Artur Gęsiarz
Marzec 22, 2024

Laboratorium nr 3
MOwNiT – Interpolacja

1. Treść zadania
   1.1. Populacja Stanów Zjednoczonych na przestrzeni lat przedstawiała się następująco

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

Istnieje dokładnie jeden wielomian ósmego stopnia, który interpoluje po- wyższe dziewięć punktów, natomiast sam wielomian może być reprezentować na różne sposoby. Rozważamy następujące zbiory funkcji bazowych ϕj(t), j = 1,...,9:

$$\varphi_j(t) = t^{j-1}$$

$$\varphi_j(t) = (t-1900)^{j-1}$$

$$\varphi_j(t) = (t-1940)^{j-1}$$

$$\varphi_j(t) = \frac{t-1940^{j-1}}{40}$$

(a) Dla każdego z czterech zbiorów funkcji bazowych utworze macierz Vandermonde'a.

(b) Obliczę współczynnik uwarunkowania każdej z powyższch macierzy

(c) Znajdę współczynniki wielomianu interpolacyjnego dla danych

(d) Dokonam ekstrapolacji wielomianu do roku 1990. Porównam otrzymaną wartość z prawdziwą wartością dla roku 1990, wynoszącą 248 709 873.

(e) Wyznaczę wielomian interpolacyjny Lagrange'a na podstawie 9 węzłów interpolacji podanych w zadaniu. Obliczę wartości wielomianu w odstępach jednorocznych.

(f) Wyznacz wielomian interpolacyjny Newtona na podstawie tych samych węzłów interpolacji i oblicz wartości wielomianu w odstępach jednorocznych.

(g) Zaokrąglę dane podane w tabeli do jednego miliona. Na podstawie takich danych wyznaczę wielomian interpolacyjny ósmego stopnia, używając najlepiej uwarunkowanej bazy z podpunktu (c). Porównam wyznaczone współczynniki z współczynnikami obliczonymi w podpunkcie (c).

2. Rozwiązanie zadania

2.1. Zapisuje jako słownik populacje Stanów Zjednoczonych na przestrzeni lat

```
population_US = {
    1900: 76212168,
    1910: 92228496,
    1920: 106021537,
    1930: 123202624,
    1940: 132164569,
    1950: 151325798,
    1960: 179323175,
    1970: 203302031,
    1980: 226542199
}

years = np.array(list(population_US.keys()))
population = np.array(list(population_US.values()))

population_US_rounded = {year: round(population, -6) for year, population in population_US.items()}
rounded_population = np.array(list(population_US_rounded.values()))
```

2.2. Wyznaczenie wszystkich podanych bazowych czterech funkcji dla j =1,..9

```
first_base_function = [lambda t, j=i: pow(t,j - 1) for i in range(1,10)]
second_base_function = [lambda t, j=i: pow(t - 1900, j - 1) for i in range(1,10)]
third_base_function = [lambda t, j=i: pow(t - 1940, j - 1) for i in range(1,10)]
fourth_base_function = [lambda t, j=i: pow((t - 1940)/40, j - 1) for i in range(1,10)]
```

2.3. Implementacja funkcji Vandermonde

```
def vandermonde_matrix(population: dict, base_functions):
    n = len(population.keys())
    m = len(base_functions)
    V = np.zeros((n, m))
    row = -1
    for year in population.keys():
        row += 1
        for j in range(m):
            V[row][j] = base_functions[j](year)
    return V
```

2.4. Wyznaczam dla każdego ze zbiorów funkcji bazowych macierz Vandermonde

```
V_first = vandermonde_matrix(population_US, first_base_function)
V_second = vandermonde_matrix(population_US, second_base_function)
V_third = vandermonde_matrix(population_US, third_base_function)
V_fourth = vandermonde_matrix(population_US, fourth_base_function)
```

2.5. Wyznaczenie współczynnika uwarunkowania każdej z powyższych macierzy

```
cond_first = np.linalg.cond(V_first)
cond_second = np.linalg.cond(V_second)
cond_third = np.linalg.cond(V_third)
cond_fourth = np.linalg.cond(V_fourth)

print(cond_first)
print(cond_second)
print(cond_third)
print(cond_fourth)
```

