Metody Numeryczne – projekt 3

“Aproksymacja profilu wysokościowego ”

Michał Soja

175793
Informatyka
semestr 4
grupa 5

08.06.2022r

1. Wstęp

Projekt polega na zaimplementowaniu i zaprezentowaniu interpolacji Lagrange’a oraz interpolacji Splainowej.

Projekt został napisany w języku python, a wykresy są rysowane za pomocą biblioteki matplotlib. Do obliczenia układów równań w metodzie interpolacji funkcjami sklejanymi wykorzystano bibliotekę numpy.

2. Implementacja

Po uruchomieniu, program prosi nas o podanie nazwy wraz z rozszerzeniem pliku umieszczonego w folderze *InputData* w katalogu projektu zawierającego dane wejściowe. Następnie należy podać ile punktów, program ma wybrać (równo rozłożonych w zbiorze danych wejściowych). Na podstawie tych punktów program później będzie wyliczał interpolacje dwoma metodami. Do tych czynności służy funkcja *get_input()*. Funkcja ta zwraca liste oryginalnych danych (wartości x oraz y), wartości y próbek, oraz nazwę pliku (przydatna do wyświetlania na wykresach).

```
if __name__ == '__main__':
    original, samples, file = get_input()
```

Jeśli użytkownik nie będzie chciał wpisywać za każdym uruchomieniem programu tych danych, wystarczy w tej funkcji odkomentować, którąś z linijek z przypisaniem do zmiennej *filename* nazwy pliku, oraz zakomentować linijkę z funkcją *input()*. Tak samo należy postąpić w przypadku zmiennej *sample_amount*.

```
def get_input():
    global original

    print("podaj nazwę pliku z danymi: ")
    filename = str(input())
    # filename = "chelm.txt"
    # filename = "ostrowa.txt"
    # filename = "rozne_wzniesienia.txt"
    # filename = "stale.txt"
    # filename = "tczew_starogard.txt"

    print("podaj ilość próbek: ")
    sample_amount = int(input())
    # sample_amount = 10

    original = input_from_file("InputData/" + filename)
    samples_result = []

    for i in range(len(original)):
        if i % int(len(original) / (sample_amount - 1)) == 0:
            samples_result.append(original[i])
    if len(samples_result) < sample_amount:
        samples_result.append(original[-1])

    return original, samples_result, filename
```

Następnym krokiem jest pobranie z oryginalnego zbioru wartości x (dalej zwane dziedziną) i przesłanie ich wraz z zbiorem próbek do funkcji obu metod. Podanie dziedziny do funkcji jest potrzebne, aby móc wyliczyć wartości interpolowane dla tych samych argumentów z dziedziny. Dzięki temu jesteśmy w stanie narysować oba wykresy na jedym układzie współrzędnych i porównać w prosty sposób wyniki otrzymane z wynikami oczekiwanymi.

```
domain = [x[0] for x in original]

results_lagrange = lagrange(samples, domain)
results_splines = splines(samples, domain)
```

Metoda Lagrange'a
Ta metoda dla każdego punktu z dziedziny wylicza wartość punktu ze wzroru:

$$F(x) = \sum_{i=1}^{4} y_i \phi_i(x)$$

$$\phi_1(x) = \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)},$$
$$\phi_2(x) = \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)},$$
$$\phi_3(x) = \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)},$$
$$\phi_4(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}.$$

Jest to metoda bardzo łatwa do zaimplementowania jak również nie wymagająca dużych zasobów obliczeniowych komputera. W programie jest zrealizowana w pętli, która dla każdej próbki liczy licznik oraz mianownik i mnoży tą wartość przez wartość próbki.

```
def lagrange_value(data, x):
    result = 0

    for i in range(len(data)):
        nominator = 1
        for j in range(len(data)):
            if i != j:
                nominator *= (x - data[j][0])

        denominator = 1
        for j in range(len(data)):
            if i != j:
                denominator *= (data[i][0] - data[j][0])

        result += data[i][1] * nominator / denominator

    return result
```

Metoda interpolacji funkcjami sklejanymi

Ta metoda jest trudniejsza do zaimplementowania w porównaniu do metody Lagrange'a. Jest ona oparta o układ równań liniowych. Najpierw tworzona jest macierz oraz wektor wartości wypełniona zerami. Następnie są one wypełniane wartościami. Dla n punktów, rozmiary macierzy kwadratowej to 4*(n – 1).

```
def splines(data, x_list):
    matrix_size = (len(data) - 1) * 4
    A = np.zeros((matrix_size, matrix_size))
    b = np.zeros(matrix_size)
```

Pierwsze n-1 wierszy jest uzupełniane zgodnie z równaniem:

$$S_j(x_j) = f(x_j)$$

```
    # S_j(x_j) = f(x_j)
    for i in range(len(data) - 1):
        A[row][i * 4 + 0] = 1              # a0
        b[row] = data[i][1]
        row += 1
```

Kolejne n-1 zgodnie ze równaniem:

$$S_j(x_{j+1}) = f(x_{j+1})$$

```
# S_j(x_j+1) = f(x_j+1)
for i in range(len(data) - 1):
    h = data[i + 1][0] - data[i][0]
    A[row][i * 4 + 0] = 1              # a0
    A[row][i * 4 + 1] = 1 * pow(h, 1)  # b0
    A[row][i * 4 + 2] = 1 * pow(h, 2)  # c0
    A[row][i * 4 + 3] = 1 * pow(h, 3)  # d0
    b[row] = data[i + 1][1]
    row += 1
```

Kolejne n-2 równania to porównanie pochodnych pierwszego stopnia w węzłach wewnętrznych.

```
# S'_j-1(x_j) = S'_j(x_j)
for i in range(1, len(data) - 1):
    h = data[i + 1][0] - data[i][0]
    A[row][i * 4 - 3] = 1              # b0
    A[row][i * 4 - 2] = 2 * pow(h, 1)  # c0
    A[row][i * 4 - 1] = 3 * pow(h, 2)  # d0

    A[row][i * 4 + 1] = -1             # b1

    row += 1
```

Następnie kolejne n-2 równań to porównanie pochodnych drugiego stopnia też węzłów wewnętrznych.

```
    # S''_j-1(x_j) = S''_j(x_j)
    for i in range(1, len(data) - 1):
        h = data[i + 1][0] - data[i][0]
        A[row][i * 4 - 2] = 2               # c0
        A[row][i * 4 - 1] = 6 * pow(h, 1)   # d0

        A[row][i * 4 + 2] = -2              # c1

        row += 1
```

Ostatnie 2 równania to porównanie pochodnych drugiego stopnia w węzłach zewnętrznych.

```
# S''_0(x_0) = 0
A[row][2] = 1
row += 1

# S''_n(x_n) = 0
h = data[-1][0] - data[-2][0]
A[row][-2] = 2
A[row][-1] = 6 * pow(h, 1)
```

Mając tak przygotowane macierze można obliczyć układ równań, a otrzymany wektor będzie zbiorem współczynników funkcji trzeciego stopnia w postaci:

$$[a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1, \dots, a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1}]$$

zgodnie z wzorem:

$$f(x) = a + b\,x + c\,x^2 + d\,x^3$$

Układ równań jest rozwiązywany za pomocą biblioteki numpy oraz funkcji *np.linalg.solve()*. Aby wyznaczyć wartość elementu z dziedziny za pomocą współczynników trzeba odnaleźć dwa punkty z próbek, pomiędzy którymi się znajduje i podstawić do wzoru korzystając z odpowiednich współczynników dla tego przedziału.

```
factors = np.linalg.solve(A, b)

result = []
temp = 0
for elem in x_list:
    for i in range(len(data) - 1):
        if data[i][0] <= elem <= data[i + 1][0]:
            temp = 0
            for j in range(4):
                h = elem - data[i][0]
                temp += factors[4 * i + j] * pow(h, j)
    result.append(temp)

return result
```

