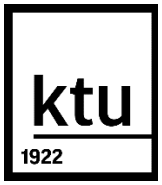


KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

Laboratorinis darbas Nr. 3

Variantas 1

Dėstytojas:
prof. BARAUSKAS Rimantas
Studentė:
Laura Capaitė IFK-0

KAUNAS, 2022

Turinys

1. Interpoliavimas daugianariu

1 užduoties užduotis(1.1 pav.)

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 interpoliuojančios funkcijos apskaičiuojamos naudojant skirtingas abscises ir gautas interpoliuojančių funkcijų išraiškas. Tame pačiame grafike vaizduokite duotąją funkciją, interpoliuojančią funkciją ir netiktį.

1.1 pav. užduotis

Funkcija užduočiai spręsti (1.2 pav.):

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

1.2 pav. funkcijos išraiška

Pirmiausia yra paskaičiuojami Niutono interpoliavimo išraiškos koeficientai (1.3 pav.):

$$a_0 = y_0$$
$$a_1 = \frac{y_1 - y_0}{x_1 - x_0} = f(x_0, x_1)$$
$$a_n = \frac{f(x_1, x_2, x_3) - f(x_0, x_1, x_2)}{x_n - x_0}$$

1.3 pav. koeficientų skaičiavimo formulė

Programos kodas skaičiuojantis koeficientus (1.4 pav.):

```
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])] -
                    range_x[np.max([i + j - (i + 1), 0])]))

    return [_a[0] for _a in a[:-1]]
```

1.4 pav. koeficientų skaičiavimas

Gavus koeficientus, galima juos įstatyti į daugianario interpoliavimo formulę:

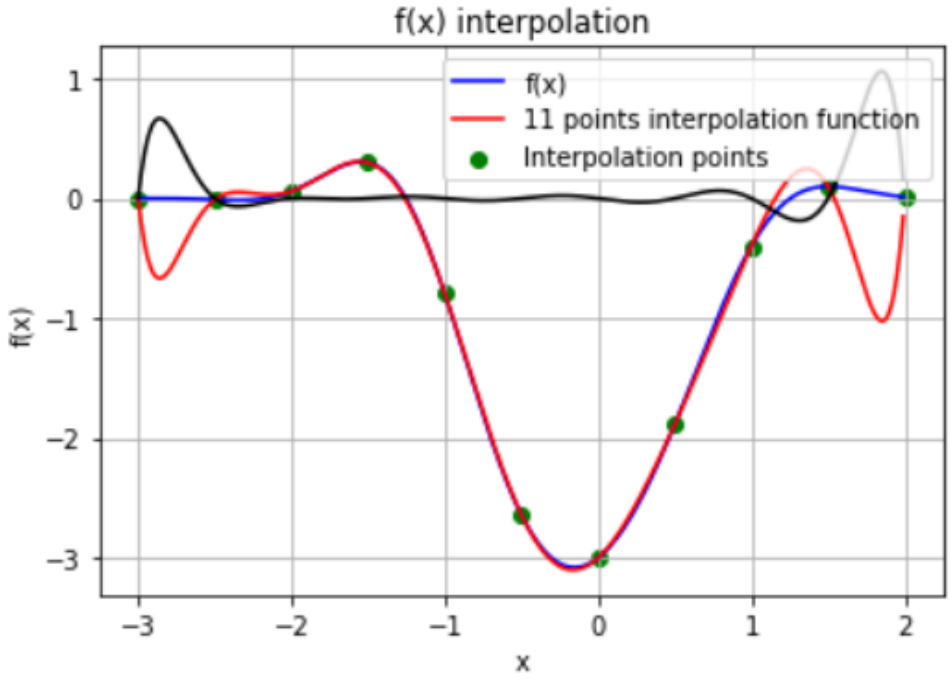
$$f(x)=a_0+a_1*(x-x_0)+a_2*(x-x_0)*(x-x_1)+...+a_n(x-x_0)*(x-x_1)*(x-x_2)...(x-x_n)$$

Programos kodas apskaičiuojantis interpoliavimo funkciją (1.5 pav.):

```
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
```

1.5 pav. interpoliavimo skaičiavimas

Gauti rezultatai pavaizduoti 1.6 pav. Mėlyna spalva rodo duotos funkcijos grafiką. Raudona spalva rodo interpoliuotos funkcijos grafiką. Interpoliavimas buvo vykdomas per 11 taškų.



