



KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

Laboratorinis darbas nr. 2

12 variantas

Dėstytojai:
Lekt. Dalia Čalnerytė

Studentas:
Justas Milišiūnas IFF-7/2

KAUNAS, 2019

Turinys

1 Įvadas.....	3
2 Uždutys.....	3
2.1 Interpoliavimas daugianariu.....	3
2.1.1 Teorinė dalis.....	3
2.1.2 A dalies rezultatai.....	4
2.1.3 B dalies rezultatai.....	5
2.2 Interpoliavimas daugianariu ir splineu per duotus taškus.....	5
2.2.1 Teorinė dalis.....	5
2.2.2 A dalies rezultatai.....	7
2.2.3 B dalis rezultatai.....	7
2.3 Parametrinis interpoliavimas.....	8
2.3.1 Teorinė dalis.....	8
2.3.2 Rezultatas.....	8
3 Išvados.....	11

1 Įvadas

Šio laboratorinio darbo esmė išmokyti sudaryti interpoliavimo, apytikslavimo funkcijas pagal duotus skaičius. Rezultatus pavaizduojant grafiškai.

2 Uždutys

2.1 Interpoliavimas daugianariu

Pagal duota interpoliuojamos funkcijos analitinę išraišką. Pateikite interpoliacinės funkcijos išraišką naudodami nurodytas bazines funkcijas, kai:

- a) Taškai pasiskirstę tolygiai.
- b) Taškai apskaičiuojami naudojant Čiobyševio abscises.

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

2.1.1 Teorinė dalis

Mano varianto užduoties funkcija:

$$\frac{\ln(x)}{(\sin(2 * x) + 1.5)} - \frac{x}{7}; 2 \leq x \leq 10$$

Interpoliavimo metodas: Niutono

Visų pirmo paskaičiavau Niutono interpoliavimo išraiškos koeficientus:

$$a_0 = y_0$$

$$a_1 = \frac{y_1 - y_0}{x_1 - x_0} = f(x_0, x_1)$$

$$a_n = \frac{f(x_1, x_2, x_3) - f(x_0, x_1, x_2)}{x_n - x_0}$$

Programos kodas skaičiuojantis koeficientus:

```
def newton_interpolation_coefficients(range_x, range_y):  
    a = [range_y]  
    for i in range(len(range_x)):  
        a.append([])  
        for j in range(1, len(range_x) - i):  
            a[i + 1].append((a[i][j] - a[i][j - 1]) / (  
                range_x[np.min([i + j, len(range_x) - 1])] - r  
                ange_x[np.max([i + j - (i + 1), 0])]))  
    return [_a[0] for _a in a[:-1]]
```

Gavus koeficientus, juos įstačiau į čia Niutono daugianario interpoliavimo formulę:

$$f(x) = a_0 + a_1 * (x - x_0) + a_2 * (x - x_0) * (x - x_1) + \dots + a_n * (x - x_0) * (x - x_1) * (x - x_2) \dots (x - x_n)$$

Programos kodas sustatantis koeficientus į formulę ir gražinantis interpoliavimo funkciją:

```
def newton_interpolation_f(range_x, range_y):
    a_coefficients = newton_interpolation_coefficients(range_x, range_y)
    def interpolation_f(_x):
        ff = a_coefficients[0]
        tmp = 1
        for ii in range(1, len(a_coefficients)):
            tmp *= (_x - range_x[ii - 1])
            ff += a_coefficients[ii] * tmp
        return ff
    return interpolation_f
```

