Aproksymacja profilu wysokościowego

Sebastian Kwaśniak

2024-05-19

Wstęp

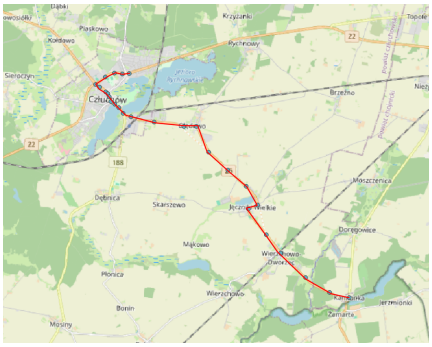
W ramach projektu wymagane jest zaimplementowanie dwóch sposobów aproksymacji interpolacyjnej do profilów wysokościowych. Metody te to: wykorzystująca wielomian Lagrange'a, oraz funkcje składane trzeciego stopnia.

Metoda Lagrange zwraca te same wyniki jak metoda Vandermonde, jednak nie jest wymagane rozwiązywanie układu równań liniowych. Metoda krzywych sklejanych trzeciego stopnia sprowadza się do rozwiązania układu równań. Do rozwiązania układu równań, zgodnie z zaleceniem z wykładu, wykrozystana zostanie metoda LU.

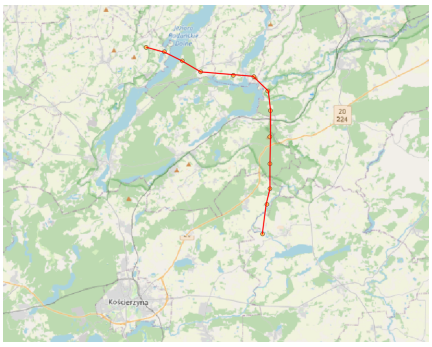
Dane

Dane zostały pobrane z serwisu OpenStreetMap i wyeksportowane za pomocą programu QGIS.

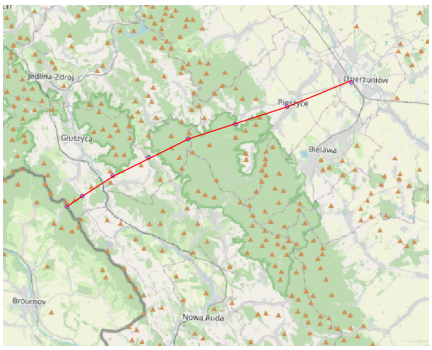
- Trasa 1 - trasa w miarę prosta, lekko poszarpana
- Trasa 2 - poszarpana
- Trasa 3 - jedno wzniesienie



Rysunek 1: Trasa 1



Rysunek 2: Trasa 2



Rysunek 3: Trasa 3

Metoda Lagrange

Funkcja F ma postać(1)

$$F(x) = \sum_{i=0}^{n} y_i \phi_i(x)$$

gdzie

$$\phi_i(x) = \prod_{j=0 \wedge j \neq i}^{n+1} \frac{x - x_j}{x_i - x_j}$$

jest bazą Lagrange'a.

W kodzie zostało to zaimplementowane w następujący sposób:

```
from numpy import linspace

def lagrange_polynomial(x, indexes, X, Y):
    result = 0
    for i in indexes:
        term = Y[i]
        for j in indexes:
            if i != j:
                term *= (x - X[j]) / (X[i] - X[j])
        result += term
    return result

def lagrange(X, Y, num_interpolation=9, num_evaluated=1000, indexes=None):
    if indexes is None:
        indexes = [int(i) for i in linspace(0, len(X) - 1, num_interpolation)]
    interpolated_X = list(linspace(X[0], X[-1], num_evaluated))
    interpolated_Y = [lagrange_polynomial(x, indexes, X, Y) for x in interpolated_X]
    return interpolated_X, interpolated_Y, indexes
```

Metoda krzywych sklejanych trzeciego stopnia

Funkcja F ma postać(2)

$$F(x) = S_i(x); x \in [x_i, x_{i+1}]$$

czyli szereg połączonych wielomianów $S_i(x)$ takich, że $\deg(S_i) = 3$. W celu uzyskania układów równań, z których pozyskane zostaną współczynniki $S_i(x)$ przyjęte są założenia:

$$\begin{aligned}
S_i(x_i) &= y_i \\
S_i(x_{i+1}) &= y_{i+1} \\
S'_{j-1}(x_i) &= S'_j(x_i); x = 1..n-1 \\
S''_{j-1}(x_i) &= S''_j(x_i); x = 1..n-1 \\
S''_0(x_0) &= 0 \\
S''_0(x_n) &= 0
\end{aligned}$$

Znalezienie wielomianów S sprowadza się do rozwiązania powyższego układu równań. Można z powyższych układów wyprowadzić wzory na b oraz d w zależności od c, a następnie rozwiązać cały układ równań dla wektora $c = [c_0, ..., c_{n-1}]$.

