Akademia Górniczo-Hutnicza w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Metody Numeryczne

Laboratorium 08: Interpolacja funkcjami sklejanymi w bazie

Andrzej Świętek

23.04.2024

Contents

1 Wstęp teoretyczny **2**

1.1 Interpolacja funkcjami sklejanymi 2

1.2 Interpolacja funkcjami sklejanymi poprzez wyznaczenie wartości drugich pochodnych w węzłach 3

1.3 Interpolacja funkcjami sklejanymi w bazie 3

2 Problem **5**

3 Implementacja **6**

4 Wizualizacja Wyników **8**

4.1 Wyniki interpolacji funkcji $f(x) = \frac{1}{1+x^2}$ 9

4.2 Wyniki interpolacji funkcji $f(x) = cos(2x)$ 12

5 Iterpretacja wyników **14**

5.1 Funckja $f_1 = \frac{1}{1+x^2}$ dla $N = 5$ 14

5.2 Funckja $f_1 = \frac{1}{1+x^2}$ dla $N = 6$ 14

5.3 Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 7$ 15

5.4 Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 10$ 15

5.5 Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 20$ 15

5.6 Funkcja $f_2 = cos(2x)$ dla $N = 6$ 15

5.7 Funkcja $f_2 = cos(2x)$ dla $N = 7$ 15

5.8 Funkcja $f_2 = cos(2x)$ dla $N = 14$ 15

6 Wnioski **16**

1. Wstęp teoretyczny

1.1. Interpolacja funkcjami sklejanymi

- Metoda ta odpowiada na główne wady metod Lagrange i Newtona: Dla małych wartości m są te metody nieskuteczne, natomiast dla większych albo mamy do czynienia z Efektem Rungego albo z oscylacjami.

- Metoda ta zakłada posługiwanie się wieloma wielomanami niskiego stopnia (najczęściej do zastosowań fizycznych wykorzystuje się 3 rzędu ze względu na ciągłość drugiej pochodnej)

- W tej metodzie zawsze rozpatrując przedział $[a, b]$ dzielimy go na mniejsze pod przedziały. W tym przedziale mamy $n + 1$ takich punktów że:

$$a = x_0 < x_1 < \cdots < x_{n_1} < x_n = b$$

  Podział ten wyznacza siatkę węzłów jak w poprzednich metodach.

  Mając n+1 punktów w sposób naturalny tworzy się n poprzedziałów $[x_i, x_{i+1}]$, gdzie $i + 1$ oznacza następny węzeł na prawo.

- Funkcja interpolująca jest kombinacją lioniową elementów bazy $\{s_i(x)\}$

$$s(x) = \sum_{i=0}^{n-1}(c_i \cdot s_i(x)), \qquad x \in [a, b]$$

  gdzie funkcja s_i jest wielomianem stopnia conajwyżej m

$$s_i(x) = c_{im} \cdot x^m + c_{im-1} \cdot x^{m-1} + \cdots + c_{i1} \cdot x^1 + c_{i0} \cdot x^0$$

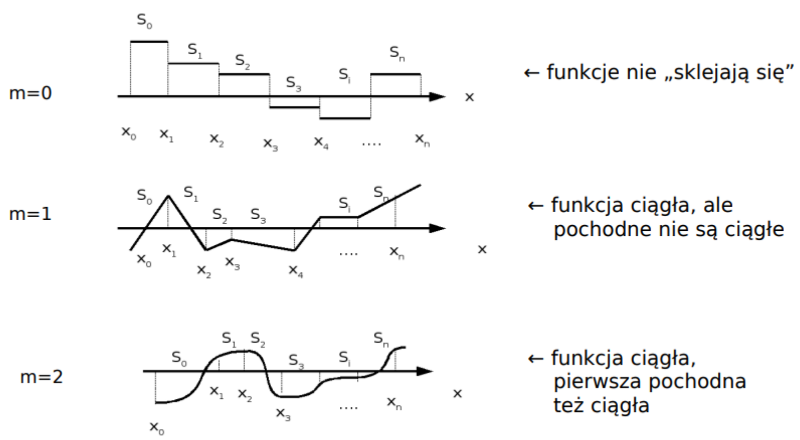

Figure 1: Kształty funkcji interpolujących w zależności od m

Poszukując funkcji interpolującej możemy natrafić na 2 warianty tego problemu:

- Szukamy postaci s_i, wówczaj przyjmujemy za wszytskie $c_i = 1$.
- Zakładamy postać $s_i(x)$ natomiast tym czego szukamy jest c_i.

1.2. Interpolacja funkcjami sklejanymi poprzez wyznaczenie wartości drugich pochodnych w węzłach

W tej metodzie, aby dokonać interpolacji funkcjami sklejanymi, wyznaczamy wartości drugich pochodnych w węzłach interpolacji. Jest to podejście oparte na kubicznych funkcjach sklejanych (ang. cubic spline interpolation), które są bardzo popularne ze względu na swoją gładkość i efektywność.