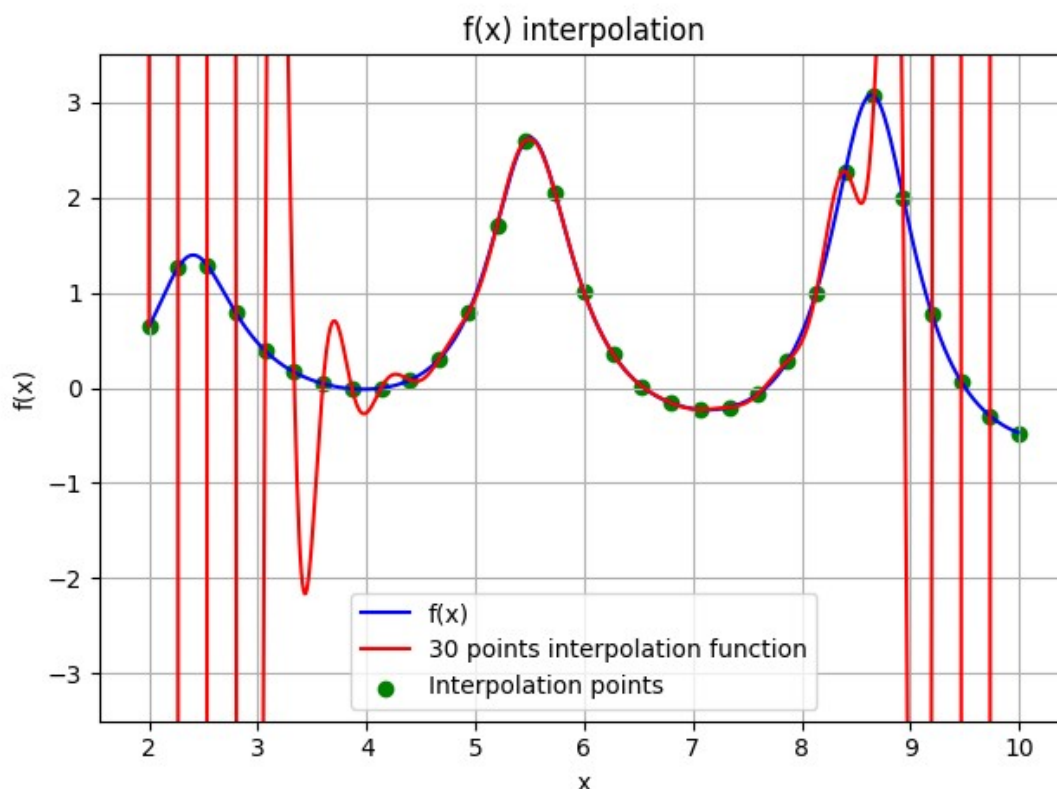
B dalyje dar reikėjo apskaičiuoti taškus naudojant Čiobyševio absceses. Formulė:

$$x_k = \frac{1}{2} * (a+b) + \frac{1}{2} * (b-a) * (\cos(\frac{2k-1}{2n} * \pi)), k=1, \dots, n$$

Programos kodas paskaičiuojantis Čiobyševio absceses:

```
def chebyshev_range(count, start, end):
    range_x = []
    for i in range(count):
        temp = (end + start) / 2 + (end - start) / 2 * np.cos((2 * i + 1) *
np.pi / (2 * count))
        range_x.append(temp)
    return range_x
```

