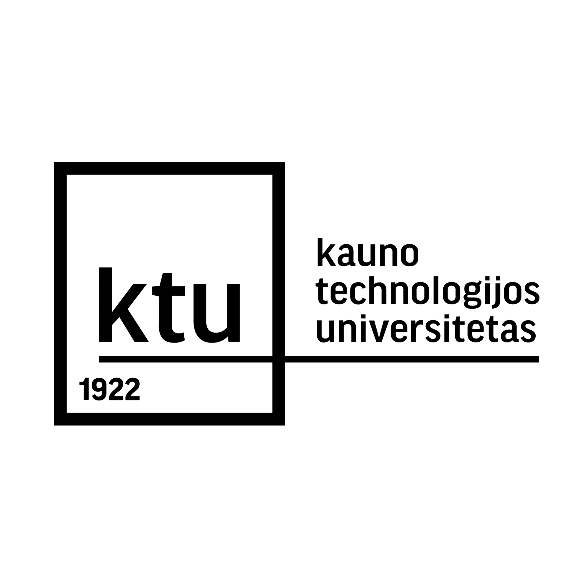
**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

**SKAITINIAI METODAI IR ALGORITMAI(P170B115)**

**3 LABORATORINIS DARBAS**

Varianto Nr. 5

**Atliko:**

IFF-1/8 gr. studentas

Matas Palujanskas**Priėmė:**

Prof. Rimantas Barauskas

Doc. Andrius Kriščiūnas

KAUNAS

2023

**Turinys**

[1. Pirma užduoties dalis 3](#_Toc151649003)

[1.1 Užduoties sąlygos 3](#_Toc151649004)

[1.2 Interpoliavimas daugianariu 4](#_Toc151649005)

[1.2.1 Sprendimo kodas 4](#_Toc151649006)

[1.2.2 Gauti rezultatai 6](#_Toc151649007)

[2. Antra užduoties dalis 8](#_Toc151649008)

[2.1. Užduoties sąlygos 8](#_Toc151649009)

[2.2. Sprendimo kodas 9](#_Toc151649010)

[2.3. Gauti rezultatai 10](#_Toc151649011)

[3. Trečia dalis 11](#_Toc151649012)

[3.1 Užduoties sąlyga 11](#_Toc151649013)

[3.2 Sprendimo kodas 11](#_Toc151649014)

[3.3 Gauti rezultatai 12](#_Toc151649015)

[3.3.1 Pirmos eilės daugianaris 12](#_Toc151649016)

[3.3.2 Antros eilės daugianaris 13](#_Toc151649017)

[3.3.3 Trečios eilės daugianaris 14](#_Toc151649018)

[3.3.4 Penktos eilės daugianaris 15](#_Toc151649019)

[4. Ketvirta dalis 16](#_Toc151649020)

[4.1 Užduoties sąlyga 16](#_Toc151649021)

[4.2 Sprendimo kodas 16](#_Toc151649022)

[4.3 Varianto šalies kontūrai 17](#_Toc151649023)

[4.4 Gauti rezultatai 18](#_Toc151649024)

[5. Literatūros sąrašas 21](#_Toc151649025)

# **Pirma** **užduoties dalis**

## Užduoties sąlygos

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, algebra

Automatiškai sugeneruotas aprašymas

pav. 1 Antrojo laboratorinio pirmos dalies užduotys

**Užduoties variantas: 5**

Paveikslėlis, kuriame yra tekstas, ekrano kopija, skaičius, Šriftas

Automatiškai sugeneruotas aprašymas

pav. 2 Interpoliuojamų funkcijų išraiškų lentelė



pav. 3 5 varianto funkcijos išraiška

## Interpoliavimas daugianariu

**Sprendimas:**

Uždavinys bus sprendžiamas Čiobyševo metodu, todėl reikšmes reikės persivesti į Čiobyševo formą ir atvirkščiai. Šiuos pervedimus daro metodai ciobysevo\_mazgas. Pagal užduoties reikalavimus taškai gali būti pasiskirstę tolygiai arba pagal Čiobyševo abscises. Apskaičiavus šias reikšmes formuojami vaizdavimo vektoriai, kurie paskui yra panaudojami braižant funkcijos reikšmes.