Kluczowym krokiem jest znalezienie funkcji sklejanych $S_i(x)$ dla każdego podprzedziału $[x_i, x_{i+1}]$. Każda funkcja $S_i(x)$ jest funkcją trzeciego stopnia (kubiczną) zdefiniowaną na danym podprzedziale. Aby uzyskać te funkcje, musimy ustalić warunki brzegowe oraz warunki wewnętrzne, co pozwoli nam na wyznaczenie wartości drugich pochodnych w węzłach interpolacji.

Warunki brzegowe mogą być różne, najczęściej stosuje się:

- **Warunki naturalne:**
  drugie pochodne na końcach przedziałów są równe 0.
- **Warunki Hermite'a**:
  oprócz wartości funkcji w węzłach, znamy także wartości pierwszych pochodnych w tych punktach.
- **Warunki sklejane pierwszego rodzaju:**
  pierwsze pochodne funkcji sklejanych są równe dla sąsiednich podprzedziałów.
- **Warunki sklejane drugiego rodzaju:**
  drugie pochodne funkcji sklejanych są równe dla sąsiednich podprzedziałów.

Po wyznaczeniu funkcji sklejanych dla każdego podprzedziału, możemy dokonać interpolacji na danym przedziale poprzez wykorzystanie odpowiedniej funkcji $S_i(x)$

1.3. Interpolacja funkcjami sklejanymi w bazie

Założenie kształtu funkcji $\implies$ Manipulujemy współczynnikami

W tej metodzie wykorzystujemy bazę funkcji, na przykład bazę B-sklejanych, aby dokonać interpolacji. B-sklejane (B-splines) są funkcjami bazowymi, które mogą być wykorzystane do konstrukcji funkcji sklejanych. Metoda ta polega na wyrażeniu funkcji interpolującej jako kombinacji liniowej funkcji bazowych.

Przykładowo, dla B-sklejanych stopnia k, funkcja interpolująca $s(x)$ może być zdefiniowana jako:

$$s(x) = \sum_{i=-1}^{n+1} c_i B_{i,k}(x)$$

gdzie $B_{i,k}(x)$ są funkcjami bazowymi B-sklejanych stopnia k a c_i to współczynniki interpolacji.

Aby wyznaczyć funkcję interpolującą musimy wyznaczyć wektor c_i z układu równań liniowych który można wyprowadzić następująco:
Korzystając z warunku interpolacji można zapisać:

$$c_{i-1} + 4c_i + c_{i+1} = y_i, \qquad i = 0, 1, \cdots, n$$

Dodatkowo należy rozważyć warunek z pierwszą pochodną to do powstałego układu równań należy dołączyć kolejne 2 równania:

$$-c_{-1} + c_1 = \frac{h}{3}\alpha_1 \implies c_{-1} = c_1 - \frac{h}{3}\alpha_1$$

$$-c_{n-1} + c_{n+1} = \frac{h}{3}\beta_1 \implies c_{n+1} = \frac{h}{3}\beta_1 + c_{n-1}$$

gdzie:

- c_{-1} wkład dodanego węzła z lewej strony przedziału (lewy sąsiad c_0)
- c_1 prawy sąsiad pierwszego węzła - c_0
- c_{n+1} Węzeł dodany z prawej strony przedziału
- c_{n-1} Przed ostatni oryginalny węzeł - lewy sąsiad ostatniego oryginalnego węzła

Po wyeliminowaniu współczynników c_{-1} i c_{n+1} otrzymujemy układ równań:

$$\begin{bmatrix} 4 & 2 & 0 & \dots & 0 \\ 1 & 4 & 1 & \dots & 0 \\ 0 & 1 & 4 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ & & 1 & 4 & 1 \\ 0 & 0 & \dots & 2 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 + \frac{h}{3}\alpha \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n - \frac{h}{3}\beta \end{bmatrix}$$

Ten układ można szybko rozwiązć metodą Rokładu LU.

2. Problem

Wyznaczyć wartości funkcji interpolującej zgodnie z wzorem:

$$s(x) = \sum_{i=0}^{n+1} c_i \Phi_{3i}(x), \quad x \in [x_{\min}, x_{\max}] \tag{1}$$

gdzie sklejki kubiczne $\Phi_{3i}(x)$ są zdefiniowane następująco:

$$\Phi_{3i}(x) = \begin{cases} \frac{1}{h^3}(x - x_{i-2})^3 & \text{dla } x \in [x_{i-2}, x_{i-1}) \\ \frac{1}{h^3}\left(h^3 + 3h^2(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3\right) & \text{dla } x \in [x_{i-1}, x_i) \\ \frac{1}{h^3}\left(h^3 + 3h^2(x_{i+1} - x) + 3h(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3\right) & \text{dla } x \in [x_i, x_{i+1}) \\ \frac{1}{h^3}(x_{i+2} - x)^3 & \text{dla } x \in [x_{i+1}, x_{i+2}) \\ 0 & \text{dla } x \notin [x_{-3}, x_{n+3}] \end{cases}$$