Tak obliczone dane są następnie rysowane na wykresach za pomocą biblioteki *matplotlib*. Wykresy obu metod rysowane są w jednym obiekcie *figure*. Z lewej strony znajduje się metody Lagrange'a, natomiast po prawej dla metody interpolacji funkcjami sklejanymi. Dodatkowo wykresy zapisywane są w formacie png w folderze *Results.*

```
def make_plots(data_input, data_samples, lagrange_output, splines_output, filename):
    figure, (ax1, ax2) = pyplot.subplots(1, 2)
    figure.set_figwidth(10)
    figure.set_figheight(5)

    figure.suptitle(filename + "\nNumber of Samples: " + str(len(data_samples)))

    draw_plot(data_input, data_samples, lagrange_output, ax1, "Lagrange")
    draw_plot(data_input, data_samples, splines_output, ax2, "Splines")

    result_filename = 'Results/' + filename.split(".")[0] + "_samples_" + str(len(data_samples)) + '.png'

    print(result_filename)
    pyplot.savefig(result_filename)
    pyplot.show()
```

3. Dane

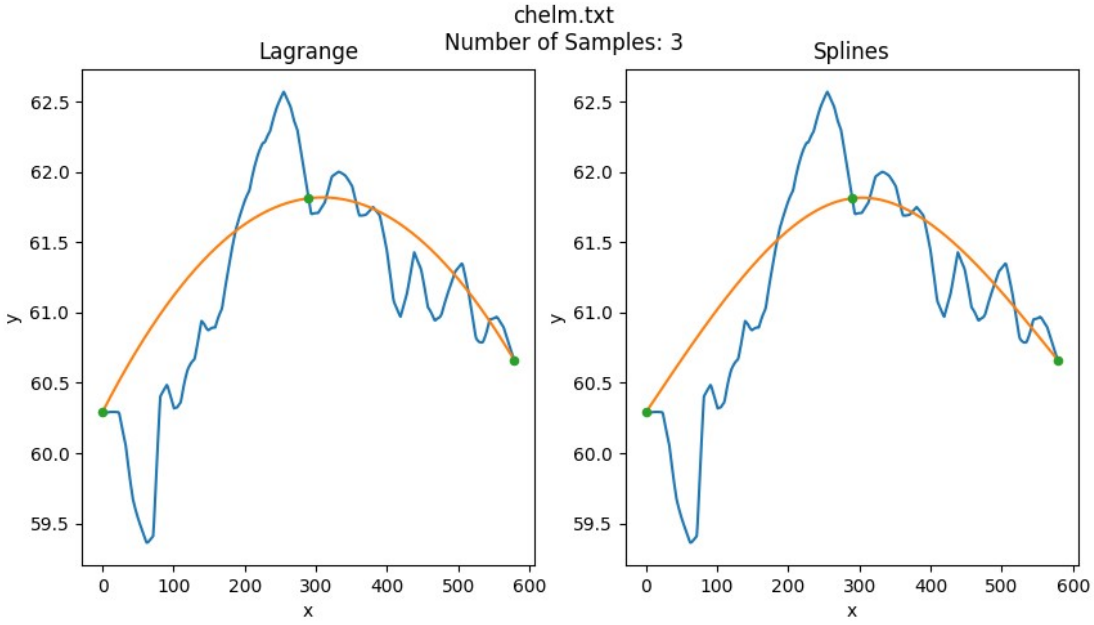
W projekcie zostały wykorzystane dane udostępnione na stronie kursu na enauczaniu, jednakże można wykorzystać dowolne dane w formacie:

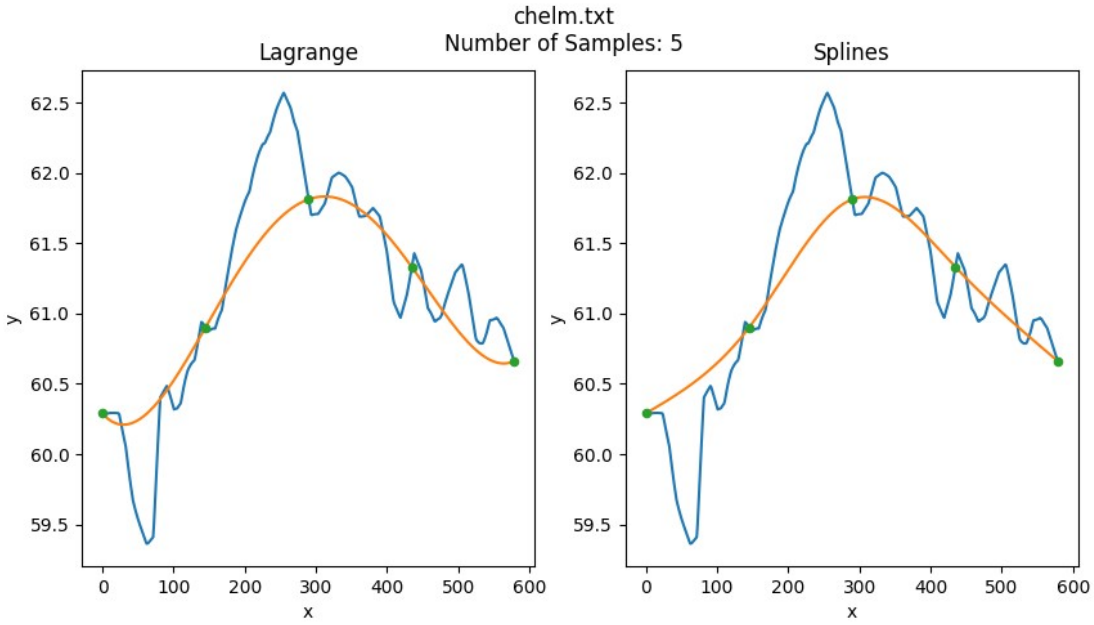
wartość_x wartość_y
wartość_x wartość_y
wartość_x wartość_y
…

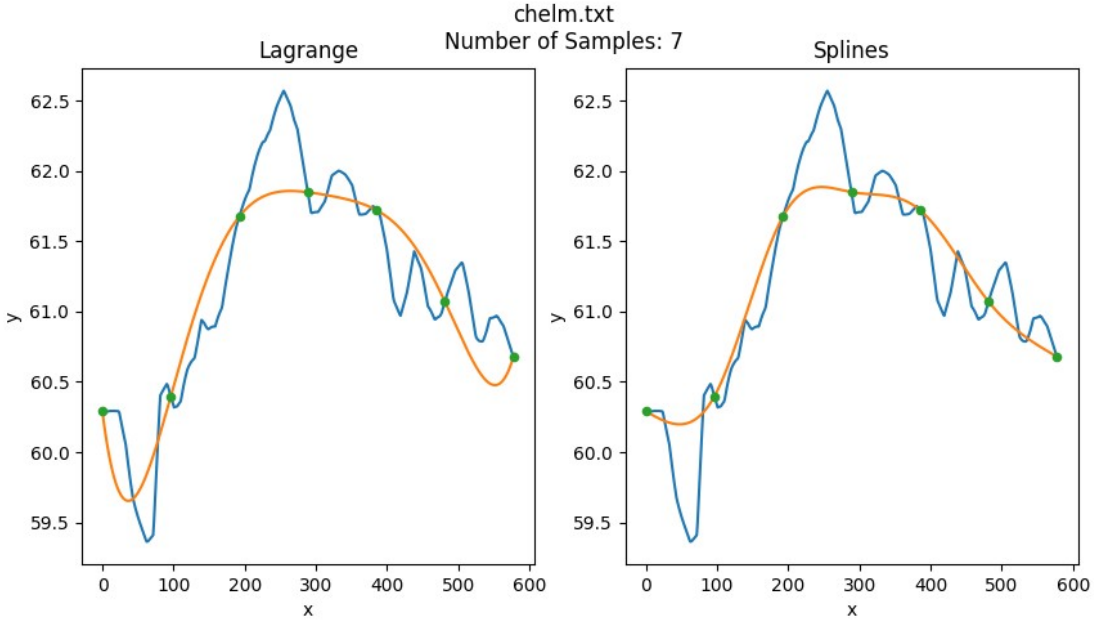
Aby dołączyć własny plik z danymi należy go umieścić w folderze *InputData* w katalogu projektu. Znajduje się tam 5 plików:

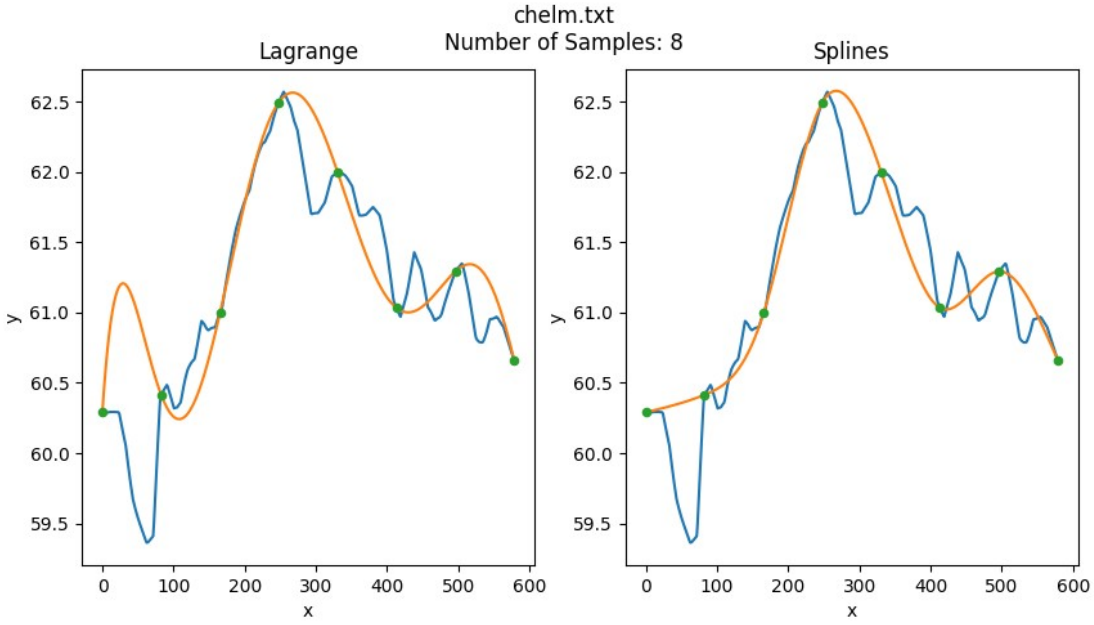
chelm.txt, ostrowa.txt, stale.txt, rozne_wzniesienia.txt, tczew_starogard.txt

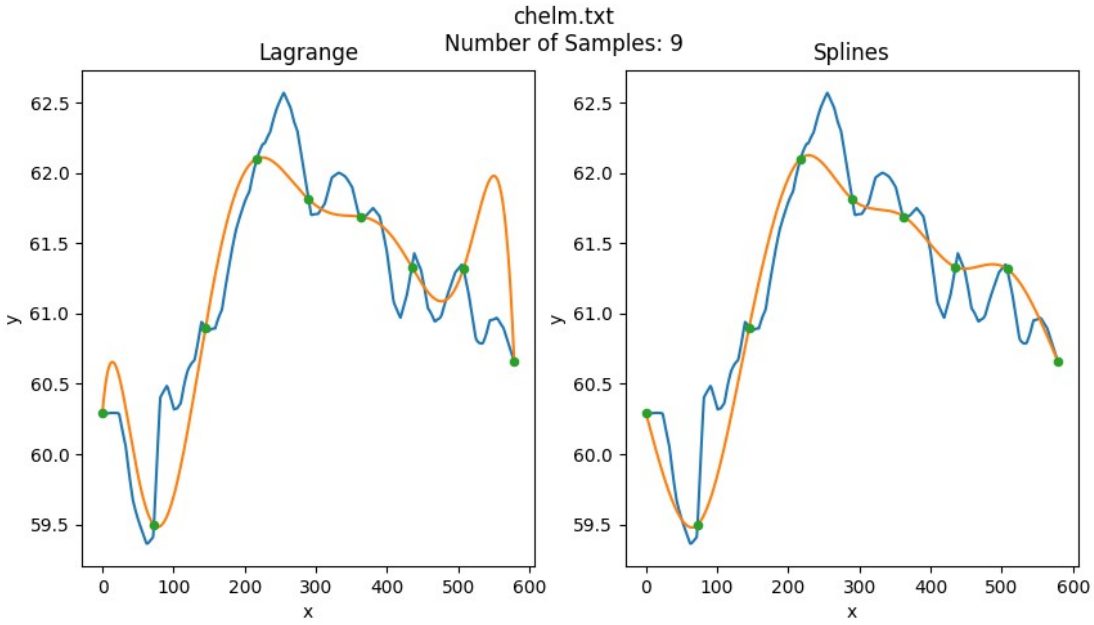
4. Wyniki

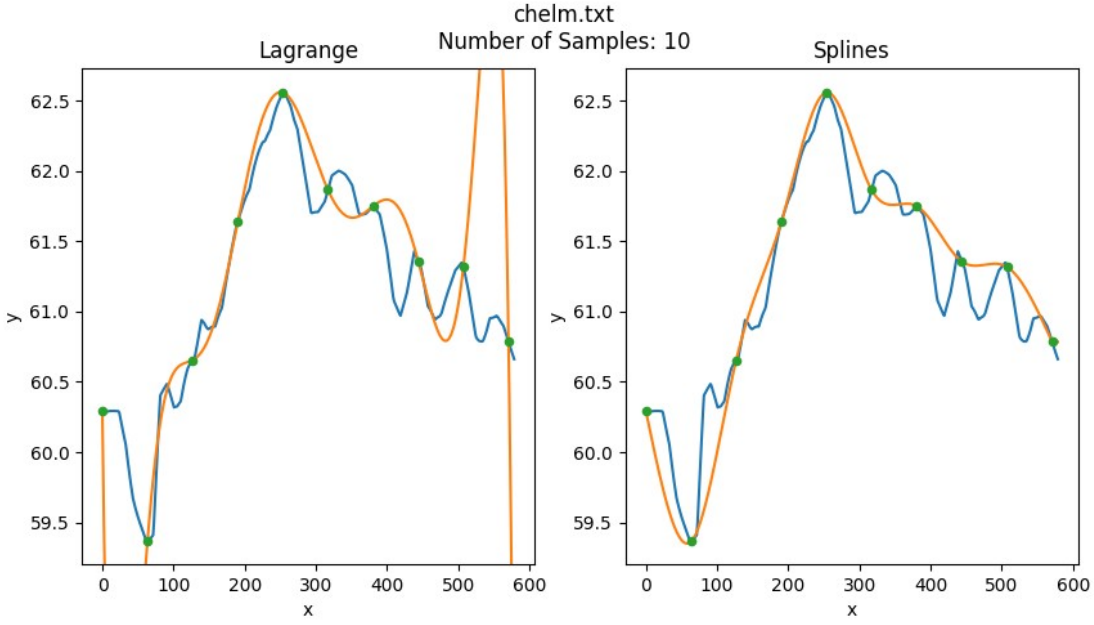

chelm.txt
Number of Samples: 3
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

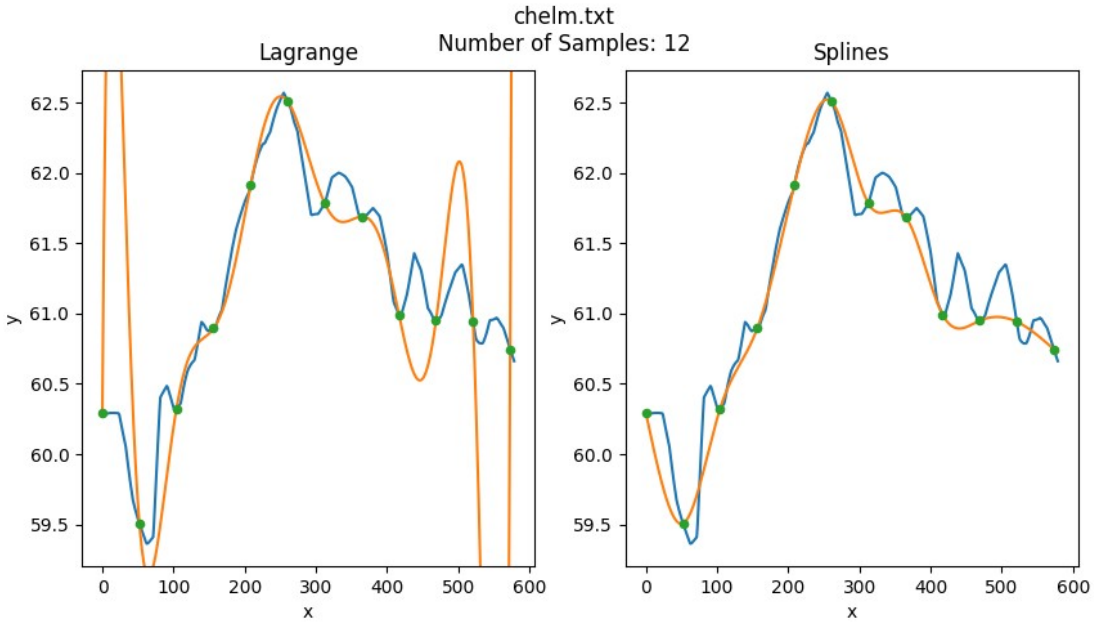

chelm.txt
Number of Samples: 5
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