W kodzie zaimplementowane jest to w następujący sposób:

```
from numpy import linspace
from lu import lu_solve

def solve_splines(x, n, indexes, X, a, b, c, d):
    ix = n-1
    for ix_num in range(len(indexes) - 1):
        if X[indexes[ix_num]] <= x < X[indexes[ix_num + 1]]:
            ix = ix_num
            break
    h = x-X[indexes[ix]]
    return a[ix] + b[ix] * h + c[ix] * h**2 + d[ix] * h ** 3
```

```
def splines(X, Y, num_interpolation=15, num_evaluated=1000, indexes=None):
    if indexes is None:
        indexes = [int(i) for i in linspace(0, len(X) - 1, num_interpolation)]
    n = len(indexes)
    a = [Y[ix] for ix in indexes]
    b = []
    d = []
    h = [X[indexes[i+1]] - X[indexes[i]] for i in range(n-1)]
    A = [[0 for _ in range(n)] for _ in range(n)]
    vec = [[0] for _ in range(n)]
    for i in range(1, n-1):
        A[i][i] = 2 * (h[i-1] + h[i])
        A[i][i-1] = h[i-1]
        A[i][i+1] = h[i]
        vec[i][0] = 3 * ((Y[indexes[i+1]] - Y[indexes[i]])/h[i] - \
                         (Y[indexes[i]] - Y[indexes[i-1]])/h[i-1])
    A[0][0] = 1
    A[n-1][n-1] = 1
    c = lu_solve(A, vec)
    for i in range(n-1):
        d.append((c[i+1] - c[i])/(3 * h[i]))
        b.append((Y[indexes[i+1]] - Y[indexes[i]])/h[i] - h[i]/3 * (2 * c[i] + c[i+1]))
    b.append(0)
    d.append(0)
    interpolated_X = list(linspace(X[0], X[-1], num_evaluated))
    interpolated_Y = [solve_splines(x, n, indexes, X, a, b, c, d) for x in interpolated_X]
    return interpolated_X, interpolated_Y, indexes
```

Pierwiastki wielomianu Czebyszewa

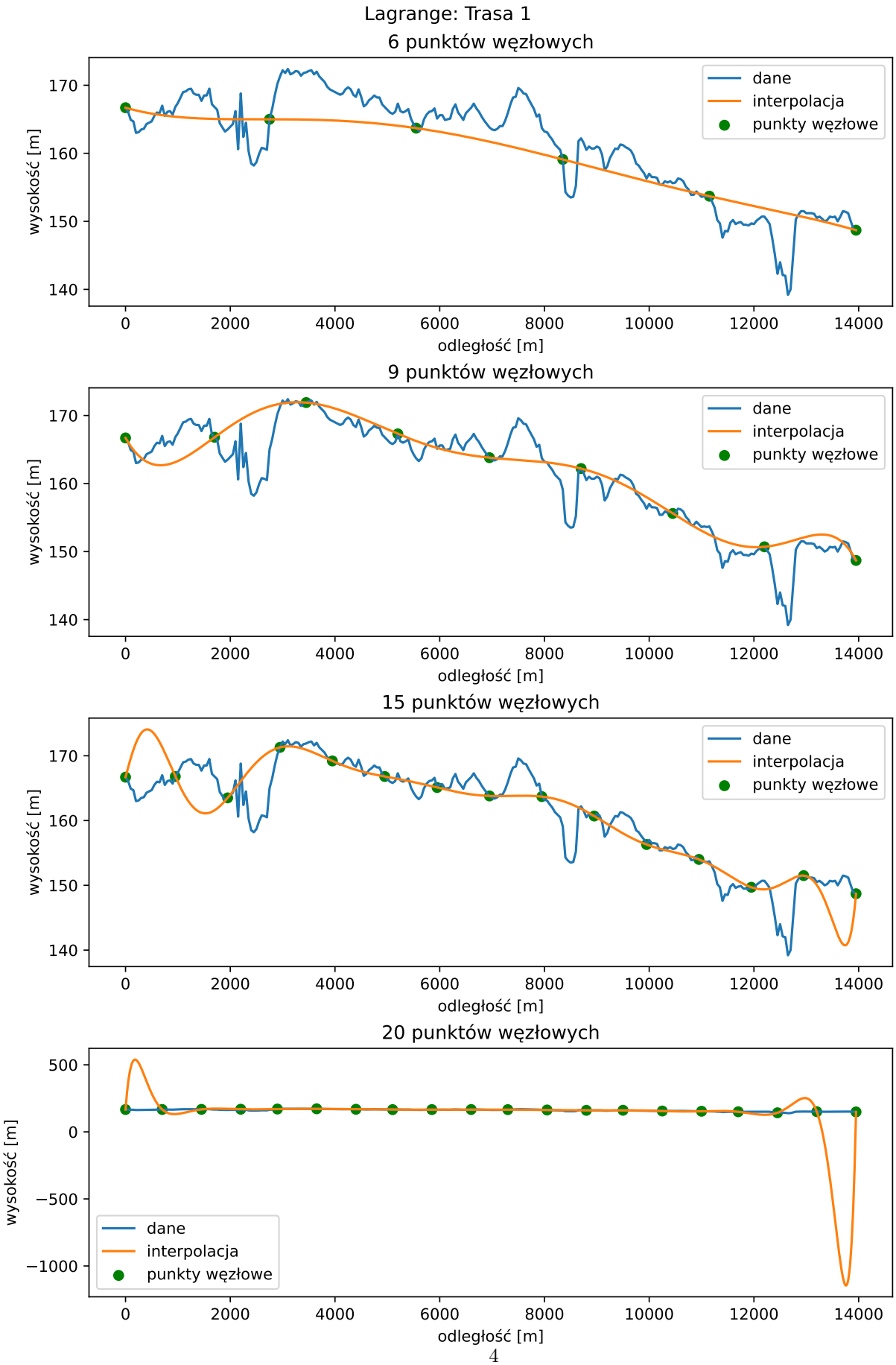
W trakcie pisania korzystane będzie także z węzłów Czebyszewa, żeby zminimalizować efekt Rungego. Pierwiastki te są w postaci:

$$x_k = \cos(\frac{2k+1}{2n}\pi), k = 0..n-1$$

Który w kodzie został zaimplementowany w następujący sposób:

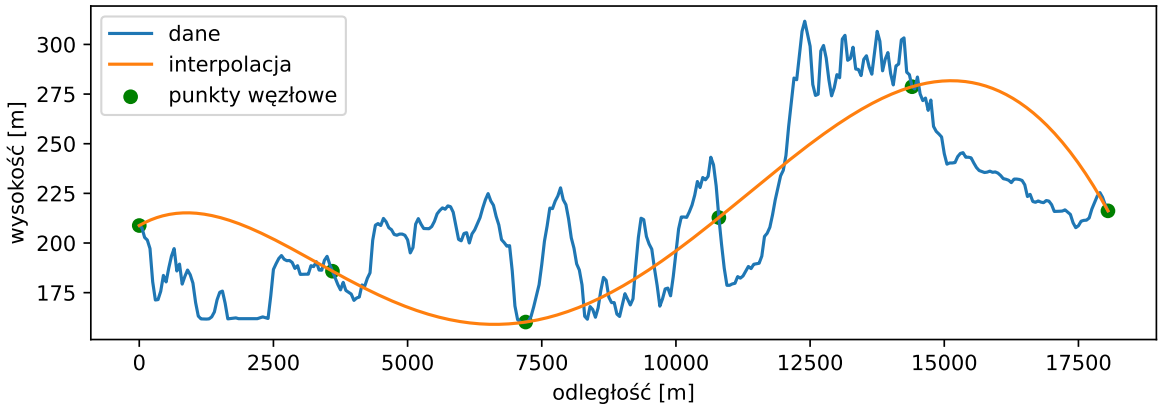
```
def generate_czebyszew(N, data_size):
    import math
    def fu(k):
        return data_size/2 * math.cos((2 * k + 1)/(2 * N) * math.pi) + data_size/2
    return [int(fu(k)) for k in range(N-1, -1, -1)] + [data_size]
```