2.6. Używając najlepiej uwarunkowanej bazy wielomianów, znajduję współczynniki wielomianu interpolacyjnego

```
V_best = V_fourth
coefficients = np.linalg.solve(V_best, population)  # wspolczynniki
```

```
def polynomial_interpolation(coefficients, base_functions, year):
    """
    Funkcja zwraca interpolacje wielomianowa dla danego roku uzywjac schmatu Hornera
    :param coefficients: Lista wspolczynnikow wielomianu interpolacyjnego
    :param base_functions: Lista funkcji bazwoych, ktora sa uzywny do konstrukcji wielomianu
    :param year: Rok dla ktorego chcemy obliczyc wartosc wielomianu interpolacyjnego
    :return:
    """
    result = 0
    for i in range(len(coefficients)):
        result += coefficients[i] * base_functions[i](year)
    return result
```

2.7. Funkcje do rysowanie wykresu wielomianu interpolacyjnego

```
x_values = np.arange(1900, 1991)
y_values = polynomial_interpolation(coefficients, fourth_base_function, x_values)

plt.plot(x_values, y_values, label='Interpolating polynomial')
plt.scatter(years, population, color='red', label='Data points')
plt.title('Interpolating Polynomial')
plt.xlabel('Year')
plt.ylabel('Population')
plt.legend()
plt.grid(True)
plt.show()
```

2.8. Ekstrapolacja wielomianu do roku 1990 z prawdziwą wartością

```python
extrapolated_population = polynomial_interpolation(coefficients, fourth_base_function, 1990)
true_population_1990 = 248709873
relative_error = abs(extrapolated_population - true_population_1990) / true_population_1990

print("Ekstrapolowana populacja w roku 1990:", round(extrapolated_population,3))
print("Prawdziwa populacja w roku 1990:", round(true_population_1990,3))
print("Błąd względny ekstrapolacji dla roku 1990 w procentach wynosi:", round(relative_error,3) * 100)
```

2.9. Wielomian interpolacyjny Lagrane'a

```python
def lagrange_interpolation(x, y, x_interp):
    """
    Funkcja ta wykonuje interpolację Lagrange'a dla zestawu punktow (x,y) w celu uzyskania wartosci
     interpolowanej  dla okerslonych punktow interpolacji x_inyerp
    :param x: Jest to tablica zawierająca współrzedne x punktów danych.
    :param y: Jest to tablica zawierająca odpowiadające współrzędne y punktów danych.
    :param x_interp: est to tablica zawierająca współrzędne x, dla których chcemy wykonać interpolację.
    :return: Wynikowa tablica result zawiera wartości interpolowane dla każdego punktu w x_interp
    """
    n = len(x)
    m = len(x_interp)
    result = np.zeros(m)

    for i in range(m):
        interpolated_value = 0
        for j in range(n):
            term = y[j]
            for k in range(n):
                if k != j:
                    term *= (x_interp[i] - x[k]) / (x[j] - x[k])
            interpolated_value += term
        result[i] = interpolated_value

    return result
```

```python
population_US_years_added = list(population_US.keys())
population_US_years_added.append(1990)

population_US_population_added = list(population_US.values())
population_US_population_added.append(true_population_1990)

x = np.array(population_US_years_added)
y = np.array(population_US_population_added)


x_interp = np.arange(1900, 2000)
y_interp = lagrange_interpolation(x, y, x_interp)

plt.plot(x_interp, y_interp, label='Lagrange Interpolating polynomial')
plt.scatter(x, y, color='red', label='Data points')
plt.title('Lagrange Interpolating Polynomial')
plt.xlabel('Year')
plt.ylabel('Population')
plt.legend()
plt.grid(True)
plt.show()
```

