Kauno technologijos universitetas
Informatikos fakultetas

Skaitiniai metodai ir algoritmai (P170B115)

3-osios projektinės užduoties ataskaita

10-as variantas

Atliko:

IFF-1/6 gr. studentas

Lukas Kuzmickas

2023 m. December 14 d.

Priėmė:

Prof. Rimantas Barauskas

Doc. Andrius Krisčiūnas

Kaunas 2023

TURINYS

1. I dalis. Interpoliavimas daugianariu

1.1. Varianto uždaviniai

1 lentelėje duota interpoliuojamos funkcijos analitinė išraiška. Pateikite interpoliacinės funkcijos išraišką naudodami **1 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 interpoliuojančiosfunkcijos apskaičiuojamos naudojant skirtingas abscises ir gautas interpoliuojančių funkcijų išraiškas. Tame pačiame grafike vaizduokite duotąją funkciją, interpoliuojančią funkciją ir netiktį.

Mano varianto numeris yra 10.

1 lentelė. Interpoliuojamos funkcijos išraiška

Var. Nr.	Funkcijos išraiška	Bazinė funkcija
1	$e^{-x^2} \cdot \sin(x^2) \cdot (x-3);\ -3 \le x \le 2$	Čiobyševo
2	$\frac{\ln(x)}{(\sin(2 \cdot x)+1{,}5)} + x/5; 2 \le x \le 10$	Vienanarių
3	$e^{-x^2} \cdot \cos(x^2) \cdot (x-3);\ -3 \le x \le 2$	Čiobyševo
4	$\cos(2 \cdot x) \cdot (\sin(2 \cdot x)+1{,}5) + \cos x; -2 \le x \le 3;$	Vienanarių
5	$e^{-x^2} \cdot \sin(x^2) \cdot (x-3);\ -3 \le x \le 2$	Čiobyševo
6	$\frac{\ln(x)}{(\sin(2 \cdot x)+1{,}5)} + x/5; 2 \le x \le 10$	Vienanarių
7	$\cos(2 \cdot x) \cdot (\sin(2 \cdot x)+1{,}5) - \cos\frac{x}{5}; -2 \le x \le 3;$	Čiobyševo
8	$\frac{\ln(x)}{(\sin(2 \cdot x)+1{,}5)} + x/5; 2 \le x \le 10$	Vienanarių
9	$\cos(2 \cdot x) \cdot (\sin(2 \cdot x)+1{,}5) + \cos x; -2 \le x \le 3;$	Čiobyševo
10	$\cos(2 \cdot x) \cdot (\sin(3 \cdot x)+1{,}5) - \cos\frac{x}{5}; -2 \le x \le 3;$	Vienanarių

1.2. Atsakymas

Interpoliavimas yra procesas, kurio metu yra ieškoma tolydžioji kreivė, kuri praeina per duotus taškus.

Taip pat buvo taikyta interpoliacija ir pagal taškus perskaičiuotus pagal **Čiobyševo abscisės** formulę (1 pav.).

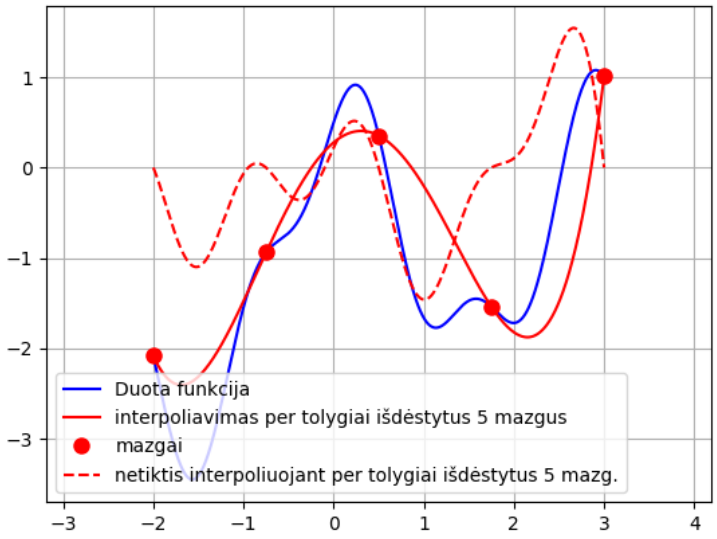
$$x_k = \frac{1}{2}\left[(a+b) + (b-a)\cos\left(\frac{(2k-1)\pi}{2N}\right)\right]$$

1 pav. Čiobyševo abscisė.

Bandymas su 5 mazgais.

Interpoliacinės funkcijos išraiška, kai taškai pasiskirstę tolygiai, gavosi tokia (vienanario bazė):

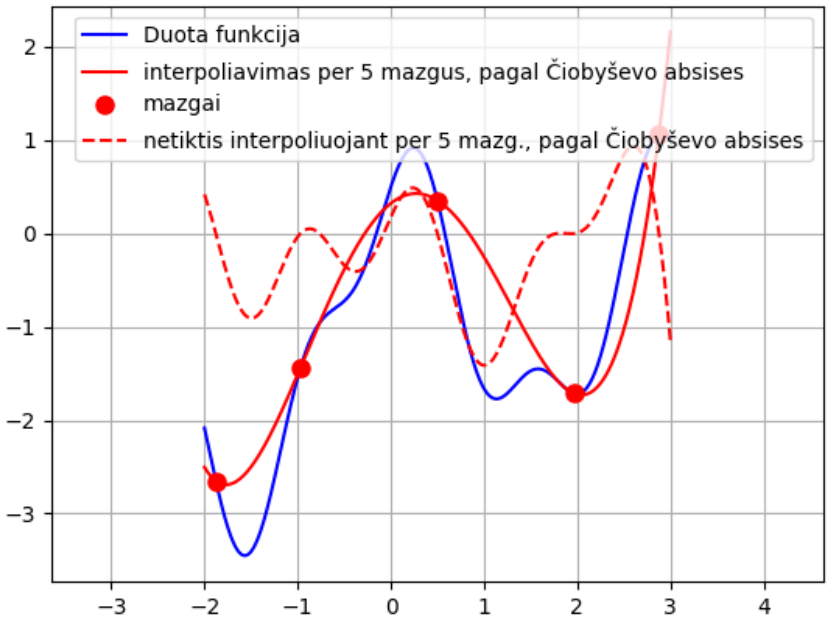
$$f(x) = 0.2789 - 0.8285x - 1.3064x^2 - 0.1908x^3 + 0.1871x^4$$



2 pav. Interpoliavimas tolygiai duotus taškus, kai duoti 5 mazgai.