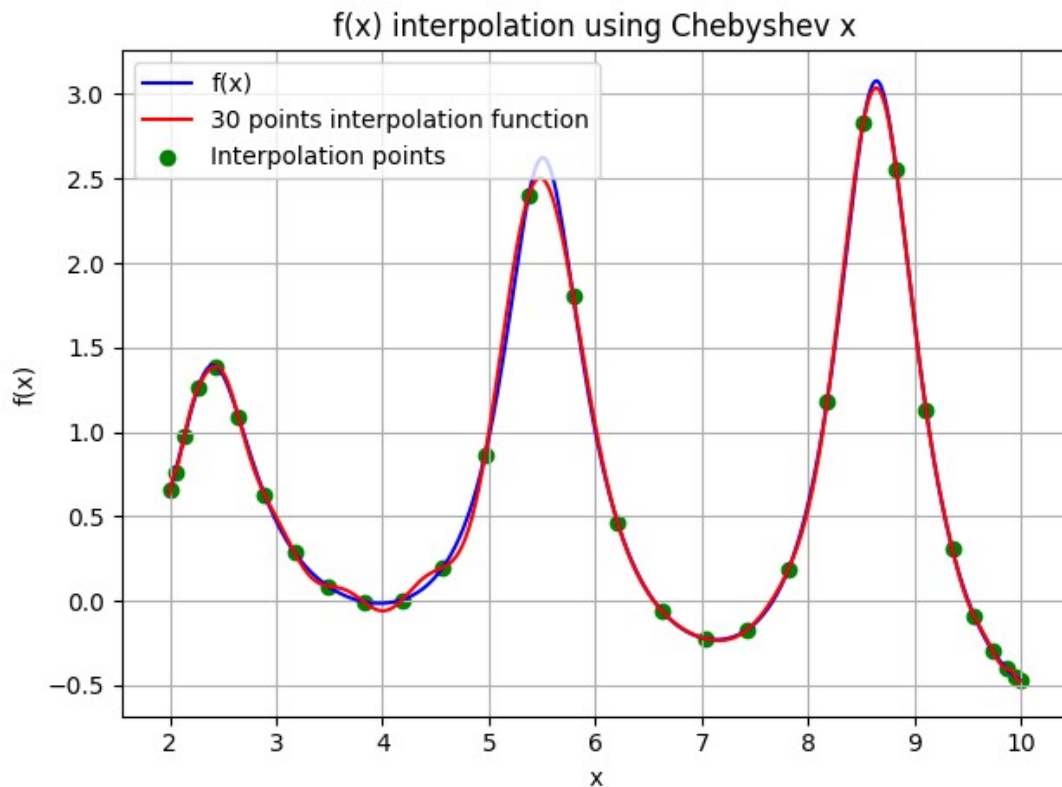
2.1.2 A dalies rezultatai



pav. 1: Duotosios funkcijos ir jos interpoliavimo funkcijos grafikas

Šiame grafike mėlyna spalva parodo duotos funkcijos grafiką. Raudona spalva parodo interpoliavimo funkcijos gautos naudojant 30 interpoliavimo taškų grafikas.

2.1.3 B dalies rezultatai



pav. 2: Funkcijos ir jos interpoliacijos funkcijas pagal 30 taškų grafikas naudojant Čiobyševo absices

Šiame grafike mėlyna spalva parodo duotos funkcijos grafiką. Raudona spalva parodo interpoliavimo funkcijos gautos naudojant 30 interpoliavimo taškų grafikas.

Matome, kad Čiobyševo absices padeda atvaizduoti interpoliuojama funkciją visame intervale. Kai A dalyje interpoliuojamas funkcijos grafikas nepasiekė intervalo galo.

2.2 Interpoliavimas daugianariu ir splainu per duotus taškus

Pagal pateiktą šalį ir metus, sudaryti interpoliuojančią kreivę 12 mėnesių temperatūroms atvaizduoti nurodytais metodais:

- a) Daugianariu, sudarytu naudojant 1 užduotyje a dalyje
- b) Nurodytu tipo splainu

2.2.1 Teorinė dalis

Pradiniai duomenys:

- Šalis : Kipras
- 12 mėnesių temperatūros: 10.5617, 11.1504, 14.4338, 16.9788, 21.0081, 24.7199, 28.526, 27.9448, 25.7463, 23.1029, 16.7226, 11.5084
- Ermito splainas

A dalyje panaudojau pirmos dalies Niutono interpoliavimo metoda su Čiobyševo abscisėmis

B dalyje reikėjo atvaizduoti temperatūros grafiką naudojant Hermito splainą. Visų pirma pasiskaičiavau kiekvieno taško išvestinę pagal formulę:

$$dy = \frac{y_2 - y_1}{x_2 - x_1}$$

Čia y_2 – intervalo galo y ; y_1 – intervalo pradžios y ; x_2 – intervalo galo x ; x_1 – intervalo pradžios x

Programos kodas paskaičiuojantis taškų išvestines:

```
def slope(x1, y1, x2, y2):
    return (y2 - y1) / (x2 - x1)
```

Gavus kiekvieno taško išvestines reikėjo susdaryti U ir V funkcijas:

$$U_j(x) = (1 - 2L_j'(x_j))(x - x_j)L_j^2(x)$$

$$V_j(x) = (x - x_j)L_j^2(x)$$

Programos kodas sukuriantis čias funkcijas:

```
def U(start, end, x):
    return (1 - 2 * (1 / (start - end)) * (x - start)) * ((x - end) / (start - end)) ** 2

def V(start, end, x):
    return (x - start) * ((x - end) / (start - end)) ** 2
```

