Akademia Górniczo-Hutnicza
Wydział Informatyki
Kierunek Informatyka

Metody Obliczeniowe w Nauce i Technice

Laboratorium 3

Interpolacja

Wojciech Michaluk, Kyrylo Iakymenko

Kraków, 15 marca 2024

1 Wprowadzenie

Podczas tego laboratorium zrealizujemy zadanie, w którym zastosujemy metodę interpolacji w praktyce. Na podstawie podanych w zadaniu danych zgrupujemy je odpowiednio, aby utworzyć węzły interpolacji. Następnie dla tych węzłów i różnych zbiorów funkcji bazowych, wyznaczymy macierze Vandermonde'a i znajdziemy wielomian interpolacyjny, a także określimy charakterystyki jego przykładowej ekstrapolacji.

W późniejszej części zadania obliczymy na podstawie tych węzłów także wielomian interpolacyjny Lagrange'a i wielomian interpolacyjny Newtona oraz porównamy uzyskane wyniki.

2 Opis zadania

W poleceniu zadania podane są informacje o populacji Stanów Zjednoczonych w wybranych latach, które prezentujemy także w tabeli poniżej.

Rok	1900	1910	1920	1930	1940	1950	1960	1970	1980
Populacja	76212168	92228496	106021537	123202624	132164569	151325798	179323175	203302031	226542199

Tabela 1: Dane dotyczące populacji USA, podane w treści zadania

Korzystając z danych w tabeli, tworzymy węzły interpolacji, przy czym każdy jest reprezentowany przez parę postaci (*Rok*, *Populacja w tym roku*). Powstałych punktów jest dokładnie 9, co oznacza, że wyznaczają one jednoznacznie wielomian ósmego stopnia. Ten wielomian można reprezentować w różny sposób - rozważamy następujące zbiory tzw. funkcji bazowych:

1. $\phi_j(t) = t^{j-1}$;
2. $\phi_j(t) = (t - 1900)^{j-1}$;
3. $\phi_j(t) = (t - 1940)^{j-1}$;
4. $\phi_j(t) = ((t - 1940)/40)^{j-1}$,

gdzie t oznacza rok, natomiast $j = 1, 2, .., 9$. Dla każdego z tych zbiorów tworzymy macierz Vandermonde'a. W następnej sekcji badamy, która z nich ma najmniejszy współczynnik uwarunkowania i na tej podstawie wybierzemy ją do wyznaczenia wielomianu interpolacyjnego.

3 Opracowanie zadania

3.1 Wykorzystanie macierzy Vandermonde'a

Macierz tę zwykle definiuje się jako

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix},$$

gdzie $x_1, x_2, ..., x_n$ oznaczają argumenty wielomianu i w naszym przypadku są to wybrane z tabeli lata, natomiast n oznacza, ile danych mamy - w naszym przypadku 9. My przyjmujemy jednak pewną alternatywną definicję - potęgi maleją wraz ze wzrostem numeru kolumny, czyli rozważane przez nas macierze mają postać

$$\begin{bmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{n-1} & x_n^{n-2} & \cdots & 1 \end{bmatrix}.$$

Stoi za tym kilka powodów - jednym z nich jest specyfika funkcji biblioteki `numpy`, bowiem używane przez nas funkcje, które będą także wspomniane później, operują na innej kolejności współczynników. Przyczyną od strony praktycznej jest z kolei fakt, że dla zwyczajnej definicji otrzymany wielomian nie przechodził przez węzły interpolacji (!!!), co jest sytuacją niedopuszczalną. Stąd przyjęty przez nas model.

