KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Skaitiniai metodai ir algoritmai (P170B115)

3 laboratorinio darbo ataskaita

Atliko:
IFF-1/9 gr. studentas
Martynas Kuliešius

Priėmė:
Doc. Dalia Čalnerytė
Doc. Andrius Kriščiūnas

KAUNAS 2023

TURINYS

1. I UŽDUOTIS. INTERPOLIAVIMAS DAUGIANARIU.

I užduotis. Interpoliavimas daugianariu.

I lentelėje duota interpoliuojamos funkcijos analitinė išraiška. Pateikite interpoliacinės funkcijos išraišką naudodami ***I lentelėje*** nurodytas bazines funkcijas, kai:

a. Taškai pasiskirstę tolygiai.
b. Taškai apskaičiuojami naudojant Čiobyševo abscises.

Interpoliavimo taškų skaičių parinkite laisvai, bet jis turėtų neviršyti 30. Pateikite du grafikus, kai interpoliuojančios funkcijos apskaičiuojamos naudojant skirtingas abscises ir gautas interpoliuojančių funkcijų išraiškas. Tame pačiame grafike vaizduokite duotąją funkciją, interpoliuojančią funkciją ir netiktį.

Gautas varianto numeris - 3. Gauta užduotis pirmoje lentelėje :

3	$e^{-x^2} \cdot \cos(x^2) \cdot (x-3);\ -3 \le x \le 2$	Čiobyševo

pav. 1 Pirmos užduoties variantas

Programos kodas :

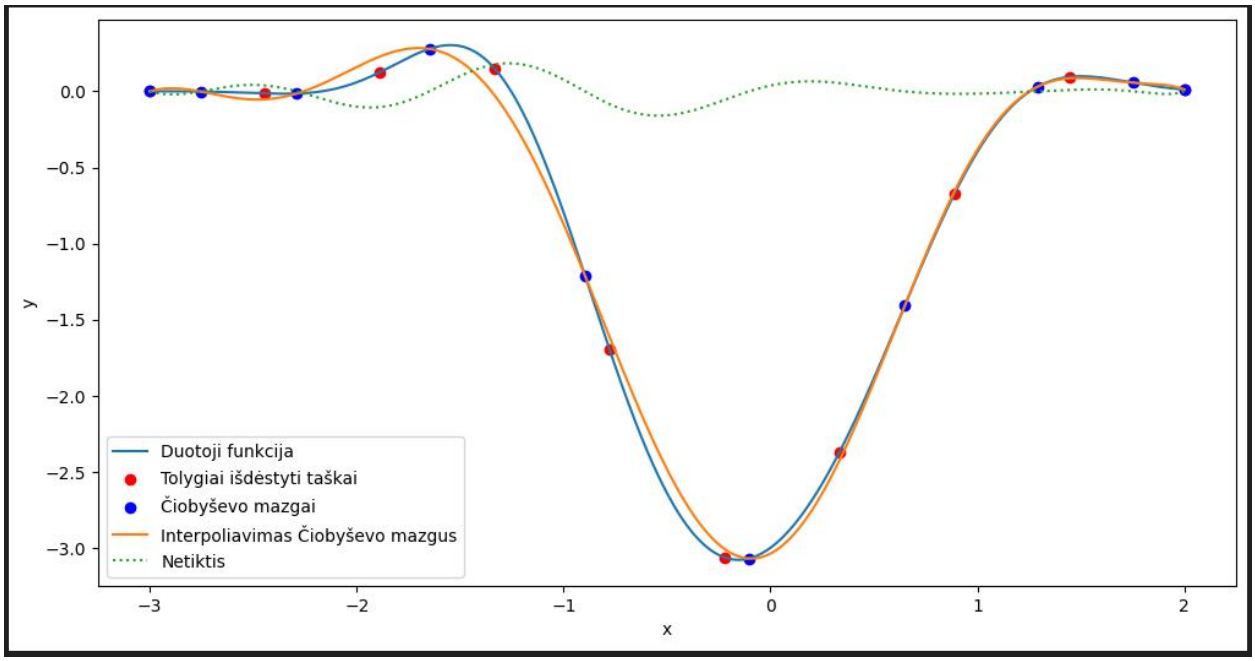
```python
import numpy as np
import matplotlib.pyplot as plt

# Given analytical function
def f(x):
    return np.exp(-x**2) * np.cos(x**2) * (x - 3)

# Function to calculate interpolating polynomial using Chebyshev nodes
def interpolate_chebyshev_nodes(x_values, y_values, x):
    result = 0
    for i in range(len(x_values)):
        term = y_values[i]
        for j in range(len(x_values)):
            if j != i:
                term = term * (x - x_values[j]) / (x_values[i] - x_values[j])
        result += term
    return result

# Generate evenly spaced points and Chebyshev nodes
num_points = 20  # You can adjust this value
evenly_spaced_x = np.linspace(-3, 2, num_points)
chebyshev_nodes_x = np.cos((2 * np.arange(1, num_points + 1) - 1) * np.pi / (2 * num_points))
chebyshev_nodes_x = np.interp(chebyshev_nodes_x, (chebyshev_nodes_x.min(), chebyshev_nodes_x.max()), (-3, 2))

# Calculate y values for the given analytical function
evenly_spaced_y = f(evenly_spaced_x)
chebyshev_nodes_y = f(chebyshev_nodes_x)

# Generate x values for plotting
x_values_for_plotting = np.linspace(-3, 2, 1000)

# Calculate y values for the analytical function
analytical_function_y = f(x_values_for_plotting)
```

