładności przybliżenia (spadek wartości błędów, jaśniejsze kolory). Widać to wyraźnie na heatmapach z adnotacjami (Rysunek 3b i 4b) oraz na heatmapach pełnego zakresu dla niższych n. Jednak po przekroczeniu pewnej optymalnej wartości $m_{opt}(n)$, dalsze zwiększanie m prowadzi do gwałtownego wzrostu błędu, zwłaszcza błędu maksymalnego E_m. Na heatmapach pełnego zakresu objawia się to nagłym

przejściem do bardzo ciemnych kolorów w prawej części wykresu dla danej wartości n. Jest to spowodowane złym uwarunkowaniem macierzy Grama $\mathbf{G}$ i niestabilnością rozwiązania układu równań normalnych dla wysokich stopni wielomianów. Program C może w takich przypadkach zwrócić błąd lub obliczyć niedokładne współczynniki, prowadząc do dużych błędów aproksymacji.

- **Wpływ liczby punktów n (dla stałego m):** Przy ustalonym stopniu wielomianu m (ruch w górę na heatmapie), zwiększanie liczby punktów n zazwyczaj poprawia dokładność aproksymacji (jaśniejsze kolory). Dostarczenie większej liczby danych o funkcji pozwala na lepsze "uśrednienie"jej przebiegu przez wielomian i potencjalnie stabilizuje problem dla danego m. Jednak dla bardzo wysokich m, gdzie dominują problemy numeryczne, zwiększanie n może nie przynieść już znaczącej poprawy lub zależność staje się nieregularna.
- **Złe uwarunkowanie i stabilność numeryczna:** Heatmapy wyraźnie ilustrują problem złego uwarunkowania związanego z rozwiązywaniem układu równań normalnych dla bazy potęgowej. Gwałtowny wzrost błędów dla m zbliżającego się do n (prawy górny róg obszaru danych na heatmapach, często z wartościami błędów rzędu 10^2 lub większymi, lub oznaczony jako 'NAN' w pliku CSV, co skutkuje białymi plamami na heatmapie) jest tego bezpośrednim skutkiem. Metoda eliminacji Gaussa z częściowym pivotingiem, choć próbuje stabilizować proces, nie jest w stanie poradzić sobie z inherentnie złym uwarunkowaniem macierzy Grama dla wysokich m. Skutkuje to albo niepowodzeniem rozwiązania, albo uzyskaniem numerycznie bezużytecznych współczynników.
- **Obszar najlepszego dopasowania:** Najniższe wartości błędów (najjaśniejsze obszary na heatmapach) koncentrują się w regionie, gdzie liczba punktów n jest stosunkowo duża (np. $n > 50 - 100$), a stopień wielomianu m jest umiarkowany (znacznie mniejszy od n). Na podstawie przedstawionych heatmap (szczególnie dla MSE), wydaje się, że optymalne stopnie m dla badanej funkcji i zakresu n do 250 leżą w przybliżeniu w zakresie $m \in [15, 35]$. Dokładne wartości najlepszych parametrów n i m zostały zidentyfikowane na podstawie danych liczbowych.
- **Porównanie metryk błędów:** Heatmapa MSE (Rysunek 4) wydaje się być nieco "gładsza"niż heatmapa błędu maksymalnego E_m (Rysunek 3). Gwałtowne wzrosty błędów i efekty numeryczne dla wysokich m są często bardziej dramatyczne na heatmapie E_m, ponieważ błąd maksymalny jest bardziej czuły na lokalne, duże odchylenia wielomianu od funkcji. MSE, jako miara średnia, lepiej oddaje ogólną jakość dopasowania na całym przedziale.

Analiza danych liczbowych z pliku `approximation_heatmap_errors.csv` potwierdza obserwacje z heatmap. Najlepsze uzyskane przybliżenie w normie supremum (E_m) znaleziono dla $n = 211$ i $m = 26$, z błędem maksymalnym rzędu 1.09×10^{-6}. Natomiast najmniejszy błąd średniokwadratowy (MSE) osiągnięto również dla $n = 211$, ale przy stopniu $m = 28$, uzyskując ekstremalnie małą wartość 2.15×10^{-15}. Pokazuje to, że optymalny wybór parametrów może nieznacznie zależeć od przyjętej metryki błędu, ale w obu przypadkach najlepsze wyniki uzyskano dla dużej liczby węzłów n i umiarkowanego stopnia wielomianu m, co jest zgodne z oczekiwaniami teoretycznymi dotyczącymi stabilności aproksymacji.