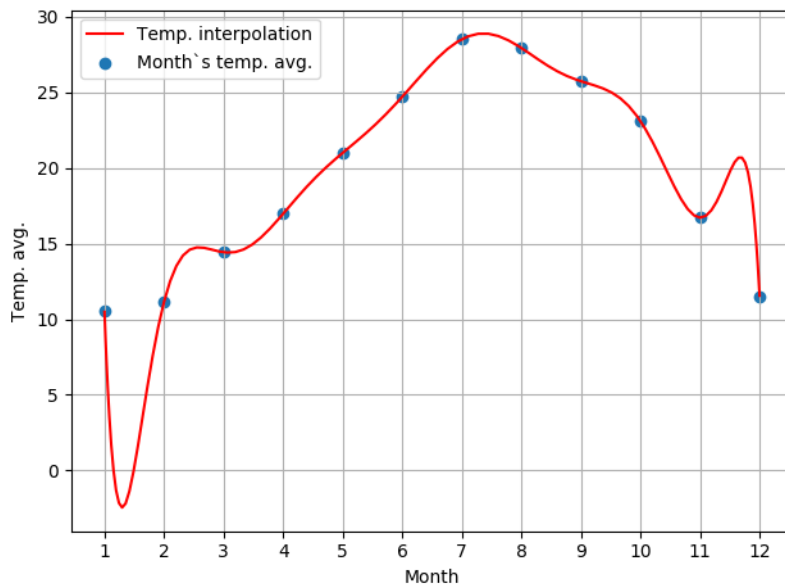
Pasidarius U ir V funkcijas, viską reikia statyti į Hermito interpoliavimo išraišką:

$$f(x) = \sum_{j=1}^n (U_j(x)y_j + V_j(x)y_j')$$

Programos kodas viską susatantis ir padarantis splaino funkciją:

```
def hermite_interpolation_spline(range_x, range_y):
    range_dy = points_slopes(range_x, range_y)
    def spline_function(x):
        index = np.searchsorted(range_x, x)
        try:
            result = U(range_x[index - 1], range_x[index], x) *
                range_y[index - 1] + V(range_x[index - 1],
                range_x[index], x) * range_dy[index - 1] \
                + U(range_x[index], range_x[index - 1], x) *
                range_y[index] \
                + V(range_x[index], range_x[index - 1], x) *
                range_dy[index]
        except TypeError:
            # Handles None data from given country's data
            return None
    return result
return spline_function
```

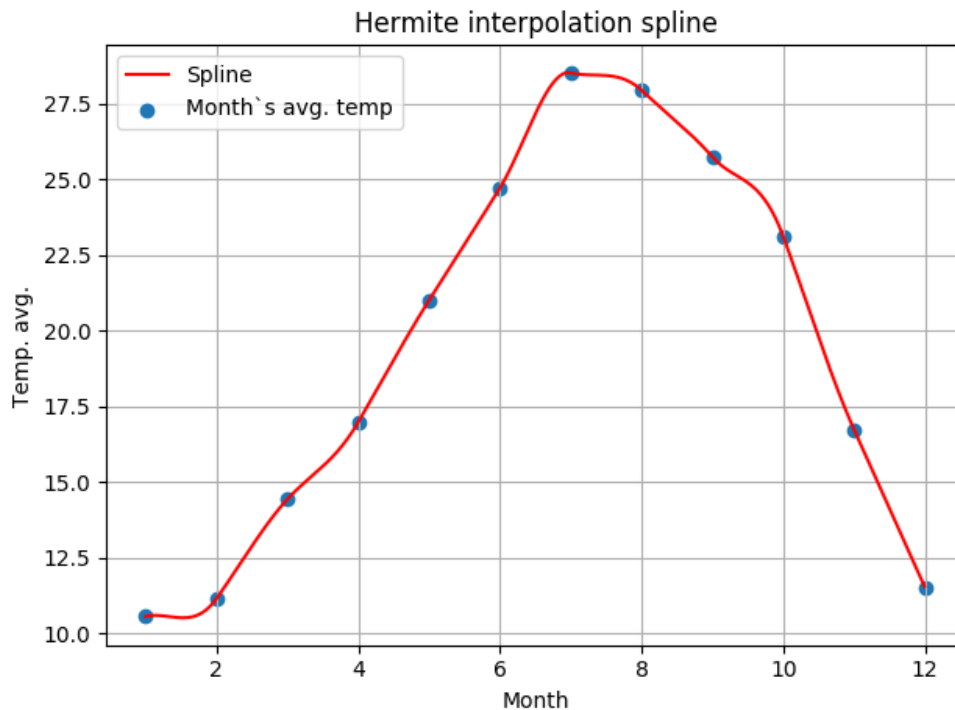
2.2.2 A dalies rezultatai



pav. 3: Metų laikotarpio temperatūros pradiniai duomenys ir interpoliavimo funkcijos grafikas

Šiame grafike mėlyni taškai parodo pradinis duomenys. Raudona kreivė parodo interpoliavimą tarp tų taškų.