Analiza interpolacji wielomianowej tras

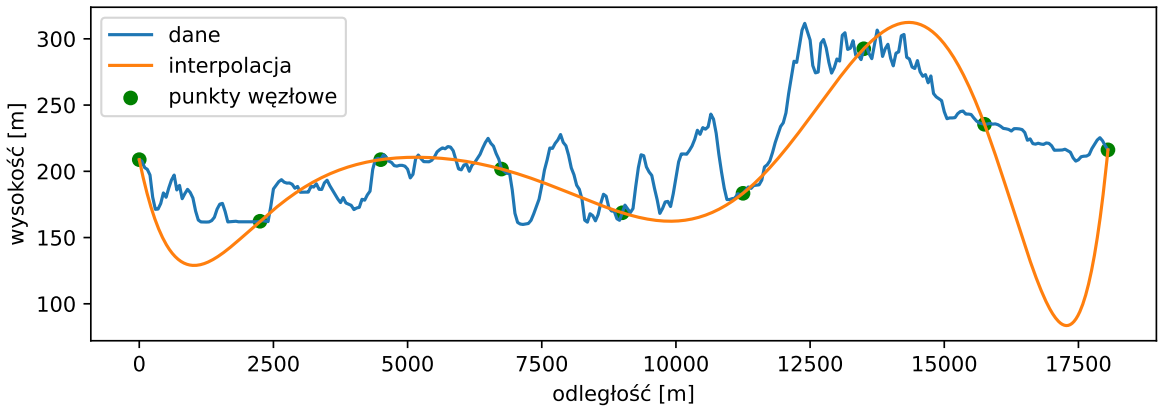


Lagrange: Trasa 2

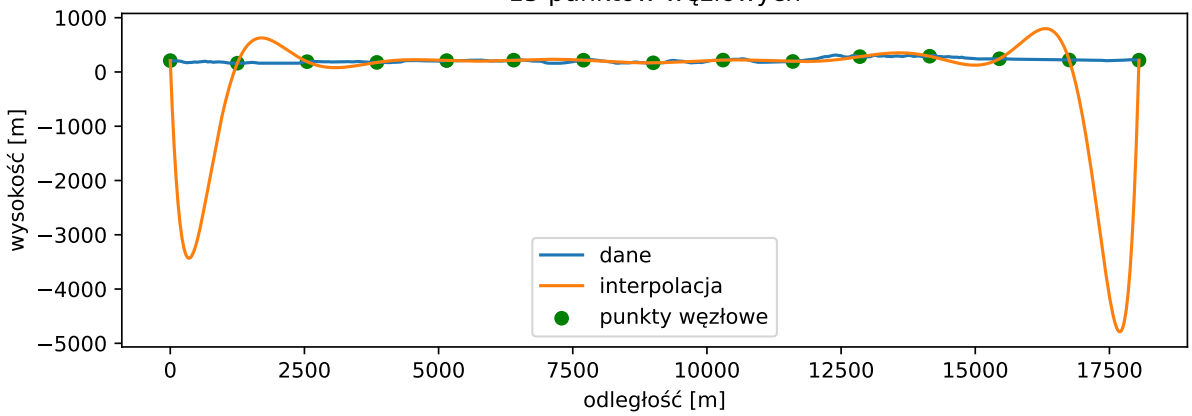
6 punktów węzłowych



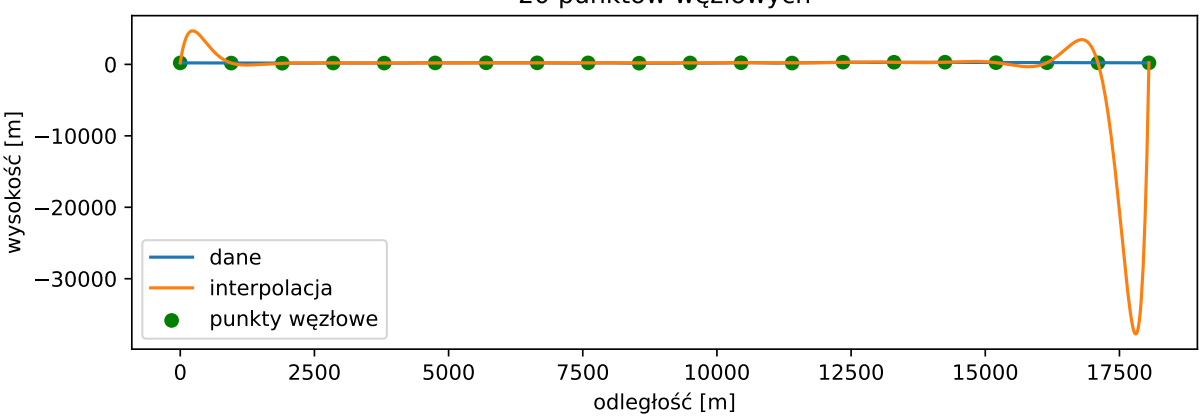
9 punktów węzłowych



15 punktów węzłowych



20 punktów węzłowych




Lagrange: Trasa 3
6 punktów węzłowych
dane
interpolacja
punkty węzłowe
wysokość [m]
odległość [m]
9 punktów węzłowych
15 punktów węzłowych
20 punktów węzłowych

Trasa 1

Dla tej trasy widać, że dla odcinku o stałym nachyleniu (np. od 9km do 11km) odzwierciedlenie jest najlepsze. Problemy są głównie przy nagłych spadkach i wzniesieniach, gdzie nawet zwiększenie liczby węzłów nie wpływa znacząco na poprawienie interpolacji. Dodatkowo, już tutaj można zauważyć efekt Rungego przy 15 węzłach, a przy 20 już znacząco wpływa na czytelność całego wykresu.