Pav. 2 Programinis kodas 1/2

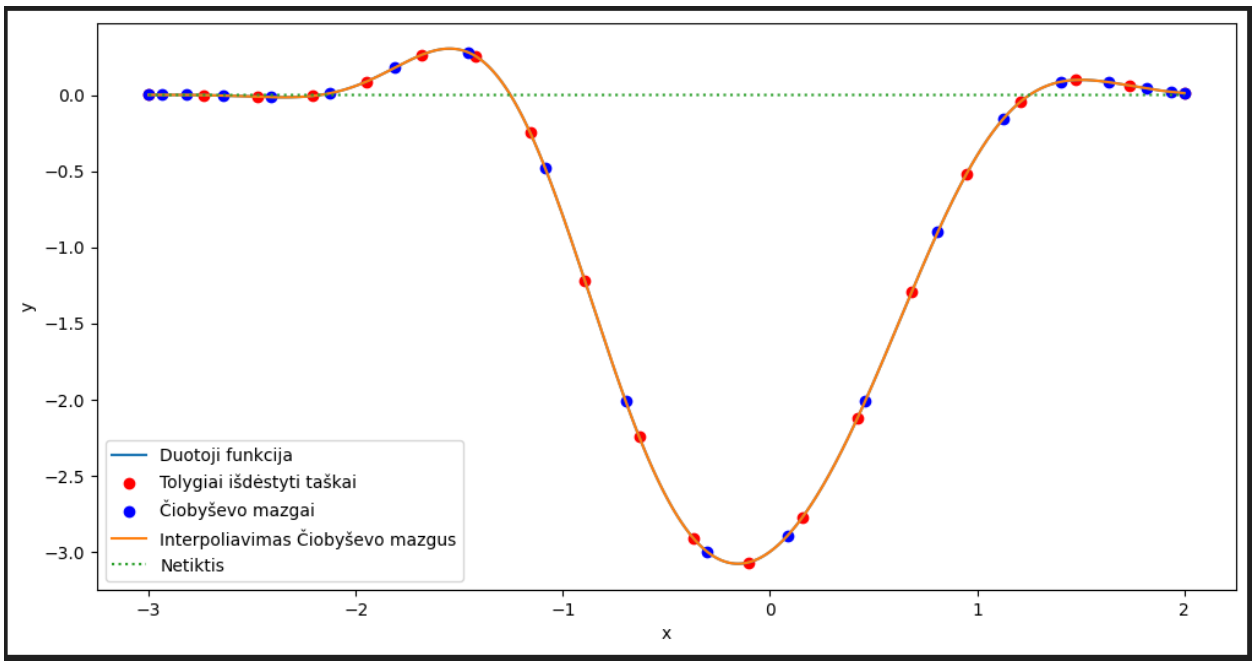
```python
# Calculate y values for the analytical function
analytical_function_y = f(x_values_for_plotting)

# Calculate y values for interpolation using Chebyshev nodes
interpolation_chebyshev_nodes = [interpolate_chebyshev_nodes(chebyshev_nodes_x, chebyshev_nodes_y, x) for x in x_values_for_plotting]

# Calculate the difference (netiktis) between the analytical function and interpolation
difference = analytical_function_y - interpolation_chebyshev_nodes

# Plot the results
plt.figure(figsize=(12, 6))

# Plot the given analytical function
plt.plot(x_values_for_plotting, analytical_function_y, label='Duotoji funkcija')

# Plot evenly spaced points
plt.scatter(evenly_spaced_x, evenly_spaced_y, color='red', marker='o', label='Tolygiai išdėstyti taškai')

# Plot Chebyshev nodes
plt.scatter(chebyshev_nodes_x, chebyshev_nodes_y, color='blue', marker='o', label='Čiobyševo mazgai')

# Plot the interpolation using Chebyshev nodes
plt.plot(x_values_for_plotting, interpolation_chebyshev_nodes, label='Interpoliavimas Čiobyševo mazgus')

# Plot the difference (netiktis) with a dotted line
plt.plot(x_values_for_plotting, difference, linestyle='dotted', label='Netiktis')

# Add labels and legend
plt.xlabel('x')
plt.ylabel('y')
plt.legend()

# Show the plot
plt.show()
```

Pav. 3 Programinis kodas 2/2

Rezultatai :



Pav. 4 Rezultatai su 10 taškų.



Pav. 5 Rezultatai su 20 taškų

Kai didiname interpoliavimo taškų skaičių, atrodo, kad netiktys artėja nuliui, tačiau kai sumažiname interpoliavimo taškų skaičių, pamatome, kad netiktis iš tiesų yra ganėtinai didelė.

2. II UŽDUOTIS. INTERPOLIAVIMAS SPLAINU PER DUOTUS TAŠKUS.

II užduotis. Interpoliavimas splainu per duotus taškus

Sudarykite ***2 lentelėje*** nurodytos šalies 1998-2018 metų šiltnamio dujų emisiją (galimo duomenų šaltinio nuoroda apačioje) interpoliuojančias kreives, kai interpoliuojama ***2 lentelėje*** nurodyto tipo splainu. Pateikite rezultatų grafiką (interpoliavimo mazgus ir gautą kreivę (vaizdavimo taškų privalo būti daugiau nei interpoliavimo mazgų)).

Gauta užduotis antroje lentelėje :

3	Ispanija	Globalus

pav. 6 Antros užduoties variantas

Programos kodas :