Čiobyševo abscisės interpoliacinės funkcijos išraiška tokia (vienanario bazė):

$$f(x) = 0.3233 + 0.7745x - 1.4016\,x^2 - \ 0.1449x^3 + 0.1980x^4$$



3 pav. Interpoliavimas Čiobyševo absises formule perskaičiuotus duotus taškus, kai duoti 5 mazgai.

Bandymas su 10 mazgų.

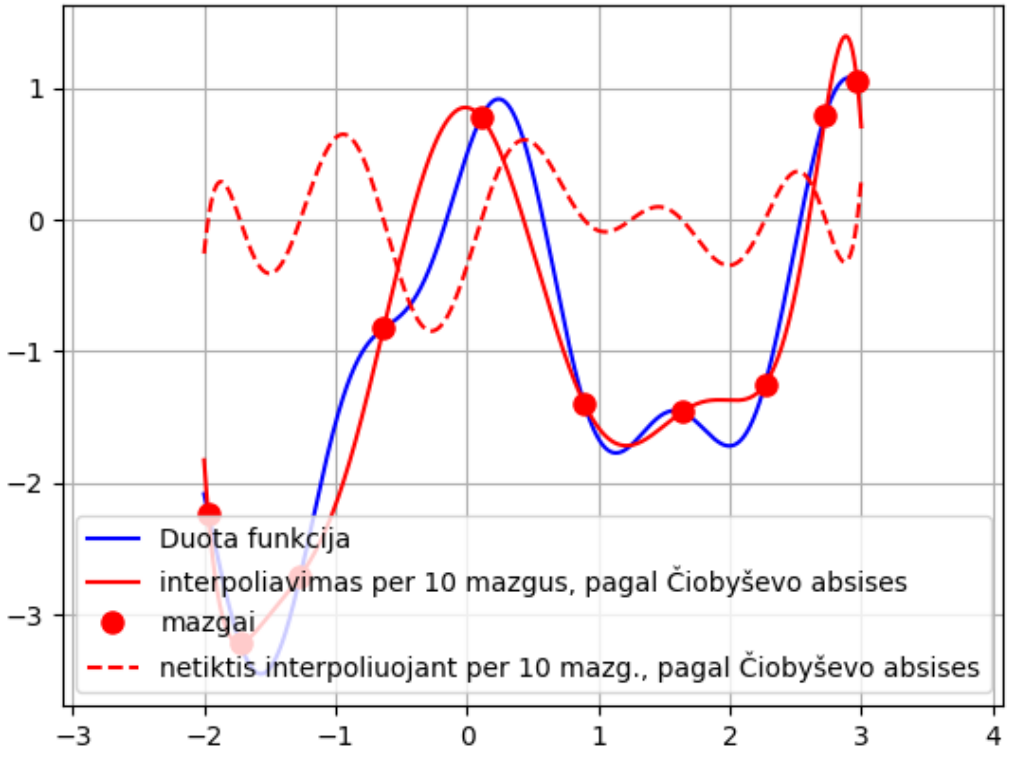
Interpoliacinės funkcijos išraiška, kai taškai pasiskirstę tolygiai, gavosi tokia (vienanario bazė):

$$f(x) = 0.7585 + 2.2799x - 6.5060\,x^2 - 4.4554x^3 + 7.3772x^4 + 1.7906x^5 - 3.3821\,x^6 + 0.1865x^7 + 0.4754x^8 - 0.0976x^9$$

4 pav. Interpoliavimas tolygiai duotus taškus, kai duoti 10 mazgų.

Čiobyševo abscisės interpoliacinės funkcijos išraiška tokia (vienanario bazė):

$$f(x) = 0.8519 - 0.1277x - 5.1628x^2 + 0.7697x^3 + 3.2726x^4 - 0.5051x^5 - 0.9362x^6 + 0.1819x^7 + 0.1006x^8 - 0.0250x^9$$



5 pav. Interpoliavimas Čiobyševo absises formule perskaičiuotus duotus taškus, kai duota 10 mazgų.

