KAUNO TECHOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Skaitiniai metodai ir algoritmai (P170B115)

Laboratorinis darbas nr. 3

Varianto nr. 9

Atliko:

IFF 8/3 gr. studentas

Dovydas Zamas

Priėmė:

doc. ČALNERYTĖ Dalia

Turinys

1. Interpoliavimas daugianariu

1.1. Užduotis

duota interpoliuojamos funkcijos analitinė išraiška. Pateikite interpoliacinės funkcijos išraišką naudodami 1 lentelėje nurodytas bazines funkcijas, kai:

- Taškai pasiskirstę tolygiai.
- 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, iterpoliuojančią funkciją ir netektį.

1.2. Interpoliuojančios funkcijos išraiška

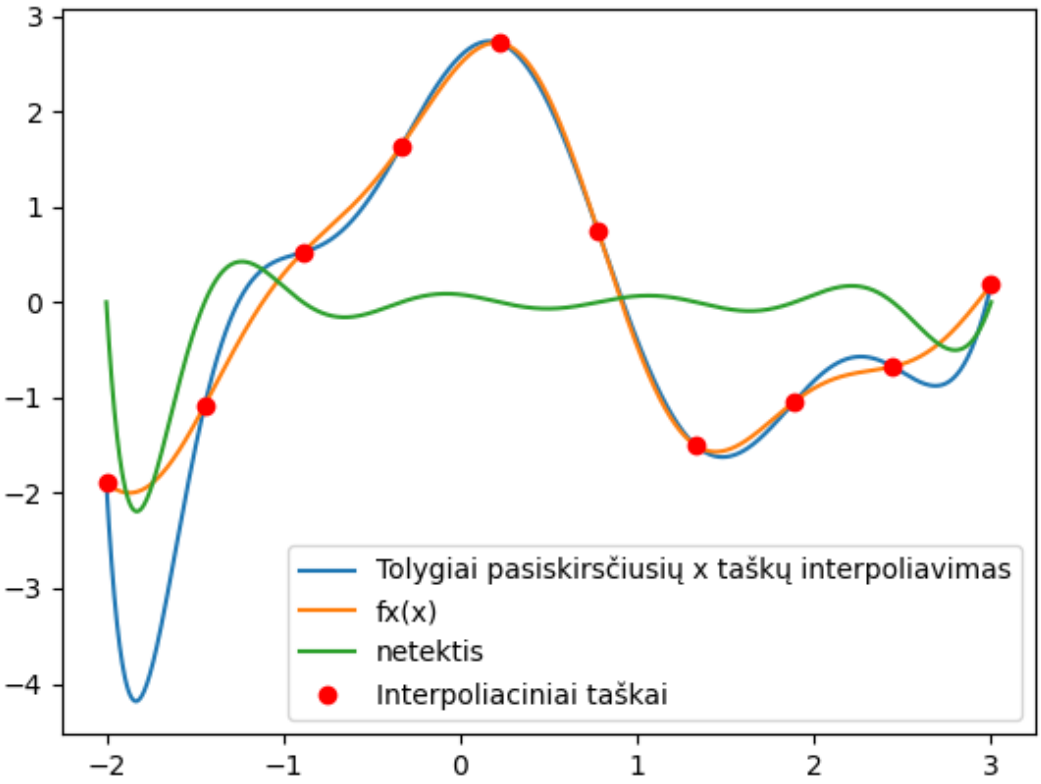
Funkcijos išraiška: $\sin(2 * x) * (\sin(2 * x) + 1{,}5) + \cos x; -2 \leq x \leq 3;$