Trasa 2

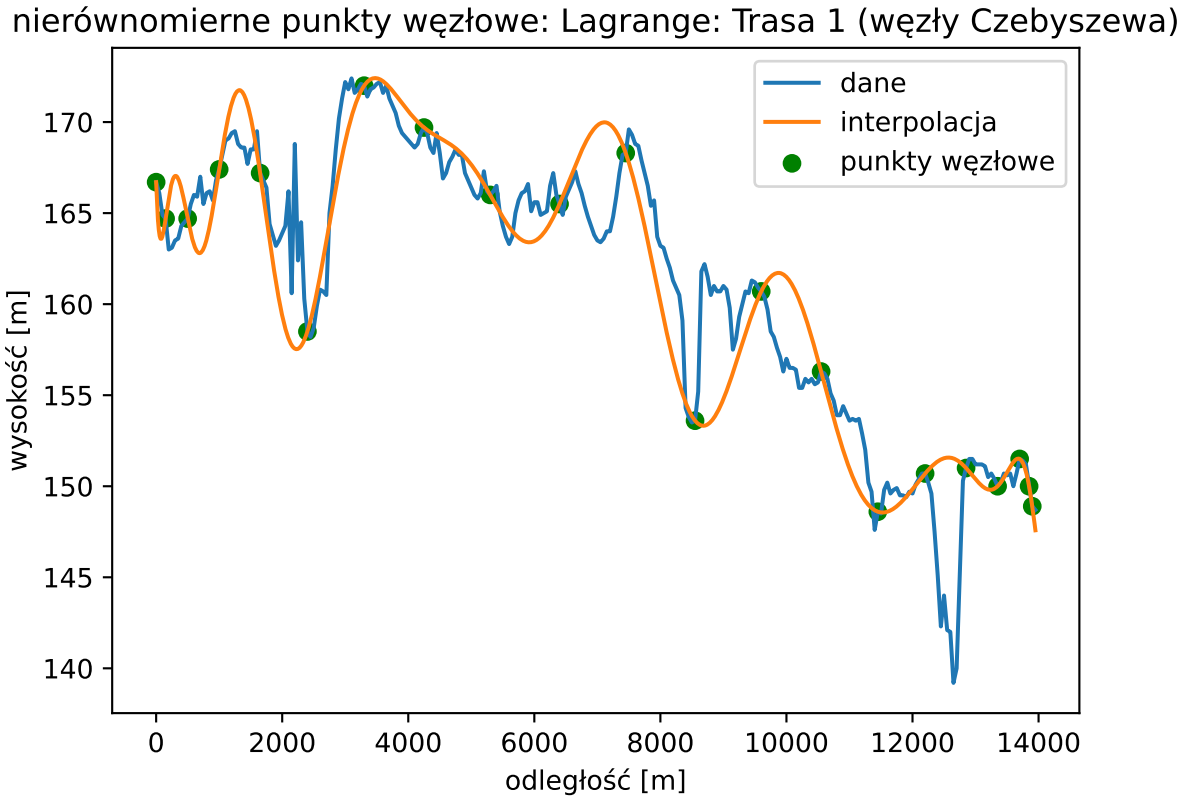
Ta trasa jest najbardziej poszarpaną. Łatwo zauważyć, że interpolacja tutaj jest bezużyteczna, a dodatkowo jeszcze trzeba sobie poradzić z efektem Rungego przy większej ilości węzłów. **Na jakość interpolacji znacząco wpływają nagłe wzniesienia, a dokładniej rozmieszczenia punktów**.

