Kauno Technologijos Universitetas

**Skaitiniai metodai ir algoritmai**

Namų darbas Nr. 2

Parengė: Tautvydas Petkus IFF-1

KAUNAS

2013

Variantas Nr. 3

1. Duota funkcija



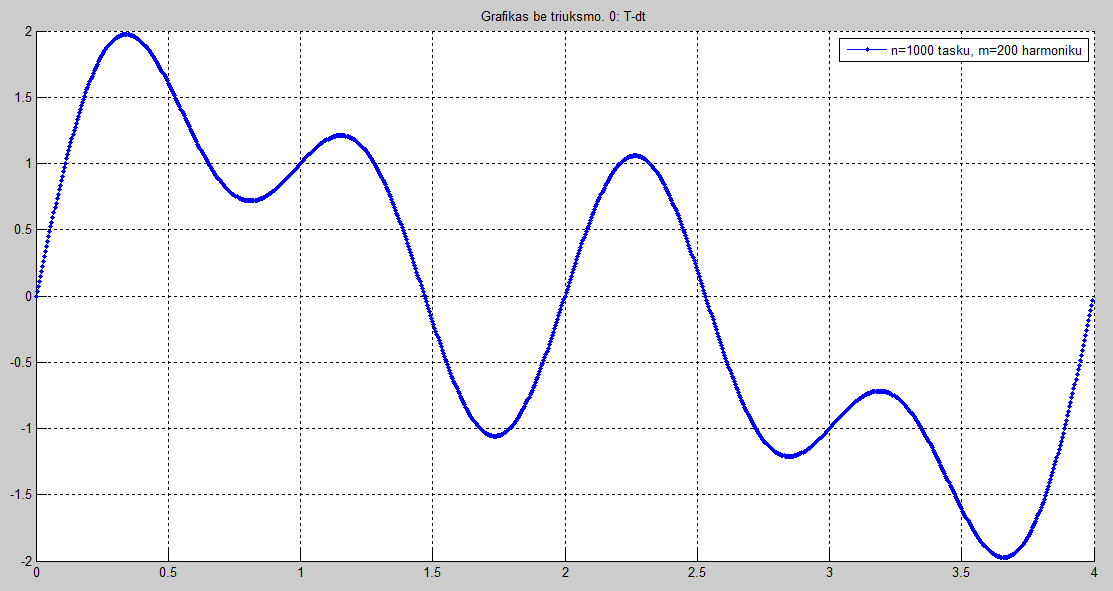


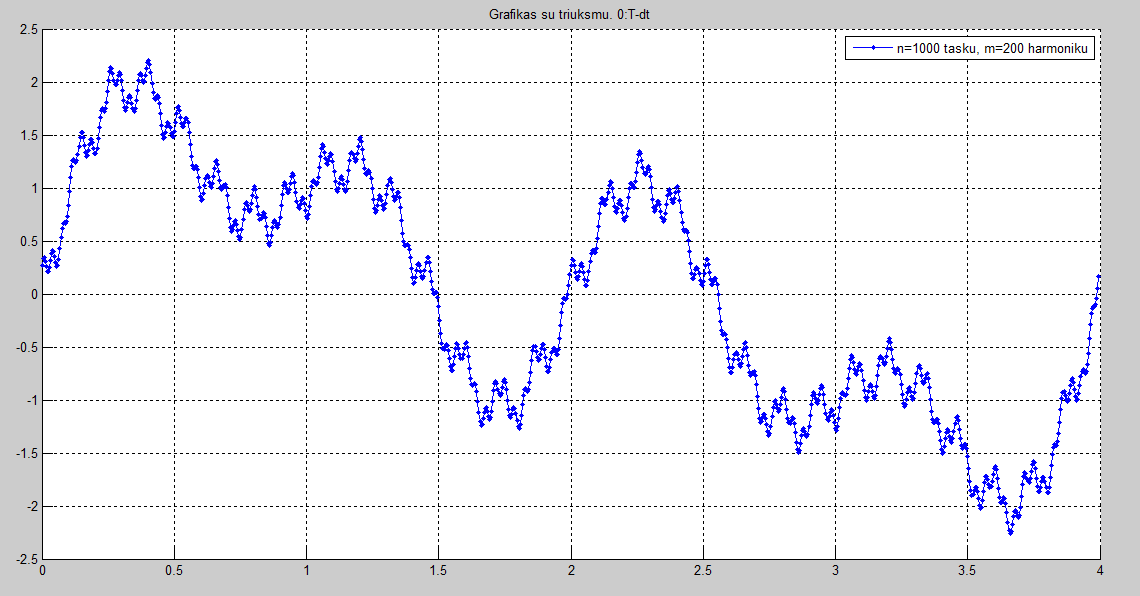
Reikia atlikti funkcijos *F*(*t*)+*R*(*t*) filtravimą ir išskirti funkciją *F*(*t*) dviem būdais:

1. atmetant harmonines dedamąsias pagal amplitudės slenksčio reikšmę;
2. atmetant harmonines dedamąsias pagal jų dažnį.