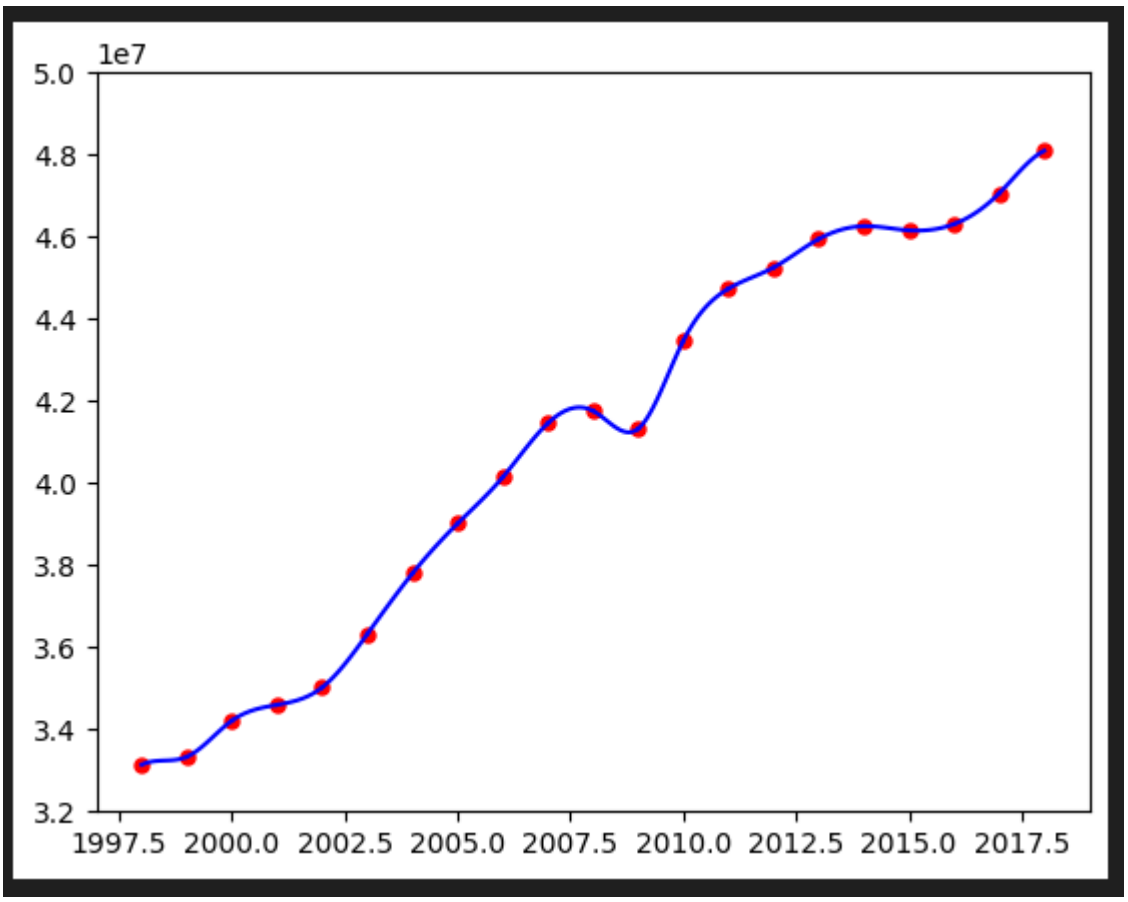
```python
import numpy as np
import matplotlib.pyplot as plt

def fx(x, f0, f1, x0, x1, y0, y1):
    return f0 * ((x - x0) ** 2) / 2 - f0 * ((x - x0) ** 3) / 6 + f1 * ((x - x0) ** 3) / 6 + \
           (y1 - y0) * (x - x0) - f0 * (x - x0) / 3 - f1 * (x - x0) / 6 + y0  # Funkcija

x = np.array(list(range(1998, 2019)))  # Metai
y = np.array([33134921, 33329417, 34208319, 34593446, 35018764, 36315562,
              37789694, 39003678, 40128915, 41442032, 41744913, 41308022,
              43442887, 44714896, 45230398, 35000000, 46235969, 46135596,
              46288080, 47035210, 48069813])

#45922200

plt.xlim([1997, 2019])
plt.ylim([32000000, 50000000])

for i in range(0, 21):
    plt.plot(x[i], y[i], markersize=5, marker='o', color='red')

mat = np.matrix(np.zeros([21, 21]))

for i in range(0, 19):
    mat[i, i] = 1 / 6
    mat[i, i + 1] = 2 / 3
    mat[i, i + 2] = 1 / 6

mat[19, 0] = 1 / 3
mat[19, 1] = 1 / 6
mat[19, 19] = 1 / 6
mat[19, 20] = 1 / 3
mat[20, 0] = 1
mat[20, 20] = -1
```

Pav. 7 Užduoties kodas 1/2

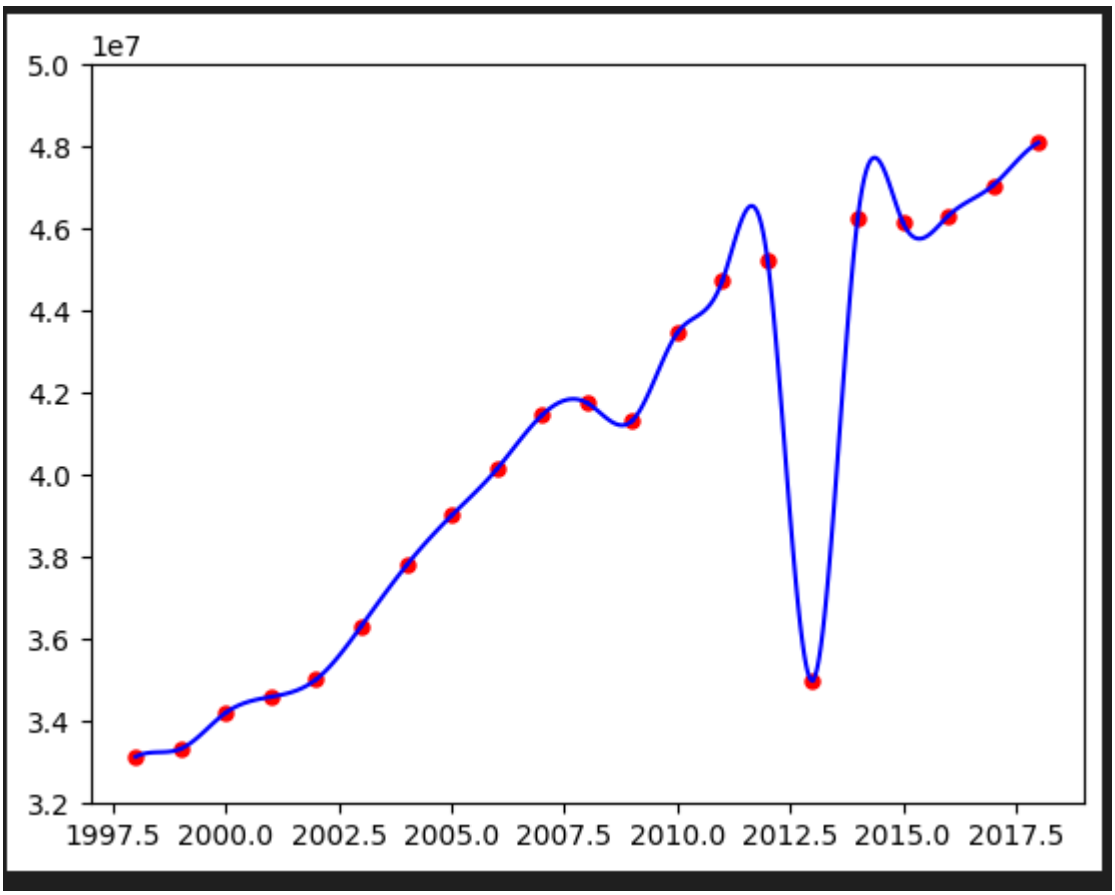
```python
b = np.matrix(np.zeros([21, 1]))

for i in range(0, 19):
    b[i, 0] = (y[i + 2] - y[i + 1]) - (y[i + 1] - y[i])

b[19, 0] = (y[1] - y[0]) - (y[20] - y[19])

ans = np.linalg.solve(mat, b)
#print(ans)

for i in range(0, 20):
    xPlot = np.linspace(x[i], x[i + 1])
    yPlot = np.asarray(fx(xPlot, ans[i], ans[i + 1], x[i], x[i + 1], y[i], y[i + 1]))
    plt.plot(xPlot, yPlot[0], color='blue')

plt.show()
```