1.3. Kodas

```
import matplotlib.pyplot as plt
import numpy as np
def funkcija(x):
    return np.cos(2 * x) * (np.sin(3 * x) + 1.5) - np.cos(x/5)
def cebysev(a, b, N):
    k = np.arange(1, N+1)
    nodes = 0.5 * (a + b) + 0.5 * (b - a) * np.cos((2 * k - 1) * np.pi / (2 * N))
    return nodes
np.set_printoptions(formatter={'float': lambda x: "{:.4f}".format(x)})
xmin = -2
xmax = 3
N = 10 #mazgu sk.
X = np.linspace(xmin, xmax, N)
X1 = cebysev(xmin, xmax, N)
leg = ['Duota funkcija',
       f'interpoliavimas per tolygiai išdėstytus {N} mazgus',
       'mazgai',
       f'netiktis interpoliuojant per tolygiai išdėstytus {N} mazg.',]
leg1 = ['Duota funkcija',
        f'interpoliavimas per {N} mazgus, pagal Čiobyševo absises',
        'mazgai',
        f'netiktis interpoliuojant per {N} mazg., pagal Čiobyševo absises']
Y = funkcija(X)
Y1 = funkcija(X1)
x = np.arange(min(X), max(X) + (max(X) - min(X)) / 1000, (max(X) - min(X)) / 1000)
# values for visualization
plt.figure(1)
plt.grid(True)
plt.axis('equal')
plt.plot(x, funkcija(x), 'b-')
A = np.vander(X, increasing=True)
coefficients = np.linalg.solve(A, Y)
print(coefficients)
A1 = np.vander(X1, increasing=True)
coefficients1 = np.linalg.solve(A1, Y1)
print(coefficients1)
F = np.vander(x, increasing=True)[:,:N] @ coefficients
F1 = np.vander(x, increasing=True)[:,:N] @ coefficients1
plt.plot(x, F, 'r-')
plt.plot(X, Y, 'ro', markerfacecolor='r', markersize=8)
plt.plot(x, funkcija(x) - F, 'r--')
plt.legend(leg)
plt.figure(2)
plt.grid(True)
plt.axis('equal')
plt.plot(x, funkcija(x), 'b-')
plt.plot(x, F1, 'r-')
plt.plot(X1, Y1, 'ro', markerfacecolor='r', markersize=8)
plt.plot(x, funkcija(x) - F1, 'r--')
plt.legend(leg1)
plt.show()
```

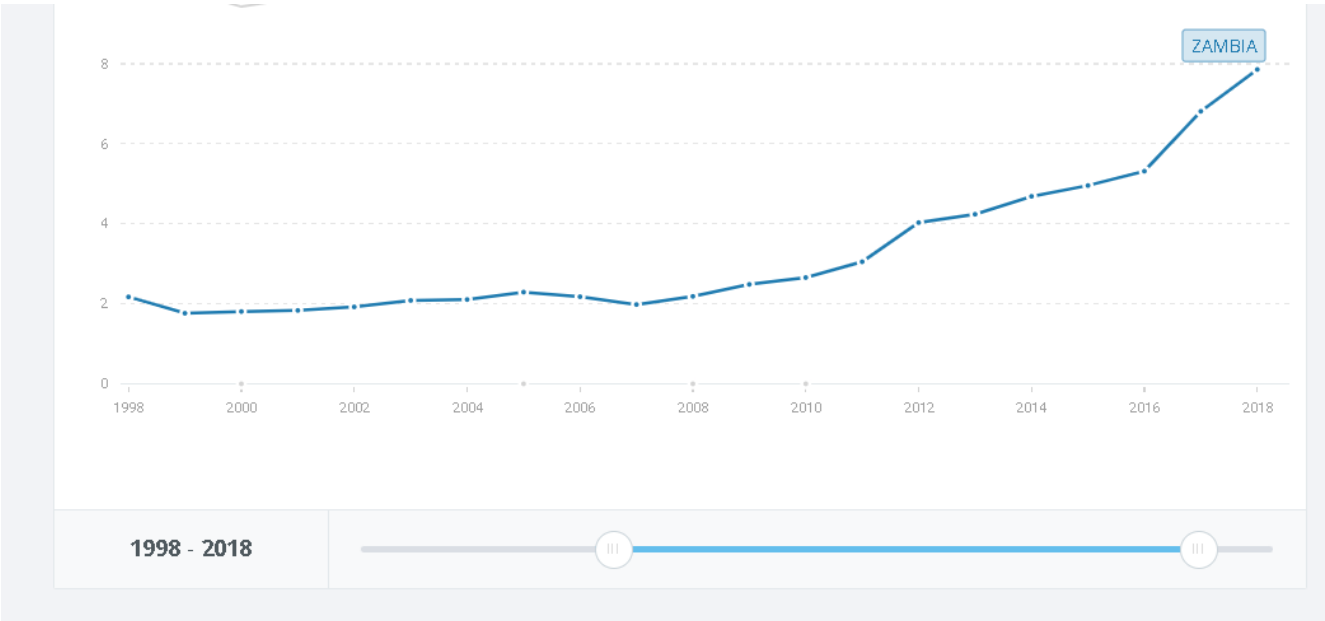
2. II dalis. Interpoliavimas splainu per duotus taškus

2.1. Varianto uždaviniai

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)).

2 lentelė. Šalys ir splaino tipas interpoliavimui.

Var. Nr.	Šalis	Splainas
1	Argentina	Globalus
2	Prancūzija	Ermito (Akima)
3	Ispanija	Globalus
4	Latvija	Ermito (Akima)
5	Kroatija	Globalus
6	Malis	Ermito (Akima)
7	Venesuela	Globalus
8	Austrija	Ermito (Akima)
9	Panama	Globalus
10	Zambija	Ermito (Akima)



6 pav. Zambijos 1998-2018 metų šiltnamio dujų emisijos grafikas.