Žemiau pateikti funkcijos grafikai be triukšmo ir su triukšmu.

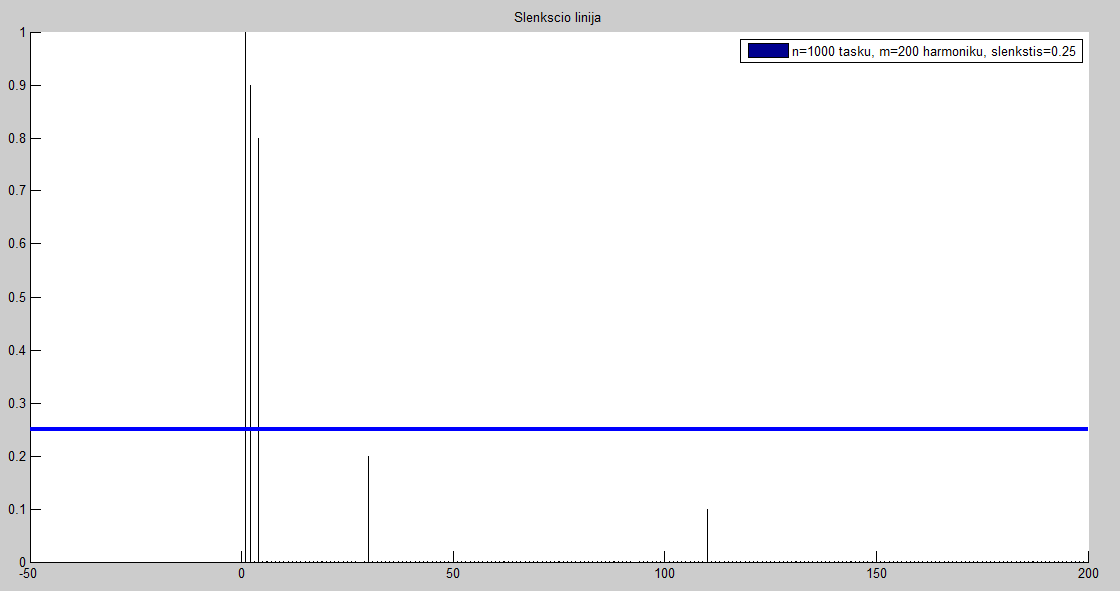
Grafikas be triukšmo:

Grafikas su triukšmais:

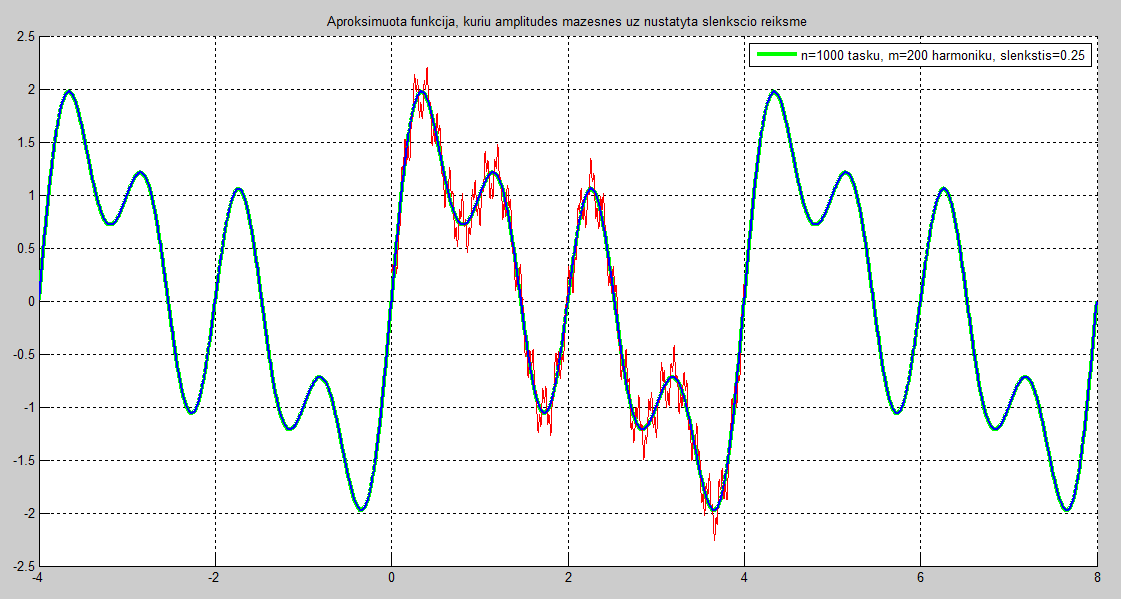


Padarytas filtravimas, kurio metu atmestos harmonikos, kurių amplitudė mažesnė nei 0,25. Pateiktas aproksimuotos f-jos grafikas

Amplitudės slenkstis:

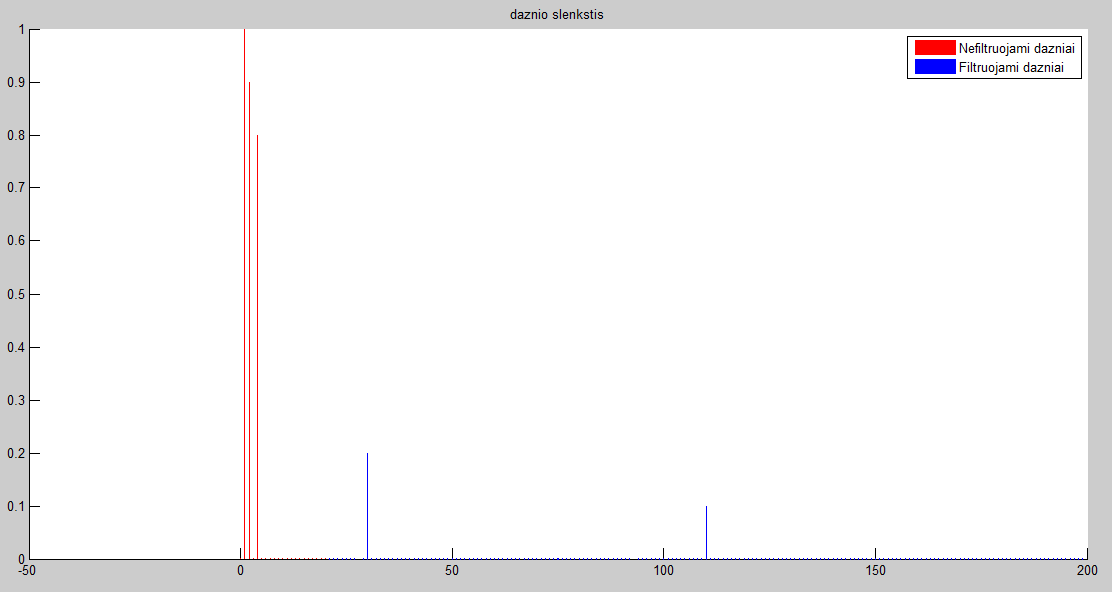


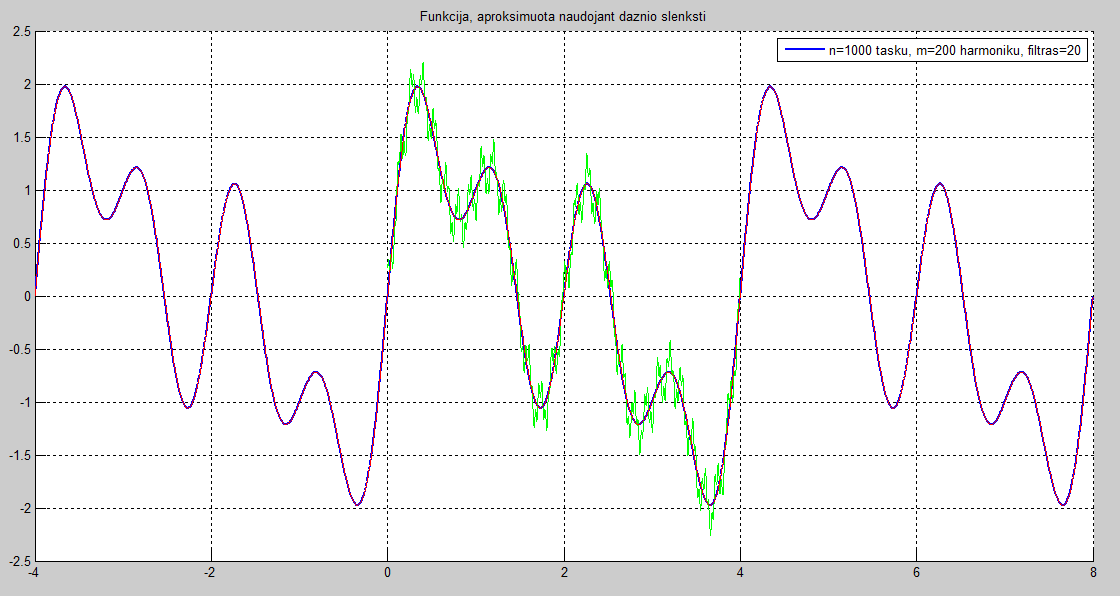
Aproksimuota f-ja:



Padarytas filtravimas, kurio metu atmestos harmonikos, kurių dažnis didesnis nei 20. Pateiktas aproksimuotos f-jos grafikas

Dažnių slenkstis:



Aproksimuota f-ja:  


**Programos kodas**

function L2\_1\_Furje

clc,close all,clear all

n=1000; % tasku skaicius

m=200;

M=2\*m-1; % m - harmoniku skaicius,M -koeficientu skaicius

if M > n, '\*\*\*\*\*\*\*\*\*per didelis harmoniku skaicius';end

T=4; %periodas

slenkstis=0.25 ; % harmoniku amplitudziu slenkstis triuksmu filtravimu

dt=T/n;