1.5 pav. f(x) ir interpoliuotos funkcijos grafikai

B dalyje reikia apskaičiuoti taškus naudojantis Čiobyševo abscises (1.6 pav.).

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

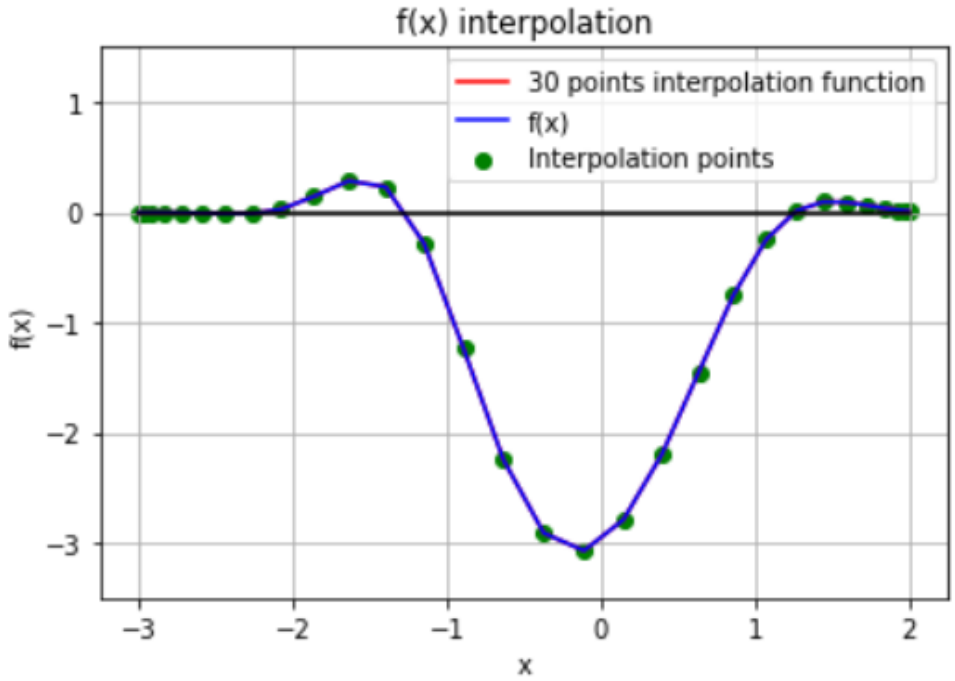
1.6 pav. Čiobyševo abscisės

Programos kodas apskaičiuojantis Čiobyševo abscises (1.7 pav.)

```
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
```

1.7 pav. Čiobyševo abscisių apskaičiavimas

Naudojantis šiais gautais taškais interpoliavimas pavyko kur kas geriau nei naudojant taškus pasiskirčiusius tolygiai (1.8 pav.). Raudona interpoliavimo funkcija gavosi daug daugiau artimesnė duotai funkcijai, kuri yra mėlyna. Čia taip pat buvo naudota 11 interpoliavimo taškų.



1.8 pav. f(x) ir interpoliuotos funkcijos grafikai

2. Interpoliavimas daugianariu ir splainu per duotus taškus

Antros užduoties užduotis (2.1 pav.)

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)).