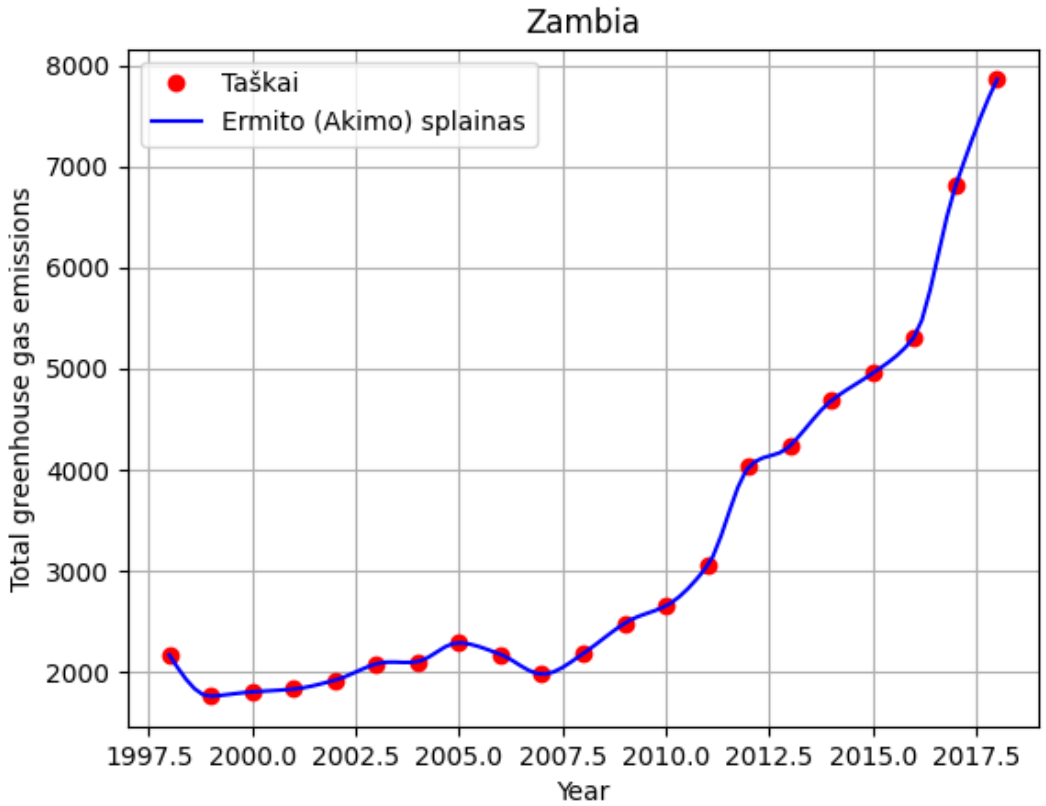
Duomenys:

Metai	Šiltnamio dujų emisiją
1998	2173
1999	1764
2000	1807
2001	1835
2002	1923
2003	2084
2004	2106
2005	2293
2006	2178
2007	1982
2008	2186
2009	2485
2010	2657
2011	3051
2012	4030
2013	4239
2014	4686
2015	4957
2016	5315
2017	6811
2018	7857

2.2. Atsakymas

Splainas yra funkcija, sudaryta iš interpoliacinių daugianarių, kurie yra sujungti taip, kad užtikrinamas sklandus perėjimas tarp interpoliacijos taškų, išvengiama aukštos eilės daugianarių problemos. Tai yra tiesioginis interpoliacijos metodas, nes jis neapibrėžia visų duotų taškų kaip kitų interpoliacijos metodų.

Trumpai, splainas sudaromas iš kelių lokalių taškų, o ne visos jų bendros visumos. Taip geriau nurodome funkcijos kryptį, gauname geresnius interpoliacijos rezultatus.



7 pav. Ermito(Akimo) splainas.

2.3. Kodas

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import Akima1DInterpolator

def spline(X, Y, n_points=100):
    akima_interpolator = Akima1DInterpolator(X, Y)
    sss = np.linspace(X[0], X[-1], n_points)
    S = akima_interpolator(sss)
    return sss, S

if __name__ == '__main__':
    X = np.array([1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007,
2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
         2016, 2017, 2018])
    Y = np.array([2173, 1764, 1807, 1835, 1923, 2084, 2106, 2293, 2178, 1982,
2186, 2485, 2657, 3051, 4030, 4239, 4686,
                  4957, 5315, 6811, 7857])

    sss, S = spline(X, Y)