Bazinė funkcija: Niutono

Vaizdavimo taškų skaičius: 1000

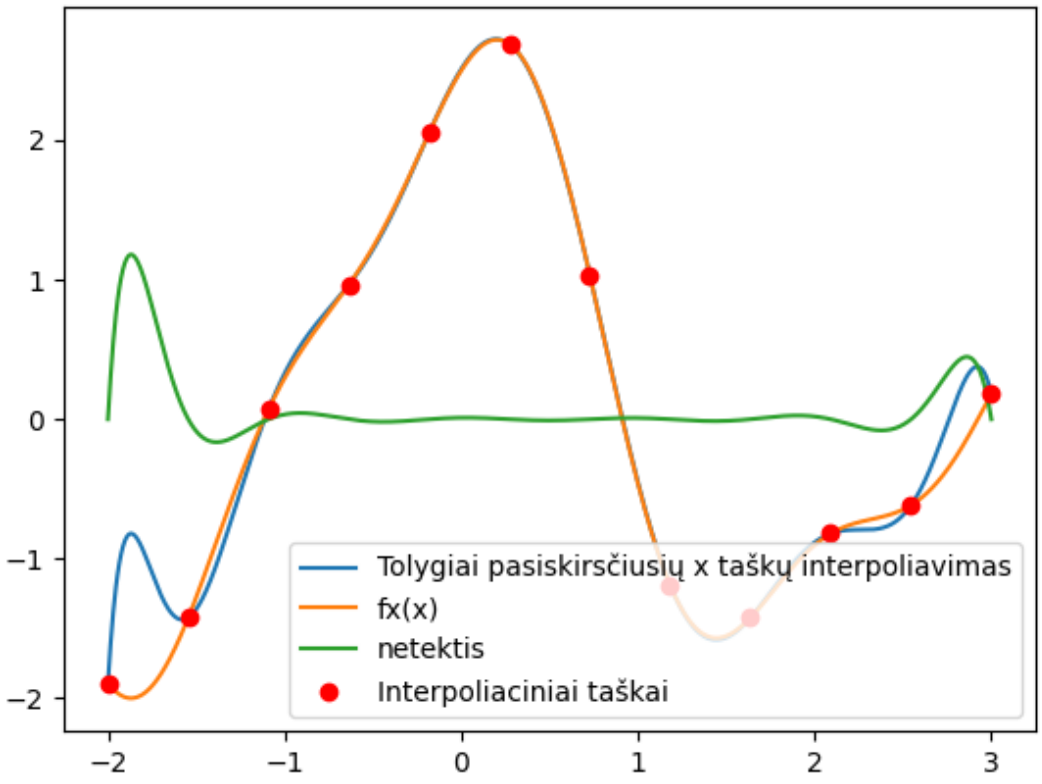
1.3. Interpoliuojančios funkcijos analitinė išraiška kai taškai pasiskirstę tolygiai