2.1 pav. užduotis

Gauta valstybė: Argentina
Splainas: Globalus

Pirmiausia duomenys apie šiltnamio dujų emisiją surašomi į masyvus (2.2 pav.). Metai į X masyvą ir emisija į Y masyvą.

```
X = np.array([1998, 1999, 2000, 2001, 2002,
              2003, 2004, 2005, 2006, 2007,
              2008, 2009, 2010, 2011, 2012,
              2013, 2014, 2015, 2016, 2017])
Y = np.array([282380, 288710, 290820, 284140, 280480,
              302570, 318700, 322740, 337250, 349820,
              354270, 333820, 339610, 350250, 355300,
              364480, 363150, 370000, 373330, 377210])
```

2.2 pav. pradiniai duomenys

Kad būtų galima atlikti interpoliavimą, pirmiausia reikia surasti splaino koeficientus. Jie randami pagal formulę (2.3 pav.):

$$f''_{i-1}\frac{d_{i-1}}{6}+f''_{i}\frac{d_{i-1}+d_i}{3}+f''_{i+1}\frac{d_i}{6}=\frac{y_{i+1}-y_i}{d_i}-\frac{y_i-y_{i-1}}{d_{i-1}}$$

2.3 pav. skaičiavimo formulė

Iš šios formulės sudaroma matrica antrų išvestinių reikšmių skaičiavimui (2.4 pav.):

$$\begin{bmatrix} \frac{d_1}{6} & \frac{d_1+d_2}{3} & \frac{d_2}{6} & 0 & \cdots & 0 \\ 0 & \frac{d_2}{6} & \frac{d_2+d_3}{3} & \frac{d_3}{6} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots\cdots & \vdots \\ 0 & 0 & \cdots & \frac{d_{n-2}}{6} & \frac{d_{n-2}+d_{n-1}}{3} & \frac{d_{n-1}}{6} \end{bmatrix} \begin{Bmatrix} f''_1 \\ f''_2 \\ \vdots \\ f''_n \end{Bmatrix} = \begin{Bmatrix} \frac{y_3-y_2}{d_2}-\frac{y_2-y_1}{d_1} \\ \frac{y_4-y_3}{d_3}-\frac{y_3-y_2}{d_2} \\ \vdots \\ \frac{y_n-y_{n-1}}{d_{n-1}}-\frac{y_{n-1}-y_{n-2}}{d_{n-2}} \end{Bmatrix}$$

2.4 pav. antrų išvestinių reikšmių skaičiavimo matricos

Programos kodas, kuris apskaičiuojamos antras išvestines splaino mazguose (2.5pav.)

```
def splaino_koeficientai(X,Y,iopt):
  n=len(X);
  d=X[1:n]-X[0:(n-1)]
  A=np.zeros(shape=(n,n), dtype=float);
  b=np.zeros(shape=(n,1), dtype=float);
  for i in range(0, n-2):
      A[i,i]=d[i]/6;
      A[i,i+1] =(d[i]+d[i+1])/3
      A[i,i+2] = d[i+1]/6;
      b[i]=(Y[i+2]-Y[i+1])/d[i+1]-(Y[i+1]-Y[i])/d[i];
  if iopt == 0:  A[n-1,1]=1;A[n,n]=1;
  DDF=A.dot(b);
  return DDF
```

2.5 pav. antrų išvestinių reikšmių apskaičiavimas

Apskaičiavus antrų išvestinių reikšmes galima skaičiuoti splainus. Splaino apskaičiavimo formulė (2.6 pav.):

$$_{(i)}f(s) = f_i'' \frac{s^2}{2} - f_i'' \frac{s^3}{6d_i} + f_{i+1}'' \frac{s^3}{6d_i} + \frac{y_{i+1} - y_i}{d_i} s - f_i'' \frac{d_i}{3} s - f_{i+1}'' \frac{d_i}{6} s + y_i;$$