    plt.plot(X, Y, 'ro', label='Taškai')
    plt.plot(sss, S, 'b-', label='Ermito (Akimo) splainas')
    plt.legend()
    plt.grid(True)
    plt.xlabel('Year')
    plt.ylabel('Total greenhouse gas emissions')
    plt.title('Zambia')
    plt.show()
```

3. III dalis. Aproksimavimas

3.1. Varianto uždaviniai

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.

3.2. Atsakymas:

Aproksimavimas yra procesas, kai vienos funkcijos ar duomenų rinkinio artinimas kitos funkcijos arba mažesnio duomenų rinkinio, siekiant supaprastinti analizę arba gauti efektyvesnį modelį.

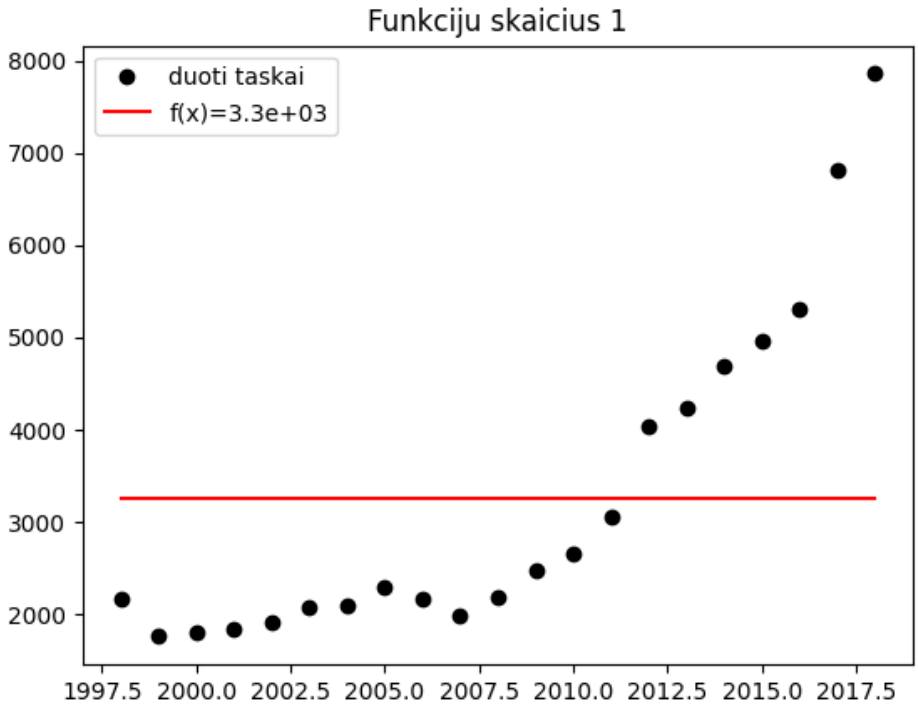
Duomenys taip pat naudojami tie patys kaip ir 2 darbo dalyje. Aproksimacija buvo atlikta su 1, 2, 3, 5 funkcijų skaičiumi.

Duomenys:

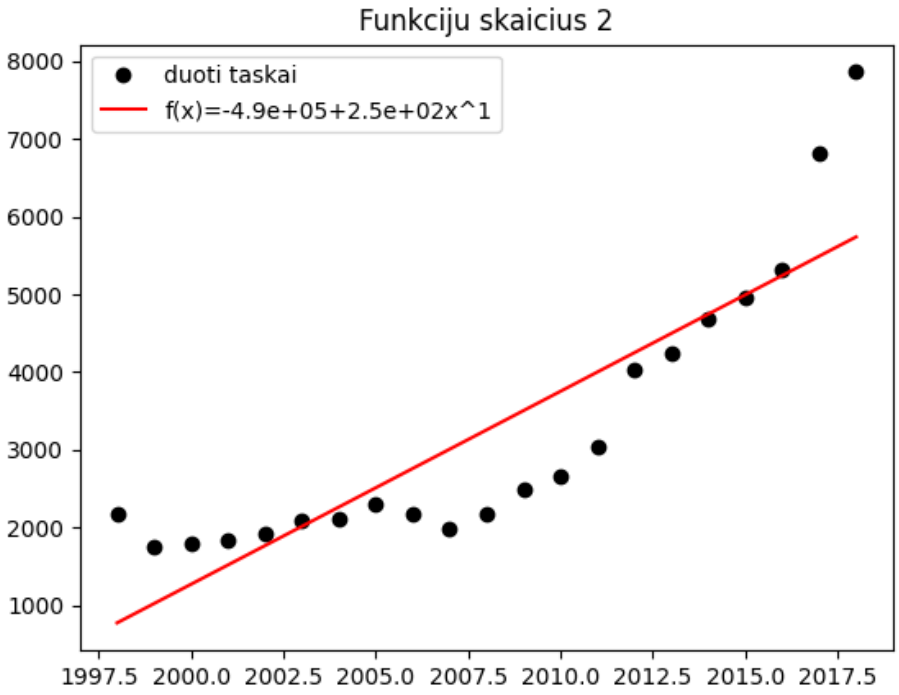
Metai	Šiltnamio dujų emisiją
1998	2173
1999	1764
2000	1807
2001	1835
2002	1923
2003	2084
2004	2106
2005	2293
2006	2178
2007	1982
2008	2186
2009	2485
2010	2657
2011	3051
2012	4030
2013	4239
2014	4686
2015	4957
2016	5315
2017	6811
2018	7857