Macierz Vandermonde'a obliczamy, korzystając z funkcji `numpy.vander` wedle następującego schematu:

```
vandermonde = np.vander(base(year))
```

gdzie *year* to tablica zawierająca lata, a *base* to zbiór jednej z funkcji bazowych oraz *np* to alias dla *numpy* - powtarzamy powyższe dla każdego zbioru. Następnie dla każdej uzyskanej macierzy, korzystając z funkcji `numpy.linalg.cond`, liczymy współczynnik uwarunkowania. Okazuje się, że najmniejszą wartość otrzymujemy dla macierzy reprezentującej zbiór nr 4. Wynosi ona $cond(\phi(x)) \approx 1605.44$. Wnioskujemy więc, że to właśnie czwarta baza wielomianów jest najlepiej uwarunkowana i to jej będziemy używać do dalszych obliczeń.

Kolejnym etapem jest obliczenie współczynników wielomianu interpolacyjnego. Możemy je znaleźć z następującego równania macierzowego:

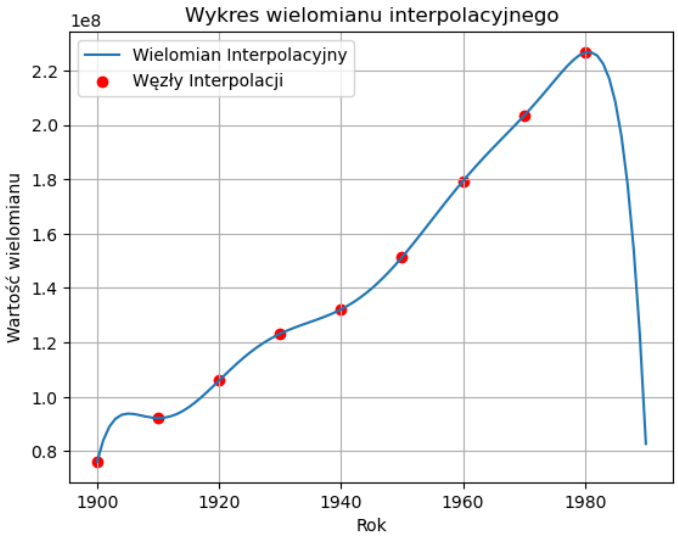
$$V \cdot C = Y, \tag{1}$$

gdzie V oznacza macierz Vandermonde'a, Y to kolumna z wartościami wielomianu w węzłach (czyli w tym przypadku populacja), natomiast $C = [c_1, c_2 \cdots c_n]^T$ to właśnie szukana kolumna współczynników, przy czym są to współczynniki stojące przy kolejno malejących potęgach. Dzieje się tak dlatego, że metoda Hornera tworzy wielomian na podstawie tej kolumny według zasady wspomnianej powyżej. Wartości wielomianu interpolacyjnego obliczamy na przedziale 1900 - 1990 w odstępach jednorocznych. Pokazujemy, jak to wygląda w kodzie.

```
coeffs = np.linalg.solve(vandermonde4, population) #kolumna współczynników
xs = np.arange(1900, 1991, 1)
ys = horner(coeffs, base4(xs))
```

Listing 1: Obliczanie kolumny współczynników i wartości wielomianu interpolacyjnego

Zgodnie z poleceniem zadania, używamy schematu Hornera do wyznaczenia wielomianu.
Poniżej prezentujemy wykres wartości wielomianu z zaznaczonymi węzłami interpolacji.



Rysunek 1: Wykres wartości wielomianu z zaznaczonymi węzłami interpolacji

Co zaskakujące, mimo tendencji wzrostowej jeżeli chodzi o liczbę populacji (i która jest widoczna, kiedy popatrzymy na populację w roku 1990), to wartości wielomianu zaczynają maleć. Rzeczywista populacja na rok 1990 wynosi 248 709 873. Tymczasem otrzymana przez nas wartość wynosi, zaokrąglając do pełnej osoby, 82 749 141 - około trzykrotnie mniej. Błąd względny ekstrapolacji dla roku 1990 obliczamy ze wzoru

$$\eta = \frac{|R_v - C_v|}{R_v},$$

gdzie R_v oznacza rzeczywistą wartość, a C_v - obliczoną ekstrapolowaną wartość. Otrzymujemy wynik ≈ 0.667.