N=1000; % vaizdavimo tasku skaicius

dttt=T/N;

t=[0:dt:T-dt];

ttt=[-T:dttt:2\*T];

fff=fnk(T,t); % apskaiciuojame ir pavaizduojame duota tasku seka

fffg=fnk2(T,t);

fff2=fnk2(T,ttt);

figure(1),hold on,grid on, plot(t,fffg,'b.-');

title('Grafikas be triuksmo. 0: T-dt')

legend(sprintf('n=%d tasku, m=%d harmoniku',n,m))

figure(7),hold on,grid on,plot(t,fff,'b.-','MarkerSize',8);

title('Grafikas su triuksmu. 0:T-dt')

legend(sprintf('n=%d tasku, m=%d harmoniku',n,m))

ac0=dot(fff,fC(0,T,t))/n;

for i=1:m-1

ac(i)=dot(fff,fC(i,T,t))\*2/n;

as(i)=dot(fff,fS(i,T,t))\*2/n;

end

figure(2),hold on

bar(0:m-1,[ac0,sqrt(ac.^2+as.^2)],0.01)

xx=axis;

plot([xx(1),xx(2)],slenkstis\*[1 1],'b-','LineWidth',3); % braizo slenkscio linija

title('Slenkscio linija')

legend(sprintf('n=%d tasku, m=%d harmoniku, slenkstis=%g ',n,m,slenkstis))

fffz=ac0\*fC(0,T,ttt)

frequencies=[1:m-1];

for i=frequencies

if sqrt(ac(i)^2+as(i)^2) > slenkstis

fffz=fffz+ac(i)\*fC(i,T,ttt)+as(i)\*fS(i,T,ttt);

end

end

figure(3),hold on,grid on, plot(ttt,fffz,'g-','LineWidth',3);

plot(t,fff,'r-');

plot(ttt,fff2,'b-','LineWidth',2);

title('Aproksimuota funkcija, kuriu amplitudes mazesnes uz nustatyta slenkscio reiksme')

legend(sprintf('n=%d tasku, m=%d harmoniku, slenkstis=%g ',n,m,slenkstis))

%--------------------------------------------------------------------------

Y = fft(fnk(T,t))

Pyy = sqrt(Y.\* conj(Y));

filt = 20; %daznis filtravimui

ind=Pyy<filt;

Y(ind)=0;

yt=ifft(Y);

%--------------------------------------------------------------------------

for i=1:filt

ac\_daz(i)=dot(fff,fC(i,T,t))\*2/n;

as\_daz(i)=dot(fff,fS(i,T,t))\*2/n;

end

for i=1:m-1-filt-1

ac\_daz1(i)=dot(fff,fC(i+1+filt,T,t))\*2/n;

as\_daz1(i)=dot(fff,fS(i+1+filt,T,t))\*2/n;

end

figure(5),title('daznio slenkstis'),hold on

bar(0:filt,[ac0,sqrt(ac\_daz.^2+as\_daz.^2)],'FaceColor', 'r','EdgeColor','r', 'BarWidth', 0.001)

bar(filt+1:m-1,[ac0,sqrt(ac\_daz1.^2+as\_daz1.^2)],'FaceColor', 'b','EdgeColor','b', 'BarWidth', 0.001)

legend('Nefiltruojami dazniai', 'Filtruojami dazniai');

%-----------------------------------------------------------------

fffz1=ac0\*fC(0,T,ttt)

frequencies=[1:filt];

for i=frequencies

fffz1=fffz1+ac(i)\*fC(i,T,ttt)+as(i)\*fS(i,T,ttt);

end

figure(6),title('Funkcija, aproksimuota naudojant daznio slenksti'),hold on,grid on, plot(ttt,fffz1,'b','LineWidth',2);plot(t,fff,'g-','LineWidth',1);plot(ttt,fff2,'r-')

legend(sprintf('n=%d tasku, m=%d harmoniku, filtras=%g ',n,m,filt))

return

end

function c=fC(i,T,t),

if i==0,

c=1\*cos(0\*t);

else,

c=cos(2\*pi\*i/T\*t);

end,

return,

end

function s=fS(i,T,t),

s=sin(2\*pi\*i/T\*t);

return,

end

function rez=fnk(T,t), %su triuksmais

rez=sin(2\*pi\*t/T) + 0.9\*sin(2\*pi\*2\*t/T) + 0.8\*sin(2\*pi\*4\*t/T) + 0.1\*sin(2\*pi\*110\*t/T + pi/4) + 0.2\*cos(2\*pi\*30\*t/T);

return,

end

function rez=fnk2(T,t), %be triuksmu

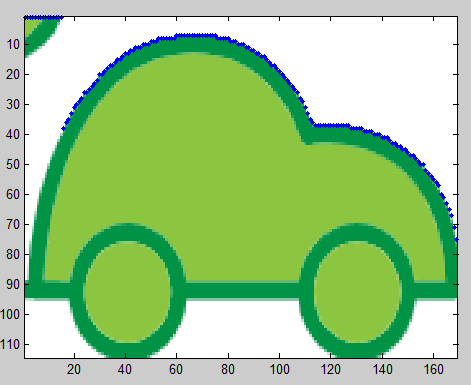
rez=sin(2\*pi\*t/T) + 0.9\*sin(2\*pi\*2\*t/T) + 0.8\*sin(2\*pi\*4\*t/T);

