# Niutono bazinės funkcijos

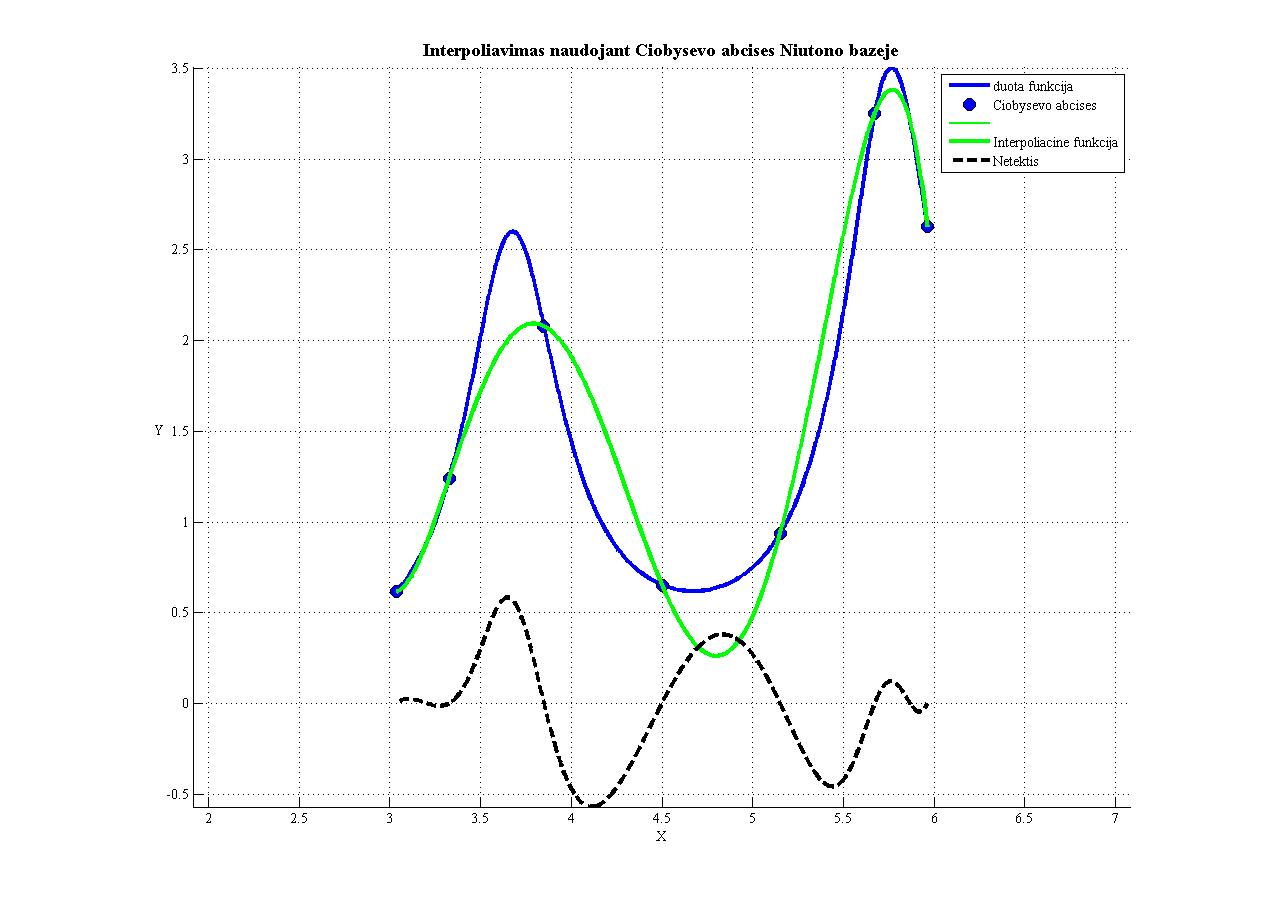
**Interpoliavimo mazgai:**

X= 5.96 5.67 5.15 4.50 3.85 3.33 3.04

Y= 2.63 3.25 0.94 0.65 2.08 1.24 0.61

**Niutono interpoliacinės išraiškos koeficientai:**

2.6278 -2.1545 -8.1242 -7.8906 -4.0970 -1.2124 -0.0122



**Programos kodas:**

function NiutonasA

clc,clear all

close all

%============================================

xmin= 3; %Intervalo pradzia

xmax= 6; %Intervalo pabaiga

n=7; %Interpoliavimo taskus skaicius

fprintf(1, 'Interpoliavimo tasku skaicius: %g', n);

%============================================================

k=[0:n-1];

X=(xmax+xmin)/2+(xmax-xmin)/2\*cos((2\*k+1)\*pi/(2\*n)); %Ciobysevo abscises

fprintf(1, '\nCiobysovo abscises: \n');

fprintf(1, ' %g ', X);

Y=fnk(X); %Ciobysevo ordinates

fprintf(1, '\nCiobysovo ordinates: \n');

fprintf(1, ' %g ', Y);