Trasa 3

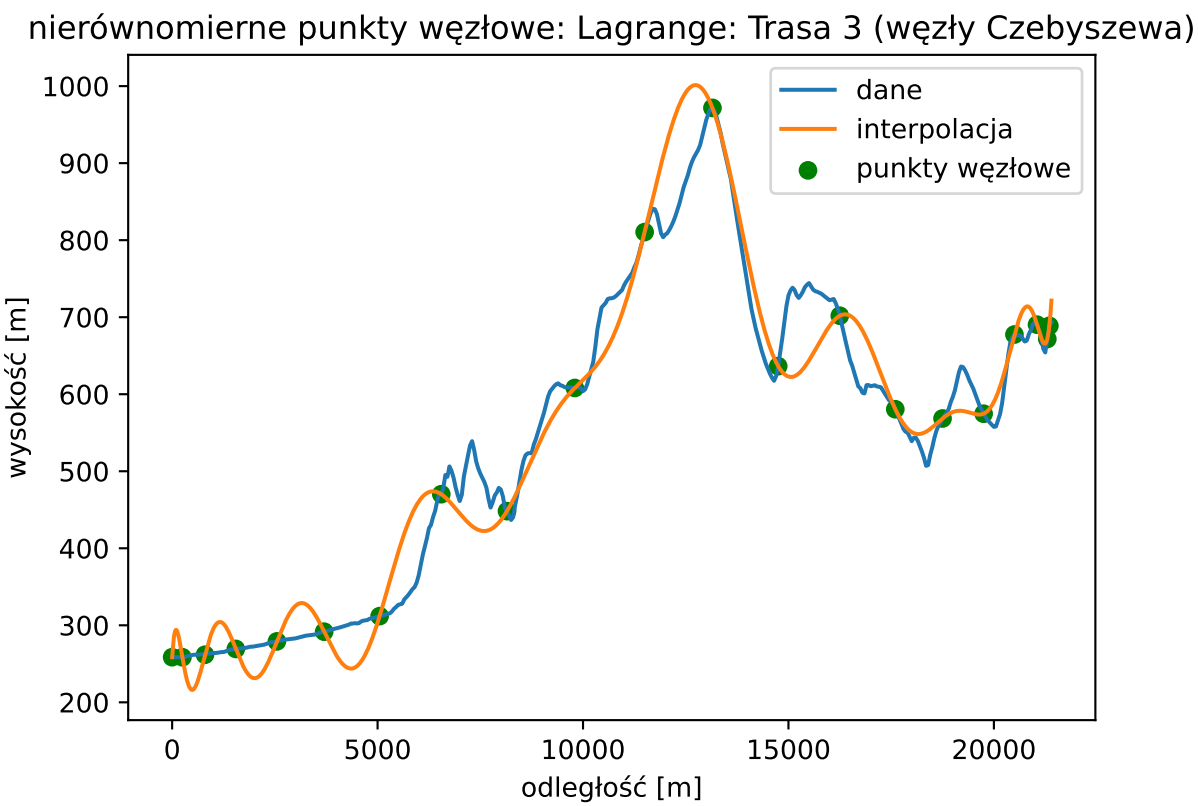
Przy tej trasie, dla 15 węzłów, gdyby pozbyć się efektu Rungego, to interpolacja wyglądałaby całkiem sensownie.

Eliminacja efektu Rungego

Przy pomocy węzłów Czebyszewa można spróbować zminimalizować efekt Rungego.