8 Tabela wyników aproksymacji

Poniższa tabela przedstawia wybrane wyniki aproksymacji dla różnych kombinacji liczby punktów dyskretyzacji n i stopnia wielomianu aproksymującego m. Pokazano wartości błędu maksymalnego (E_m) oraz błędu średniokwadratowego (MSE), odczytane z pliku CSV wygenerowanego przez program C. Wybrano reprezentatywne wartości n oraz stopnie m ilustrujące trend: od niskich stopni, przez zakres uznany za bliski optymalnemu (okolice $m \approx 20 - 35$), aż po wyższe stopnie, gdzie pojawiają się problemy numeryczne (duże błędy lub wartości 'NAN' - oznaczone jako '—').

Tabela 1: Błędy aproksymacji (E_m i MSE) dla wybranych n i m.

L. pkt n	**Stopień m**	**Błąd Maks E_m**	**Błąd MSE**
25	5	8.51	8.42
	10	2.52e−1	1.84e−2
	15	1.13e−2	3.53e−5
	20	5.78e−4	1.02e−7

Ciąg dalszy na następnej stronie

Tabela 1 – ciąg dalszy z poprzedniej strony

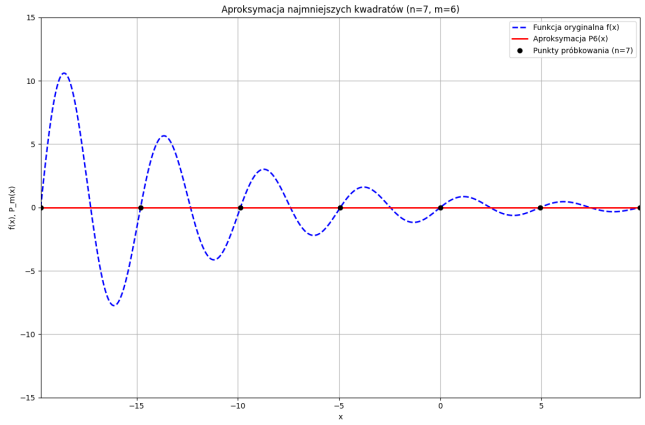
L. pkt n	**Stopień** m	**Błąd Maks** E_m	**Błąd MSE**
	24	1.03e2	1.93e2
	25	—	—
50	5	8.81	8.75
	10	1.21e−1	4.46e−3
	15	1.16e−3	3.45e−7
	20	5.21e−6	8.61e−12
	25	1.03e−6	2.11e−14
	30	8.09e−7	1.20e−14
	40	7.11e−4	1.43e−7
	48	7.99e1	6.00e2
100	5	8.81	8.73
	10	1.23e−1	4.47e−3
	15	1.06e−3	2.94e−7
	20	4.13e−6	5.33e−12
	25	1.01e−6	1.96e−14
	30	3.92e−7	3.82e−15
	40	6.22e−6	1.06e−11
	98	3.19e2	1.81e3
150	5	8.81	8.73
	10	1.23e−1	4.47e−3
	15	1.06e−3	2.93e−7
	20	4.12e−6	5.29e−12
	25	1.01e−6	1.95e−14
	30	3.90e−7	3.78e−15
	40	6.18e−6	1.05e−11
	100	1.04e1	6.01
200	5	8.81	8.73
	10	1.23e−1	4.47e−3
	15	1.05e−3	2.93e−7
	20	4.12e−6	5.29e−12
	25	1.01e−6	1.95e−14
	30	3.89e−7	3.77e−15
	40	6.17e−6	1.04e−11
	100	7.75	5.83
211	5	8.81	8.73
	15	1.05e−3	2.93e−7
	20	4.12e−6	5.29e−12
	25	1.01e−6	1.95e−14
	26	1.09e−6	2.21e−14
	28	1.55e−6	2.15e−15
	30	3.89e−7	3.77e−15
	40	6.17e−6	1.04e−11
	100	7.75	5.79
250	5	8.81	8.73
	10	1.23e−1	4.47e−3
	15	1.05e−3	2.93e−7
	20	4.12e−6	5.29e−12
	25	1.01e−6	1.95e−14
	30	3.89e−7	3.77e−15
	40	6.17e−6	1.04e−11
	100	7.75	5.78