*Paveikslėlis, kuriame yra Šriftas, linija, tekstas, baltas

Automatiškai sugeneruotas aprašymas*

pav. 4 Čiobyševo abscisės

### Sprendimo kodas

**Lab3-1.py:**

import numpy as np  
import matplotlib  
matplotlib.use('TkAgg')  
from numpy import sin, cos, arccos, pi, exp  
import matplotlib.pyplot as plt  
import sympy as sym  
  
plt.style.use('ggplot')  
  
*# Duota funkcija*def funkcija(x):  
 return (exp(-x\*\*2)) \* sin(x\*\*2) \* (x-3)  
  
*# Čiobyševo polinomo formulė*def ciobysevo\_daugianaris(x, i):  
 return cos(arccos(x) \* i)  
  
*# Čiobyševo polinomo funkcijos formulė spausdinimui*def sym\_ciobysevo\_daugianaris(x, i):  
 return sym.cos(sym.acos(x) \* i)  
  
*# Čiobyševo intervalo formulė*def ciobysevo\_intervalas(x, a, b):  
 return (2 \* x - (b + a)) / (b - a)  
  
*# Čiobyševo mazgo transformacija*def ciobysevo\_mazgas(i, pr, pb, n):  
 return ((pb - pr) / 2) \* cos(pi \* (2 \* i + 1) / (2 \* n)) + ((pb + pr) / 2)  
  
*# Spausdinimo funkcija*def spausdinti\_ciobysevo(i, koffs):  
 x = sym.Symbol('x')  
 koffs = koffs.flatten()  
 xc = ciobysevo\_intervalas(x, -3, 2)  
 for i in range(len(koffs)):  
 A = sym\_ciobysevo\_daugianaris(xc, i)  
 if i == 0:  
 print(str(koffs[i]) + ' +')  
 elif i == 1:  
 print(str(koffs[i]) + ' \* (' + str(A) + ') +')  
 else:  
 print(str(koffs[i]) + ' \* ' + str(A) + ' +')  
  
print('Pasirinkite taškų išdėstymo tipą:')  
print('Tolygiai - tl')  
print('Čiobyševo - cb')  
  
n = 15 *# interpoliavimo taškų skaičius*i = np.arange(n)  
  
if str(input("Pasirinkta: ")) == "tl":  
 x = np.linspace(-3, 2, n).reshape(-1, 1)  
 plot\_name = 'Taškai tolygiai pasiskirstę'  
else:  
 x = ciobysevo\_mazgas(i, -3, 2, n).reshape(-1, 1)  
 plot\_name = ' Taškai naudojant Čiobyševo abscises'  
  
int\_ciobysevo = ciobysevo\_intervalas(x, -3, 2)  
cb\_daugianaris = ciobysevo\_daugianaris(int\_ciobysevo, i)  
koffs = np.linalg.solve(cb\_daugianaris, funkcija(x))  
  
x = np.linspace(-3, 2, 100).reshape(-1, 1)  
  
cb\_i\_intervalas = ciobysevo\_intervalas(x, -3, 2)   
cb\_i\_daugianaris = ciobysevo\_daugianaris(cb\_i\_intervalas, i)  
int = np.dot(cb\_i\_daugianaris, koffs)  
  
plt.plot(x, int, label='Interpoliuota, n=15')  
plt.plot(x, funkcija(x), label='Duota')  
plt.plot(x, funkcija(x) - int, label='Netiktis')  
plt.legend()  
plt.title(plot\_name)  
plt.show()

### Gauti rezultatai

Pirmame grafike atvaizduojamas interpoliuojamos funkcijos rezultatas, kai taškai pasiskirstę tolygiai:

Paveikslėlis, kuriame yra tekstas, Grafikas, linija, diagrama

Automatiškai sugeneruotas aprašymas

pav. 5 Daugianario interpoliacija tolygiai

Mėlyna spalva pažymėta duota funkcija, violetine netiktis, interpoliuota raudona, buvo naudota 15 taškų.