3.1.1 Dygresja - interpolacja dla zaokrąglonych danych

Rozważmy powyższe kroki, ale dla przypadku, kiedy zaokrąglamy dane dotyczące liczby mieszkańców do jednego miliona.

Rok	1900	1910	1920	1930	1940	1950	1960	1970	1980
Populacja [mln]	76	92	106	123	132	151	179	203	227

Tabela 2: Dane dotyczące populacji USA, zaokrąglone do jednego miliona

Używamy tej samej macierzy, co dla oryginalnych danych. W analogiczny sposób obliczamy współczynniki wielomianu dla zaokrąglonych danych. Wydaje się, że przy tej skali takie zaokrąglenie nie powinno wpłynąć znacząco na uzyskane wyniki. Tymczasem rezultaty okazuja się zaskakujące.

Porównajmy najpierw wyznaczone współczynniki:

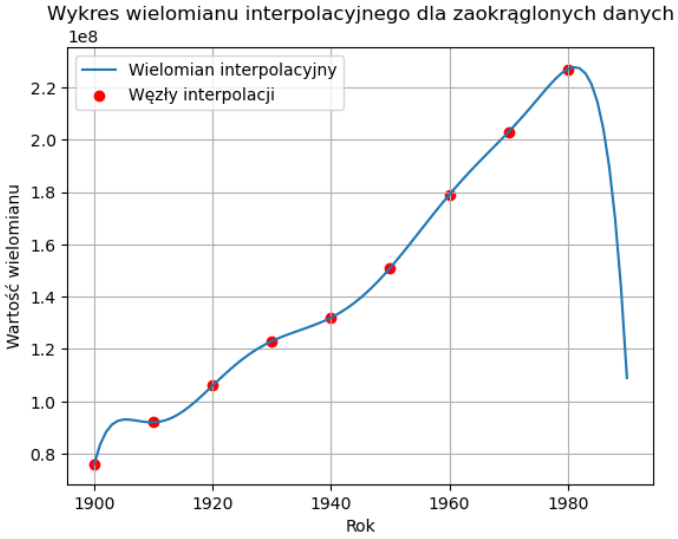
Oryginalne dane [$\cdot 10^8$]	Zaokrąglone dane [$\cdot 10^8$]
-3.15180235	-2.94196825
1.89175576	1.86920635
6.06291250	5.70311111
-3.42668456	-3.38488889
-3.74614715	-3.56755556
1.82527130	1.81111111
1.02716315	1.00141270
0.461307656	0.459571429
1.32164569	1.32000000

Tabela 3: Porównanie współczynników wielomianu

Przy czym należy pamiętać, że współczynniki odnoszą się kolejno do malejących potęg, tzn. w pierwszym właściwym wierszu mamy wyraz stojący przed ósmą potęgą, w drugim ten przed siódmą itd.

Jak widać, są różnice, ale nie są one znaczne. Mając wyznaczone współczynniki postępujemy tak samo jak w poprzednim przypadku (kod jest identyczny z dokładnością do nazw, więc nie pokazujemy go tutaj).

Uzyskujemy poniższy wykres. Kształt wykresu jest podobny, ale można zauważyć, że



Rysunek 2: Wykres wartości wielomianu z zaznaczonymi węzłami interpolacji, zaokrąglone dane

wartość wielomianu dla roku 1990 jest nieco większa (co za tym idzie, bliższa rzeczywistej wartości). Zatem błąd względny powinien być mniejszy. Istotnie, te obserwacje znajdują uzasadnienie w liczbach: ekstrapolowana wartość wielomianu wynosi 109 000 000, a błąd względny (obliczany identycznie) wynosi ≈ 0.562.
Można próbować wyjaśnić dane zjawisko na kilka sposobów:

1. Zaokrąglone dane mogą pomóc w redukcji nieregularności występujących w oryginalnych danych. Czasami dane mogą zawierać drobne fluktuacje, które nie odzwierciedlają rzeczywistego trendu. Zaokrąglając dane do jednego miliona, możemy eliminować te fluktuacje. W szczególności błąd modelu wynikający z szumu w danych wejściowy jest wiarygodny w przypadku małej ilości danych wejściowych (w naszej predykcji stosowaliśmy tylko 9 punktów).