```
gauta funkcija, m f(x)=3.3e+03 1
gauta funkcija, m f(x)=-4.9e+05+2.5e+02x^1 2
gauta funkcija, m f(x)=9.7e+07-9.7e+04x^1+   24x^2 3
gauta funkcija, m f(x)=2.9e+08+4.5e+06x^1-2.3e+03x^2+ 0.38x^3-5.2e-08x^4 5
```

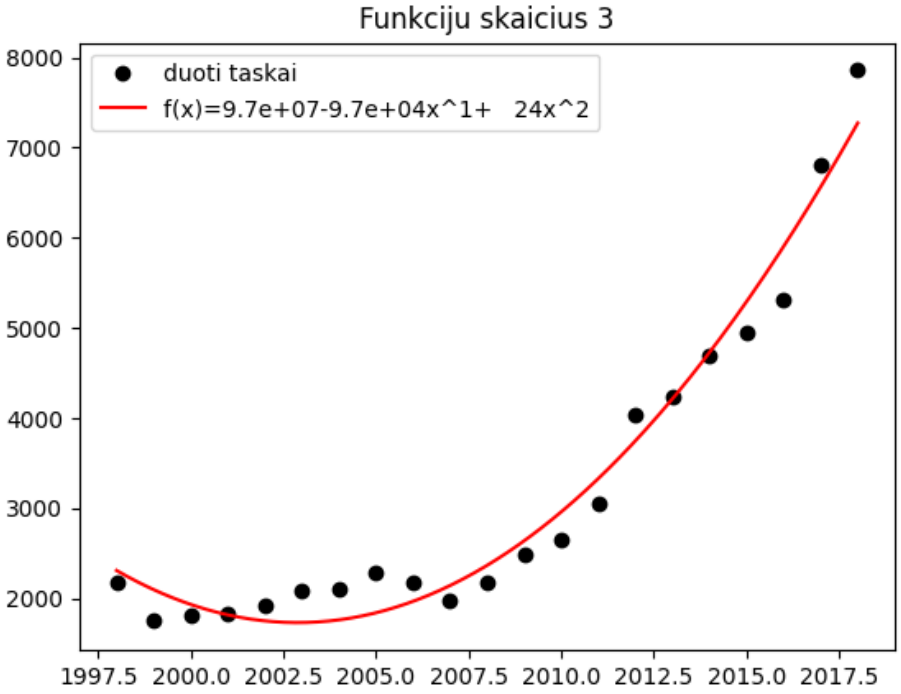
8 pav. Gautos 1,2,3,5 funkcijų daugianario išraiškos.



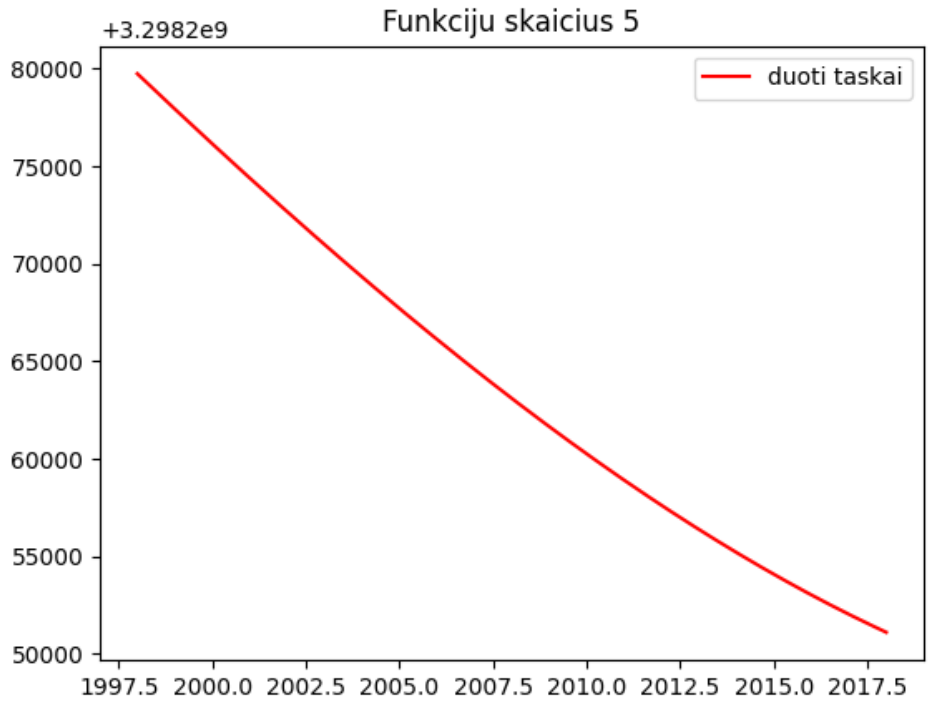
9 pav. Aproksimuojančios kreivės pirmos eilės daugianarių grafikas.



10 pav. Aproksimuojančios kreivės antros eilės daugianarių grafikas.



11 pav. Aproksimuojančios kreivės trečios eilės daugianarių grafikas.



12 pav. Aproksimuojančios kreivės penktos eilės daugianarių grafikas.

3.3. Kodas

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

def base(m, x):
    G = np.zeros((len(x), m))
    for i in range(m):
        G[:, i] = x**(i)
    return G
def plot_function(X, Y, m_values):
    xmin, ymin, xmax, ymax = 1998, -90000, 2018, 0

    plt.figure(1)
    plt.axis([xmin, xmax, ymin, ymax])
    plt.grid(True)
    plt.ion()
    plt.clf()
    hcurve = None
    for m in m_values:
        n = len(X)
        plt.plot(X, Y, 'ko')
        #plt.show()
        G = base(m, X)
        c = np.linalg.solve(np.dot(G.T, G), np.dot(G.T, Y))

        sss = f'{c[0]:5.2g}'
        for i in range(1, m):
            sss += f'+{c[i]:5.2g}x^{i}'

        sss = sss.replace('+-', '-')
```

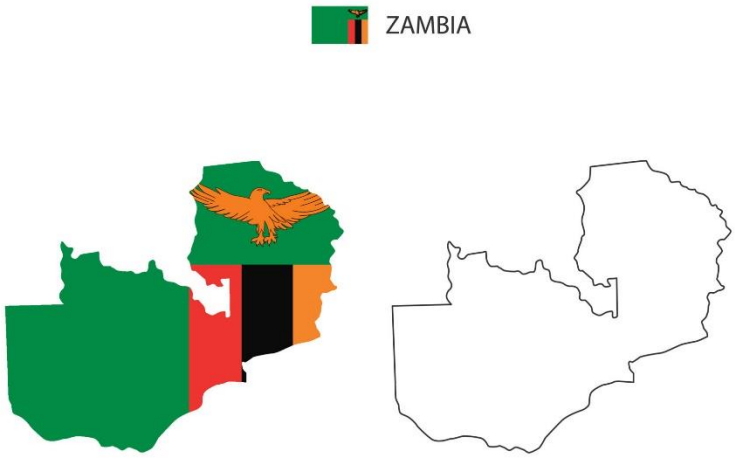
```
        print('gauta funkcija, m', 'f(x)=' + sss, m)

        nnn = 200
        xxx = np.linspace(xmin, xmax, nnn)
        Gv = base(m, xxx)
        fff = np.dot(Gv, c)

        if hcurve is not None:
            hcurve.remove()

        hcurve, = plt.plot(xxx, fff, 'r-')
        plt.legend(['duoti taskai', f'f(x)={sss}'])
        plt.title(f'Funkciju skaicius {m}')

        plt.pause(1)  # Pause for 1 second


    plt.ioff()
    plt.show()


if __name__ == "__main__":
    X = np.array([1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007,
2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
         2016, 2017, 2018])
    Y = np.array([2173, 1764, 1807, 1835, 1923, 2084, 2106, 2293, 2178, 1982,
2186, 2485, 2657, 3051, 4030, 4239, 4686,
                  4957, 5315, 6811, 7857])
    m_values = [1,2,3,5]
    plot_function(X, Y, m_values)
```