2.2.3 B dalis rezultatai



pav. 4: Splaino funkcijos grafikas

Melyna spalva - pradiniai temperatūros duomenys. Raudona - splaino funkcija.

2.3 Parametrinis interpoliavimas

Naudodami parametrinio interpoliavimo metodą nurodytu splainu suformuokite nurodytos šalies kontūrą. Pateikite pradinis duomenis ir rezultatus, gautus naudojant **10, 20, 50, 100** interpoliavimo taškų.

2.3.1 Teorinė dalis

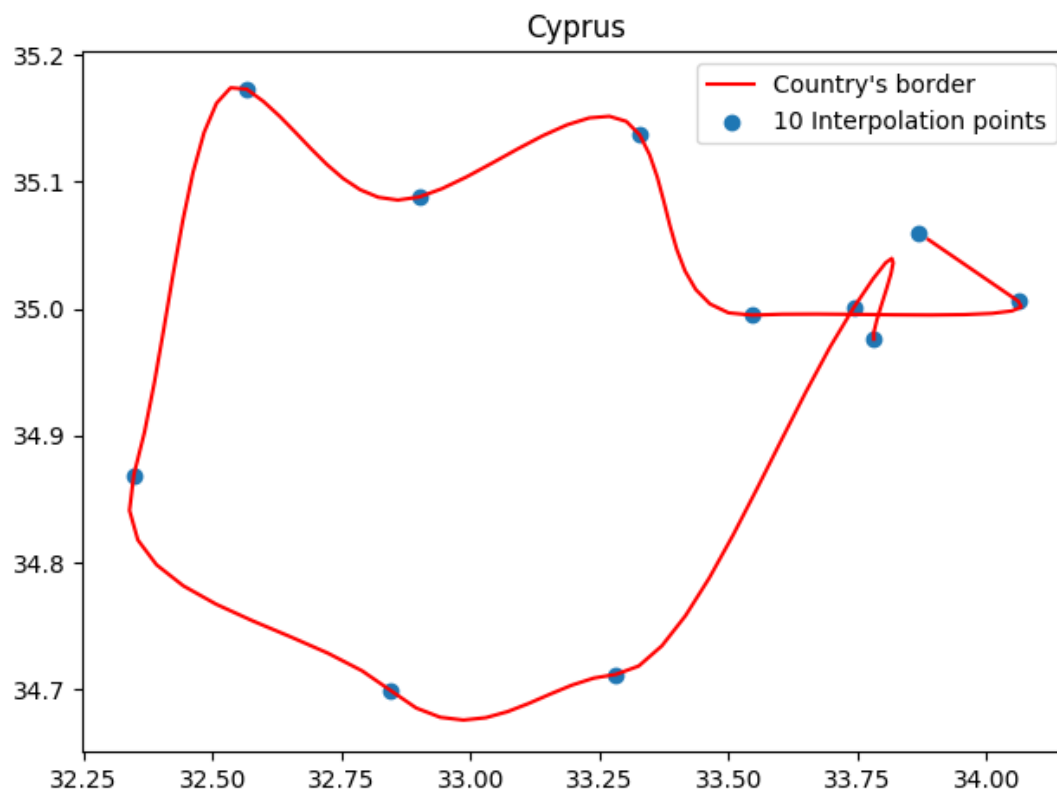
Mano varianto šalis: Kipras

Visų pirma kiekvienam taškui priskyriau laiko parametą. Tada naudodamas Hermito interpoliavimo splaino metodą gavau funkcijas f_x ir f_y padavęs $x=x(t)$ ir $y=y(t)$. Apjungęs f_x ir f_y funkcijas gavau funkciją, kuri paskaičiuoja duotos šalies sienų kontūrų koordinatas:

```
def parametric_interpolation(fx, fy, range_t):  
    x_results = []  
    y_results = []  
    for t in range_t:  
        x_results.append(fx(t))  
        y_results.append(fy(t))  
    return x_results, y_results
```