Interpoliavimo taškų skaičius: 10



pav. 1 Tolygiai pasiskirsčiusių x taškų interpoliavimo grafikas, kai n = 10

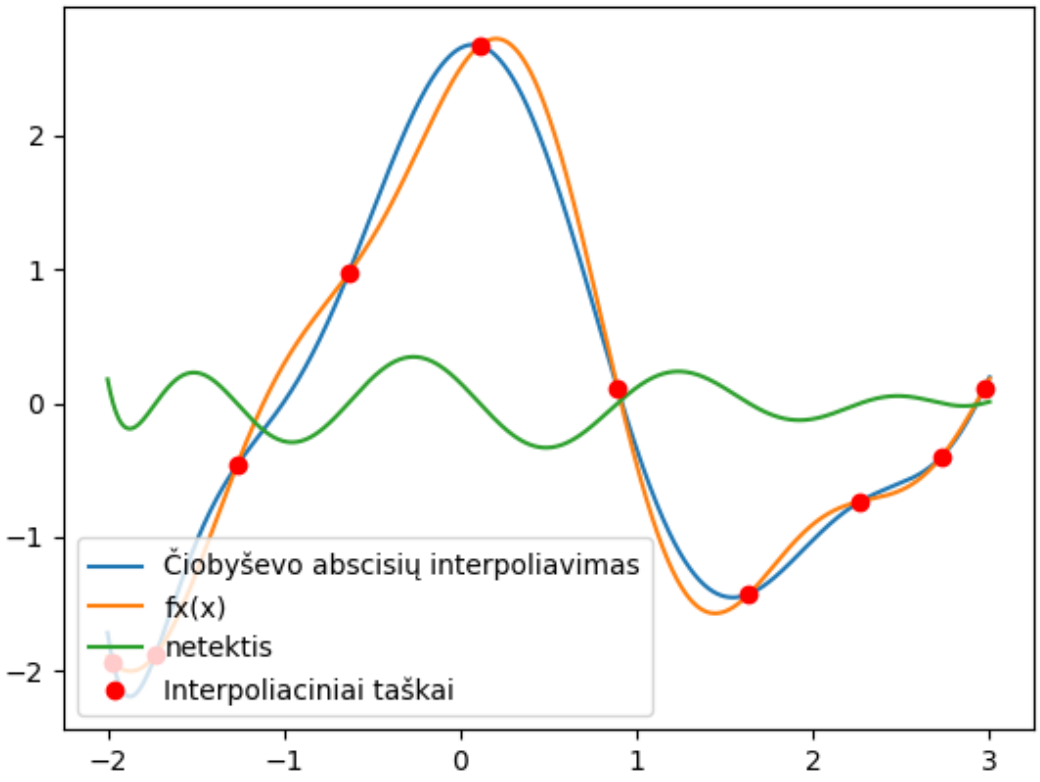
Interpoliavimo taškų skaičius: 12