Pav. 8 Užduoties kodas 2/2

Rezultatai :



Pav. 9 Rezultatai



Pav. 10 Užduoties grafikas, bet viena reikšmė žymiai pakeista

Teorija pasitvirtino : vieno interpoliavimo taško pakeitimas pakeitė ir aplinkinius intervalus – ne tik savo.

3. III UŽDUOTIS. APROKSIMAVIMAS.

III užduotis. Aproksimavimas

Mažiausių kvadratų metodu sudarykite ***2 lentelėje*** nurodytos šalies 1998-2018 metų šiltnamio dujų emisiją (galimo duomenų šaltinio nuoroda apačioje) aproksimuojančias kreives (**pirmos, antros**, **trečios** ir **penktos** eilės daugianarius). Pateikite gautas daugianarių išraiškas ir grafinius rezultatus.

Gauta užduotis antroje lentelėje :

3	Ispanija	Globalus

pav. 11 Trečios užduoties variantas

Programos kodas :

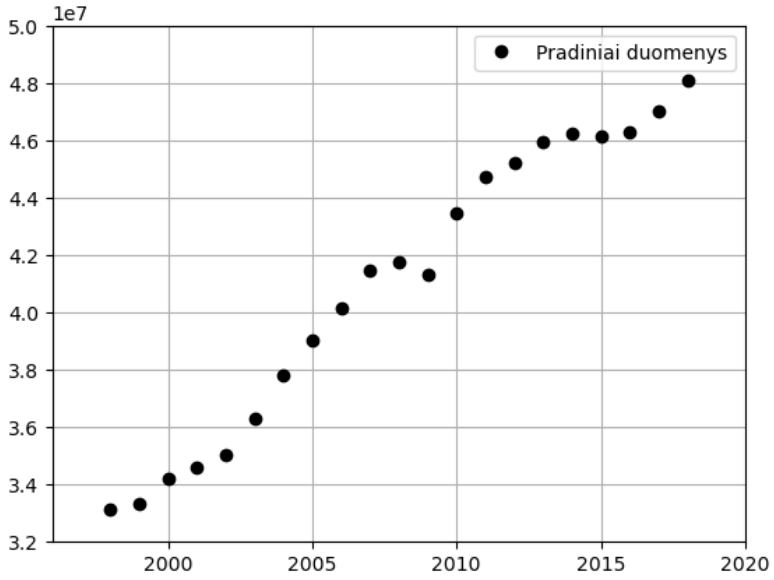
```
import numpy as np
import matplotlib.pyplot as plt

# Function to create the basis matrix
def generate_basis_matrix(degree, x_values):
    basis_matrix = np.column_stack([x_values**(i) for i in range(degree + 1)])
    return basis_matrix

def main():
    # Define plot boundaries
    x_min, y_min, x_max, y_max = 1996, 32000000, 2020, 50000000
    plt.figure(1)
    plt.axis([x_min, x_max, y_min, y_max])
    plt.grid(True)
    degree = 6

    # Data points
    x_data = np.array([1998, 1999, 2000, 2001, 2002, 2003, 2004,
                       2005, 2006, 2007, 2008, 2009, 2010, 2011,
                       2012, 2013, 2014, 2015, 2016, 2017, 2018])
    y_data = np.array([33134921, 33329417, 34208319, 34593446, 35018764, 36315562,
                       37789694, 39003678, 40128915, 41442032, 41744913, 41308022,
                       43442887, 44714896, 45230398, 45922200, 46235969, 46135596,
                       46288080, 47035210, 48069813])

    plt.plot(x_data, y_data, 'ko', label='Pradiniai duomenys')

    # Calculate middle number and set it as 'i'
    number = int((y_max - y_min) / 2)
    i = number + 1

    plt.legend()
    plt.show()

    curve = None
    for degree_value in range(1, 7):
```

Pav. 12 Trečios užduoties programinis kodas 1/2

```
        num_data_points = len(x_data)
        basis_matrix = generate_basis_matrix(degree_value, x_data)
        coefficients = np.linalg.lstsq(basis_matrix, y_data, rcond=None)[0]
        equation_string = f'{coefficients[0]:5.2g}' + ''.join([f'+{coefficients[i]:5.2g}x^{i}' for i in range(1, degree_value + 1)])
        equation_string = equation_string.replace('+-', '-')

        num_interpolation_points = 50
        x_interpolation = np.linspace(x_min, x_max, num_interpolation_points)
        basis_matrix_interp = generate_basis_matrix(degree_value, x_interpolation)
        interpolated_values = basis_matrix_interp @ coefficients

        print(f'\nRezultatai funkcijoms ={degree_value}')
        print('Duoti duomenys:')
        for x, y in zip(x_data, y_data):
            print(f'({x:.2f}, {y:.2f})')

        print('Koeficientai:', coefficients)
        print('Aproksimuojanti funkcija:', equation_string)
        print('')

        if curve is not None:
            curve.remove()
        curve, = plt.plot(x_interpolation, interpolated_values, 'r-')
        plt.plot(x_data, y_data, 'ko', label='Duoti duomenys')
        plt.legend()
        plt.title(f'Taškų skaičius : {num_data_points},  Funkcijų skaičius : {degree_value}')
        plt.show()

if __name__ == "__main__":
    main()
```