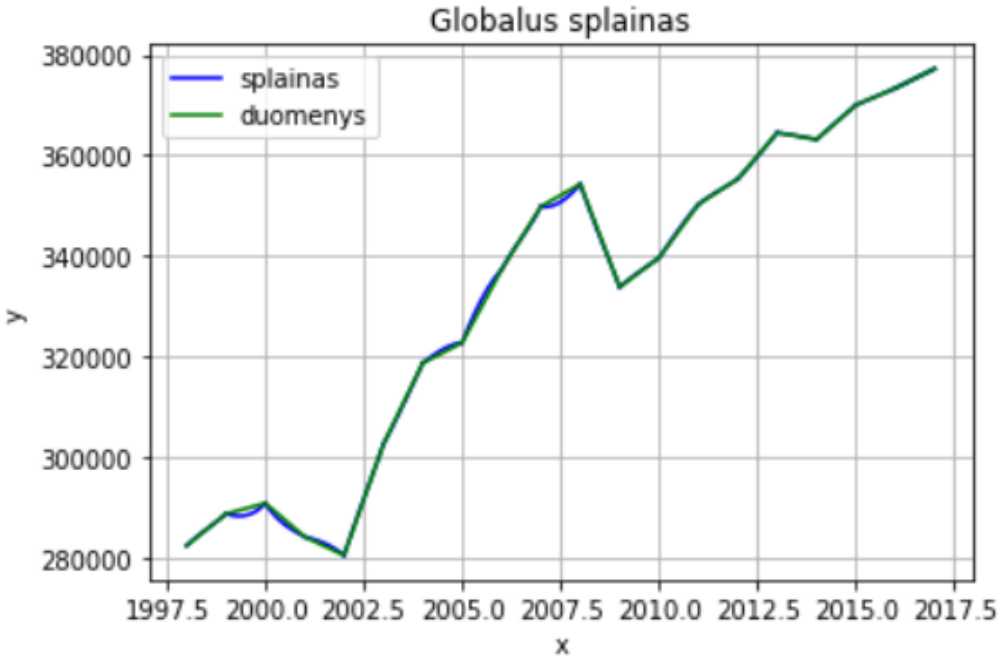
2.6 pav. antrų išvestinių reikšmių apskaičiavimas

Pagal formulę yra užrašomas programos kodas (2.7 pav.):

```
def splainas(X,Y,DDF,nnn):
  d=X[1]-X[0];
  sss = np.arange(X[0], X[1], (X[1]-X[0])/(nnn-1))
  S=DDF[0]/2*(sss-X[0])**2+(DDF[1]-DDF[0])/(6*d)*(sss-X[0])**3+(sss-X[0])*((Y[1]-Y[0])/d-DDF[0]*d/3-DDF[1]*d/6) +Y[0];
  return S, sss
```

2.7 pav. splaino apskaičiavimas

Gauti rezultatai parodyti 2.8 pav. Žalia spalva rodo duomenis iš masyvų, mėlyna spalva rodo gautą splainą. Interpoliavimo taškų skaičius: 20. Vaizdavimo taškų skaičius: 50. Splainas turi tik minimalius nukrypimus nuo tikro grafiko., todėl galima teigti, kad splainas buvo surastas teisingai



2.8 pav. rezultatų grafikas

3. Aproksimavimas

Trečios dalies užduotis (3.1 pav.)

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.

3.1 pav. užduotis

Pradiniai duomenys naudojami tokie patys kaip ir antroje užduotyje (3.2 pav.).
Metai surašyti į X masyvą ir emisija į Y masyvą.

```
X = np.array([1998, 1999, 2000, 2001, 2002,
              2003, 2004, 2005, 2006, 2007,
              2008, 2009, 2010, 2011, 2012,
              2013, 2014, 2015, 2016, 2017])
Y = np.array([282380, 288710, 290820, 284140, 280480,
              302570, 318700, 322740, 337250, 349820,
              354270, 333820, 339610, 350250, 355300,
              364480, 363150, 370000, 373330, 377210])
```

3.2 pav. pradiniai duomenys

Pradedant spręsti uždavinį pirmiausia surandamos G matricos reikšmės (3.3 pav.).
Jos aprašomos nuo nulinės eilės iki m-1 eilės vienanariais.

$$\mathbf{G} = \begin{bmatrix} g_1(x_1) & g_2(x_1) & \cdots & g_m(x_1) \\ g_1(x_2) & g_2(x_2) & \cdots & g_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(x_n) & g_2(x_n) & \cdots & g_m(x_n) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{m-1} \\ 1 & x_2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{m-1} \end{bmatrix}$$