Antrame grafike vaizduojamas interpoliuojamos funkcijos rezultatas, kai taškai pasiskirstę pagal Čiobyševo abscises:

Paveikslėlis, kuriame yra tekstas, Grafikas, ekrano kopija, linija

Automatiškai sugeneruotas aprašymas

pav. 6 Daugianario interpoliacija pagal Čiobyševą

Mėlyna spalva pažymėta duota funkcija, violetine netiktis, interpoliuota raudona, buvo naudota taip pat 15 taškų.

**Išvada:** galima pastebėti, kad netiktis prie Čiobyševo interpoliacijos yra sumažinta kraštiniuose intervalo taškuose. Tai būtent yra Čiobyševo abscisės tikslas - optimaliai pasiskirstyti taškus taip, kad sumažėtų interpoliacijos netiktis kraštiniuose intervalo taškuose. Taigi, galima patvirtinti, kad Čiobyševo metodas taškus paskirsto geriau nei tolyginis paskirstymas.

# Antra užduoties dalis

## Užduoties sąlygos

Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, algebra

Automatiškai sugeneruotas aprašymas

pav. 7 Antros dalies sąlygos

5 varianto šalis:



pav. 8 5 varianto šalis ir splaino metodas

Paveikslėlis, kuriame yra Šriftas, linija, rankraštis, baltas

Automatiškai sugeneruotas aprašymas

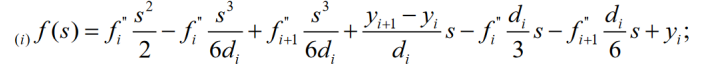
pav. 9 Splaino koeficientai

Paveikslėlis, kuriame yra diagrama, linija, tekstas, Planas

Automatiškai sugeneruotas aprašymas

pav. 10 Antrų išvestinių skaičiavimų matricos

Apskaičiavus antrų išvestinių reikšmes galima skaičiuoti splainus:



pav. 11 Antrų išvestinių reikšmių gavimas

## Sprendimo kodas

**Lab3-2.py:**

import numpy as np  
import matplotlib  
matplotlib.use('TkAgg')  
import matplotlib.pyplot as plt  
from scipy.interpolate import UnivariateSpline  
import pandas as pd  
  
plt.style.use('ggplot')  
  
def interpoliuoti\_globalu(x, y):  
 *# Surandame interpoliacinį splainą* spline = UnivariateSpline(x.ravel(), y.ravel(), s=0)  
  
 *# Rekonstruojame reikšmes naujuose taškuose* x\_n = np.linspace(x.min(), x.max(), 100)  
 y\_n = spline(x\_n)  
  
 plt.title('Kroatijos emisija 1998-2018 (Globalus)')  
 plt.scatter(x, y, label='Esami', color='red') *# Scatter plot for existing data points* plt.plot(x\_n, y\_n, 'g-', label='Interpoliuoti')  
 plt.legend()  
 plt.xticks(np.arange(20), np.arange(1, 21))  
 plt.show()  
  
n = 20 *# 20 metų*data = pd.read\_csv('Croatia\_Emissions.csv')  
x = data.index.to\_numpy().reshape(-1, 1)  
y = data.iloc[:, 0].to\_numpy().reshape(-1, 1)  
del data  
  
interpoliuoti\_globalu(x, y)

## Gauti rezultatai

Atvaizduota interpoliuojanti kreivė Globaliu splaino tipu:

*Paveikslėlis, kuriame yra tekstas, diagrama, Grafikas, ekrano kopija

Automatiškai sugeneruotas aprašymas*

pav. 12 Kroatijos emisija

**Išvada**: globalaus metodo splainas yra pakankamai tikslus, taip pat buvo patogu naudoti scipy.interpolate biblioteką.

# Trečia dalis

## Užduoties sąlyga

Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, linija

Automatiškai sugeneruotas aprašymas

pav. 13 Trečios dalies sąlyga

## Sprendimo kodas

**Lab3-3.py:**

import numpy as np  
import matplotlib  
matplotlib.use('TkAgg')  
import pandas as pd  
import matplotlib.pyplot as plt  
  
plt.style.use('ggplot')  
  
  
def kelti(x, laipsnis):  
 x = x.flatten() *# iš 2d į 1d* return np.array([x \*\* i for i in range(laipsnis)]).T.astype(float) *# skaičius pakeliamas tam tikru laipsniu*def b(x, koffs): *# aproksimavimui x pakelti ir su koffu sudaugint* koffs = koffs.flatten() *# iš 2d į 1d* ats = 0  
 for i in range(len(koffs)):  
 ats += koffs[i] \* x \*\* i  
 print(f'{koffs[i]} \* x ^ {i} +')  
 return ats  
  
laipsnis = int(input('Daugianario laipsnis: '))  
laipsnis += 1  
duomenys = pd.read\_csv('Croatia\_Emissions.csv')  
x = duomenys.index.to\_numpy().reshape(-1, 1) *# indeksai/mėnesiai*y = duomenys.iloc[:, 0].to\_numpy().reshape(-1, 1) *# temperatūros*G = kelti(x, laipsnis)  
koffs = np.linalg.solve(G.T.dot(G), G.T.dot(y)) *# išsprendžia lygčių sistema*xx = np.arange(0, 21, 0.01).reshape(-1, 1) *# į 2d*plt.plot(x, y, 'D', label=f'esami = {len(x)}')  
plt.plot(xx, b(xx, koffs), 'g-', label=f'aproksimavimas = {laipsnis-1}')  
plt.xticks(np.arange(22), np.arange(1, 23))  
plt.title('Kroatijos emisija 1998-2018')  
plt.legend()  
plt.show()

## Gauti rezultatai

### 3.3.1 Pirmos eilės daugianaris

Paveikslėlis, kuriame yra tekstas, ekrano kopija, diagrama, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 14 Pirmos eilės daugianario grafinis atvaizdavimas

Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, tipografija

Automatiškai sugeneruotas aprašymas

pav. 15 Pirmos eilės daugianario išraiška

### 3.3.2 Antros eilės daugianaris

Paveikslėlis, kuriame yra tekstas, ekrano kopija, diagrama, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 16 Antros eilės daugianario grafinis atvaizdavimas

*Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, tipografija

Automatiškai sugeneruotas aprašymas*

pav. 17 Antros eilės daugianario išraiška

### 3.3.3 Trečios eilės daugianaris

Paveikslėlis, kuriame yra tekstas, diagrama, ekrano kopija, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 18 Trečios eilės daugianario grafinis atvaizdavimas

Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, tipografija

Automatiškai sugeneruotas aprašymas

pav. 19 Trečios eilės daugianario išraiška

### 3.3.4 Penktos eilės daugianaris

Paveikslėlis, kuriame yra tekstas, diagrama, Grafikas, ekrano kopija

Automatiškai sugeneruotas aprašymas

pav. 20 Penktos eilės daugianario grafinis atvaizdavimas

Paveikslėlis, kuriame yra tekstas, Šriftas, ekrano kopija, dizainas

Automatiškai sugeneruotas aprašymas

pav. 21 Penktos eilės daugianario išraiška

**Išvada:** mažiausių kvadratų metodu buvo sudarytos aproksimuojančios kreivės, rezultatai pasitvirtino, esant vis aukštesnei eilei aproksimavimas darosi tikslesnis.