2.10. Wielomian interpolacyjny Newtona

```python
def divided_differences(x, y):
    """
    Funkcja oblicza różnice dzielone dla zestawu punktów (x, y) za pomocą metody Newtona.
    Różnice dzielone wykorzystywane są później do konstrukcji wielomianu interpolacyjnego
    :param x: Jest to tablica zawierająca współrzędne x punktów danych.
    :param y: Jest to tablica zawierająca odpowiadające współrzędne y punktów danych.
    :return:
    """
    n = len(x)
    F = np.zeros((n, n))  # tworze macierz n na n
    F[:,0] = y  # wypelniam pierwsza kolumne wartosciami y

    for j in range(1, n):
        for i in range(j, n):
            F[i, j] = (F[i, j - 1] - F[i - 1, j - 1]) / (x[i] - x[i - j])

    return np.diag(F)

def newton_interpolation(x, y, x_interp):
    """
    Funkcja wykonuje interpolację Newtona dla zestawu punktów (x, y) w celu uzyskania wartości interpolowanej
     dla określonych punktów interpolacji x_interp.
    :param x: Jest to tablica zawierająca współrzędne x punktów danych.
    :param y: Jest to tablica zawierająca odpowiadające współrzędne y punktów danych.
    :param x_interp: Jest to tablica zawierająca współrzędne x, dla których chcemy wykonać interpolację.
    :return:  Wynikowa tablica result zawiera wartości interpolowane dla każdego punktu w x_interp
    """
    n = len(x)
    m = len(x_interp)
    coefficients = divided_differences(x, y)
    result = np.zeros(m)

    for i in range(m):
        interpolated_value = coefficients[0]
        term = 1
        for j in range(1, n):
            term *= (x_interp[i] - x[j - 1])
            interpolated_value += coefficients[j] * term
        result[i] = interpolated_value

    return result
```

```python
population_US_years_added = list(population_US.keys())
population_US_years_added.append(1990)

population_US_population_added = list(population_US.values())
population_US_population_added.append(true_population_1990)

x = np.array(population_US_years_added)
y = np.array(population_US_population_added)

x_interp = np.arange(1900, 2000)
y_interp = newton_interpolation(x, y, x_interp)

plt.plot(x_interp, y_interp, label='Newton Interpolating polynomial')
plt.scatter(x, y, color='red', label='Data points')
plt.title('Newton Interpolating Polynomial')
plt.xlabel('Year')
plt.ylabel('Population')
plt.legend()
plt.grid(True)
plt.show()
```

2.11. Wielomian interpolacyjny z zaokrąglonymi danymi do miliona

```
V_rounded = vandermonde_matrix(population_US_rounded, fourth_base_function)
coefficients_rounded = np.linalg.solve(V_rounded, rounded_population)

print("Współczynniki wielomianu interpolacyjnego z zaokraglonymi danymi:\n", coefficients_rounded, "\n")
print("Współczynniki wielomianu interpolacyjnego:\n", coefficients)

x_values = np.arange(1900, 1991)
y_values = polynomial_interpolation(coefficients_rounded, fourth_base_function, x_values)

plt.plot(x_values, y_values, label='Interpolating polynomial')
plt.scatter(years, rounded_population, color='red', label='Data points')
plt.title('Interpolating Polynomial')
plt.xlabel('Year')
plt.ylabel('Population')
plt.legend()
plt.grid(True)
plt.show()
```