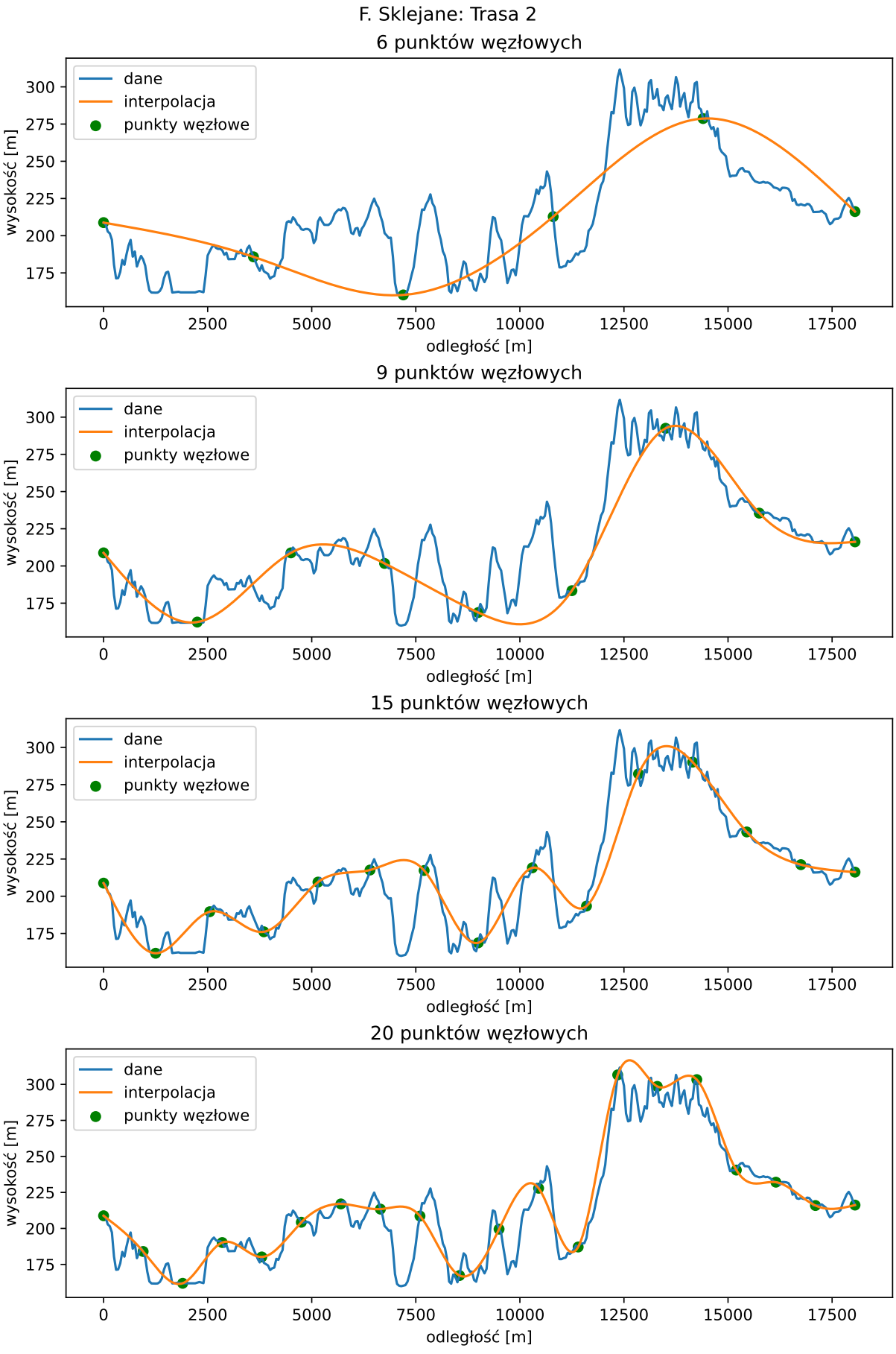
Efekt Rungego dla 20 węzłów został wyeliminowany, lecz nadal interpolacja nie jest w pełni poprawna i nie daje zadowalających wyników. Szczególnie widać to po 12 kilometrze.