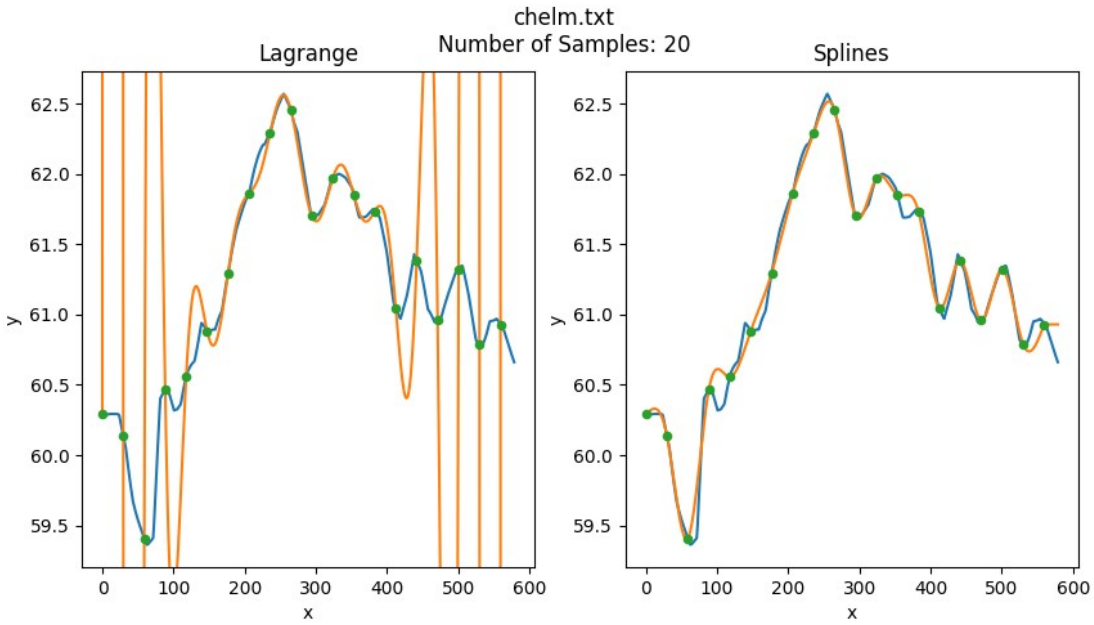

chelm.txt
Number of Samples: 7
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

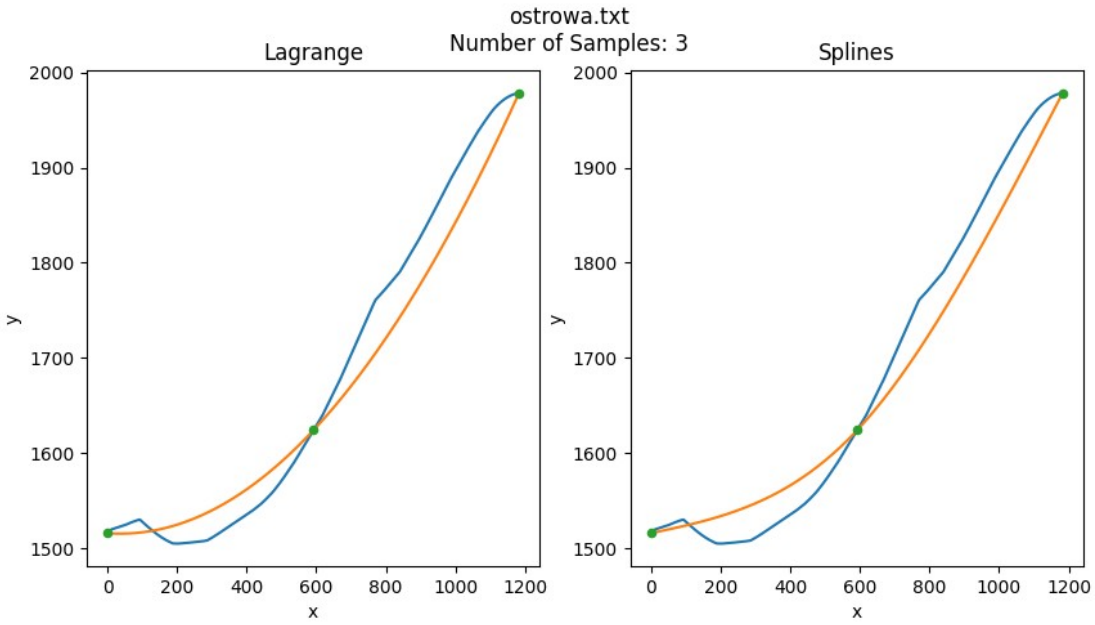

chelm.txt
Number of Samples: 8
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

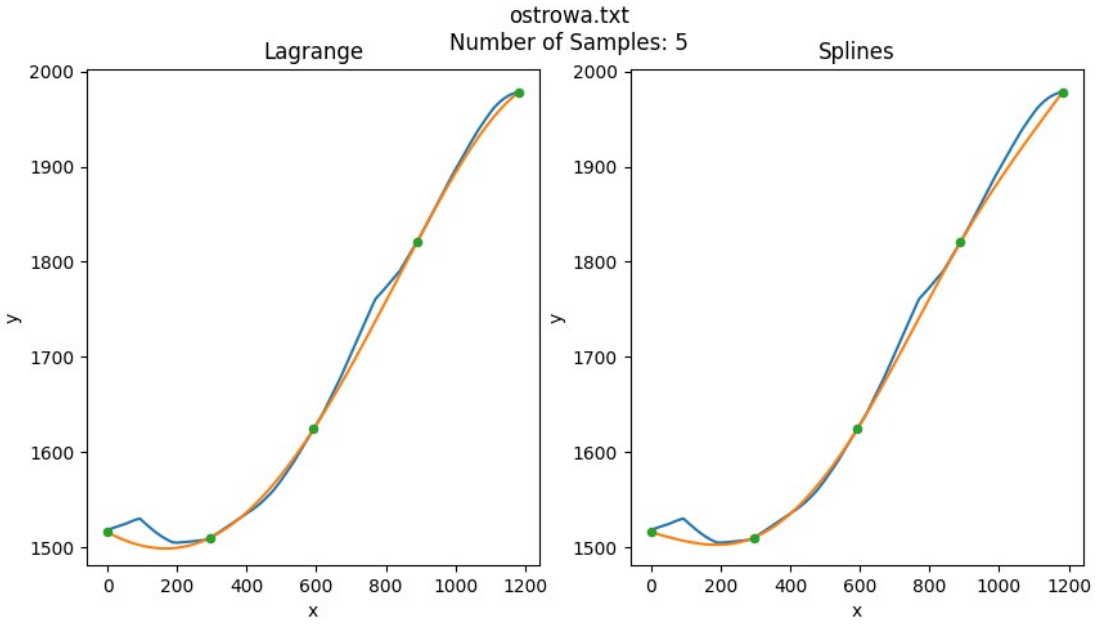

chelm.txt
Number of Samples: 9
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