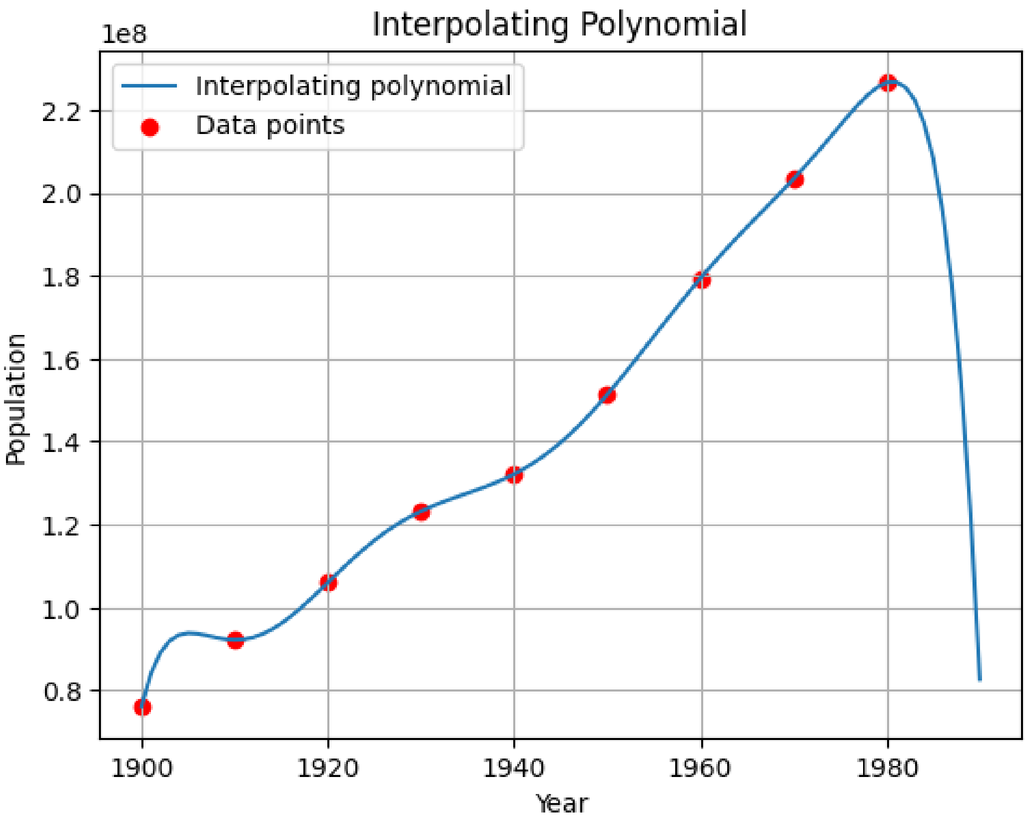
2.12. Ekstrapolacja wielomianu przybliżonego do roku 1990 z prawdziwą wartością

```
extrapolated_population = polynomial_interpolation(coefficients_rounded, fourth_base_function, 1990)
true_population_1990 = 248709873
relative_error = abs(extrapolated_population - true_population_1990) / true_population_1990

print("Ekstrapolowana populacja w roku 1990:", round(extrapolated_population,3))
print("Prawdziwa populacja w roku 1990:", round(true_population_1990,3))
print("Błąd względny ekstrapolacji dla roku 1990 w procentach wynosi:", round(relative_error,3) * 100)
```