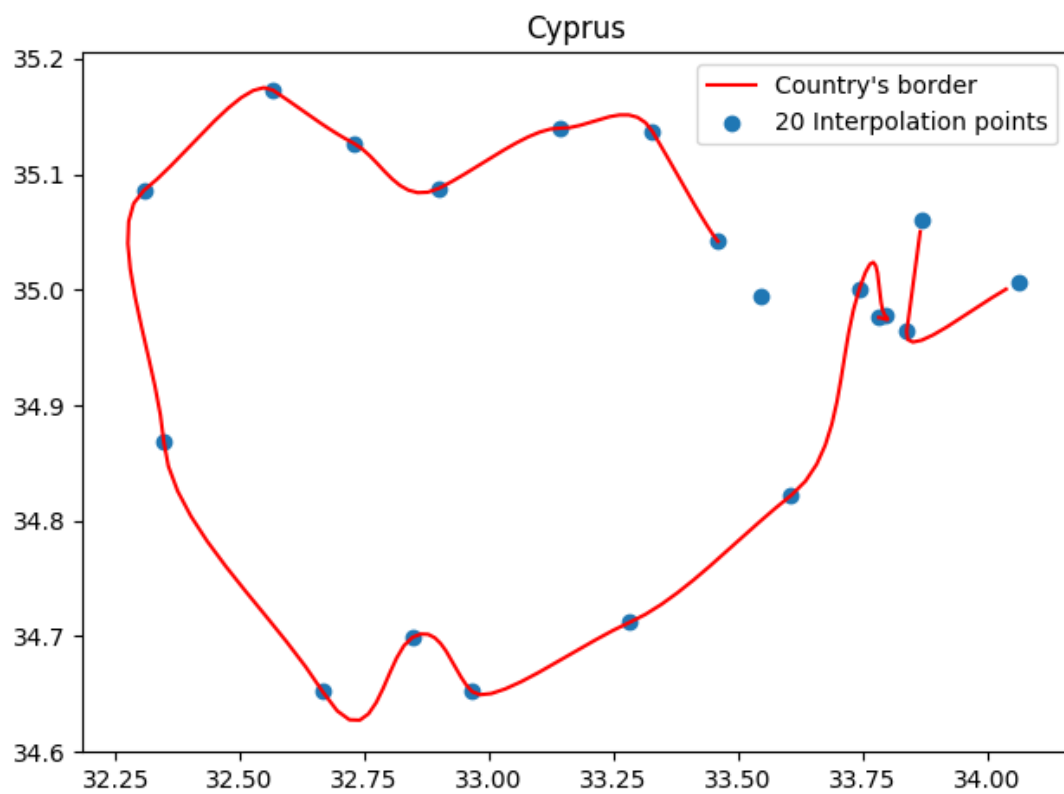
2.3.2 Rezultatas

Naudojant 10 interpoliavimo taškus:



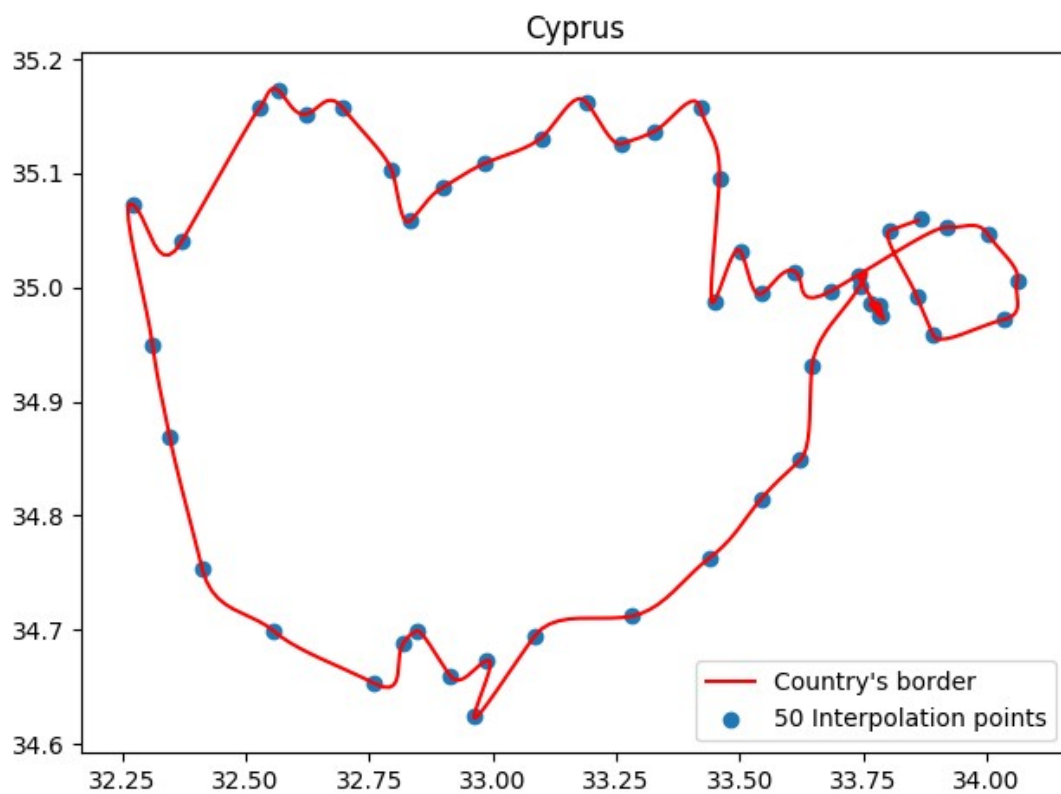
pav. 5: Kipro šalies kontrūras naudojant 10 interpoliavimo taškų

Naudojant 20 interpoliavimo taškus:



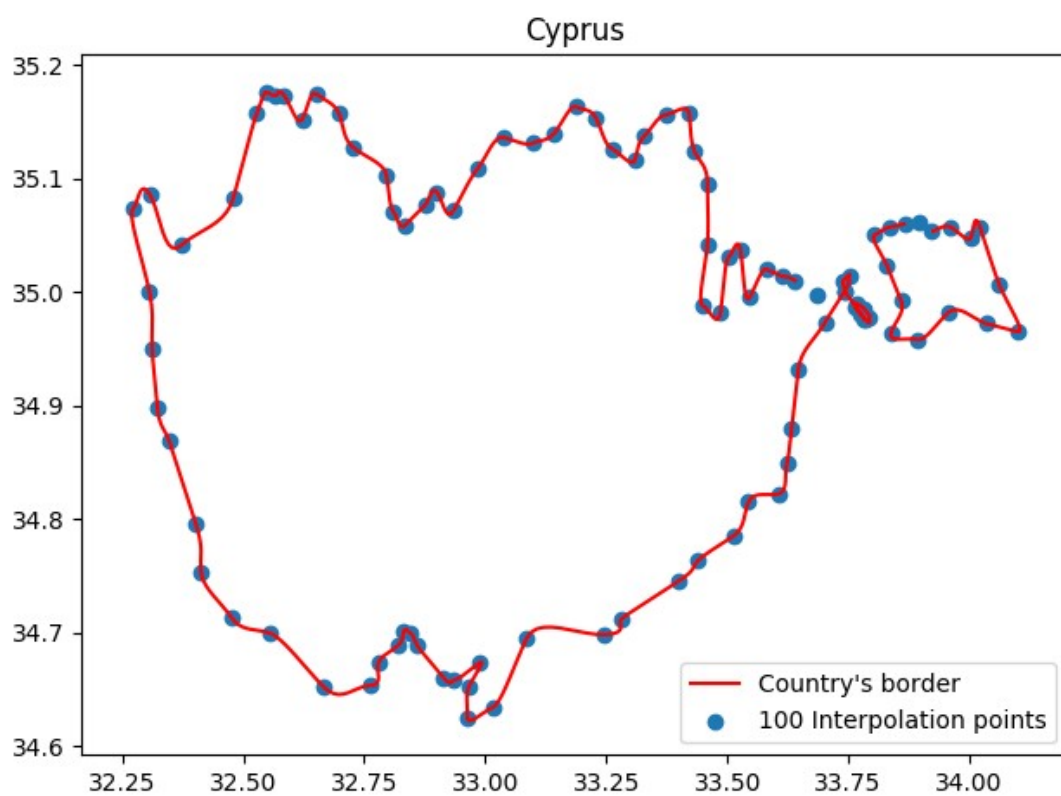
pav. 6: Kipro šalies kontūras naudojant 20 interpoliavimo taškus