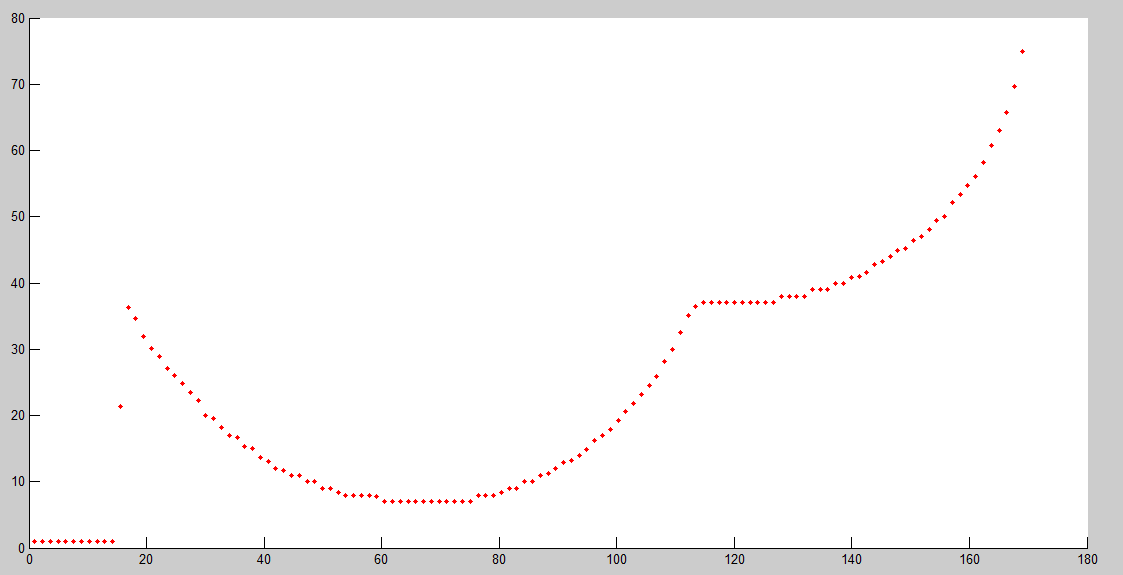
return,

end

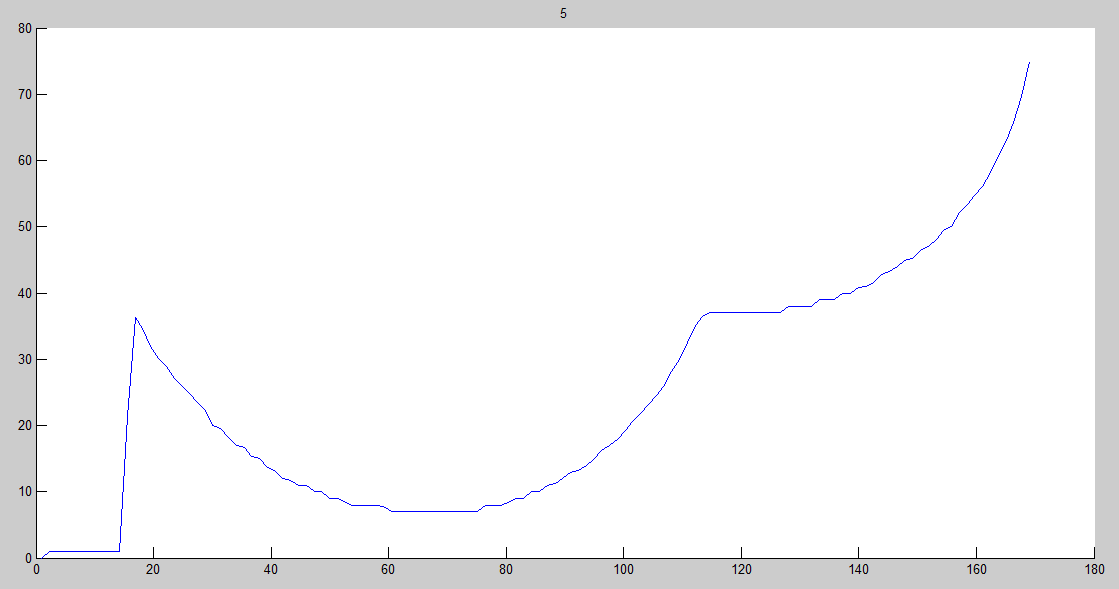
**Haro bangelių aproksimacija**

Tikslas: pagal duoto paveikslėlio viršutinius kontūrus rasti koordinates X ir Y bei atlikti funkcijos aproksimaciją naudojant Hario bangeles, suskaičiuoti bangų koficientus