%===============================================================

x=min(X):(max(X)-min(X))/1000:max(X); %x asies reiksmes brezimui

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

plot(x,fnk(x), 'b-', 'LineWidth', 3) %Pradine funkcija

%============================================================

n=length(X);set(gca,'Fontname','Times New Roman Baltic');

fprintf('\n');

fprintf('\*\*\*Interpoliavimas naudojant Ciobysevo abcises Niutono bazėje\*\*\*\n\n')

fprintf('Interpoliavimo mazgai:\n')

fprintf('\nX= ')

for i=1:n

fprintf('\t%4.2f',X(i))

end

fprintf('\nY= ')

for i=1:n

fprintf('\t%4.2f',Y(i))

end

fprintf('\n')

xx=zeros(n,n);

xx(:,1)=1;

for j=2:n

for i=j:n

san=1;

for k=1:j-1

san=san\*(X(i)-X(k));

xx(i,j)=san;

end

end

end

xx;

fprintf('\nBazinių funkcijų reikšmės interpoliavimo mazguose:\n\n')

for i=1:n

for j=1:n

fprintf('\t%9.4f',xx(i,j));

end

fprintf('\n')

end

A=inv(xx)\*Y';

fprintf('\nNiutono interpoliacinės išraiškos koeficientai:\n\n');

for i=1:n

fprintf('\t%9.4f',A(i))

end

fprintf('\n')

%=================================================================

title('Interpoliuota pagal Ciobysevo abcises')

x=min(X):(max(X)-min(X))/1000:max(X);

f=A(1);

for i=2:n

sand=1;

for k=1:i-1

sand=sand.\*(x-X(k));

end

f=f+A(i).\*sand;

end

g=cos(2\*x)/(sin(x)+1.5)-(x/5);

%===============================================================

%===============================================================

plot(X,Y,'o','MarkerEdgeColor','k','MarkerFaceColor','b','MarkerSize',10);

hold on; grid on; plot(x,f,'g','LineWidth',2);

set(gca,'Fontname','Times New Roman Baltic');xlabel('X');ylabel('Y');

set(get(gca,'YLabel'),'Rotation',0.0);

title('Interpoliavimas naudojant Ciobysevo abcises Niutono bazeje','FontWeight','Bold','Fontsize',14);

legend('Pradine funkcija','Interpoliavimo mazgai','Interpoliacine funkcija','Location','NorthWest')

%====================================================

plot(x, f, 'g-', 'LineWidth', 3) %Braizoma funkcija interpoliuota pagal tolygiai paskirstytus taskus

plot(x, fnk(x)-f, 'k--', 'LineWidth', 3) %Braizoma liekana

legend({'duota funkcija','Ciobysevo abcises','','Interpoliacine funkcija','Netektis', ''})

end

%======================================================

function f=fnk(x)

f=log(x)./(sin(3.\*x)+1.5);

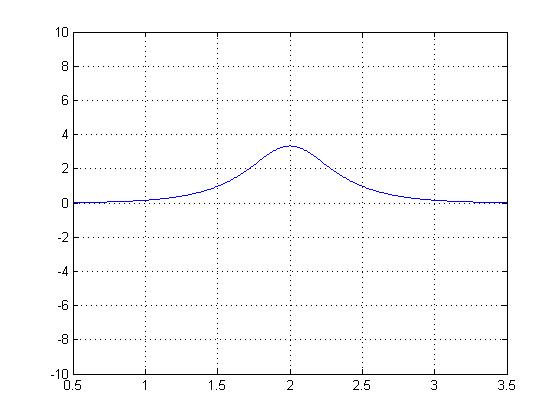
return

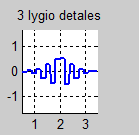
end

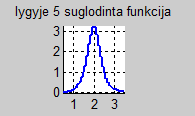
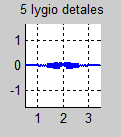
%=======================================================

# Haaro bangelės

**Funkcija:**







**Koeficientai:**

details 1 : -0.00367006 -0.00367006 -0.00367006 -0.00527107 -0.00527107 -0.00527107 -0.0170406 -0.0170406 -0.0170406 -0.0356398 -0.0356398 -0.0495968 -0.0723433 -0.0715027 -0.0606973 -0.0248741 0.0179217 0.069613 0.0698343 0.070354 0.0459299 0.0383067 0.0383067 0.014323 0.014323 0.014323 0.00606634 0.00606634 0.00606634 0.000958377 0.000958377 0.000958377

details 2 : -0.0103805 -0.0121717 -0.0149089 -0.048198 -0.0751387 -0.110674 -0.204023 -0.146347 0.148289 0.198624 0.113738 0.0791458 0.0405116 0.0171582 0.0101837 0.0027107

details 3 : -0.0306271 -0.0969111 -0.273948 -0.501344 0.523978 0.289616 0.0813978 0.0129512

details 4 : -0.147081 -1.23998 1.2511 0.128373

details 5 : -1.66241 1.63026

details 6 : 0.000558851

smooth 6 : 2.70462

**Programos kodas:**

% Raimundas Stankevičius IFK-1

% Haro bangeliu aproksimacija

%

function main

clc;close all;clear all;