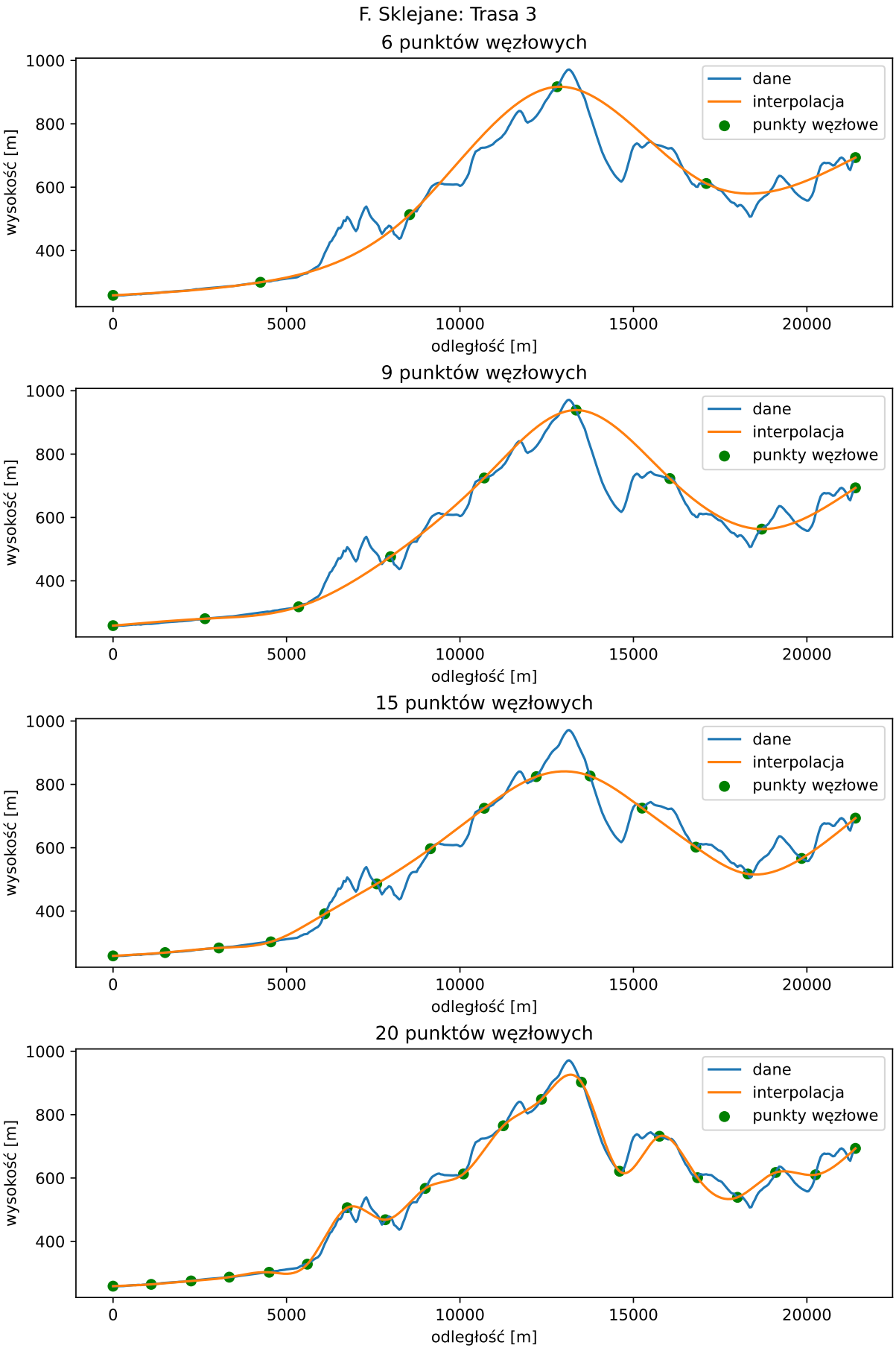


Podsumowanie metody Lagrange'a

Wszystkie te trasy mają problem z efektem Rungego. Można spróbować wyeliminować ten efekt, poprzez skorzystanie z węzłów Czebyszewa, lecz wyniki nadal nie są wystarczająco zadowalające.

Analiza interpolacji funkcjami sklejanymi trzeciego stopnia tras


F. Sklejane: Trasa 2
6 punktów węzłowych
9 punktów węzłowych
15 punktów węzłowych
20 punktów węzłowych
dane
interpolacja
punkty węzłowe
wysokość [m]
odległość [m]
175
200
225
250
275
300
0
2500
5000
7500
10000
12500
15000
17500


F. Sklejane: Trasa 3
6 punktów węzłowych
9 punktów węzłowych
15 punktów węzłowych
20 punktów węzłowych
dane
interpolacja
punkty węzłowe
wysokość [m]
odległość [m]
1000
800
600
400
0
5000
10000
15000
20000

Trasa 1

W porównaniu do metody wielomianowej, tutaj są od razu lepsze wyniki. Także na pierwszy rzut oka widać brak efektu Rungego.

Trasa 2

Na tej trasie, przez liczne poszarpania, metoda wielomianowa od początku miała problemy. Tutaj widać, że interpolacja jest znacząco lepsza i użyteczna.

Trasa 3

W tym wypadku jest idealnie zobrazowane zjawisko, że w tej metodzie im więcej węzłów, tym lepsza interpolacja.

Podsumowanie interpolacji funkcjami sklejanymi

Jak widać na każdym wykresie przy zwiększaniu ilość węzłów wyniki są coraz lepsze. Tutaj nie ma problemu z efektem Rungego, dlatego można uzyskać coraz dokładniejsze interpolacje. Jednak, przy np. przetwarzaniu sygnałów, niepożądane jest bycie aż tak dokładnym i **"wygładzenie" może być pożądanym efektem**. Niemniej, każda trasa jest lepiej interpolowana, w szczególności można spojrzeć na 20 węzłów przy trasie trzeciej, gdzie jest to najbardziej widoczne.

Wnioski

Obie metody są przydatne, ale interpolacja z wykorzystaniem wielomianu Lagrange'a jest ograniczona przez efekt Rungego. Zjawisko to powoduje oscylacje na krańcach przedziału, wraz ze wzrostem ilości punktów węzłowych, co pogarsza jakość interpolacji.

Krzywe sklejane za to, radzą sobie zarówno z terenami o wolnozmiennym nachyleniu jakość i o nagłych zmianach nachylenia, a metoda wielomianowa ma problem z ostrymi krawędziami.

Efekt Rungego można wyeliminować za pomocą węzłów Czebyszewa, przez co wyniki są lepsze. Im bardziej nieregularna powierzchnia, tym więcej węzłów trzeba do poprawnej interpolacji.

Źródła

1. Wikipedia. Interpolacja wielomianowa — Wikipedia, the free encyclopedia. http://pl.wikipedia.org/w/index.php?title=Interpolacja%20wielomianowa&oldid=70764886; 2024.

2. Wikipedia. Interpolacja funkcjami sklejanymi — Wikipedia, the free encyclopedia. http://pl.wikipedia.org/w/index.php?title=Interpolacja%20funkcjami%20sklejanymi&oldid=70098639; 2024.