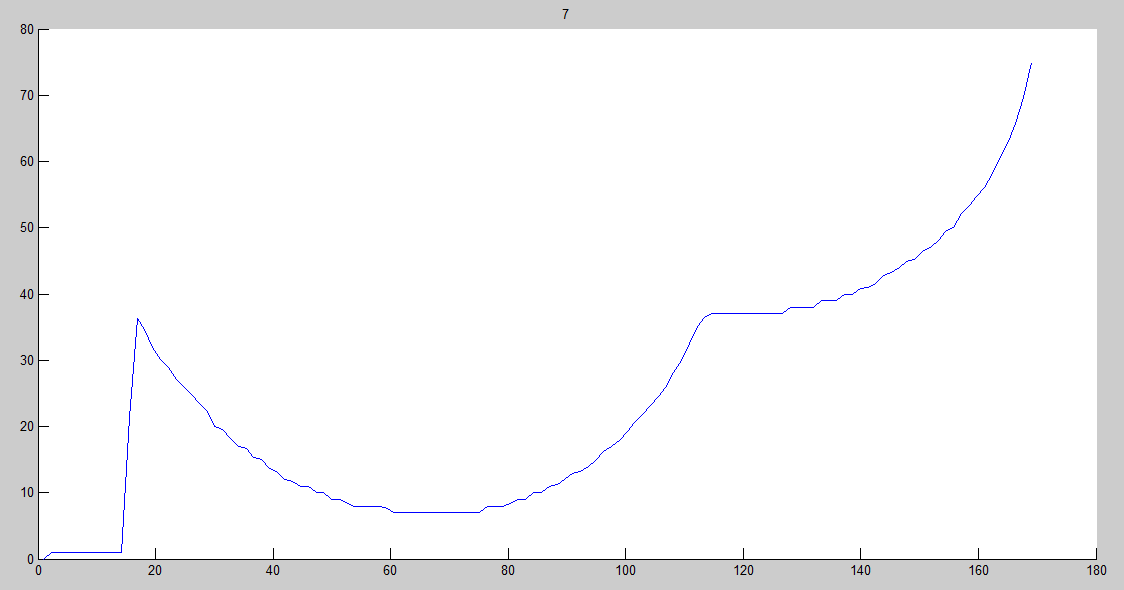
Viršutinio kontūro grafikas:  


Suinterpoliuotas viršūtinis konturas. Funkcija:  


Lygyje 5 suglodinta f-ja:



Lygyje 7 suglodinta f-ja:



Koficientai, kai detalumo lygis – 7:

1 : 0 0 0 0 0 -214.134 17.2795 18.9331 19.0157 11.9055 13.8898 4.46457 13.1457 13.8898 13.8898 10.5 7.02756 10.5 9.92126 6.20079 0 0 8.35039 0 0 0 0 0 -9.67323 0 -6.61417 -10.3346 -10.3346 -7.27559 -3.38976 -9.67323 -7.93701 -13.8898 -13.8898 -13.8898 -24.3898 -27.7795 -13.8898 0 0 0 0 0 -0.0826772 0 0 -10.5 -9.4252 -5.70472 -3.38976 -8.76378 -13.8898 -11.8228 -5.7874 -13.8898 -13.8898 -26.7874 -27.7795 -55.5591

2 : 0 0 -151.416 64.8924 37.0646 45.9508 24.9631 26.7169 12.3939 12.2185 0 5.90462 0 0 -6.84001 -12.2185 -17.2462 -19.9939 -28.6462 -39.2862 -63.5477 -18.1231 0 0 -0.0584616 -7.42462 -12.2185 -20.6369 -25.8985 -45.0154 -59.8647 -118.326

3 : 0 -570.059 112.276 71.1024 37.8248 16.8248 0 -27.8622 -57.0059 -103.594 -152.498 0 -26.2913 -60.1063 -95.5335 -230.008

4 : -554.508 264.773 71.7908 -42.56 -226.392 -125.955 -108.914 -413.645

5 : -387.012 45.8445 -782.787 -649.822

6 : 396.808 -1344.34

7 : -2123.06

Programos kodas:

function L2\_2\_Haro

M = imread('picture.png');

image(M)

figure(1); hold on;

X=[]; Y=[];

X=1:size(M, 2);

%k=0;

for j=1:size(M, 2)

for i=1:size(M, 1)

if sum(M(i, j, :)) < 700

Y(j)=i;

break

else

Y(j)=i;

end

end

end

fid=fopen('SX.txt', 'w');

fprintf(fid, '%g\n', X);

fclose(fid);

fid=fopen('SX.txt', 'r');

SX = fscanf(fid, '%g');

fclose(fid);

SY = Y;

n=7; nnn=2^n; a=min(SX); b=max(SX);

t=a:(b-a)/(nnn-1):b; ts=interp1(SX,SY,t);