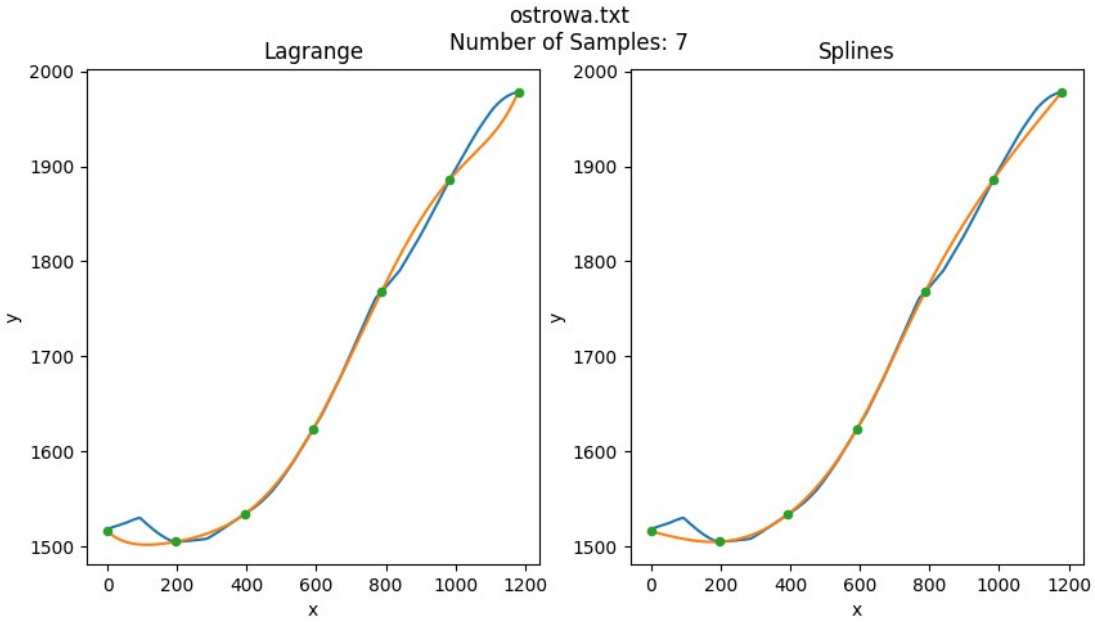

chelm.txt
Number of Samples: 10
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

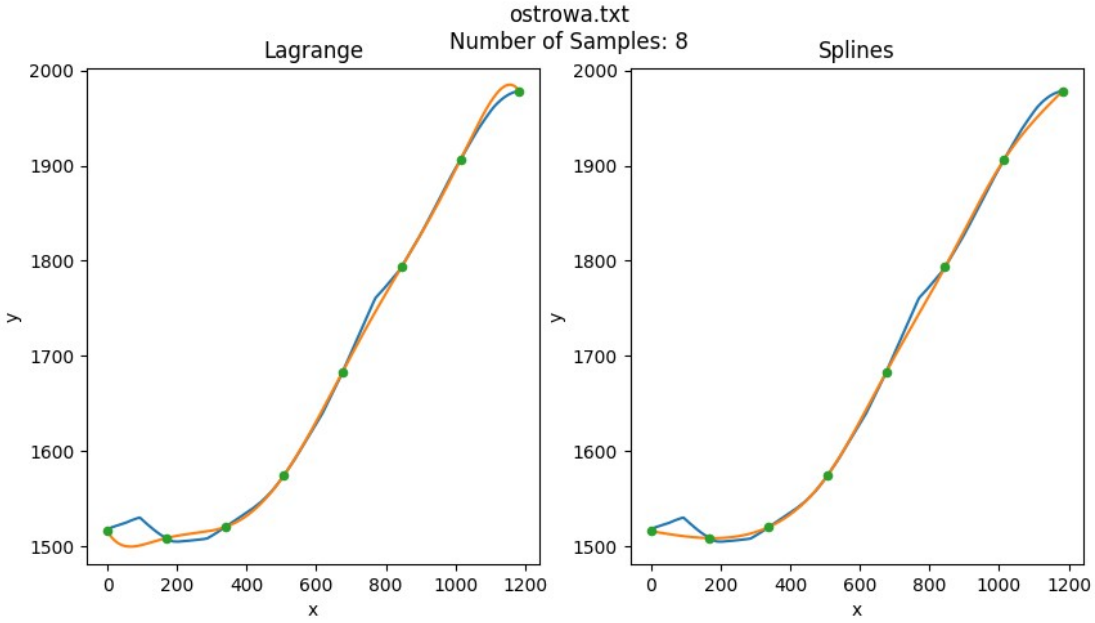

chelm.txt
Number of Samples: 12
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

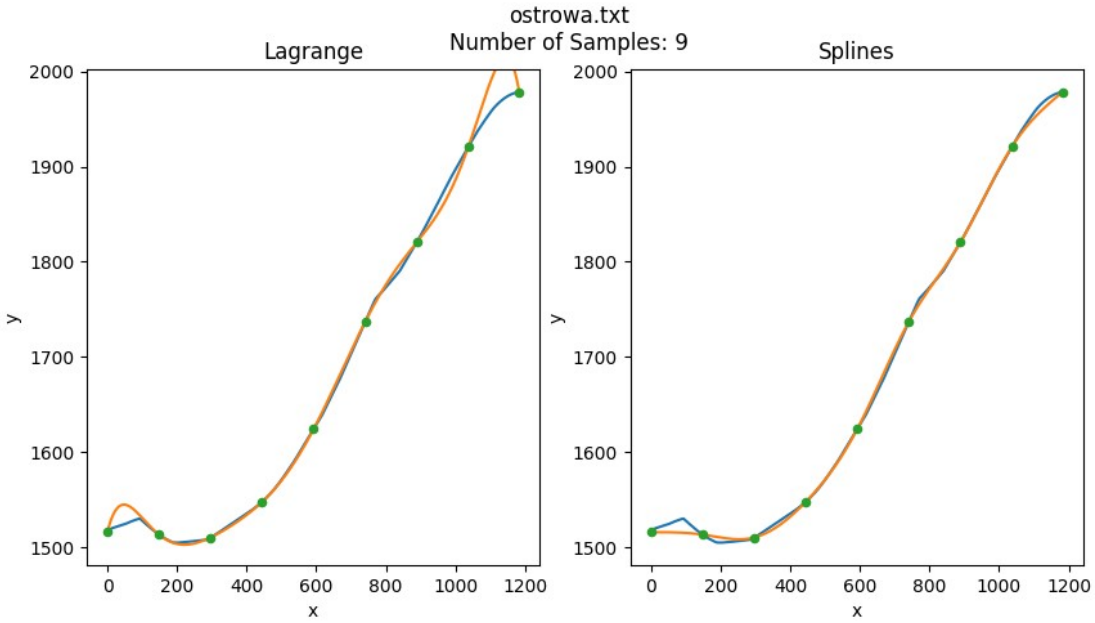

chelm.txt
Number of Samples: 20
Lagrange
Splines
y
x
62.5
62.0
61.5
61.0
60.5
60.0
59.5
0
100
200
300
400
500
600

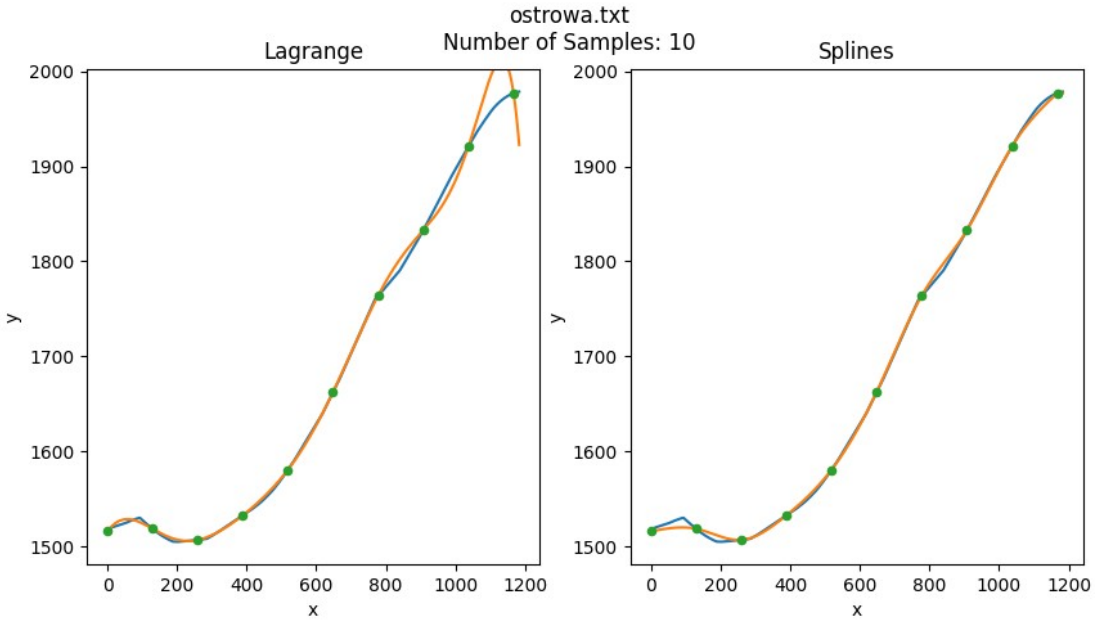

ostrowa.txt
Number of Samples: 3
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