gdzie h jest odległością pomiędzy sąsiednimi węzłami. Warunki interpolacyjne dla pierwszej pochodnej $\left(\alpha = \frac{df}{dx}\Big|_{x=x_{\min}}, \beta = \frac{df}{dx}\Big|_{x=x_{\max}}\right)$ są zadane równaniami:

$$-c_0 + c_2 = \frac{h}{3}\alpha \tag{2}$$

$$-c_{n-1} + c_{n+1} = \frac{h}{3}\beta \tag{3}$$

Układ równań (2) i (3) można zapisać w postaci macierzowej:

$$\begin{bmatrix} 4 & 2 & 0 & \dots & 0 \\ 1 & 4 & 1 & \dots & 0 \\ 0 & 1 & 4 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ & & 1 & 4 & 1 \\ 0 & 0 & \dots & 2 & 4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 + \frac{h}{3}\alpha \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n - \frac{h}{3}\beta \end{bmatrix}$$

Do rozwiązania układu równań można wykorzystać odpowiednią metodę numeryczną, na przykład z pakietu Numerical Recipes lub GSL.

Zadania do wykonania

1. Przy użyciu programu należy przeprowadzić interpolację funkcji

$$f_1(x) = \frac{1}{1 + x^2}$$

oraz

$$f_2(x) = \cos(2x)$$

W przedziale $x \in [-5, 5]$

2. Wykonać interpolację dla funkcji $f_1(x)$ dla liczby węzłów równych $n = 5, 6, 10, 20$. Dla każdego przypadku sporządzić wykresy funkcji interpolowanej i interpolującej na jednym rysunku.
3. Wykonać interpolację funkcji $f_2(x)$ dla liczby węzłów $n = 6, 7, 14$. Dla każdego przypadku sporządzić wykresy funkcji interpolowanej i interpolującej na jednym rysunku.

3. Implementacja

```
double phi(
    double x, double xi_minus_2, double xi_minus_1,
    double xi, double xi_plus_1, double xi_plus_2, double h
) {
    if( x>= xi_minus_2 && x < xi_minus_1 )
        return 1/pow(h,3) * pow(x - xi_minus_2, 3);
    else if( x >= xi_minus_1 && x < xi )
        return 1/pow(h,3) *
            (
                pow(h,3)
                + 3*pow(h,2)*(x - xi_minus_1)
                + 3*h* pow((x - xi_minus_1),2)
                - 3* pow((x - xi_minus_1),3)
            );
    else if( x >= x_i && x < xi_plus_1)
        return 1/pow(h,3) *
            (
                pow(h,3)
                + 3*pow(h,2)*(xi_plus_1 - x)
                + 3*h* pow(((xi_plus_1 - x),2)
                - 3* pow((xi_plus_1 - x),3)
            );
    else if ( x >= xi_plus_1 && x < xi_plus_2)
        return 1/pow(h,3) * pow((xi_plus_2 - x) ,3);
    return  0.0;
}
```

Listing 1: Implementacja funkcji Φ

```
double interpolate(double x, const std::vector<double>& c, const std::vector<
    double>& xx, double h) {
    double sum = 0.0;
    int n = c.size(); // N+1 - po spuszowaniu C_0 i C_n+1
    cout << endl << endl;
    for (int i = 0; i < n ; ++i) {
        sum += c[i] * phi(x, xx[i], xx[i+1], xx[i+2], xx[i+3], xx[i+4], h);
    }
    return sum;
}
```

Listing 2: Implementacja funkcji interpolującej

Inicjalizacja warunków początkowych oraz wektorów mających przechowywać siatke węzłów. Wartości α i β wyliczamy z pochodnej w punkcie na skrajach przedziałów. Implementacja ilorazu różnicowego - taka jak zawsze.
W kolejnych liniach iterujemy przez kolekcje reprezentującą ilości rozpatrywanych węzłów.

```
   const int num_nodes[] = {5,6, 7,10, 20};
    const double x_min = -5.0; const double x_max = 5.0;

    double alpha = dfdx(f1, x_min);
    double beta  = dfdx(f1, x_max);

     for( int N : num_nodes) {
        cout << "===============[ " << N << " ]===============\n";
        double h = (x_max - x_min) / (N - 1);
        std::vector<double> xx(N + 6); // z dolozonymi wezlami
        std::vector<double> xw(N + 4);
        std::vector<double> yw(N + 4);

        //... ciag dalszy
    }
```