pav. 2 Tolygiai pasiskirsčiusių x taškų interpoliavimo grafikas, kai n = 12

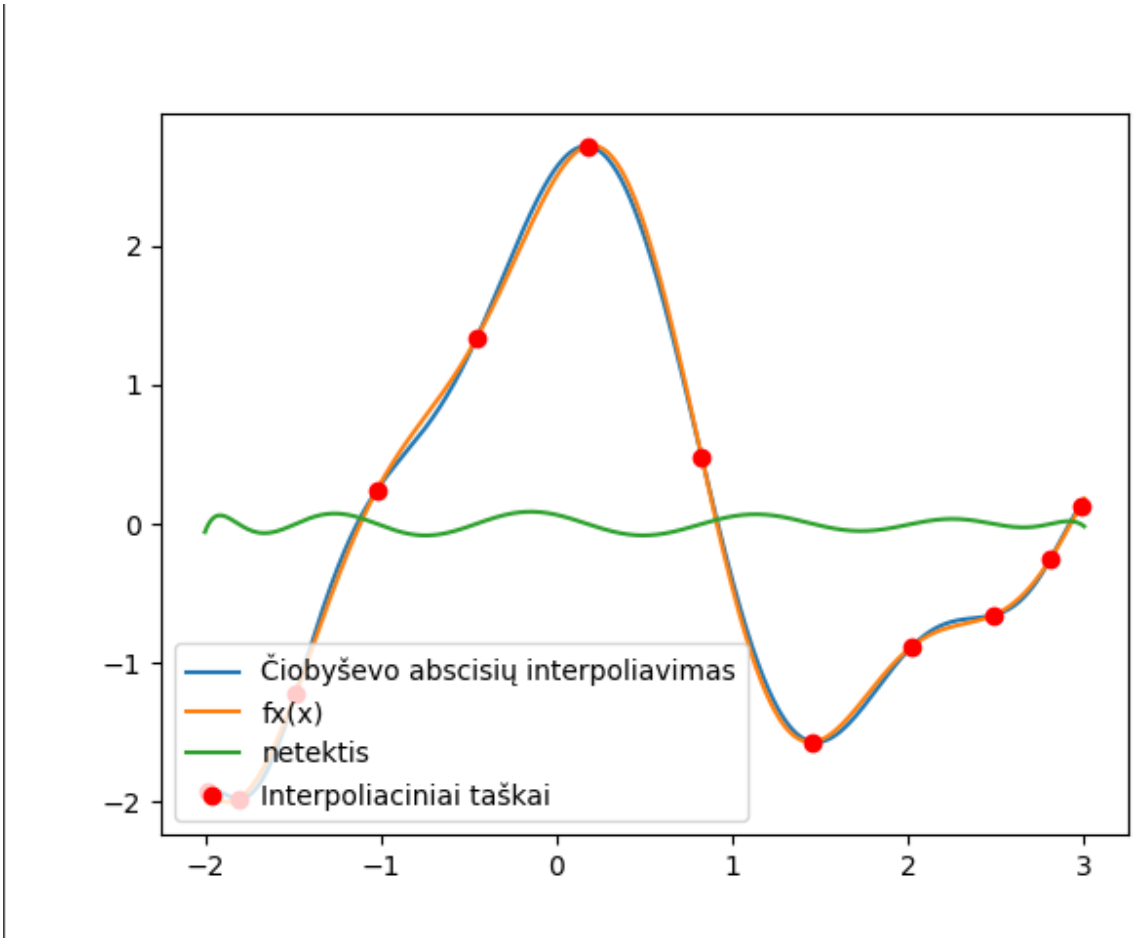
1.4. Interpoliuojančios funkcijos analitinė išraiška naudojant Čiobyševo abscises