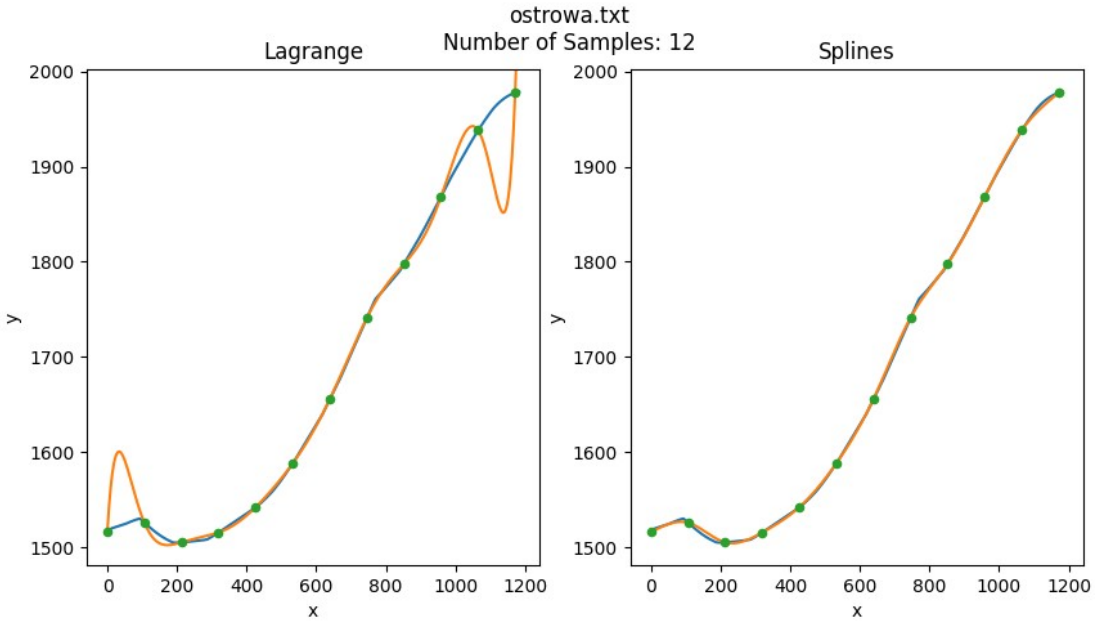

ostrowa.txt
Number of Samples: 5
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

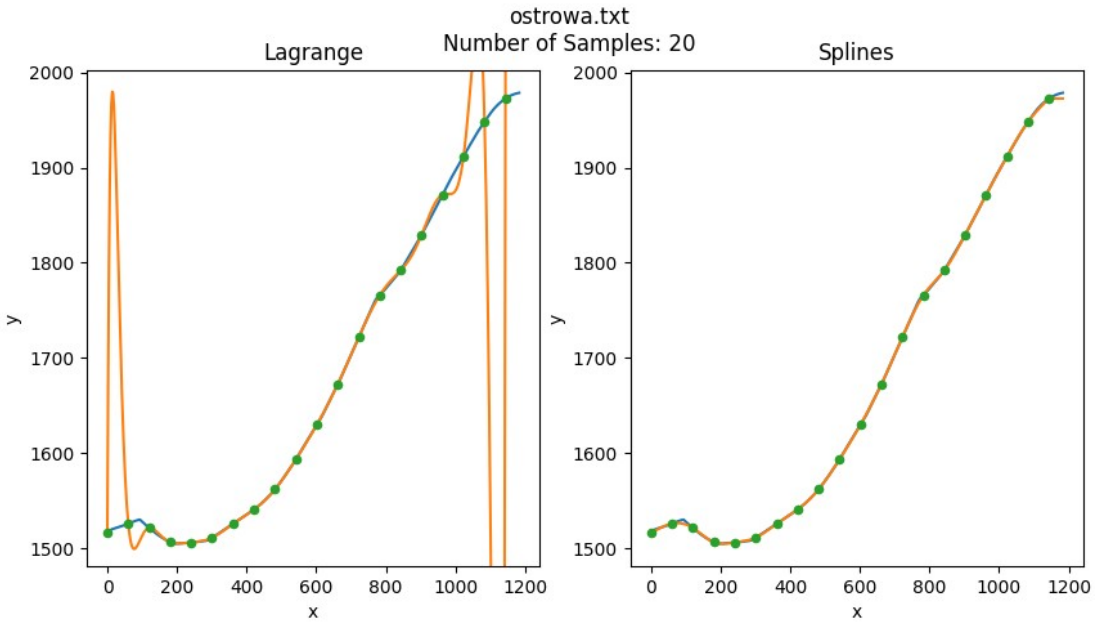

ostrowa.txt
Number of Samples: 7
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

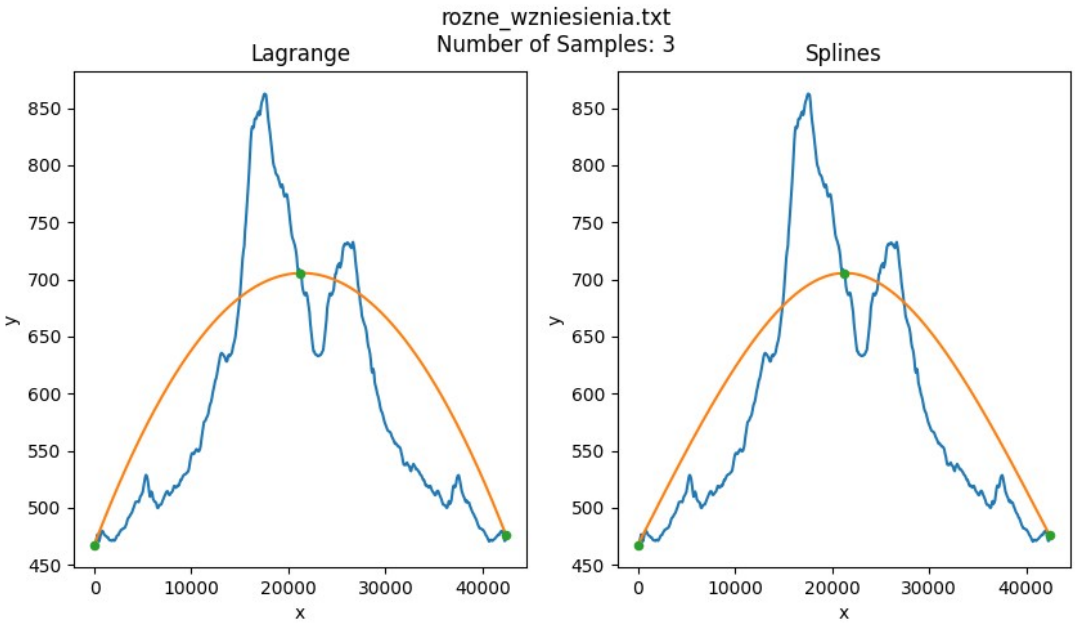

ostrowa.txt
Number of Samples: 8
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