Listing 3: Inicjalizacja

Następnie w środku pętli, poniżej wypełniamy wektory siatki odpowiendnimi wartościami z uwzględnieniem dodatkowych elementów siatki na skrajach przedziału.

```
        // ROZWIAZNANIE UKL ROWNAN
        gsl_matrix *A = gsl_matrix_alloc(N, N);
        gsl_vector *b = gsl_vector_alloc(N);
        gsl_vector *c_gsl = gsl_vector_alloc(N);

        // Wypelnienie macierzy A i wektora b
        for (int i = 0; i < N; ++i) {
            if (i == 0 || i == N-1) {
                gsl_matrix_set(A, i, i, 4.0);
                if (i == 0) {
                    gsl_matrix_set(A, i, i + 1, 2.0);
                } else {
                    gsl_matrix_set(A, i, i - 1, 2.0);
                }
                gsl_vector_set(
                        b, i, i == 0 ?
                            f1(xw[i+1]) + alpha*h / 3
                            :
                            f1(xw[i+1]) - beta*h / 3
                );
            } else {
                gsl_matrix_set(A, i, i, 4.0);
                gsl_matrix_set(A, i, i - 1, 1.0);
                gsl_matrix_set(A, i, i + 1, 1.0);
                gsl_vector_set(b, i, f1(xw[i+1]));
            }
        }
```

Listing 4: Wypełnienie węzłów

Kolejnym - kluczowym krokiem jest usalenie wielkości współczynników (wektora) c_i. W tym celu rozwiązujemy układ równań liniowych dowolną metodą np LU itd.

```
        // Rozwiazanie ukladu rownan
        gsl_linalg_HH_solve(A, b, c_gsl);
```

Listing 5: Rozwiązanie układu równań z użyciem GSL

Alternatywnie można wykorzystać rozkłąd LU w prost:

```
        gsl_permutation *p = gsl_permutation_alloc(N);
        int signum;
        gsl_linalg_LU_decomp(A, p, &signum);
        gsl_linalg_LU_solve(A, p, b, c_gsl)
```

Listing 6: Rozwiązanie układu równań z użyciem GSL

```
        double step = 0.1;
        double c0 = c[1] - h/3*alpha;
        double cn_plus_1 = c[ c.size() -2] + h/3*beta;

        c.insert(c.begin(), c0);
        c.push_back(cn_plus_1);
```

Listing 7: Dodanie z przodu i z tyłu wektora c_i wartości odpowiadających dodanym węzłom

```
        for (double x = x_min; x <= x_max; x += step) {
            double s_x = interpolate(x, c, xx, h);
            cout << x << " " << s_x << "\n";
            file << x << " " << s_x << "\n";
        }
```

Listing 8: Wykonanie samej interpolacji dla kolejnych punktów z zadanym krokiem

4. Wizualizacja Wyników

Wszystkie otrzymane wyniki spełniają podstawowe założenia interpolacji w tym warunek $f(x_i) = F(x_i)$, gdzie $f(x)$ to funkcja oryginalna, $F(x)$ to funkcja interpolowana a x_i to argument danego węzła.