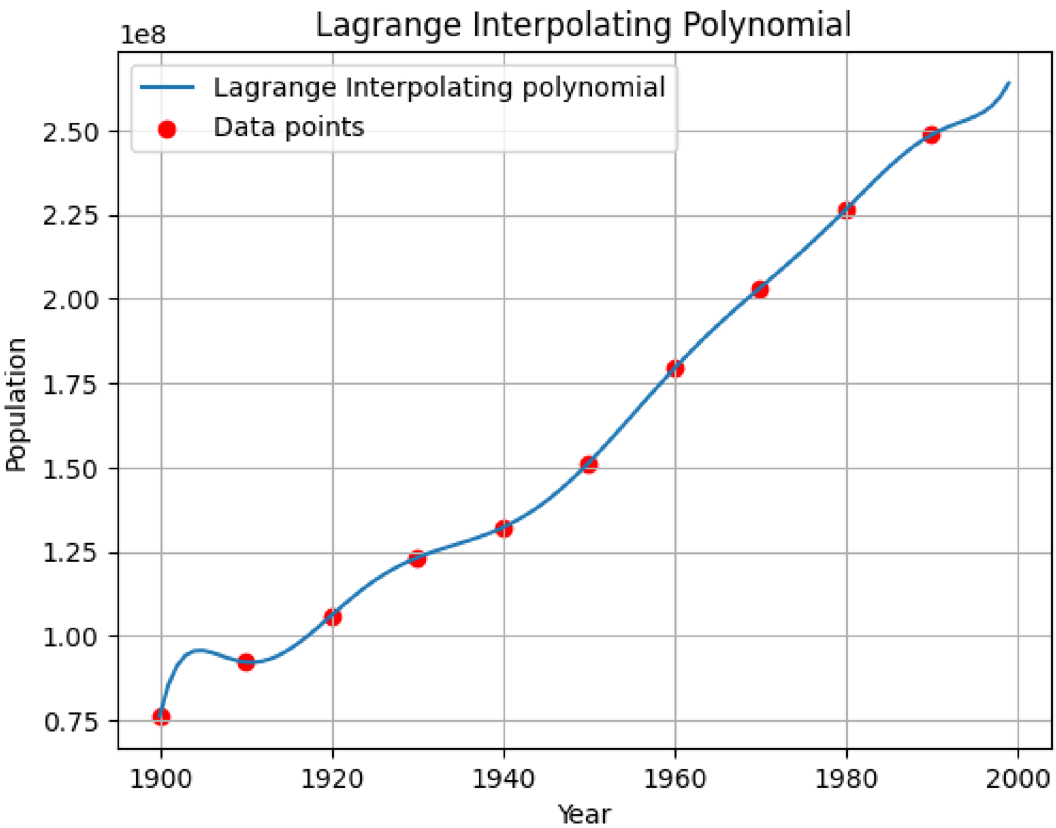
3. Wykresy

3.1. Wykres wielomianu interpolacyjnego



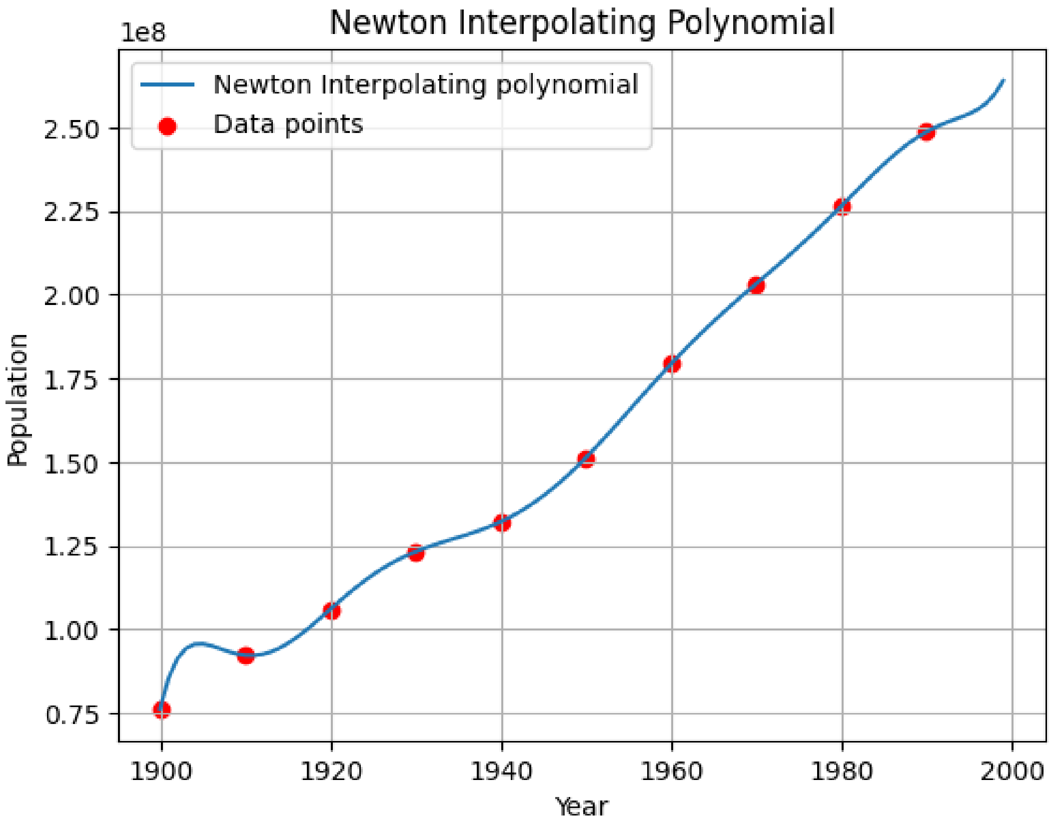
Wykres 1. Wykres przedstawiający wielomian interpolacyjny