4. IV dalis. Parametrinis aproksimavimas

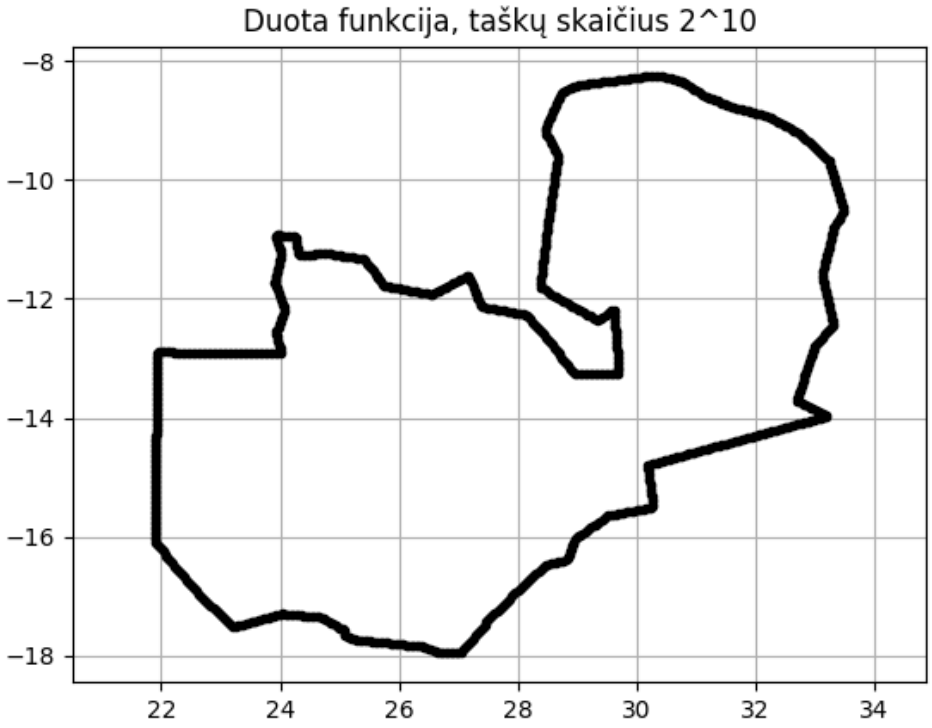
4.1. Varianto uždaviniai

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.

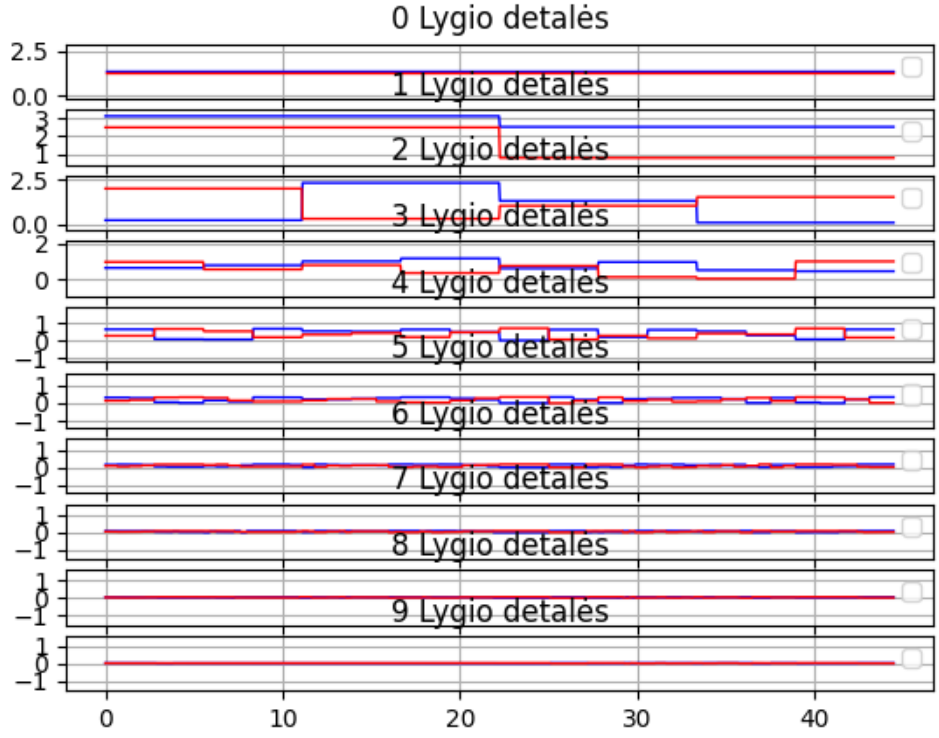
4.2. Atsakymas:



13 pav. Zambijos šalies kontūras palyginimui.