4.1. Wyniki interpolacji funkcji $f(x) = \frac{1}{1+x^2}$

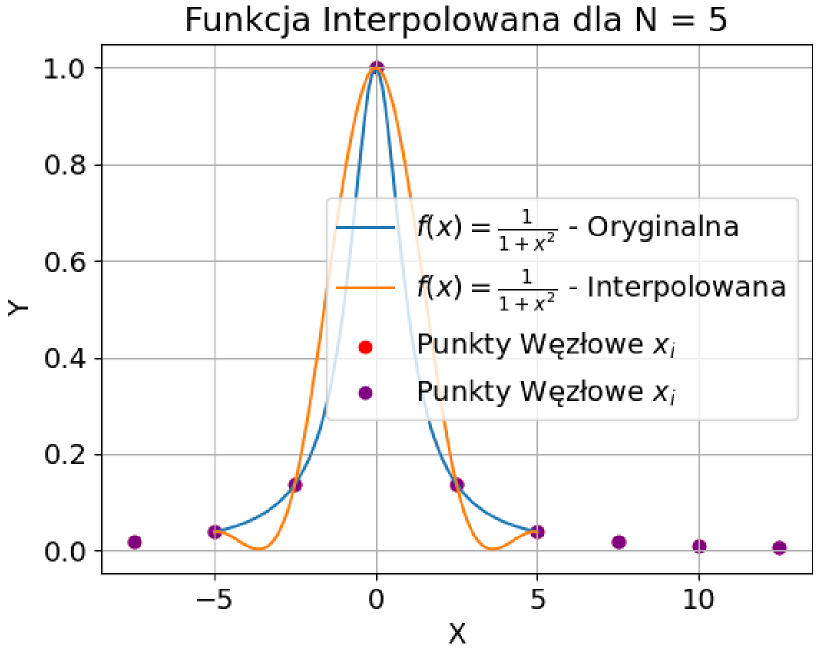

Figure 2: Wrtości funckji interpolowanej i dokładnej dla N = 5

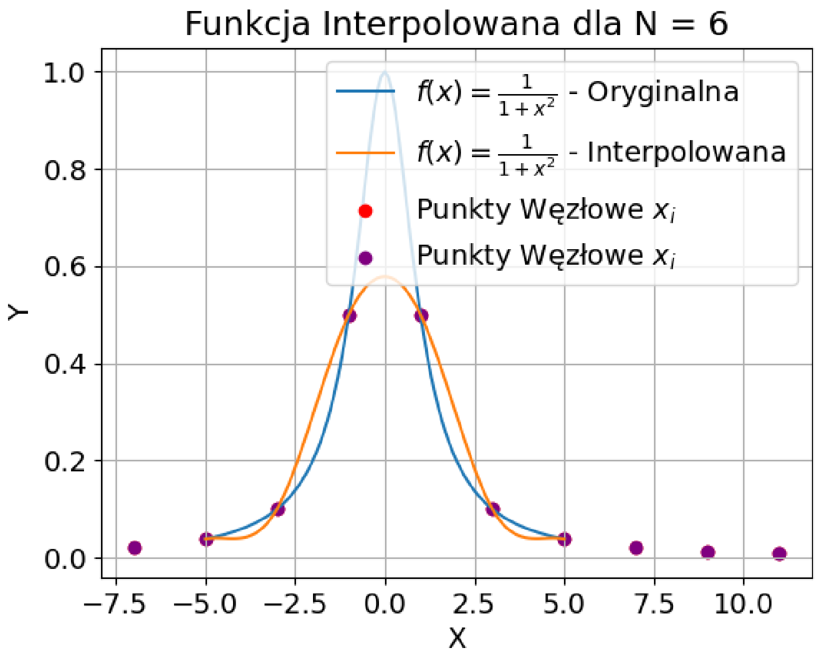

Figure 3: Wrtości funckji interpolowanej i dokładnej dla N = 6

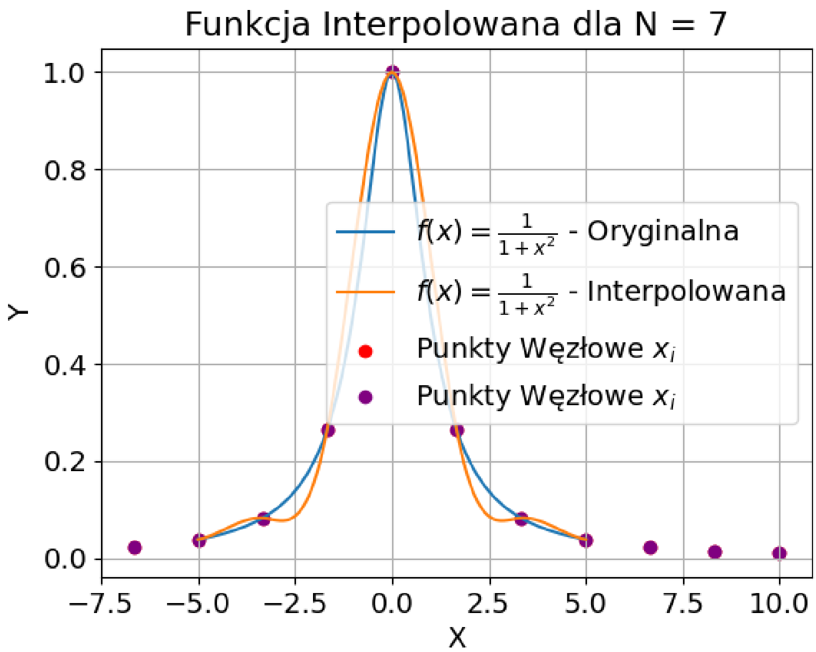

Figure 4: Wrtości funckji interpolowanej i dokładnej dla N = 7