3.3 pav. vienanarių matrica

Parašomas metodas šiai matricai sukurti (3.4 pav.)

```
def function (m, x):
    G = np.zeros(shape=(len(x),m));
    for i in range(m):
        G[:,i]=x**(i);
    return G
```

3.3 pav. vienanarių matricos sukūrimo metodas

C koeficientų apskaičiavimui reikalinga G matrica, G transponiota matrica ir y vektorius (taškai y ašyje) (3.4 pav.).

$$\left(\left(\mathbf{G}^T\right)_{m\times n}\mathbf{G}_{n\times m}\right)_{m\times m}\mathbf{c}_{m\times 1} = \left(\mathbf{G}^T\right)_{m\times n}\mathbf{y}_{n\times 1}$$

3.4 pav. c koeficientų skaičiavimo formulė

Koeficientų c skaičiavimo funkcija (3.5 pav.)

```
c=scipy.linalg.solve((Gg.T.conj().dot(G)),(Gg.T.conj().dot(fff.T.conj())))
print(c)
```

3.5 pav. c koeficientų skaičiavimas

Vėliau sukuriama nauja G matrica, kuri yra padauginama iš c koeficientų ir yra gaunama aproksimuojanti kreivė (3.6 pav.).

```
Gv=function(m,ttt);
fff1=np.matmul(Gv, c);
plt.plot(ttt,fff1,'b');
```

3.6 pav. gaunama aproksimuojanti kreivė

Raudona spalva žymima duota funkcija, mėlyna – aproksimuojanti kreivė

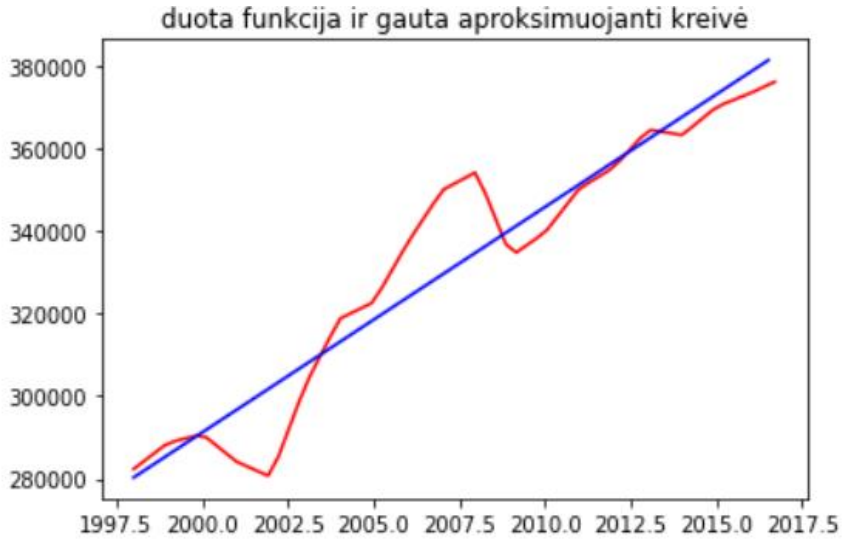
Pirmos eilės aproksimuojanti kreivė (3.7 pav.):

3.7 pav. pirmos eilės kreivė

Gauta funkcija:

$F(x) = 331317.06$

Antros eilės aproksimuojanti kreivė (3.8 pav.):