2. Również eliminacja szumu poprzez zaokrąglenie mogła mieć wpływ na wynik ze względu na to, że wielomian interpolacyjny ma dokładnie przechodzić przez dane punkty, więc mała zmiana danych wejściowych powodująca zmiany współczynników wielomianu ma większe znaczenie wraz ze wzrostem argumentu.

3. Nie warto wykluczać też zwykłego przypadku. Nie możemy robić wiarygodnych wniosków na podstawie jednego modelu.

3.2 Wielomian interpolacyjny Lagrange'a

Idea metody interpolacji Lagrange'a (jak i w przypadku metody Newtona) polega na konstrukcji wielomianu, który przechodzi przez zadane punkty. Wielomian ten jest wyrażony jako suma iloczynów funkcji bazowych, zwanych funkcjami Lagrange'a, z odpowiednimi współczynnikami wagowymi. Każda funkcja Lagrange'a jest skonstruowana tak, aby była równa 1 w jednym z punktów danych i równa 0 we wszystkich pozostałych punktach. Dzięki temu, sumując te funkcje dla każdego punktu, otrzymujemy wielomian interpolacyjny, który dokładnie przechodzi przez wszystkie punkty danych.

Metoda interpolacji Lagrange'a jest prostsza w implementacji, w porównaniu do innych metod interpolacyjnych i dobrze się sprawdza podczas interpolowania funkcji o niewielkiej liczbie punktów danych. Jednakże, dla dużych zestawów danych lub dla funkcji o dużej złożoności, może być bardziej wydajne zastosowanie innych technik interpolacji - o tym warto pamiętać.

W kolejnych częściach tego wprowadzenia omówimy kroki niezbędne do zastosowania metody interpolacji Lagrange'a oraz przyjrzymy się jej zastosowaniom i ograniczeniom.

Metoda Lagrange'a polega na wyznaczeniu współczynników Lagrange'a, a potem na ich podstawie wielomianu interpolacyjnego:

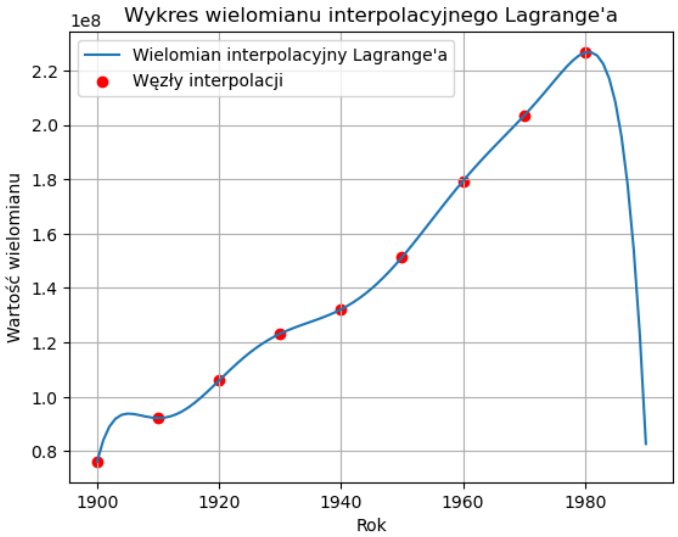
1. wielomiany bazowe $\ell_j(t) = \frac{\prod_{k=1,k\neq j}^{n}(t-t_k)}{\prod_{k=1,k\neq j}^{n}(t_j-t_k)}$ $j = 1, ..., n$.
2. wielomian interpolacyjny $p_{n-1}(t) = y_1\ell_1(t) + ... + y_n\ell_n(t)$.