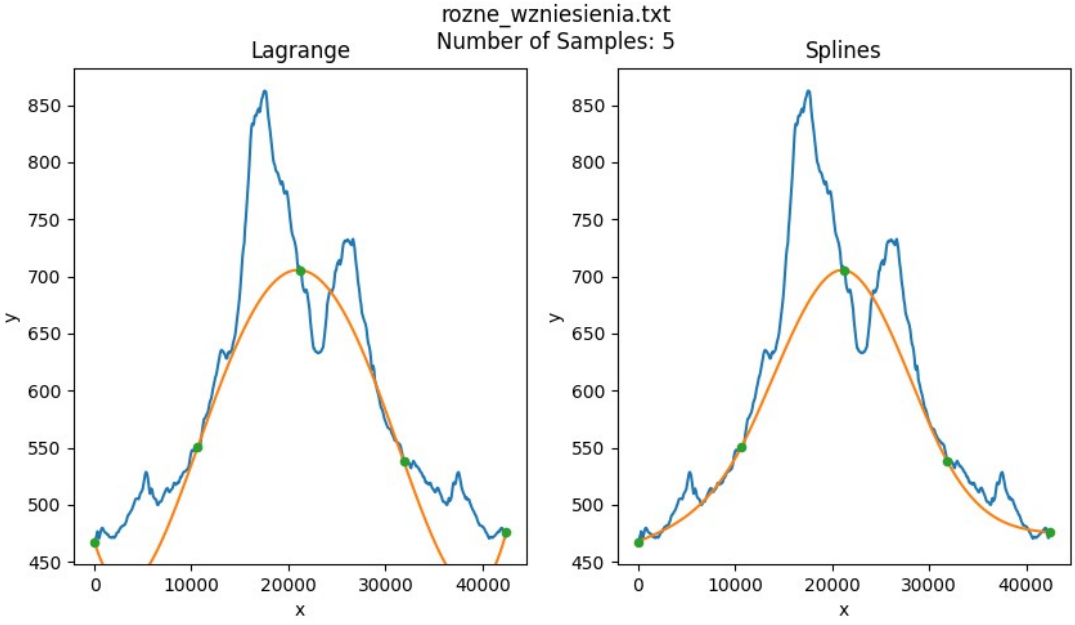

ostrowa.txt
Number of Samples: 9
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

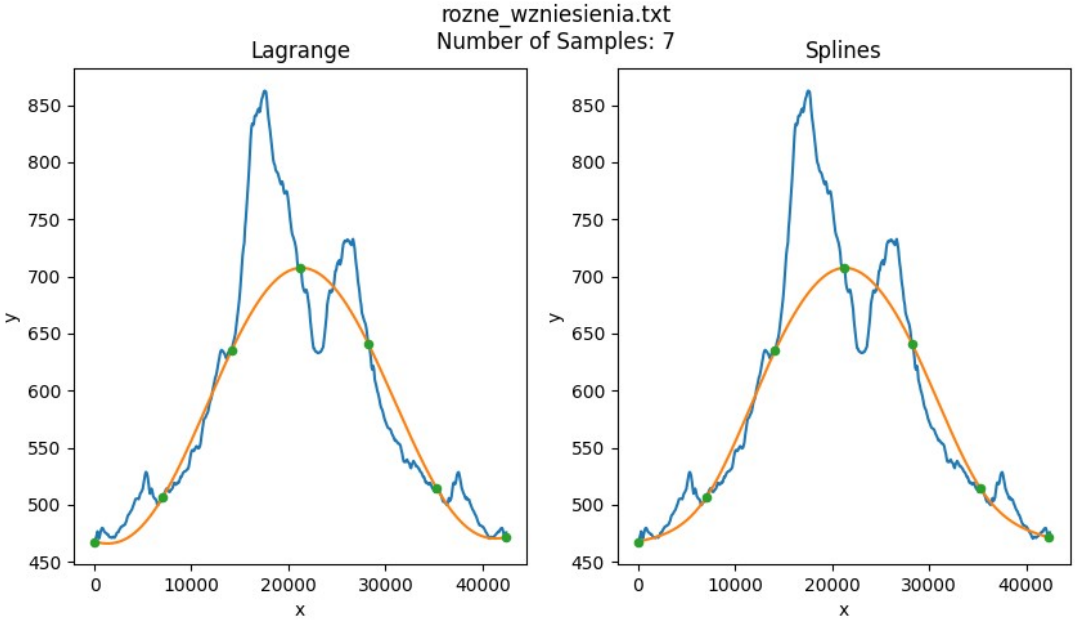

ostrowa.txt
Number of Samples: 10
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

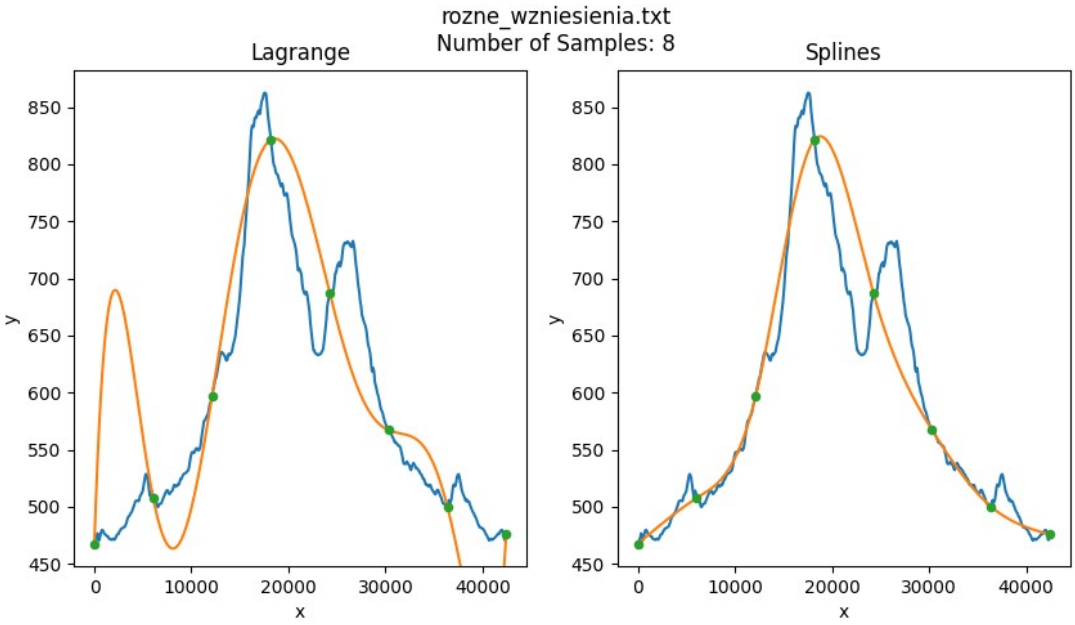

ostrowa.txt
Number of Samples: 12
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

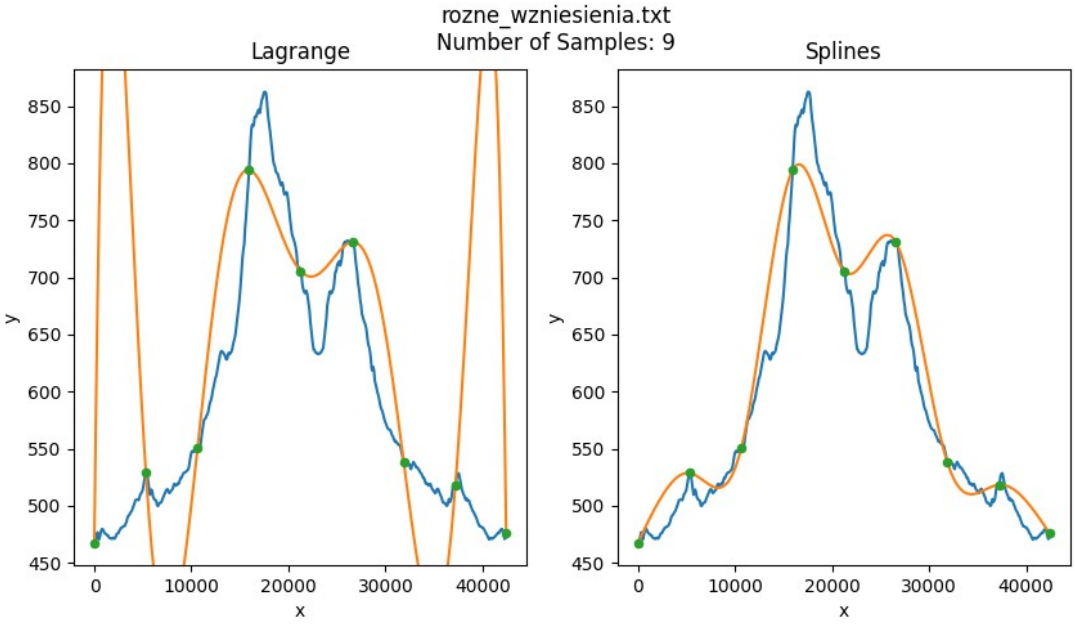

ostrowa.txt
Number of Samples: 20
Lagrange
Splines
y
x
2000
1900
1800
1700
1600
1500
0
200
400
600
800
1000
1200