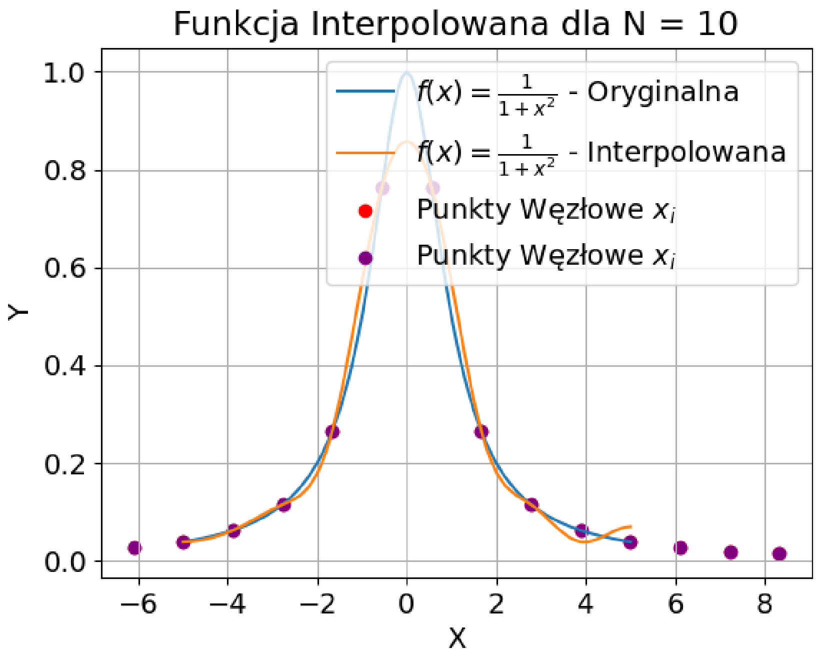

Figure 5: Wrtości funckji interpolowanej i dokładnej dla N = 10

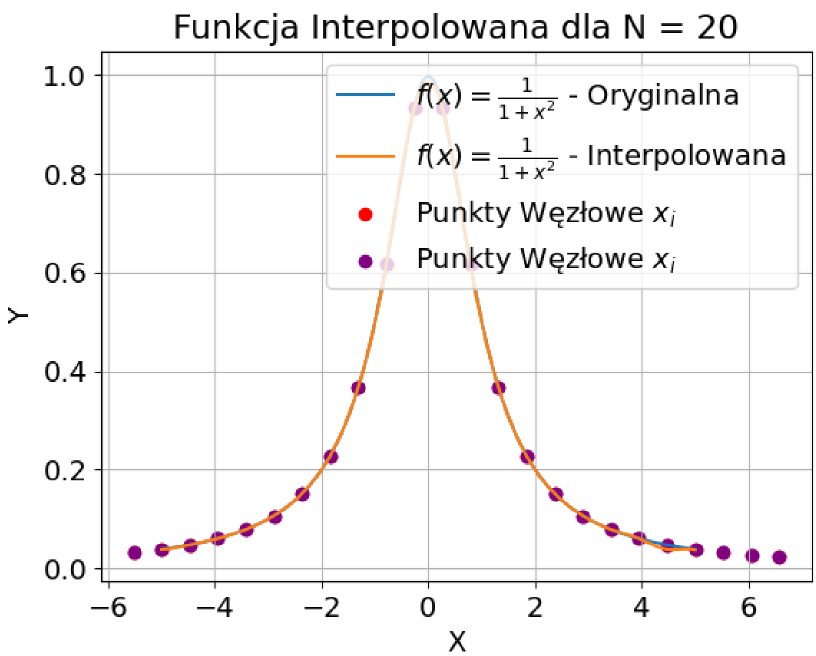

Figure 6: Wrtości funckji interpolowanej i dokładnej dla N = 20

4.2. Wyniki interpolacji funkcji $f(x) = cos(2x)$

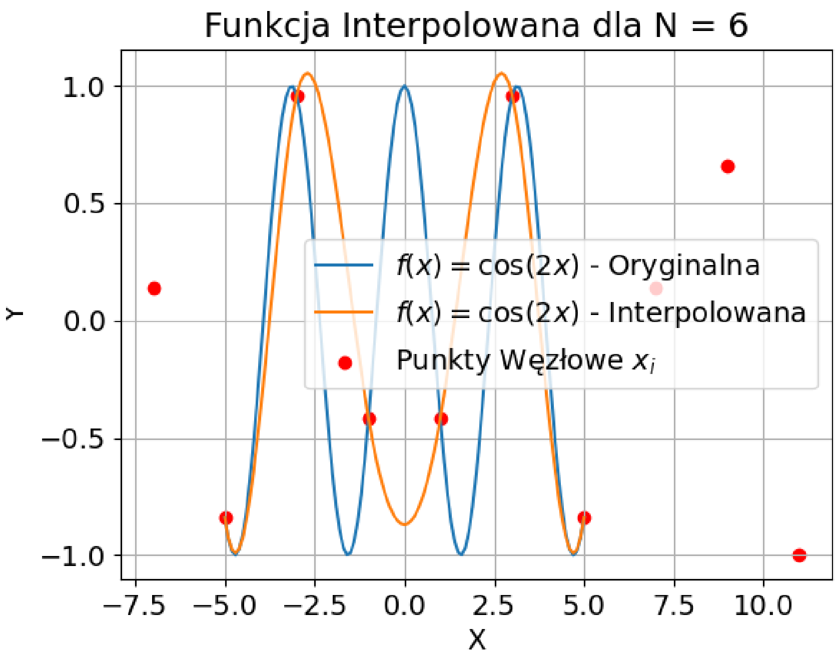

Figure 7: Wrtości funckji interpolowanej i dokładnej dla N = 6 dla funkcji $cos(2x)$