3.2. Wykres wielomianu interpolacyjnego Lagrange'a



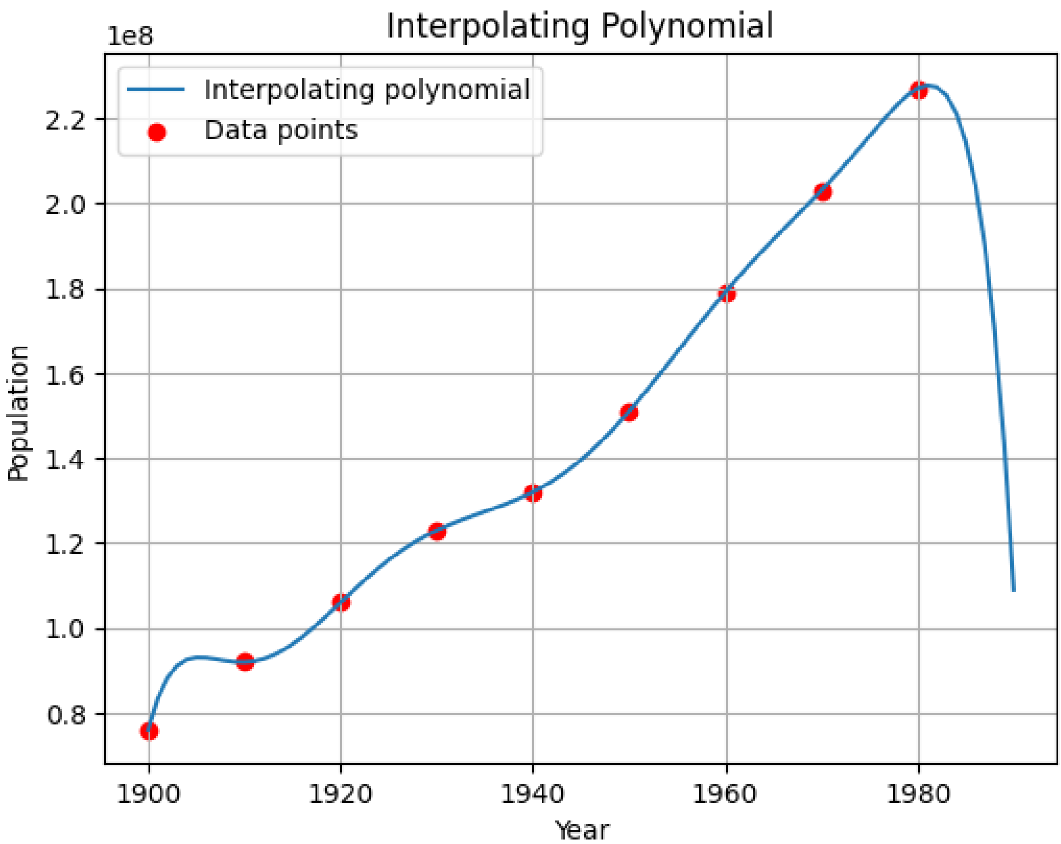
Wykres 2. Wykres przestawiający wielomian interpolacyjny Lagrange'a wraz z dodatkowym punktem

3.3. Wykres wielomianu interpolacyjnego Newtona



Wykres 3. Wykres przedstawiający wielomian interpolacyjny Newtona wraz z dodatkowym punktem

3.4. Wykres wielomianu interpolacyjnego z zaokrąglonymi danymi do miliona



Wykres 4. Wykres przedstawiający wielomian interpolacyjny z zaokrąglonymi danymi

4. Tabele

4.1. Współczynniki uwarunkowania każdej z wyznaczonej macierzy Vandermone'a

Macierz	Współczynnik uwarunkowania macierzy
Korzystająca* z pierwszej funkcji bazowej	$4.377811242696482 * 10^{37}$
Korzystająca* z drugiej funkcji bazowej	6211148482504961
Korzystająca* z trzeciej funkcji bazowej	9315536040586.04
Korzystająca* z czwartej funkcji bazowej	1605.44

4.2. Porównanie ekstrapolowanych wielomianów dla roku 1990

Wielomian	Ekstrapolowania populacja w roku 1990:	Prawdziwa populacja w 1990:	Błąd względny ekstrapolacji dla roku 1990:
Niezaokrąglony*	82749141	248709873	$66{,}7\%$
Zaokrąglony*	109000000	248709873	$56{,}2\%$