9 Przypadek szczególny: Punkty próbkowania w pobliżu miejsc zerowych

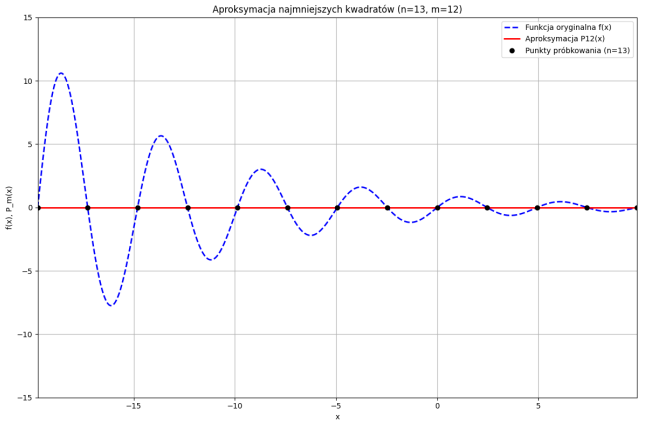
Podczas analizy wyników zaobserwowano szczególny przypadek dla pewnych wartości liczby punktów dyskretyzacji n, takich jak $n = 7$ i $n = 13$. Przy równomiernym rozmieszczeniu węzłów w przedziale $[-2\pi^2, \pi^2]$ (zgodnie z funkcją `uniformNodes`), dla tych konkretnych n, punkty próbkowania (x_i, y_i) trafiają bardzo blisko miejsc zerowych funkcji $f(x) = \sin(4x/\pi)\exp(-0.4x/\pi)$. W rezultacie, wartości $y_i = f(x_i)$ są bardzo bliskie zeru dla wszystkich $i = 0, \ldots, n-1$.

Gdy algorytm aproksymacji najmniejszych kwadratów (`leastSquaresApprox`) otrzymuje taki zestaw danych $(x_i, y_i \approx 0)$, stara się znaleźć wielomian $P_m(x)$, który minimalizuje sumę $\sum(P_m(x_i) - y_i)^2$. Ponieważ wszystkie y_i są bliskie zeru, wielomianem, który najlepiej minimalizuje tę sumę (szczególnie dla m bliskiego $n-1$, gdzie problem staje się zbliżony do interpolacji, jak w pokazanych przykładach $m = 6, n = 7$ oraz $m = 12, n = 13$), jest wielomian tożsamościowo równy zeru lub bardzo bliski zeru: $P_m(x) \approx 0$.

Ilustrują to poniższe wykresy (wygenerowane przez Gnuplot na podstawie danych z C):



(a) Aproksymacja dla $n = 7$ i $m = 6$. Punkty próbkowania (czarne kropki) leżą blisko osi OX, a wynikowy wielomian (czerwona linia) jest bliski zeru.



(b) Aproksymacja dla $n = 13$ i $m = 12$. Podobnie jak dla $n = 7$, punkty próbkowania trafiają w pobliże miejsc zerowych, prowadząc do aproksymacji $P_{12}(x) \approx 0$.

Rysunek 5: Ilustracja przypadku, gdy równomierne próbkowanie trafia w pobliże miejsc zerowych funkcji, prowadząc do trywialnej i błędnej aproksymacji.

Ten przypadek dobitnie pokazuje, że samo zwiększanie liczby punktów n lub stopnia wielomianu m nie gwarantuje dobrej aproksymacji, jeśli punkty próbkowania są rozmieszczone w sposób "niefortunny". Równomierne próbkowanie jest szczególnie podatne na takie sytuacje w przypadku funkcji oscylujących. Trafienie punktów w pobliże miejsc zerowych powoduje utratę informacji o amplitudzie i kształcie funkcji między tymi punktami, co prowadzi do całkowicie błędnej aproksymacji (o dużym błędzie maksymalnym i MSE w porównaniu do funkcji oryginalnej). Podkreśla to znaczenie strategii doboru węzłów w procesie aproksymacji – węzły powinny adekwatnie "próbkować"zmienność funkcji w całym przedziale.