spalvos={'r-','g-','m-','c-','k-','y-','r.','g.','m.','c.','k.','y.'};

x = 0.5:0.01:3.5;

y = (1./(0.3 + 2.\*(x-2).^2)).\*exp(-(x-2).^2);

plot(x,y)

grid on % Turn on grid lines for this plot

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Ivedami taskai, spaudziant pele ant grafiko

xmin=0.5;ymin=-10;xmax=3.5;ymax=10; % koordinaciu sistemos ribos

%subplot(2,2,1);

figure(1),hold on, axis([xmin,xmax,ymin,ymax]);grid on

m=6; % aproksimuojanciu funkciju skaicius bazeje;

%Pele ivedami taskai. Baigiama, kai taskas parenkamas uz koord. sistemos ribu

% 15 tasku +- ivedineti is grafiko 15 geriau nevirsyti

X=[];Y=[];

while 1

[X(end+1,1),Y(end+1,1)]=ginput(1); % ,1 rasome, kad gautume vektorius-stulpelius

if X(end) < xmin || X(end) > xmax || Y(end) < ymin || Y(end) > ymax,

X(end)=[];Y(end)=[]; break;

end

plot(X(end),Y(end),'ko');

end

cla, plot(X,Y,'ko');

fileX = fopen('carx2.txt','w');

fileY = fopen('cary2.txt','w');

fprintf(fileX,'%3.6f \n',X);

fprintf(fileY,'%6.2f \n',Y);

disp('Taškų skaičius: ');

%n=length(X) % tasku skaicius

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Is failu ivedami duomenys:

n=6

disp('Aproksimuojančių funkcijų skaičius bazėje 2^n: ');

nnn=2^n

fclose all; fhx=fopen('carx2.txt','r'); fhy=fopen('cary2.txt','r');

%subplot(2,2,1);

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

SX=fscanf(fhx,'%g '); SY=fscanf(fhy,'%g '); fclose all;

plot(SX,SY);

a=min(SX),b=max(SX),t=[a:(b-a)/(nnn-1):b];

ts=interp1(SX,SY,t);

clear SX SY, SX=t;SY=ts;plot(SX,SY,'r.');

title(sprintf('duota funkcija, tasku skaicius 2^%2d',n));

xmin=min(SX);xmax=max(SX);

ymin=min(SY);ymax=max(SY);

% Aproksimavimas Haro bangelemis:

disp('Detalumo lygių skaičius: ');

m=6 % detalumo lygiu skaicius

smooth=(b-a)\*SY\*2^(-n/2); % auksciausio detalumo suglodinimas (pagal duota funkcija)

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 details %d : ',i);fprintf('%g ', details{i});

smooth=smooth1;

end

fprintf(1,'\n smooth %d : ',i);fprintf('%g ', smooth);fprintf('\n');

% Funkcijos rekonstrukcija:

h=zeros(1,nnn); for k=0:2^(n-m)-1, h=h+smooth(k+1)\*Haar\_scaling(SX,n-m,k,a,b); end % suglodinta funkcija

leg={sprintf('suglodinta funkcija, detalumo lygmuo %d',n-m)};

figure(2);subplot(m+1,1,1),axis equal,axis([xmin xmax ymin ymax]); hold on,grid on, plot(SX,h,'Linewidth',2);title(sprintf('lygyje %d suglodinta funkcija',0));

for i=0:m-1 %detalumo didinimo ciklas

% apskaiciuojamos funkcijos detales:

h1=zeros(1,nnn); for k=0:2^(n-m+i)-1, h1=h1+details{m-i}(k+1)\*Haar\_wavelet(SX,n-m+i,k,a,b); end

figure(3),subplot(m,1,i+1), axis equal,hold on,grid on

yshift=(ymin+ymax)/2;axis([xmin xmax ymin-yshift ymax-yshift]), plot(SX,h1,'b-','Linewidth',2);title(sprintf('%d lygio detales',i));

leg={leg{1:end},sprintf('lygmens %d detales',n-m+i)};

h=h+h1; % detales pridedamos prie ankstesnio suglodinto vaizdo

figure(2);subplot(m+1,1,i+2),axis equal,axis([xmin xmax ymin ymax]), hold on,grid on, plot(SX,h,'Linewidth',2);title(sprintf('lygyje %d suglodinta funkcija' ,i+1));

end

return

end

function h=Haar\_scaling(x,j,k,a,b) % \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

eps=1e-9;

xtld=(x-a)/(b-a); % (a,b) intervale duota kintamojo reiksme perskaiciuojama i "standartini"

% intervala (0,1), kuriame uzrasyta bangeles formule

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

return

end

function h=Haar\_wavelet(x,j,k,a,b) % \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

eps=1e-9;

xtld=(x-a)/(b-a); % (a,b) intervale duota kintamojo reiksme perskaiciuojama i "standartini"

% intervala (0,1), kuriame uzrasyta bangeles formule

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