Pav. 13 Trečios užduoties programinis kodas 2/2

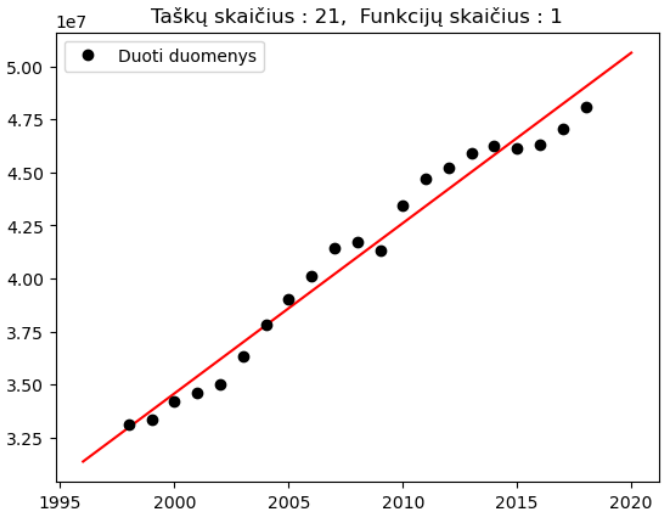
Rezultatai :



Pav. 14 Pradiniai duomenys grafike

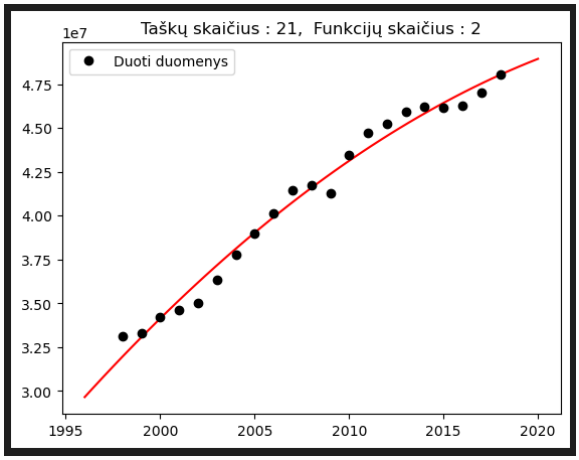
```
Rezultatai funkcijoms =1
Duoti duomenys:
(1998.00, 33134921.00)
(1999.00, 33329417.00)
(2000.00, 34208319.00)
(2001.00, 34593446.00)
(2002.00, 35018764.00)
(2003.00, 36315562.00)
(2004.00, 37789694.00)
(2005.00, 39003678.00)
(2006.00, 40128915.00)
(2007.00, 41442032.00)
(2008.00, 41744913.00)
(2009.00, 41308022.00)
(2010.00, 43442887.00)
(2011.00, 44714896.00)
(2012.00, 45230398.00)
(2013.00, 45922200.00)
(2014.00, 46235969.00)
(2015.00, 46135596.00)
(2016.00, 46288080.00)
(2017.00, 47035210.00)
(2018.00, 48069813.00)
Koeficientai: [-1.57285653e+09  8.03715609e+05]
Aproksimuojanti funkcija: -1.6e+09+8e+05x^1
```

Pav. 15 Pirmos eilės funkcijos gauti duomenys



Pav. 16 Pirmos eilės funkcija grafike

```
Rezultatai funkcijoms =2
Duoti duomenys:
(1998.00, 33134921.00)
(1999.00, 33329417.00)
(2000.00, 34208319.00)
(2001.00, 34593446.00)
(2002.00, 35018764.00)
(2003.00, 36315562.00)
(2004.00, 37789694.00)
(2005.00, 39003678.00)
(2006.00, 40128915.00)
(2007.00, 41442032.00)
(2008.00, 41744913.00)
(2009.00, 41308022.00)
(2010.00, 43442887.00)
(2011.00, 44714896.00)
(2012.00, 45230398.00)
(2013.00, 45922200.00)
(2014.00, 46235969.00)
(2015.00, 46135596.00)
(2016.00, 46288080.00)
(2017.00, 47035210.00)
(2018.00, 48069813.00)
Koeficientai: [-6.55526168e+10  6.45291560e+07 -1.58678885e+04]
Aproksimuojanti funkcija: -6.6e+10+6.5e+07x^1-1.6e+04x^2
```

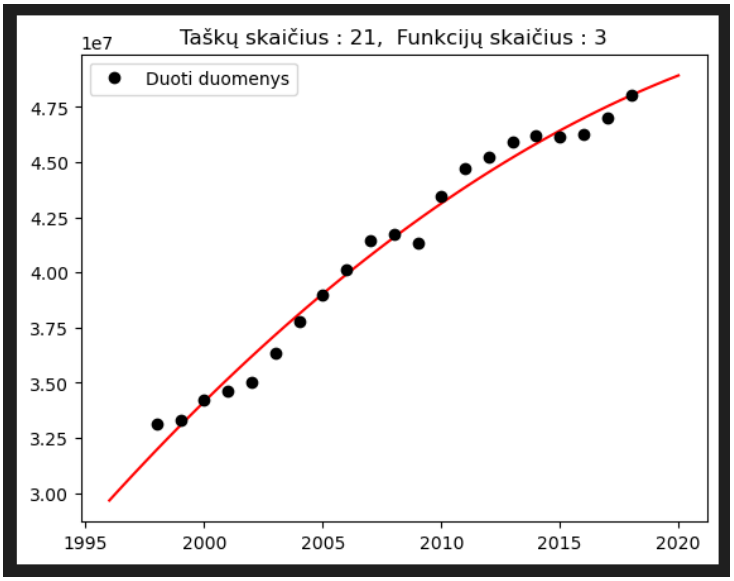


Pav. 17 Antros eilės funkcija grafike

Pav. 18 Antros eilės funkcijos gauti duomenys

```
Rezultatai funkcijoms =3
Duoti duomenys:
(1998.00, 33134921.00)
(1999.00, 33329417.00)
(2000.00, 34208319.00)
(2001.00, 34593446.00)
(2002.00, 35018764.00)
(2003.00, 36315562.00)
(2004.00, 37789694.00)
(2005.00, 39003678.00)
(2006.00, 40128915.00)
(2007.00, 41442032.00)
(2008.00, 41744913.00)
(2009.00, 41308022.00)
(2010.00, 43442887.00)
(2011.00, 44714896.00)
(2012.00, 45230398.00)
(2013.00, 45922200.00)
(2014.00, 46235969.00)
(2015.00, 46135596.00)
(2016.00, 46288080.00)
(2017.00, 47035210.00)
(2018.00, 48069813.00)
Koeficientai: [-5.00052452e+04 -3.34698735e+07  3.29669842e+04 -8.11174998e+00]
Aproksimuojanti funkcija: -5e+04-3.3e+07x^1+3.3e+04x^2+ -8.1x^3
```