4.3. Współczynniki wielomianów interpolacyjnych:

Wielomian	Wsp. 1.	Wsp. 2.	Wsp. 3.	Wsp. 4.	Wsp. 5.	Wsp. 6.	Wsp. 7.	Wsp. 8.	Wsp. 9.
Niezaokrąglony*	$1.32 * 10^8$	$4.61 * 10^7$	$1.02 * 10^8$	$1.82 * 10^8$	$-3.74 * 10^8$	$-3.42 * 10^8$	$6.06 * 10^8$	$1.89 * 10^8$	$-3.15 * 10^8$
Zaokrąglony*	$1.32 * 10^8$	$4.59 * 10^7$	$1 * 10^8$	$1.811 * 10^8$	$-3.56 * 10^8$	$-3.38 * 10^8$	$5.7 * 10^8$	$1.86 * 10^8$	$-2.94 * 10^8$

**poprzez „zaokrąglony" – mam na myśli wielomian, który jest utworzony poprzez wartości populacji Stanów Zjednoczonych, ale z zaokrągleniem do miliona*

**poprzez „niezaokrąglony" – mam na myśli zwykłe dane*

**poprzez „korzystająca z … funkcji bazowej" – mam na myśli macierz Vandermone'a która jest utworzona z … funkcji bazowej*

5. Wnioski

Podsumowując, możemy zauważyć, że, wybór odpowiedniej bazy funkcji bazowych ma istotny wpływ na jakość interpolacji. W zadaniu podano cztery funkcje bazowe, a wyniki pokazały, że wybór najlepiej uwarunkowanej bazy może znacząco wpłynąć na dokładność interpolacji.

Współczynniki wielomianu interpolacyjnego różnią się w zależności od bazy funkcji bazowych oraz danych wejściowych. Współczynniki wielomianu interpolacyjnego z zaokrąglonymi danymi mogą różnić się od tych uzyskanych bez zaokrąglania danych wejściowych, co może wpłynąć na dokładność wyników interpolacji.

Interpolacja Lagrange'a i Newtona to dwa popularne podejścia do interpolacji wielomianowej. Oba podejścia mogą być skutecznie wykorzystywane do interpolacji danych, ale mogą różnić się efektywnością i stabilnością numeryczną w zależności od konkretnych przypadków.

Warto zauważyć, że wyniki interpolacji mogą być wrażliwe na skrajne wartości danych wejściowych oraz na stopień wielomianu interpolacyjnego. Zbyt wysoki stopień wielomianu może prowadzić do zjawiska nadmiernego dopasowania, co może prowadzić do nieprawidłowych wyników dla danych spoza zakresu interpolacji.

W przypadku ekstrapolacji danych zaokrąglonych, wartość ekstrapolowanej populacji w roku 1990 wynosiła około 109 mln, podczas gdy prawdziwa populacja w tym roku wynosiła 248 709 873. Błąd względny ekstrapolacji dla roku 1990 wynosił około 56.2%.

Dla danych wejściowych niezaokrąglonych, wynik ekstrapolacji może jest inny. Jest to ważne zjawisko, ponieważ zaokrąglenie danych może wpłynąć na obliczenia i prowadzić do różnych wyników w porównaniu z danymi niezaokrąglonymi.

Zaokrąglenie danych może prowadzić do utraty dokładności i zwiększenia błędu w wynikach interpolacji i ekstrapolacji. Warto zauważyć, że im większa różnica między wartościami rzeczywistymi a wynikami ekstrapolacji, tym większy wpływ może mieć zaokrąglenie danych na wyniki.

W praktyce, dla zadań wymagających wysokiej dokładności, należy unikać lub minimalizować zaokrąglanie danych, aby uzyskać jak najbardziej dokładne wyniki interpolacji i ekstrapolacji.

Podsumowując, odpowiedni wybór metody interpolacji i bazy funkcji bazowych jest kluczowy dla uzyskania dokładnych i stabilnych wyników interpolacji. W praktyce warto eksperymentować z różnymi metodami i parametrami, aby znaleźć optymalne rozwiązanie dla konkretnego problemu. Wnioski te podkreślają znaczenie uwzględnienia precyzji danych oraz dokładnego zarządzania nimi podczas wykonywania operacji numerycznych, zwłaszcza w przypadku zadań, gdzie dokładność jest kluczowa.

6. Bibliografia
Wykład MOwNiT - prowadzony przez dr. Inż. K. Rycerz
Prezentacje – dr. Inż. M. Kuta

7. Dodatkowe informacje
Rozwiązanie zadania znajduje się w pliku ex1.pynb