Interpoliavimo taškų skaičius: 10



pav. 3 Čiobyševo abscisių interpoliavimo grafikas, kai n = 10

Interpoliavimo taškų skaičius: 12



pav. 4 Čiobyševo abscisių interpoliavimo grafikas, kai n = 12

1.5. Pirmosios užduoties programinis kodas

```
# %% first task


CONST_xMin = -2
CONST_xMax = 3
CONST_pointCount = 10


def ChebyshevAbscissa(n):
    x1 = (CONST_xMax - CONST_xMin) / 2
    x2 = (CONST_xMax + CONST_xMin) / 2
    i = np.array(range(n))
    TmpAbscissa = np.cos(np.pi * (2 * i + 1) / (2 * n))
    return x1 * TmpAbscissa + x2


def fx(x):
    return np.cos(2 * x) * (np.sin(2 * x) + 1.5) + np.cos(x)


# CONST_xPoints = ChebyshevAbscissa(CONST_pointCount)

fixed_step_xPoints = np.linspace(CONST_xMin, CONST_xMax, CONST_pointCount)
chebyshev_abscissa_xPoints = ChebyshevAbscissa(CONST_pointCount)
fixed_step_yMatrix = np.matrix(fx(fixed_step_xPoints)).transpose()
chebyshev_abscissa_yMatrix =
np.matrix(fx(chebyshev_abscissa_xPoints)).transpose()


def NewtonsFunction(x, xPoints, yMatrix):
    A = np.zeros((CONST_pointCount, CONST_pointCount))
    A[:, 0] = 1
    for i in range(1, CONST_pointCount):
        tmp = 1
        for j in range(0, i):
            tmp *= xPoints[i] - xPoints[j]
            A[i, j + 1] = tmp
    a = np.linalg.solve(A, yMatrix)
    y = 0
    tmp = 0
    for i in range(0, len(a)):
        if i == 0:
            y += a[i]
        else:
            tmp = a[i]
            for j in range(0, i):
                tmp *= (x - xPoints[j])
        y += tmp
    return y[0, 0]


def plot(y, label, points_count):
    x = np.linspace(-2, 3, points_count)
    plt.plot(x, y,
             label=label)
```

```
def ExecuteFirstTask():
    plot_chebyshev_abscissa_y = []
    plot_fixed_step_y = []
    for x in np.linspace(-2, 3, 1000):
        plot_fixed_step_y.append(NewtonsFunction(x, fixed_step_xPoints,
fixed_step_yMatrix))
        plot_chebyshev_abscissa_y.append(NewtonsFunction(x,
chebyshev_abscissa_xPoints, chebyshev_abscissa_yMatrix))
    fxy = fx(np.linspace(CONST_xMin, CONST_xMax, 1000))

    plot(plot_fixed_step_y, "Tolygiai pasiskirsčiusių x taškų
interpoliavimas", 1000)
    plot(fxy, "fx(x)", 1000)
    plot(plot_fixed_step_y - fxy, "netektis", 1000)
    plt.plot(np.linspace(CONST_xMin, CONST_xMax, CONST_pointCount),
             fx(np.linspace(CONST_xMin, CONST_xMax, CONST_pointCount)), 'o',
color='r', label="Interpoliaciniai taškai")
    plt.legend(loc='best')
    plt.show()

    plot(plot_chebyshev_abscissa_y, "Čiobyševo abscisių interpoliavimas",
1000)
    plot(fxy, "fx(x)", 1000)
    plot(plot_chebyshev_abscissa_y - fxy, "netektis", 1000)
    plt.plot(chebyshev_abscissa_xPoints, fx(chebyshev_abscissa_xPoints), 'o',
color='r',
             label="Interpoliaciniai taškai")
    plt.legend(loc='best')
    plt.show()
```