Realizacja metody Lagrange'a w Pythonie:

```
def lagrange_interpolation_values(x, x_nodes, y_nodes):
    n = len(x_nodes)
    result = np.zeros(len(x))
    l_coeffs_denom = np.zeros(n)

    #denominators are dependent only on given data, so we prepare it in advance
    for i in range(n):
        l_coeffs_denom[i] = np.prod(x_nodes[:i] - x_nodes[i])
          * np.prod(x_nodes[i + 1:] - x_nodes[i])

    for i in range(n):
        term = y_nodes[i]
        for j in range(n):
            if j != i:
                term *= (x - x_nodes[j])
        result += term / l_coeffs_denom[i]

    return result
```

Listing 2: Obliczanie kolumny współczynników i wartości wielomianu interpolacyjnego Lagrange'a

Spójrzmy na uzyskany wykres wielomianu interpolacyjnego:



Rysunek 3: Wykres wartości wielomianu interpolacyjnego Lagrange'a z zaznaczonymi węzłami interpolacji

Patrząc na otrzymany wykres, nie widzimy dużych różnic w porównaniu do innych wykresów. Można było tego się spodziewać, gdyż wszystkie metody interpolacji wielomianowej wyznaczają ten sam wielomian i mogą się różnić, jeżeli chodzi o błędy wynikające ze specyfiki arytmetyki komputerowej (które z reguły są stosunkowo małe w porównaniu do innych możliwych czynników).

Patrząc na wartość ekstrapolowaną dla roku 1990, dostajemy $p(1990) \approx 82\ 749\ 141$. Wartość błędu względnego wynosi ≈ 0.667. Są to identyczne rezultaty jak w pierwszym przypadku.

3.3 Wielomian interpolacyjny Newtona

Wielomian interpolacyjny Newtona dla n węzłów interpolacji jest wyrażony wzorem

$$f[t_1]\pi_1(t) + f[t_1, t_2]\pi_2(t) + \cdots + f[t_1, t_2, ..., t_n]\pi_n(t) \tag{2}$$

Omówmy znaczenie poszczególnych symboli występujących w równaniu (2).

- Wyrazy $f[t_1], f[t_1, t_2]$ itd. to tzw. ilorazy różnicowe (ang. *divided differences*), które pełnią rolę współczynników i są obliczane następująco [2]:

$$f[t_i] = f(t_i) \mid \text{0-wy iloraz różnicowy}$$
$$f[t_i, t_{i+1}] = \frac{f[t_{i+1}] - f[t_i]}{t_{i+1} - t_i} \mid \text{1-szy iloraz różnicowy}$$

  Gdy określone są ilorazy aż do $(k-1)$, to wtedy k-ty iloraz różnicowy definiujemy jako

$$f[t_i, t_{i+1}, ..., t_{i+k}] = \frac{f[t_{i+1}, t_{i+2}, ..., t_{i+k}] - f[t_i, t_{i+1}, ..., t_{i+k-1}]}{t_{i+k} - t_i}.$$

- $\pi_j(t)$ to wielomiany "bazowe" w metodzie interpolacji Newtona. Są określone wzorem

$$\pi_j(t) = \prod_{k=1}^{j-1}(t - t_k), \text{ dla } j = 1, 2, ..., n$$

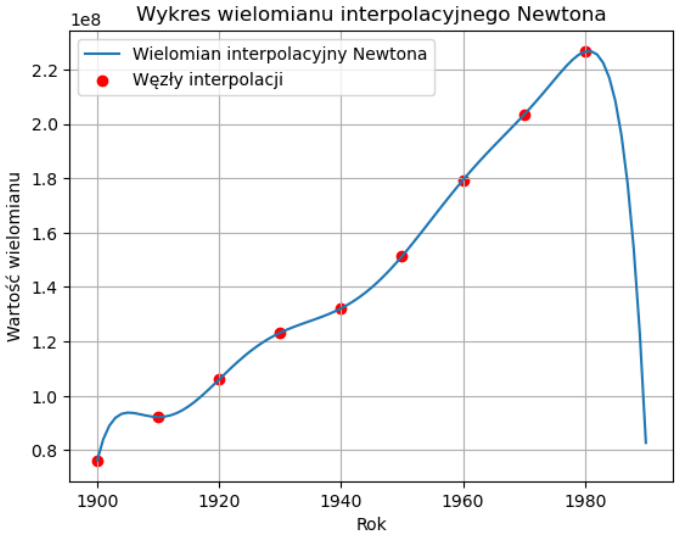
  Przyjmujemy umownie, że $\pi_1(t) \equiv 1$. Zauważmy, że $\pi_j(t_i) = 0$ dla $i < j$.