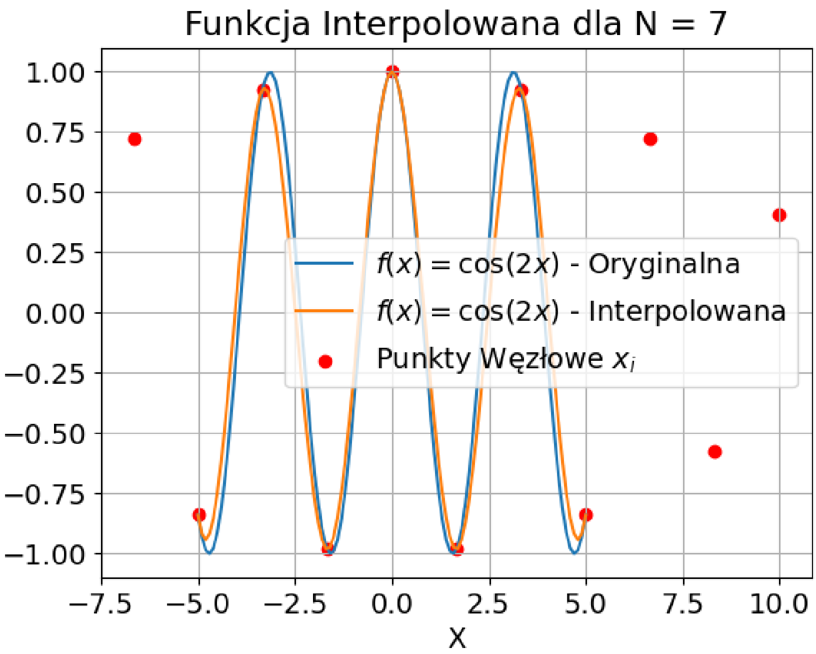

Figure 8: Wrtości funckji interpolowanej i dokładnej dla N = 7 dla funkcji $cos(2x)$

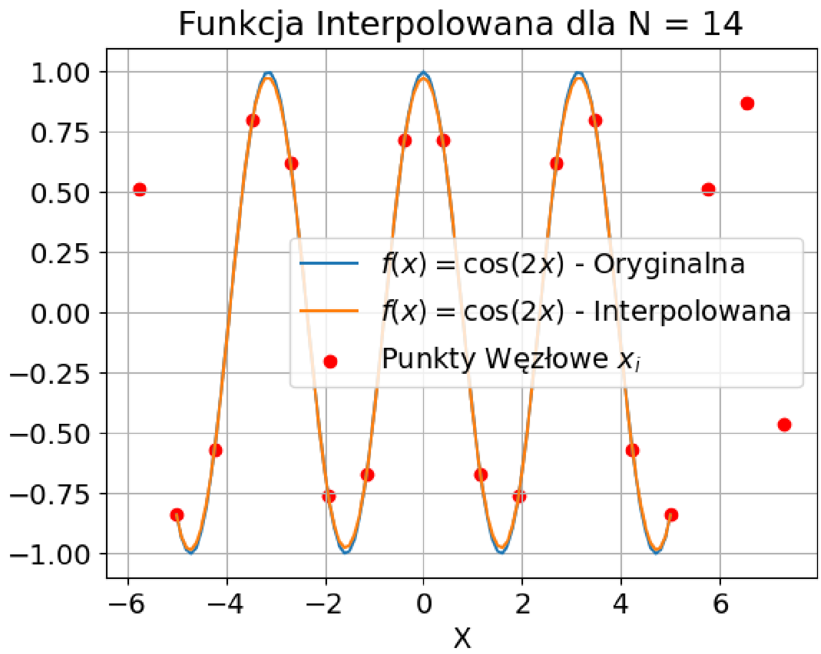

Figure 9: Wrtości funckji interpolowanej i dokładnej dla N = 14 dla funkcji $cos(2x)$

5. Iterpretacja wyników

5.1. Funckja $f_1 = \frac{1}{1+x^2}$ dla $N = 5$

Dla tej ilości węzłów funkcja interpolowana wyraźnie różni się od funkcji oryginalnej. W okolicy węzłów brzegowych błąd względem oryginału jest najbardziej znaczący ponieważ funkcja interpolowana w tym przedziale ma całkowicie inną monotoniczność. W okolicy środkowych węzłów funckcja mimo ogranicaniania oryginalnej od góry, przybliwża jej wartości z dokładnością która z delszej perspektywy może być zadawalająca, niemniej jeśli precyzja ma znaczenie (nie tylko ogólny przebieg) to nie jest to wystarcznająco dobra fukcja.

5.2. Funckja $f_1 = \frac{1}{1+x^2}$ dla $N = 6$

Przy zwiększeniu liczby węzłów z $N = 5$ do $N = 6$ widoczny staje się efekt charakterystyczny dla funkcji parzystych. Wykres funkcji interpolowanej jest zdecydowanie wypłaszczony w środku i odbiega znacząco od oryginału. Krańce przedziału zostały natomiast zinterpolowane dokładnie i zgodnie z kierunkiem przebiegu oryginału.

Aby unikać tego typu niewłaściwej interpolacji dla funkcji parzystych należy używać nieparzystej liczby węzłów.

5.3. Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 7$

Funkcja w dosyć dokładny sposób przybliża funkcję oryginalną na całym przedziale z jedyną oscylacją na wyskości 2 i przed ostatniego węzła. Dla reszty przedziału wartości zinterpolowane między węzłami widocznie stały się bliższe oryginałowi - wynik ten może już być wystarczający dla wielu zastosowań.