2. Interpoliavimas daugianariu ir splainu per duotus taškus

2.1. Užduotis

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)).

2.2. Duomenys interpoliavimui daugianariu ir splainu

Šalis: Panama

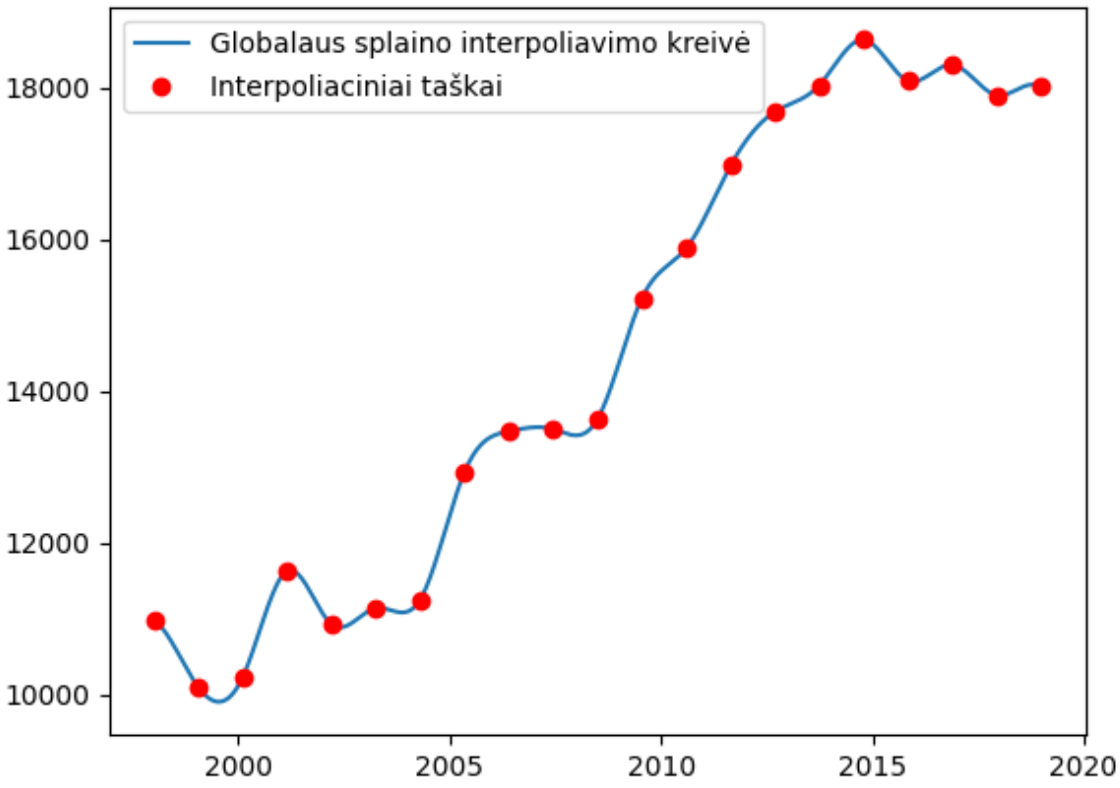
Splainas: Globalus

Vaizdavimo taškų skaičius: 1000

Metai: 1998-2018

Duomenys: šiltnamio dujų emisija

2.3. Rezultatas



pav. 5 Globalaus splaino interpoliavimo grafikas

2.4. Antrosios užduoties programinis kodas

```
# %% second task
def Panama():
    def process(data_from_csv):
        result = data_from_csv.astype(np.int32)
        return result

    with open("data.csv") as tmp_data:
        data = csv.reader(tmp_data, delimiter=',')
        for line in data:
            if line[0] == "Panama":
                res = np.array(line[42:63])
                year = np.array(np.linspace(1998, 2019, 22))
                return process(res), process(year)


# s = x-xi
# di = xi+1-xi
def calculate_d(year):
    res = []
    for i in range(0, len(year) - 1):
        res.append(year[i + 1] - year[i])
    return res


def calculate_A(data):
    A = np.zeros((len(data), len(data)))
    for i in range(0, len(data) - 2):
        for j in range(i, i + 1):
            A[i, j] = data[i] / 6
            A[i, j + 1] = (data[i] + data[i + 1]) / 3
            A[i, j + 2] = data[i + 1] / 6
    A[-2, 0] = A[-2, -1] = 1 / 3
    A[-2, 1] = A[-2, -2] = 1 / 6
    A[-1, 0] = 1
    A[-1, -1] = -1
    return A


def calculate_b(y, d):
    b = np.zeros((len(y), 1))
    for i in range(0, len(y)):
        try:
            b[i, 0] = (y[i + 2] - y[i + 1]) / d[i + 1] - (y[i + 1] - y[i]) /
d[i]
        except:
            b[-2, 0] = (y[1] - y[0]) / d[0] - (y[-1] - y[-2]) / d[-2]
            b[-1, 0] = 0
            return b
    return b


def func(f, d, xi, y):
    res = []
    for x in np.linspace(xi[0], xi[-1], 1000):
        i = np.where(xi == math.floor(x))[0][0]
```

```
        s = x - xi[i]
        try:
            res.append(f[i, 0] * s ** 2 / 2 - f[i, 0] * s ** 3 / (6 * d[i]) +
f[i + 1, 0] * s ** 3 / (6 * d[i]) + (
                    y[i + 1] - y[i]) / d[i] * s - f[i, 0] * (
                                (d[i]) / 3) * s - f[i + 1, 0] * ((d[i] / 6) *
s) + y[i])
        except:
            res.append(y[i])
            break
    return res


def ExecuteSecondTask():
    y, x = Panama()
    d = calculate_d(x[:])
    A = calculate_A(d)
    b = calculate_b(y, d)
    f = np.linalg.solve(A, b)
    res = func(f, d, x, y)
    np.set_printoptions(threshold=np.inf)
    plt.plot(np.linspace(x[0], x[-1], len(res)), res, label="Globalaus
splaino interpoliavimo kreivė")
    plt.plot(np.linspace(x[0], x[-1], len(y)), y, 'o', color='r',
label="Interpoliaciniai taškai")
    plt.legend(loc='best')
    plt.show()
```

3. Mažiausių kvadratų aproksimavimas

3.1. Užduotis

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.