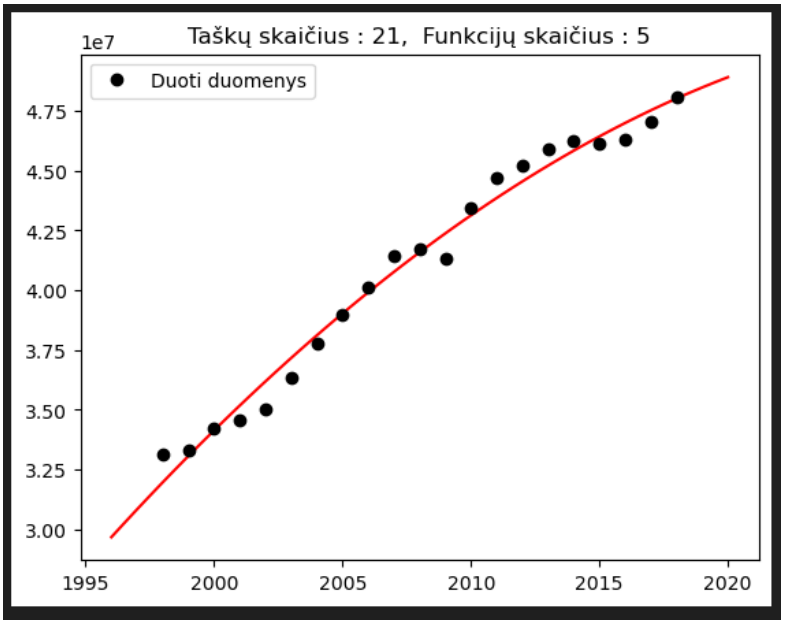
Pav. 19 Trečios eilės funkcijos gauti duomenys



Pav. 20 Trečios eilės funkcija grafike

```
Rezultatai funkcijoms =5
Duoti duomenys:
(1998.00, 33134921.00)
(1999.00, 33329417.00)
(2000.00, 34208319.00)
(2001.00, 34593446.00)
(2002.00, 35018764.00)
(2003.00, 36315562.00)
(2004.00, 37789694.00)
(2005.00, 39003678.00)
(2006.00, 40128915.00)
(2007.00, 41442032.00)
(2008.00, 41744913.00)
(2009.00, 41308022.00)
(2010.00, 43442887.00)
(2011.00, 44714896.00)
(2012.00, 45230398.00)
(2013.00, 45922200.00)
(2014.00, 46235969.00)
(2015.00, 46135596.00)
(2016.00, 46288080.00)
(2017.00, 47035210.00)
(2018.00, 48069813.00)
Koeficientai: [-1.07384514e-08 -1.31625942e-05 -1.29892986e-02 -8.69417150e+00
  8.57286202e-03 -2.11181969e-06]
Aproksimuojanti funkcija: -1.1e-08-1.3e-05x^1-0.013x^2+ -8.7x^3+0.0086x^4-2.1e-06x^5
```

Pav. 21 Penktos eilės funkcijos gauti rezultatai



Pav. 22 Penktos eilės funkcija grafike

4. IV UŽDUOTIS. PARAMETRINIS APROKSIMAVIMAS.

IV užduotis. Parametrinis aproksimavimas.

Naudodami **parametrinį aproksimavimą Haro bangelėmis** suformuokite ***2 lentelėje*** nurodytos šalies kontūrą. Analizuokite bent 10 detalumo lygių. Pateikite aproksimavimo rezultatus (aproksimuotą kontūro kreivę) ne mažiau kaip 4 skirtinguose lygmenyse. Jei šalis turi keletą atskirų teritorijų (pvz., salų), pakanka analizuoti didžiausią iš jų.

3	Ispanija	Globalus

pav. 23 Ketvirtos užduoties variantas

Programos kodas :

```python
import matplotlib.pyplot as plt
import shapefile
import numpy as np
from shapely import geometry

def custom_wavelet_reconstruction(x, j, k, a, b):
    eps = 1e-9
    xtld = (x - a) / (b - a)
    xx = np.power(2, j) * xtld - k
    h = np.power(2, j / 2) * (np.sign(xx + eps) + np.sign(xx - 1 - eps) - 2 * np.sign(xx - 0.5)) / (2 * (b - a))
    return h

def custom_pyramid_algorithm(ValueArray, NL, a, b):
    details = []
    smooth = (b - a) * ValueArray * 2 ** (-NL / 2)
    for i in range(NL):
        smooth_temp = (smooth[::2] + smooth[1::2]) / np.sqrt(2)
        details.append((smooth[::2] - smooth[1::2]) / np.sqrt(2))
        smooth = smooth_temp
    return smooth, details

shape = shapefile.Reader("./ne_10m_admin_0_countries_esp.shp")
id_val = -1
for i in range(len(shape)):
    feature = shape.shapeRecords()[i]
    if feature.record.NAME_EN == "Spain":
        id_val = i
        break

if id_val == -1:
    print("Country not found")
else:
    print("ID: " + str(id_val))
```