5.4. Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 10$

Dla 10 węzłów funkcja interpolowana na skrajach przedziału jest prawie identyczna z funkcją oryginalną. Widoczny natomiast jest problem analogiczny do tego co wystąpił przy $N = 6$. Ponieważ nie ma węzłów w okolicy maximum funkcji wartości interpolowane róœnież zostały spłaszczone i nie osiągają oryginalnych wartości.

5.5. Funkcja $f_1 = \frac{1}{1+x^2}$ dla $N = 20$

Dla 20 węzłów problematyczny dla parzystej ilości węzłów dalej występuje, niemniej jest zdecydowanie mniej widoczny a występujące wypłaszczanie jest nieznaczne. Aby rozwiązać ten problem można albo dodać albo odjąć jeden węzeł.
Dla całej reszty przedziału wartości funkcji interpolowanej niemal całkowicie pokrywają się z funkcją oryginalną a obecna ilość węzłów jest całkowicie wystarczająca aby uzyskać dokładne przybliżenie.

5.6. Funkcja $f_2 = cos(2x)$ dla $N = 6$

Interpolowana funkcja $F(x)$ dla sześciu węzłów znacząco różni się od $cos(2x)$. Choć jest również okresowa i spełnia założenia interpolacji to wewnątrz przedziału zupełnie rozbiega się z funkcją oryginalną a nawet różni się okresem. Przyczynami tego jest parzystość liczby węzłów dla funkcji parzystej jak również 6 okazuje się być niewystarczającą liczbą do skutecznego zbliżenia funkcji interpolowanej do oryginału.

5.7. Funkcja $f_2 = cos(2x)$ dla $N = 7$

Widoczne jest znaczące porpawienie jakości funkcjie interpolowanej. Dla 7 węzłów przybliżenie jest bardzo dokładne nielicząc drobnych odchyłów w okolicy extrów. Do wielu zastosowań może to być wystarczjąco dobry wynik.

5.8. Funkcja $f_2 = cos(2x)$ dla $N = 14$

Dla 14 węzłów uzyskujemy bardzo dobre przybliżenie funkcji $cos(2x)$. Przy odpowiednio dużej skali można dostrzec lekkie spłaszczenie w okolicach wartości największych i najmniejszych charakterystyczne dla parzystych ilości węzłów dla funkcji parzystych. Poza tym drobnym szczegółem funkcja jest prawie identyczna i wystarczająco dobra do prawie wszystkich zastosowań. Jedynym usprawnieniem jakie można by poczynić na rzecz jeszcze większej dokładności - jeżeli takowa wogóle jest potrzebna - to zwiększyć N do 15.

6. Wnioski

Metoda interpolacji funkcjami sklejanymi w bazie:

- **Zalety**:
  - Zapewnia gładkie interpolacje nawet dla dużych zbiorów danych.
  - Efektywnie radzi sobie z problemem efektu Rungego dzięki wykorzystaniu lokalnych funkcji bazowych.
  - Umożliwia kontrolę stopnia interpolacji w poszczególnych fragmentach funkcji.
- **Wady**:
  - Wymaga rozwiązania układu równań, co może być czasochłonne dla dużych zbiorów danych.
  - Wybór odpowiedniej liczby węzłów i stopnia interpolacji może być nieintuicyjny.

Metoda interpolacji Lagrange'a:

- **Zalety**:
  - Prosta w implementacji i zrozumieniu.
  - Nie wymaga rozwiązania układu równań, co może być wydajne dla małych zbiorów danych.
  - Może być stosowana do interpolacji funkcji o dowolnym stopniu.
- **Wady**:
  - Narażona na efekt Rungego, który może prowadzić do oscylacji wartości interpolowanej funkcji na końcach przedziału interpolacji.
  - Wrażliwa na nierównomierny rozkład węzłów interpolacji, co może prowadzić do niepożądanych wyników.

Metoda interpolacji Newtona:

- **Zalety**:
  - Zapewnia dobrą dokładność interpolacji, szczególnie dla równomiernie rozłożonych węzłów interpolacyjnych.
  - Prosta w implementacji i zrozumieniu.
  - Może być stosowana do interpolacji funkcji o dowolnym stopniu.
- **Wady**:
  - Wrażliwa na nierównomierny rozkład węzłów interpolacji, co może prowadzić do błędów interpolacji.
  - Wymaga obliczenia ilorazów różnicowych, co może być trudne dla dużych stopni interpolacji.

Porównując te metody, interpolacja funkcjami sklejanymi w bazie jest często preferowana ze względu na jej zdolność do zapewnienia gładkich interpolacji nawet dla dużych zbiorów danych. Jednakże, metoda interpolacji Lagrange'a i Newtona mogą być bardziej przydatne w przypadku prostych zastosowań lub dla małych zbiorów danych, gdzie czas obliczeń nie jest krytyczny. Warto również zauważyć, że wybór odpowiedniej metody interpolacji zależy od specyfiki problemu oraz preferencji użytkownika.