10 Wnioski końcowe

Przeprowadzone eksperymenty z aproksymacją średniokwadratową funkcji $f(x) = \sin(4x/\pi)\exp(-0.4x/\pi)$ wielomianami algebraicznymi, zrealizowane w języku C z wykorzystaniem metody równań normalnych i eliminacji Gaussa, oraz wizualizowane za pomocą Pythona i Gnuplot, pozwalają sformułować następujące wnioski:

1. **Zależność od parametrów n i m:** Jakość aproksymacji silnie zależy od liczby punktów n i stopnia wielomianu m. Zwiększanie stopnia m dla ustalonego n poprawia dokładność tylko do pewnego momentu, po czym dominować zaczynają problemy związane z niestabilnością numeryczną metody równań normalnych. Zwiększanie liczby punktów n generalnie poprawia dokładność dla odpowiednio dobranego, umiarkowanego stopnia m.

2. **Optymalne parametry i najlepsza aproksymacja:** Dla badanej funkcji i równomiernych punktów, istnieje optymalny zakres stopni m, który minimalizuje błędy. Stosowanie stopni m bliskich $n-1$ jest wysoce niezalecane z powodu gwarantowanych problemów numerycznych (złe uwarunkowanie macierzy

Grama). Na podstawie przeprowadzonych symulacji (dla n do 250 i m do 100), najlepsze wyniki uzyskano dla $n = 211$. Najmniejszy błąd maksymalny ($E_m \approx 1.09 \times 10^{-6}$) osiągnięto dla $m = 26$, a najmniejszy błąd średniokwadratowy (MSE $\approx 2.15 \times 10^{-15}$) dla $m = 28$. Potwierdza to, że optymalny stopień jest znacznie niższy niż maksymalna możliwa wartość $n - 1$.

3. **Złe uwarunkowanie i stabilność numeryczna:** Problem złego uwarunkowania macierzy Grama $\mathbf{G}$ jest kluczowym ograniczeniem przy stosowaniu bazy potęgowej i metody równań normalnych dla wielomianów wysokiego stopnia. Jest to główna przyczyna gwałtownego wzrostu błędów (lub niepowodzenia obliczeń), gdy m zbliża się do n. Zastosowana w kodzie C eliminacja Gaussa z częściowym pivotingiem jest standardową techniką rozwiązywania układów, ale nie jest w stanie przezwyciężyć inherentnie złego uwarunkowania macierzy $\mathbf{G}$ dla wysokich m.

4. **Wizualizacja (Heatmapy i wykresy indywidualne):** Heatmapy błędów (generowane przez Python) okazały się bardzo efektywnym narzędziem do zrozumienia globalnych zależności między n, m a błędami aproksymacji, pozwalając szybko zidentyfikować obszary optymalne i problematyczne. Indywidualne wykresy (generowane przez Gnuplot) dostarczają wglądu w jakość dopasowania dla konkretnych par (n, m) i pozwalają zaobserwować np. oscylacje wielomianu.

5. **Równomierne punkty i potencjalne problemy:** Użycie równomiernie rozmieszczonych punktów, choć proste w implementacji, przyczynia się do problemów ze złym uwarunkowaniem. Dodatkowo, jak pokazano w Sekcji 9 (dla $n = 7$ i $n = 13$), może prowadzić do bardzo słabych, wręcz trywialnych wyników, jeśli punkty próbkowania przypadkowo trafią w pobliże miejsc zerowych funkcji. Podkreśla to, że dla funkcji oscylujących, równomierne próbkowanie może nie być najlepszą strategią.

6. **Porównanie z innymi metodami (sugestia):** Zaimplementowana metoda (równania normalne + Gauss) jest klasycznym podejściem, ale numerycznie mniej stabilnym niż alternatywy. W przyszłych badaniach warto byłoby porównać wyniki z metodami opartymi na rozkładzie QR lub SVD macierzy Vandermonde'a, a także zbadać wpływ użycia węzłów Czebyszewa lub ortogonalnych baz wielomianowych na stabilność i dokładność aproksymacji.

Podsumowując, aproksymacja średniokwadratowa wielomianami algebraicznymi jest potężną techniką, ale jej skuteczne zastosowanie wymaga świadomego doboru parametrów n i m oraz zrozumienia ograniczeń numerycznych metody obliczeniowej i potencjalnych problemów związanych z rozmieszczeniem węzłów, zwłaszcza przy użyciu wielomianów wysokiego stopnia i równomiernego próbkowania. Zastosowana w projekcie kombinacja narzędzi (C do obliczeń, Python i Gnuplot do wizualizacji) pozwoliła na kompleksową analizę zagadnienia.