Pav. 24 Ketvirtos užduoties programinis kodas 1/3

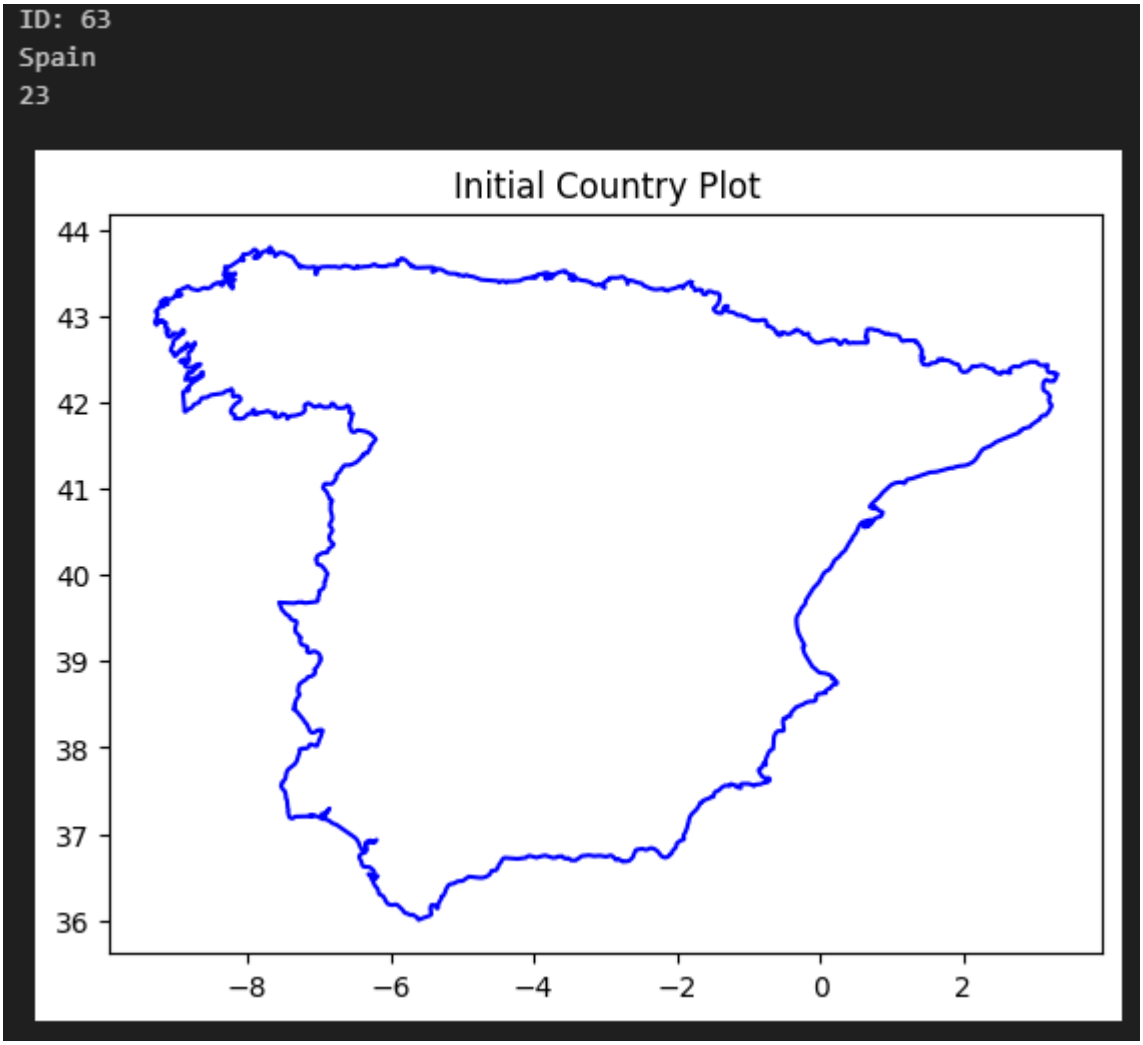
```
feature = shape.shapeRecords()[id_val]
print(feature.record.NAME_EN)
largestAreaID = 0
if feature.shape.__geo_interface__['type'] == 'MultiPolygon':
    print(len(feature.shape.__geo_interface__['coordinates']))
    area = 0
    for i in range(len(feature.shape.__geo_interface__['coordinates'])):
        points = feature.shape.__geo_interface__['coordinates'][i][0]
        polygon = geometry.Polygon(points)
        if polygon.area > area:
            area = polygon.area
            largestAreaID = i
    xxyy = feature.shape.__geo_interface__['coordinates'][largestAreaID][0]
else:
    xxyy = feature.shape.__geo_interface__['coordinates'][0]

xy = list(zip(*xxyy))
X = xy[0]
Y = xy[1]

plt.title("Initial Country Plot")
plt.plot(X, Y, 'b')
plt.show()

NL = 10

starter_point_count = len(X)
interpolated_points_count = 2 ** NL

t = np.zeros(starter_point_count)
for i in range(1, starter_point_count):
    diff = np.array([X[i] - X[i - 1], Y[i] - Y[i - 1]])
    t[i] = t[i - 1] + np.linalg.norm(diff)
```

Pav. 25 Ketvirtos užduoties programinis kodas 2/3

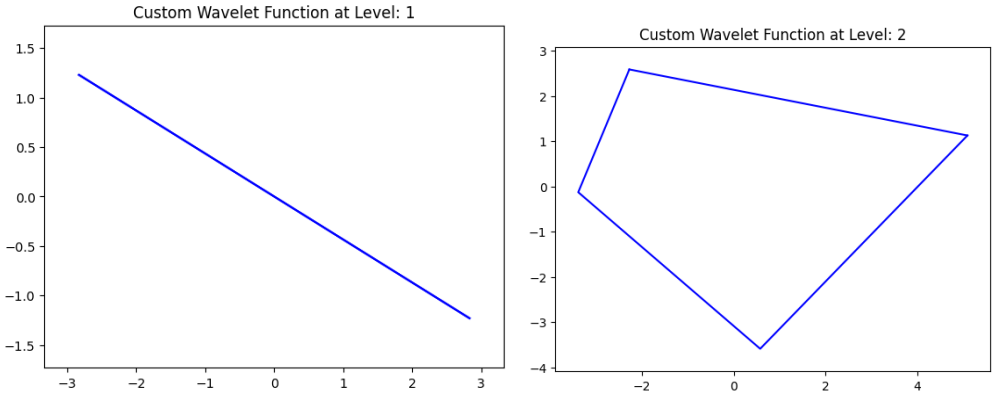
```
min_t = min(t)
max_t = max(t)
print(f"Wavelet ranges: [{min_t} - {max_t}]")

t1 = np.linspace(min_t, max_t, interpolated_points_count)
interpolated_X = np.interp(t1, t, X)
interpolated_Y = np.interp(t1, t, Y)

t = t1

plt.plot(interpolated_X, interpolated_Y)
plt.title("Interpolated New Points")
plt.show()

smooth_X, details_X = custom_pyramid_algorithm(interpolated_X, NL, min_t, max_t)
smooth_Y, details_Y = custom_pyramid_algorithm(interpolated_Y, NL, min_t, max_t)

hx = np.zeros(interpolated_points_count)
hy = np.zeros(interpolated_points_count)

for i in range(NL):
    h1x, h1y = np.zeros(interpolated_points_count), np.zeros(interpolated_points_count)
    for k in range(2 ** i):
        h1x += details_X[NL - i - 1][k] * custom_wavelet_reconstruction(t, i, k, min_t, max_t)
        h1y += details_Y[NL - i - 1][k] * custom_wavelet_reconstruction(t, i, k, min_t, max_t)

    hx += h1x
    hy += h1y
    plt.title(f'Custom Wavelet Function at Level: {i + 1}')
    plt.plot(hx, hy, color="blue")
    plt.plot([hx[0], hx[-1]], [hy[0], hy[-1]], color="blue")
    plt.ylim(min(hy) - 0.5, max(hy) + 0.5)
    plt.xlim(min(hx) - 0.5, max(hx) + 0.5)
    plt.show()
```