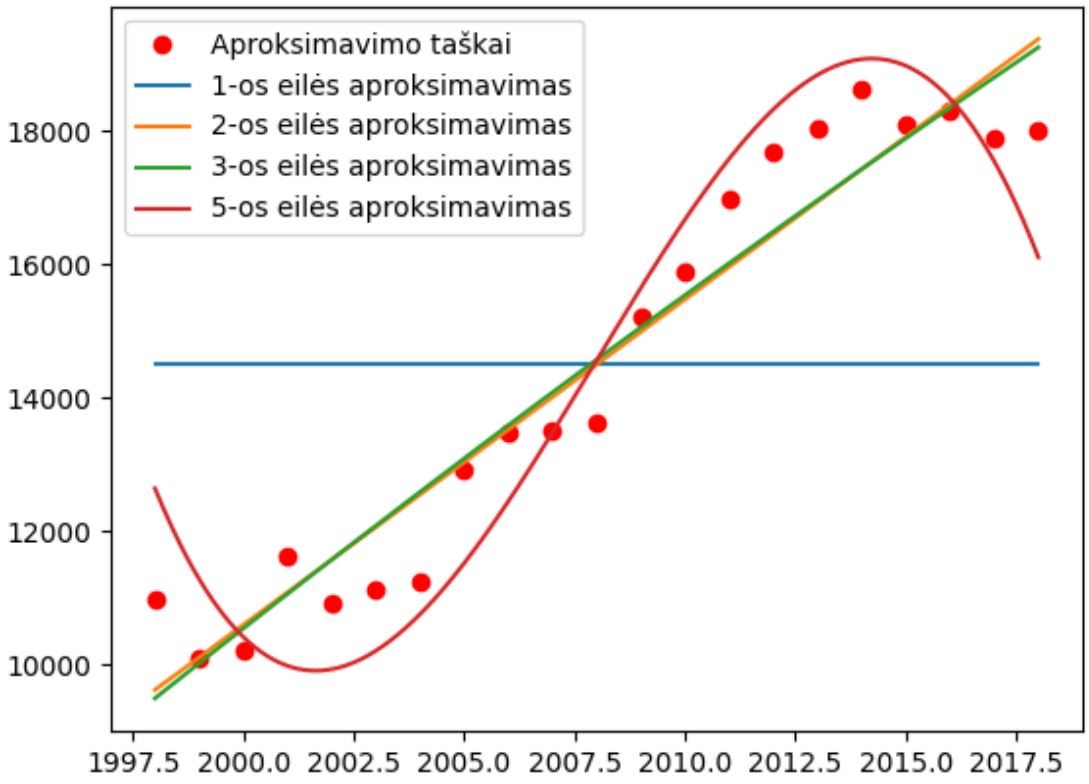
3.2. Duomenys mažiausių kvadratų aproksimavimui

Šalis: Panama

Duomenys: šiltnamio dujų emisija

Eilė: 1,2,3,5

3.3. Rezultatas



pav. 6 Skirtingų eillių aproksimavimo kreivių grafikas

3.4. Trečiosios užduoties programinis kodas

```
# %% Third task
def GetData():
    def process(data_from_csv):
        result = data_from_csv.astype(np.int32)
        return result

    with open("data.csv") as tmp_data:
        data = csv.reader(tmp_data, delimiter=',')
        for line in data:
            if line[0] == "Panama":
                res = np.array(line[42:63])
                year = np.array(np.linspace(1998, 2018, 21))
                return year, process(res)

def calculate_G(x, order):
    G = np.zeros((len(x), order))
    for i in range(0, len(x)):
        for j in range(0, order):
            G[i, j] = x[i] ** j
    return G
def calculate_c(g,Y):
    g_transposed = g.transpose()
    G = np.dot(g_transposed, g)
    y = np.dot(g_transposed, np.transpose(Y))
    return np.linalg.solve(G, y)

def Approximate(X, Y, order, depict_dots_n):
    G_interpolation = calculate_G(X,order)
    c = calculate_c(G_interpolation,Y)
    x = np.linspace(X[0], X[-1], depict_dots_n)
    G_depict = calculate_G(x, order)
    y = np.dot(G_depict, c)
    return x, y


def ExecuteThirdTask():
    data = GetData()
    plt.plot(data[0], data[1], 'o', label="Aproksimavimo taškai", color='r')
    for i in [1, 2, 3, 5]:
        x, y = Approximate(data[0], data[1], i, 1000)
        plt.plot(x, y, label=f"{i}-os eilės aproksimavimas")
    plt.legend(loc='best')
    plt.show()
```

4. Išvados

Pirma ir trečia užduotys realizuotos sėkmingai, antra užduotis realizuota, tačiau ne iki galo. Antroje užduotyje susidūriau su problema, kai dedame interpoliavimo taškų y koordinates į masyvą, iki 2017 metų sudeda teisingai, tačiau nuo 2017 meta klaidą kadangi formulėje yra naudojamas y_{i+1} išeina iš ribų, šiai problemai išspresti pasinaudojau python funkcija (try: except:). Kai atvaizdavimo taškų skaičius parenkamas 1000, programa sudeda 953 narius į rezultatų masyvą. Ketvirta užduotis nerealizuota.

Paveikslėlių sąrašas

Šaltiniai

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

https://moodle.ktu.edu/