Możemy teraz przystąpić do obliczania wartości wielomianu interpolacyjnego Newtona. Najpierw przedstawiamy kod odpowiadający za obliczenie wielomianu:

```
#Obliczenie współczynników - ilorazów różnicowych
def newton_polynomial_coeffs(x_nodes, y_nodes):
    n = len(x_nodes)
    dd = np.array(y_nodes, copy = True, dtype = np.double)

    #building divided differences matrix
    for i in range(1, n):
        dd[i : n] = (dd[i : n] - dd[i - 1]) / (x_nodes[i : n] - x_nodes[i - 1])

    return dd

#Obliczanie wartości wielomianu
def newton_polynomial_values(x_nodes, y_nodes, x):
    a = newton_polynomial_coeffs(x_nodes, y_nodes)
    n = len(x_nodes)
    p = np.ones(shape = (n, len(x)))

    for i in range(1, n):
        for j in range(i):
            p[i] *= (x - x_nodes[j])

    return np.dot(a, p)
```

Listing 3: Obliczanie ilorazów różnicowych i wartości wielomianu interpolacyjnego Newtona

Na podstawie obliczonych wartości rysujemy wykres, który przedstawiamy poniżej.



Rysunek 4: Wykres wartości wielomianu interpolacyjnego Newtona z zaznaczonymi węzłami interpolacji

Jak widać, wygląda on praktycznie identycznie jak w poprzednich przypadkach. Porównajmy, czy statystyki ekstrapolacji do roku 1990 istotnie są takie same. Obliczona dla roku 1990 wartość wielomianu wynosi 82 749 141 - czyli jest identyczna! Oczywiście oznacza to także taką samą wartość błędu względnego, wynoszącą ≈ 0.667.

4 Podsumowanie i wnioski

Porównując działanie metod Newtona i Lagrange'a, warto najpierw zwrócić uwagę na to, że wyniki interpolacji wielomianowej są jednoznaczne. Czyli mając n punktów początkowych (x_1, y_1), ... , (x_n, y_n) możemy jednoznacznie wyznaczyć wielomian stopnia $n-1$ przez te punkty przechodzący (twierdzenie interpolacyjne).

Zatem na tej podstawie możemy patrzeć wyłącznie na dokładność obliczeń, złożoność oraz na specyfikę liczenia współczynników. Jeżeli chodzi o dokładność obliczeń, to metody Newtona i Lagrange'a za bardzo się niczym nie różnią. Dostajemy bardzo podobną dokładność, natomiast teoretycznie Lagrange jest trochę bardziej podatny na błąd obliczeniowy ze względu na większą liczbę operacji arytmetycznych (zwłaszcza mnożeń), które trzeba wykonać, żeby dostać współczynniki wielomianu albo jego wartość dla danego punktu. Z tego wynika, że dla wielomianów interpolacyjnych o tym samym stopniu, różnice w wydajności między metodami Lagrange'a i Newtona są zazwyczaj niewielkie. Obie metody mogą być stosowane z sukcesem w praktyce, szczególnie przy interpolacji funkcji o niewielkiej liczbie punktów danych.

Literatura

[1] Materiały pomocnicze do laboratorium zamieszczone na platformie Teams w katalogu *lab03/lab3-intro.pdf*.

[2] Treść przedstawiona na wykładzie o interpolacji.