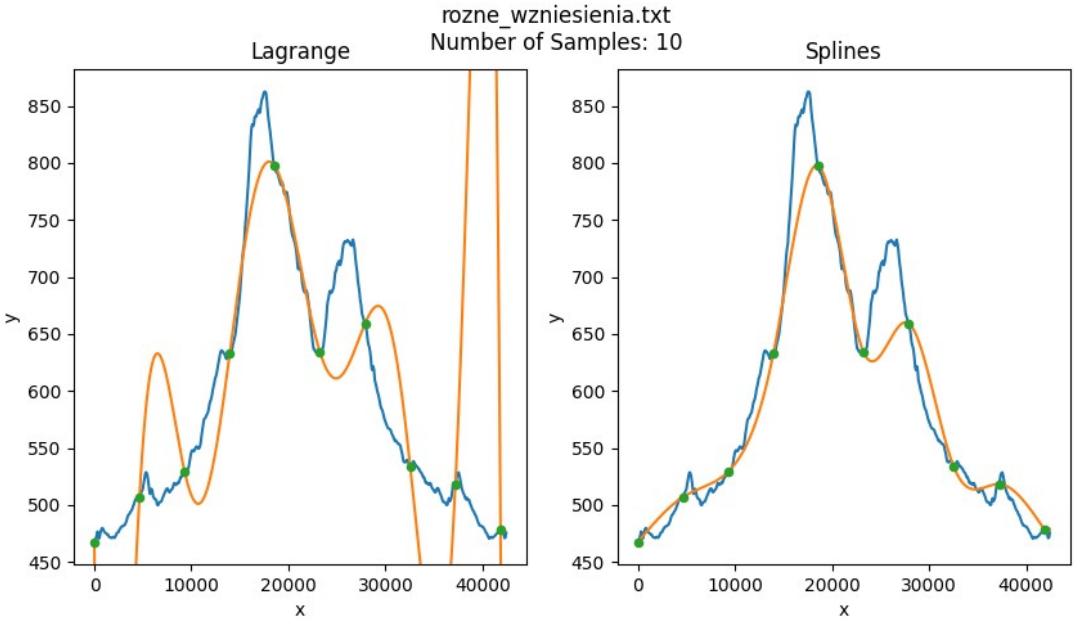

rozne_wzniesienia.txt
Number of Samples: 3
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000

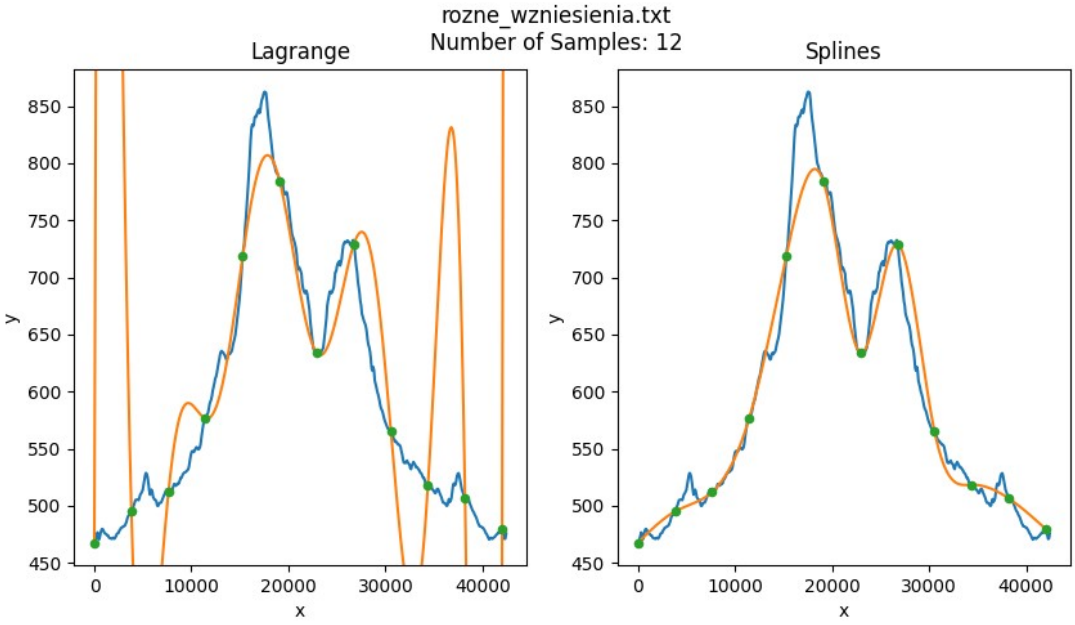

rozne_wzniesienia.txt
Number of Samples: 5
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000

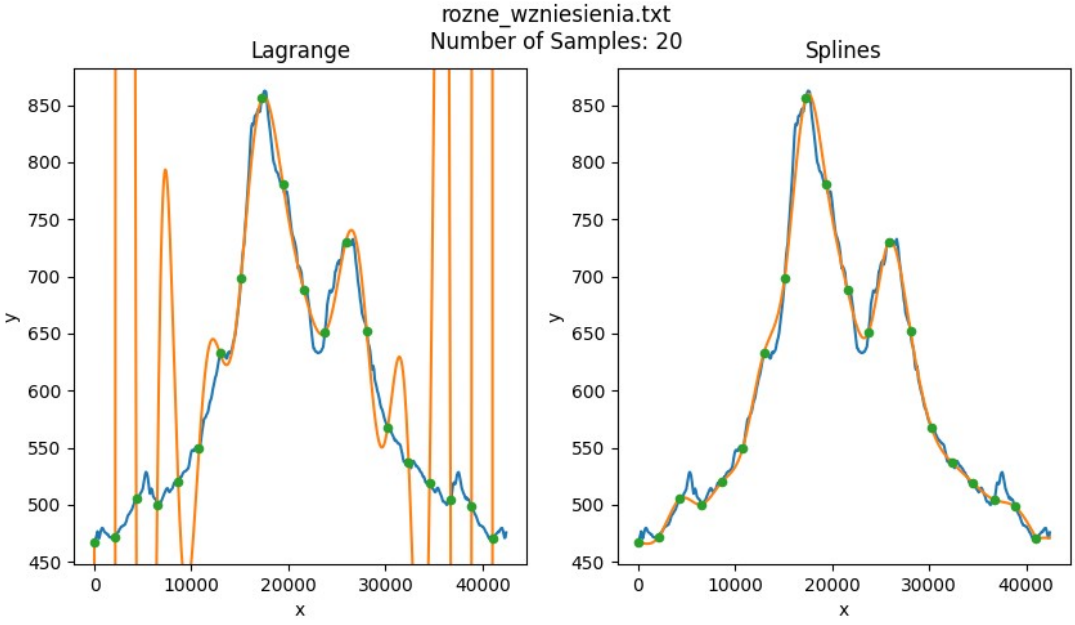

rozne_wzniesienia.txt
Number of Samples: 7
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000


rozne_wzniesienia.txt
Number of Samples: 8
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000


rozne_wzniesienia.txt
Number of Samples: 9
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000


rozne_wzniesienia.txt
Number of Samples: 10
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000


rozne_wzniesienia.txt
Number of Samples: 12
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000


rozne_wzniesienia.txt
Number of Samples: 20
Lagrange
Splines
y
x
850
800
750
700
650
600
550
500
450
0
10000
20000
30000
40000

5. Wnioski

Otrzymane wyniki pokazują przewagę metody funkcji sklejanych nad metodą Lagrange’a. Metoda Lagrange’a oferuje szybszą implementację oraz szybsze obliczenia, ale płaci za to dokładnością wyników. Przy niewielkich ilościach próbek obie metody mają podobne wyniki. Jednakże podczas metody funkcji sklejanych wraz z zwiększaniem ilości próbek wykres zbliża się do prawdziwego. Dzięki temu, przy odpowiedniej ilości próbek, jesteśmy w stanie otrzymać wykres o niewielkiej różnicy od oryginalnego. Sytuacja wygląda dużo gorzej przy metodzie Lagrange’a. Podczas tej metody po osiągnięciu zbliżonego wykresu przy pewnej ilości próbek, zwiększanie ilości próbek spowoduje pogorszenie się wyniku w postaci oscylacji na początku oraz końcu wykresu. Dalsze zwiększanie ilości próbek spowoduje zwiększenie się amplitudy oraz ilości okresów oscylacji.