SX=t; SY=ts;

ts=interp1(SX,SY,t); SX=t;SY=ts;

plot(SX, SY, 'r.');

m\_all=[5 7]; current\_figure=2;

for m=m\_all;

smooth=(b-a)\*SY\*2^(-n/2);

for i=1:m

smooth1=(smooth(1:2:end)+smooth(2:2:end))/sqrt(2);

details{i}=(smooth(1:2:end)-smooth(2:2:end))/sqrt(2);

fprintf(1,'\n |||series %d : ',i);fprintf('%g ', details{i});

smooth=smooth1;

end

h=zeros(1,nnn);

for k=0:2^(n-m)-1

h=h+smooth(k+1)\*Haro\_mastelis(SX,n-m,k,a,b);

end

for i=0:m-1

h1=zeros(1,nnn);

for k=0:2^(n-m+i)-1

h1=h1+details{m-i}(k+1)\*Haro\_bangele(SX,n-m+i,k,a,b);

end

h=h+h1;

if i+1==m

figure(current\_figure);

current\_figure = current\_figure + 1;

hold on;

plot(SX,h);

title(sprintf('%d' ,i+1));

end

end

end

end

function h=Haro\_bangele(x, j, k, a, b)

eps=1e-9;

xtld=(x-a) ./ (b-a);

xx=2^j\*xtld-k;

h=2^(j/2)\*(sign(xx-eps)-2\*sign(xx-0.5)+sign(xx-1-eps))/(2\*(b-a));

return

end

function h=Haro\_mastelis(x, j, k, a, b)

eps=1e-9;

xtld=(x-a) ./ (b-a);

xx=2^j\*xtld-k;

h=2^(j/2)\*(sign(xx-eps)-sign(xx-1-eps))/(2\*(b-a));

return

end

**Interpoliavimas**

Duotos interpoliavimo taškų koordinatės ir dvi bazinės funkcijos:





Reikia apskaičiuoti interpoliarines funkcijas naudojant dvi bazines funkcijas. Pavaizduoti grafiškai

**Vienanarių**

Baz. Reikšmių matrica interpoliavimo mazguose:

0.0001 -0.0002 0.0004 -0.0008 0.0016 -0.0032 0.0064 -0.0128 0.0256 -0.0512 0.1024

0.0001 -0.0001 0.0002 -0.0003 0.0005 -0.0008 0.0011 -0.0017 0.0026 -0.0038 0.0058

0.0001 -0.0001 0.0001 -0.0001 0.0001 -0.0001 0.0001 -0.0001 0.0001 -0.0001 0.0001

0.0001 -0.0001 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000

0.0001 0 0 0 0 0 0 0 0 0 0

0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001

0.0001 0.0001 0.0002 0.0003 0.0005 0.0008 0.0011 0.0017 0.0026 0.0038 0.0058

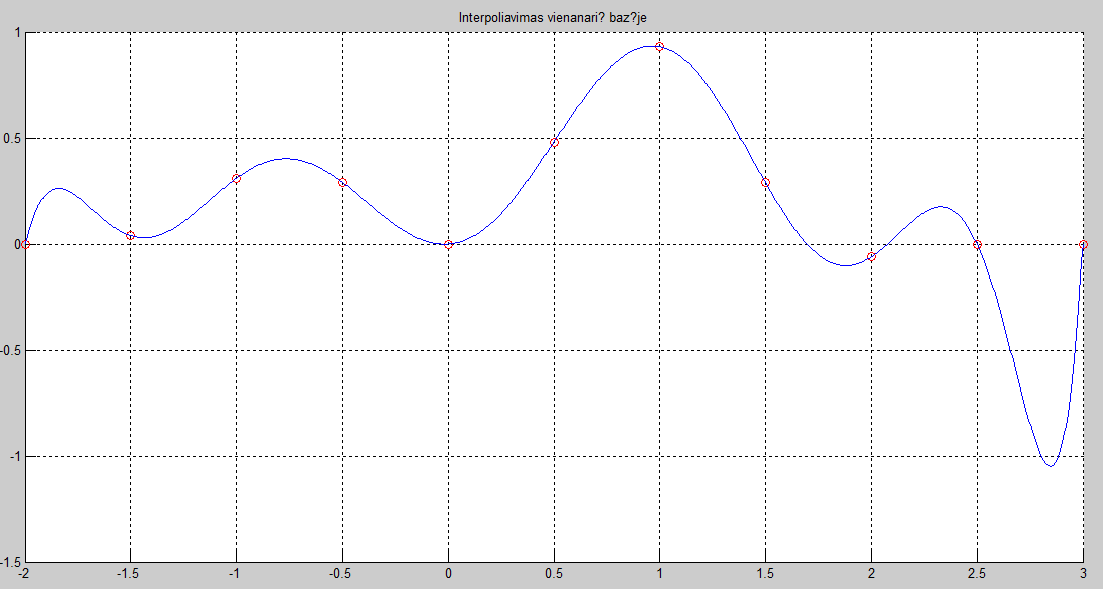
0.0001 0.0002 0.0004 0.0008 0.0016 0.0032 0.0064 0.0128 0.0256 0.0512 0.1024

0.0001 0.0003 0.0006 0.0016 0.0039 0.0098 0.0244 0.0610 0.1526 0.3815 0.9537

0.0001 0.0003 0.0009 0.0027 0.0081 0.0243 0.0729 0.2187 0.6561 1.9683 5.9049

Vienanariu interpoliacines f-jos koeficientai:

-0.0000 0.0726 1.9949 0.5630 -1.9922 -0.3904 0.7164 0.0661 -0.1035 -0.0013 0.0044



function L2\_3\_Vienanariai

clc, close all

fprintf(1, 'Interpoliavimo mazgai:');

X = [-2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3]

Y = [0 0.04 0.31 0.29 0 0.48 0.93 0.29 -0.06 0 0]

% interpoliavimo laipsnis

n = size(X, 2);

nn=1000; a=min(X); b=max(X);

% sudarome baziniu f-ciju matrica B

for i = 1:n

B(:,i)=X.^(i-1)

end

% randame f-cijos koeficientus

A= B\Y';

disp('Vienanariu interpoliacines f-jos koeficientai:');

disp(A');

figure(1), grid on, hold on;

x=[a:(b-a)/(nn-1):b];

plot(X, Y, 'ro', x, foo(x,A));

title 'Interpoliavimas vienanariu bazeje'

end

function y = foo(x, A)

y = 0;

for i = 1:size(A, 1)

y = y + A(i).\* x.^(i-1);

end

return

end

**Čiobyševo**

Baz. Funkcijų reikšmių matrica iterpoliavimo mazguose:

0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0001 -0.0005 0.0019 -0.0070 0.0262

0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0001 -0.0003 0.0008

0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000

0.0000 -0.0000 -0.0000 0.0000 -0.0000 -0.0000 0.0000 -0.0000 -0.0000 0.0000 -0.0000

0.0000 0 -0.0000 0 0.0000 0 -0.0000 0 0.0000 0 -0.0000

0.0000 0.0000 -0.0000 -0.0000 -0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0003 0.0008

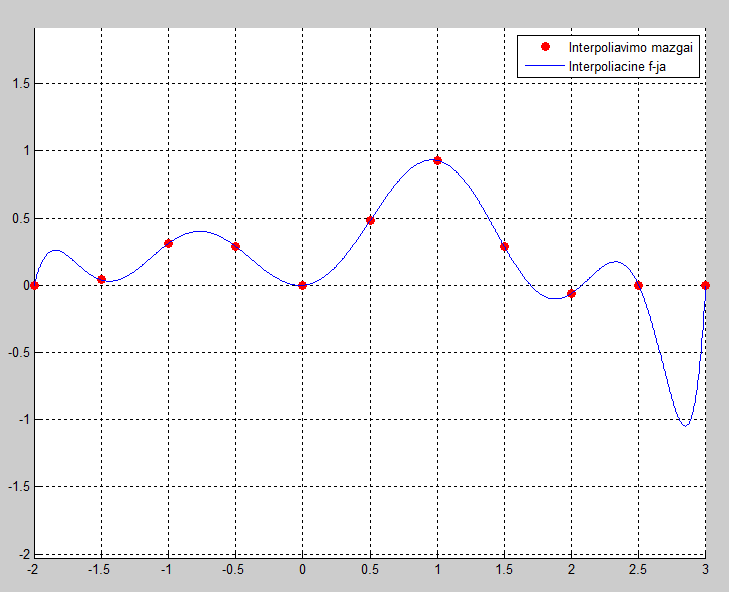
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0005 0.0019 0.0070 0.0262

0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0006 0.0029 0.0139 0.0665 0.3188

0.0000 0.0000 0.0000 0.0000 0.0001 0.0003 0.0020 0.0114 0.0666 0.3881 2.2620

Ciobysevo interpoliacines israiskos koeficientai:

0.0000 -0.0000 -0.0007 0.0010 0.0163 -0.0173 0.1363 0.0400 0.2937 0.2864 0.4470



function L2\_4\_Cebysevas

clc, clear all, close all

fprintf(1, 'Interpoliavimo mazgai:');

X=[-2.0 -1.5 -1.0 -0.5 0 0.5 1.0 1.5 2.0 2.5 3.0]

Y=[0 0.04 0.31 0.29 0 0.48 0.93 0.29 -0.06 0 0]

n = length(X);

T = zeros(n, n);

for i=1:n

T(i,1) = 1;

T(i,2) = X(i);

for j=3:n

T(i,j) = 2 \* X(i) \* T(i,j-1) - T(i,j-2);

end

end

fprintf(1, 'Baziniu funkciju reiksmes interpoliavimo mazguose:');

T

fprintf(1, 'Ciobysevo interpoliacines israiskos koeficientai:');

a = flipud(T\Y')

figure(1), hold on, grid on, axis equal

plot(X,Y,'ro','MarkerFaceColor','r')

XC=min(X):(max(X)-min(X))/(n\*91):max(X);

YC = klensou(a,XC);

plot(XC,YC,'b-');

legend('Interpoliavimo mazgai','Interpoliacine f-ja');

return

end

function px=klensou(a,x)

n=numel(a);

bk2=0; bk1=0;

for k=1:n

bk=a(k)+2\*x.\*bk1-bk2;

bk2=bk1; bk1=bk;

end;

px=bk-x.\*bk2;

return

end