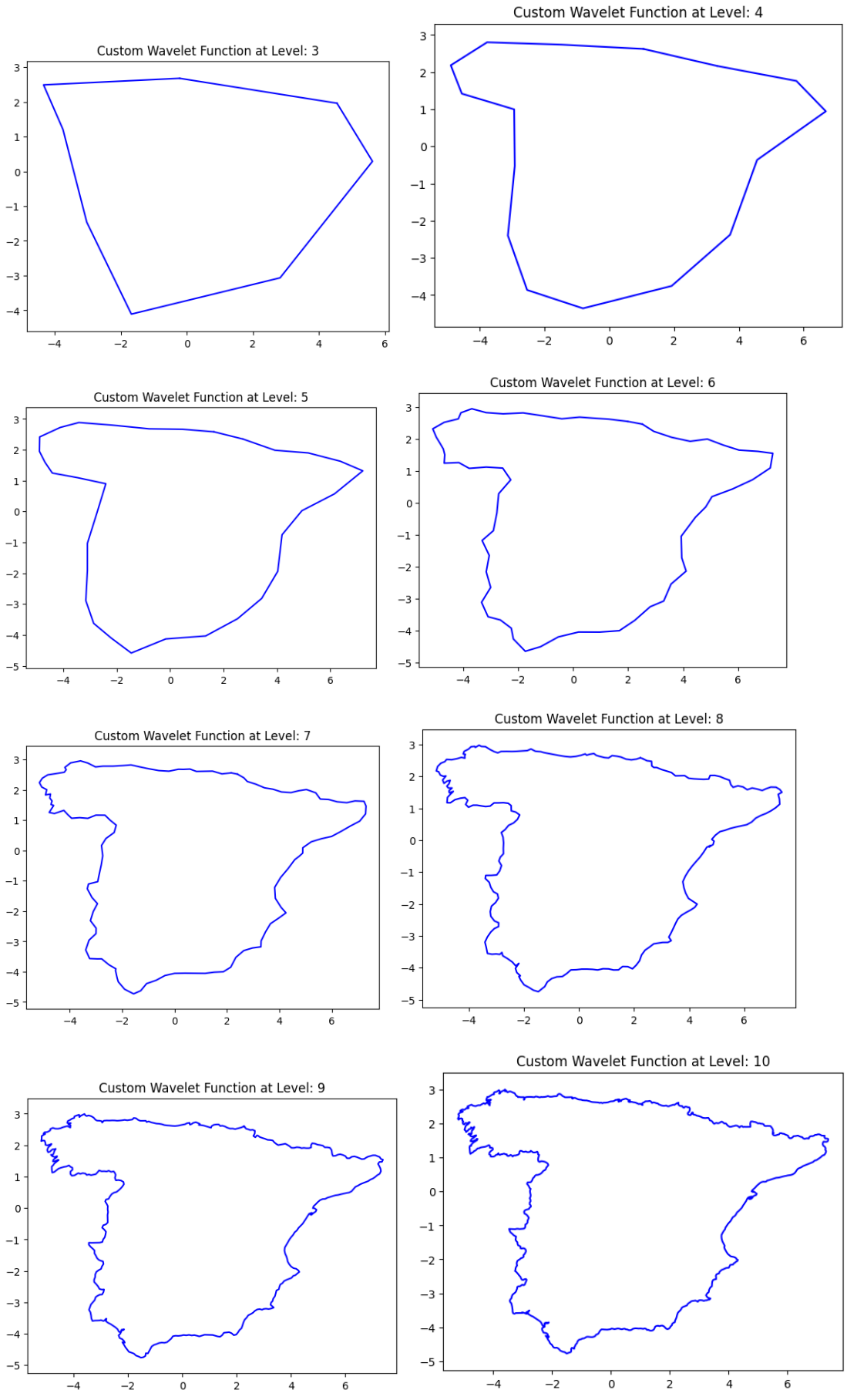
Pav. 26 Ketvirtos užduoties programinis kodas 3/3

Rezultatai :



Pav. 27 Pradiniai duomenys - Ispanijos kontūras





Pav. 28 Ispanijos kontūras 10-tyje skirtingų lygių.

IŠVADOS:

Atliekant laboratorinį darbą sužinojau, kas yra globalus splainas, interpoliacija, aproksimacija:

Globalus splainas – matematinė funkcija, dažnai naudojama interpoliacijos arba aproksimacijos užduotyse. Splainai – sudėtinės polinomo funkcijos, kurios yra sujungtos tam tikruose taškuose. Globalus splainas apima tam tikrą intervalą arba visus duotus duomenis. Splainus paprastai apibrėžia kontroliniai taškai, kurie nustato splaino formą.

Interpoliacija – matematinė ir statistinė sąvoka, kuri nusako procesą, kai yra sukuriama nauja funkcija arba tiesė, kuri praeina per žinomus taškus arba duomenis.

Lagranžo interpoliacija – bene populiariausias interpoliacijos metodas, leidžiantis sukurti polinominę funkciją, kuri praeina pro duotus taškus. Labai efektyviai veikia kai yra mažas kiekis duomenų.