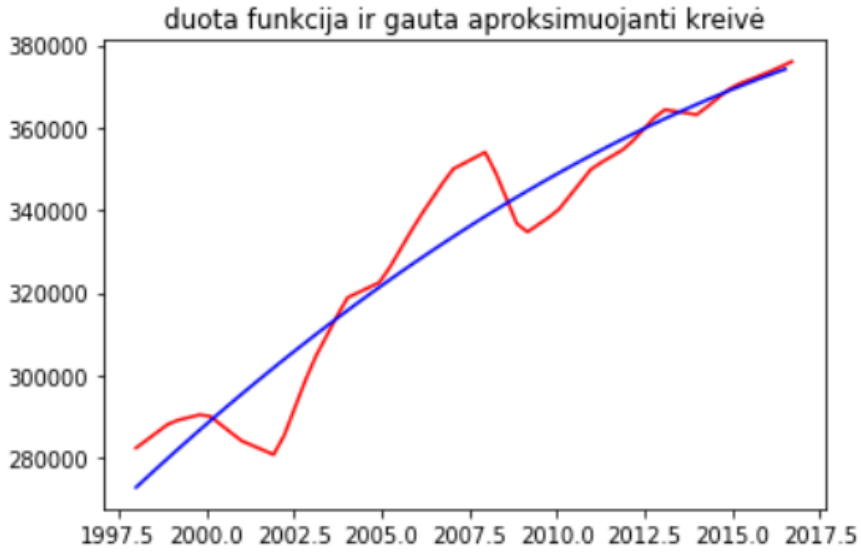


3.8 pav. antros eilės kreivė

Gauta funkcija:

$F(x) = -1.06e+07 + 5.45+03x$

Trečios eilės aproksimuojanti kreivė (3.9 pav.):

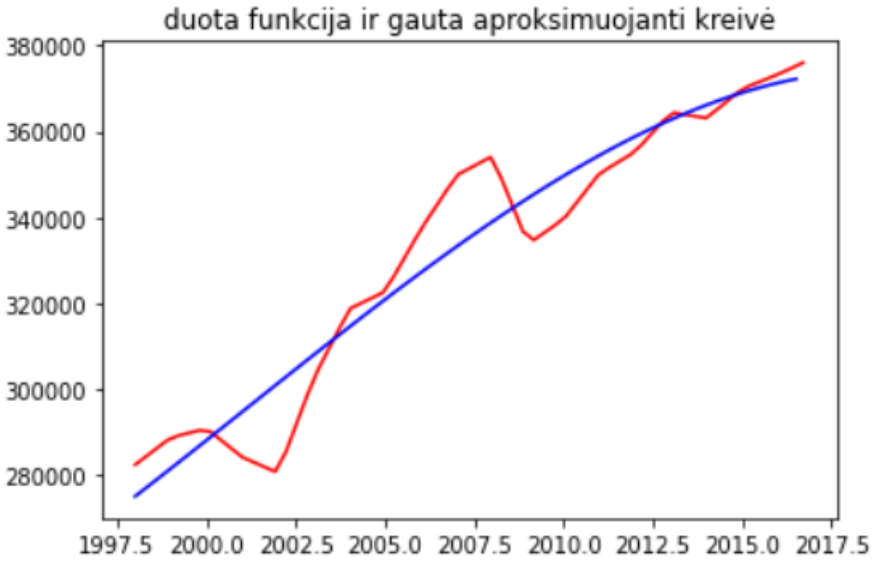


3.9 pav. trečios eilės kreivė

Gauta funkcija:

$F(x) = -5.39e+08 + 5.31e+05x - 1.31e+02x^2$

Penktos eilės aproksimuojanti kreivė (3.10 pav.):



3.10 pav. penktos eilės kreivė

Gauta funkcija:

$F(x) = 3.58e+10 - 4.32e+07x + 1.11e+04 * x^2 + 3.35x^3 - 1.30e-03x^4$

Išvados

Pirmoje užduotyje galima pastebėti, kad naudojant Čiobyševo abscises interpoliavimas pavyko daug geriau nei naudojant tolygiai didėjančias.

Antroje užduotyje naudojant globatų splainą interpoliavimas pavyko gana gerai, gauta interpoliuojanti kreivė yra artima duotai kreivei.

Atliekant trečia užduotį iškilo keletas sunkumu. Pirmos, antros ir trečios eilių aproksimacijos kreivės gautos teisingos. Penktos eilės koeficientai gauti šiek tiek su paklaida (funkcijos buvo lyginamos su rezultatais gautais naudojant matlab).