Naudojant 50 interpoliavimo taškus:



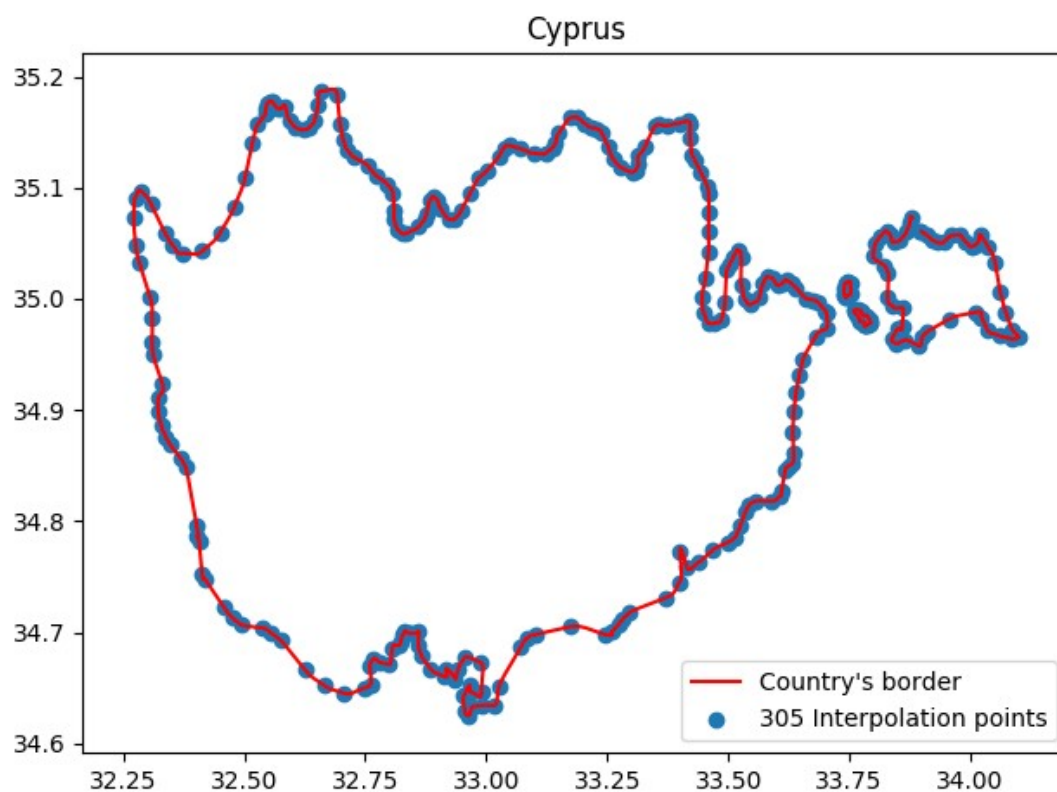
pav. 7: Kipro šālies kontūras naudojant 50 interpoliavimo taškus

Naudojant 100 interpoliavimo taškus:



pav. 8: Kipro šālies kontūras naudojant 100 interpoliavimo taškus

Naudojant 305(visus) interpoliavimo taškus:



pav. 9: Kipro šalies kontūras naudojant 305 interpoliavimo taškus

3 Išvados

1. Išmokau gauti funkcija turint interpoliavimo taškus.
2. Naudojant Čiobyševo absceses grafikas pilnai nupiešiamas duotame intervale.
3. Splainų naudojimas padeda išvengti funkcijos neatitikimų intervalo galuose.
4. Parametriniuose daugianariuose yra svarbu pasirinkti gerą parametą norint gauti tinkamą šalies grafiką.
5. Iš 3 užduoties matome, kad interpoliavimo funkcijas tikslumas labai priklauso nuo duotų pradinių taškų.