# Ketvirta dalis

## Užduoties sąlyga

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, linija

Automatiškai sugeneruotas aprašymas

pav. 22 4 dalies sąlyga

## Sprendimo kodas

**Lab3-4.py:**

import matplotlib.pyplot as plt  
import matplotlib  
matplotlib.use('TkAgg')  
import numpy as np  
from interpolation import parametric\_interpolation, hermite\_interpolation\_spline  
from lab3\_4\_data import x\_range, y\_range  
  
n = 500 *# Number of interpolation points*step = 0.1 *# Graph's precision  
  
# Reducing interpolation points to selected*t = range(n + 1)  
x\_range = x\_range[::(2359 // n)]  
y\_range = y\_range[::(2359 // n)]  
  
ff = hermite\_interpolation\_spline(t, x\_range)  
ff2 = hermite\_interpolation\_spline(t, y\_range)  
  
xx, yy = parametric\_interpolation(ff, ff2, np.arange(0, n, step))  
  
*# Plot country borders as a line*plt.plot(x\_range, y\_range, 'r-', label="Šalies ribos")  
  
*# Scatter plot for interpolation points*plt.scatter(x\_range, y\_range, color='blue', marker='o', label=f"{n} Aproksimavimo taškai")  
  
plt.title('Kroatija')  
plt.legend()  
plt.show()

## Varianto šalies kontūrai

Gauto 5 varianto šalis yra Kroatija, jos kontūrai iš countries.zip:

Paveikslėlis, kuriame yra tekstas, diagrama, žemėlapis, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 23 Originalūs Kroatijos kontūrai

## Gauti rezultatai

**Tikslas yra gauti kuo tikslesnius šalies kontūrus.**

**Grafiko tikslumas, pradinis žingsnis yra 0,1.**

Pradžioje buvo aproksimuojama su 10 taškų:

Paveikslėlis, kuriame yra tekstas, linija, ekrano kopija, diagrama

Automatiškai sugeneruotas aprašymas 

pav. 24 Šalies kontūrai naudojant 10 taškų

Naudojant 20 taškų su 0.1 žingsniu:

Paveikslėlis, kuriame yra linija, diagrama, tekstas, Grafikas

Automatiškai sugeneruotas aprašymas



pav. 25 Šalies kontūrai naudojant 20 taškų

Naudojant 75 taškus ir žingsnį pakeitus į 0.5:

Paveikslėlis, kuriame yra linija, tekstas, diagrama, ekrano kopija

Automatiškai sugeneruotas aprašymas



pav. 26 Šalies kontūrai naudojant 75 taškus

Ties tokiu taškų skaičiumi galime pastebėti, jog ryškėja šalies kontūrai. Taip pat su didesniu žingsniu programa veikia greičiau.

Naudojant 200 taškų ir žingsnį pakeitus į 0.5:

Paveikslėlis, kuriame yra ekrano kopija, linija, Grafikas, diagrama

Automatiškai sugeneruotas aprašymas



pav. 27 Šalies kontūrai naudojant 200 taškų

Naudojant 750 taškų su 0.1 žingsniu:

Paveikslėlis, kuriame yra žemėlapis, ekrano kopija, tekstas, linija

Automatiškai sugeneruotas aprašymas



pav. 28 Šalies kontūrai naudojant 750 taškų

Su tokiu kiekiu taškų programa jau dirba ilgiau. Galime pastebėti, jog yra jau aiškūs šalies kontūrai, tačiau dar yra netikslumų.

Naudojant 1500 taškų su 0.1 žingsniu:

Paveikslėlis, kuriame yra tekstas, ekrano kopija, diagrama, linija

Automatiškai sugeneruotas aprašymas



pav. 29 Šalies kontūrai naudojant 1500 taškų

Su tokiu taškų kiekiu jau galime aiškiai matyti Kroatijos kontūrus.

**Išvada**: didėjant taškų kiekiui, buvo tikslinamas ir šalies kontūras, tačiau ilgėjo ir programos vykdomas laikas.

# Literatūros sąrašas

1. „Skaitiniai metodai ir algoritmai“ „Moodle“ aplinkoje [HTTPS://MOODLE.KTU.EDU/COURSE/VIEW.PHP?ID=7639](https://moodle.ktu.edu/course/view.php?id=7639)