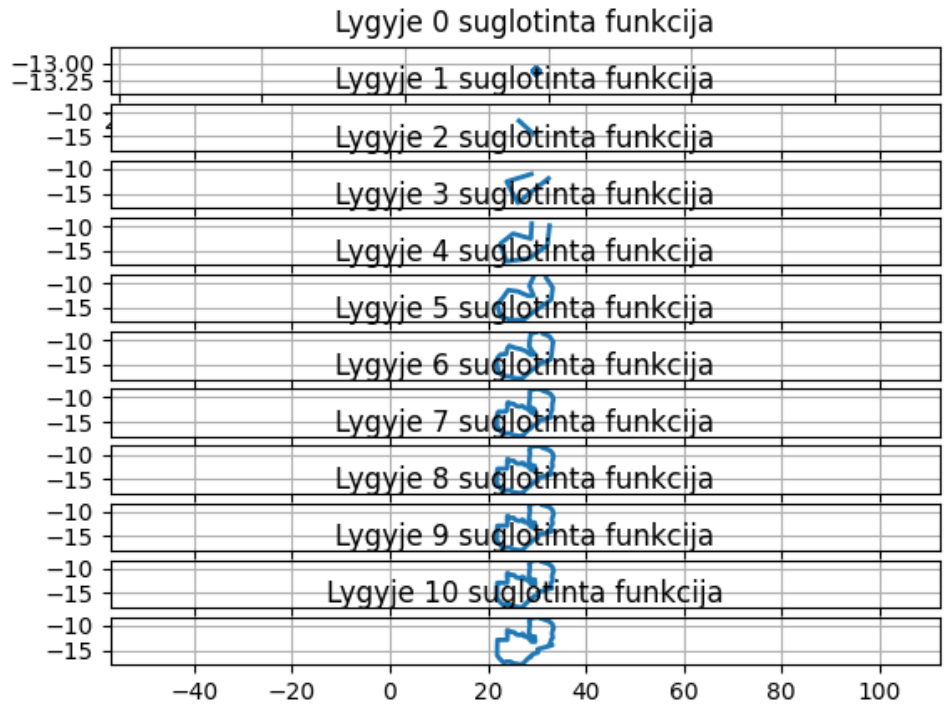


14 pav. Zambijos šalies kontūras aproksimuojant parametriškai Hara bangelėmis.



15 pav. Suglotinimai lygiuose

.



16 pav. Detalės lygiuose.

4.3. Kodas

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

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

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

def Haar_wavelet_approximation(SX, SY, n, m):
    a, b = min(SX), max(SX)
    nnn = 2**n
    smooth = (b - a) * SY * 2**(-n / 2)

    details = {}
    for i in range(1, m + 1):
        smooth1 = (smooth[::2] + smooth[1::2]) / np.sqrt(2)
        details[i] = (smooth[::2] - smooth[1::2]) / np.sqrt(2)
```

```
        #print(f"\ndetales {i}:", details[i])
        smooth = smooth1

    print(f"\nsmooth {m}:", smooth, "\n")
    return smooth, details

def main():
    plt.close('all')
    spalvos = ['r-', 'g-', 'm-', 'c-', 'k-', 'y-', 'r.', 'g.', 'm.', 'c.', 'k.',
'y.']
    n = 10
    nnn = 2**n

    with open('X-Zambia.txt', 'r') as fhx, open('Y-Zambia.txt', 'r') as fhy:
        SX = np.loadtxt(fhx)
        SY = np.loadtxt(fhy)

    nP = len(SX)
    t = np.zeros(nP)
    t[0] = 0
    for i in range(1, nP):
        t[i] = t[i - 1] + np.linalg.norm(np.array([SX[i], SY[i]]) - np.array([SX[i
- 1], SY[i - 1]]))

    plt.figure(1)
    plt.axis('equal')
    plt.grid(True)
    plt.plot(SX, SY, 'c')


    a, b = min(t), max(t)
    t1 = np.linspace(a, b, nnn)

    ts = interp1d(t, SX, kind='linear', fill_value='extrapolate')(t1)
    SX = ts

    ts = interp1d(t, SY, kind='linear', fill_value='extrapolate')(t1)
    SY = ts
    t = t1

    plt.plot(SX, SY, 'k.')
    #plt.plot(t, SX, 'b.')
    #plt.plot(t, SY, 'g.')
    plt.title(f"Given function, number of points 2^{n}")


    plt.title(f"Duota funkcija, taškų skaičius 2^{n}")

    xmin, xmax = min(SX), max(SX)
    ymin, ymax = min(SY), max(SY)

    m = 10
    smoothx, detailsx = Haar_wavelet_approximation(t, SX, n, m)
    smoothy, detailsy = Haar_wavelet_approximation(t, SY, n, m)

    print("smoothx:", smoothx)
    print("smoothy:", smoothy)

    hx = np.zeros(nnn)
    hy = np.zeros(nnn)
    for k in range(2**(n - m)):
        hx += smoothx[k] * Haar_scaling(t, n - m, k, a, b)
        hy += smoothy[k] * Haar_scaling(t, n - m, k, a, b)

    plt.figure(2)
```

```
    plt.subplot(m + 1, 1, 1)
    plt.axis('equal')
    plt.axis([xmin, xmax, ymin, ymax])
    plt.grid(True)
    plt.plot(hx, hy, '.', linewidth=2)
    plt.title(f"Lygyje {0} suglotinta funkcija")

    for i in range(m):
        h1x = np.zeros(nnn)
        h1y = np.zeros(nnn)
        for k in range(2**(n - m + i)):
            h1x += detailsx[m - i][k] * Haar_wavelet(t, n - m + i, k, a, b)
            h1y += detailsy[m - i][k] * Haar_wavelet(t, n - m + i, k, a, b)


        plt.figure(3)
        plt.subplot(m, 1, i + 1)
        plt.axis('equal')
        plt.grid(True)
        plt.plot(t, np.abs(h1x), 'b-', linewidth=1)
        plt.plot(t, np.abs(h1y), 'r-', linewidth=1)
        plt.title(f"{i} Lygio detalės")
        plt.legend()

        hx += h1x
        hy += h1y

        plt.figure(2)
        plt.subplot(m + 1, 1, i + 2)
        plt.axis('equal')
        plt.axis([xmin, xmax, ymin, ymax])
        plt.grid(True)
        plt.plot(hx, hy, linewidth=2)
        plt.title(f"Lygyje {i + 1} suglotinta funkcija")

    plt.show()

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

5. Išvados

Susipažinta su interpoliacija ir aproksimacija bei parametrinę aproksimacija.

6. Literatūra.

Galimas duomenų šaltinis: Šiltnamio dujų emisijos duomenys:

https://data.worldbank.org/indicator/EN.ATM.GHGT.KT.CE?end=2018&start=1998

Šalių kontūrai:

http://www.naturalearthdata.com/downloads/10m-cultural-vectors