Kauno Technologijos Universitetas

**Skaitiniai metodai ir algoritmai**

Namų darbas Nr. 3

Parengė: Kęstutis Česnavičius IFK-0

KAUNAS

2012

Variantas Nr. 10

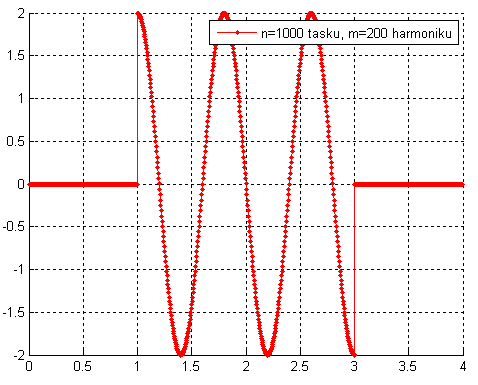
a) Duota funkcija ) ir triukšmas

.

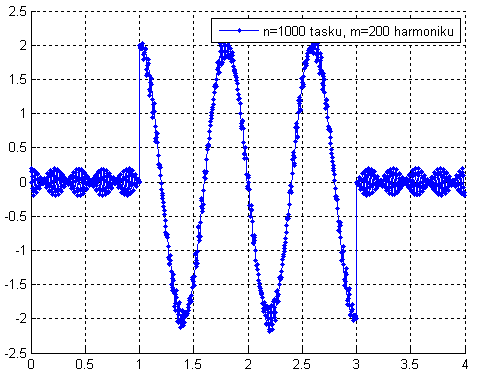
Reikia atlikti funkcijos *F*(*t*)+*R*(*t*) filtravimą ir išskirti funkciją *F*(*t*) dviem būdais:

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

Žemiau pateikti funkcijos grafikai be triukšmo ir su triukšmu.

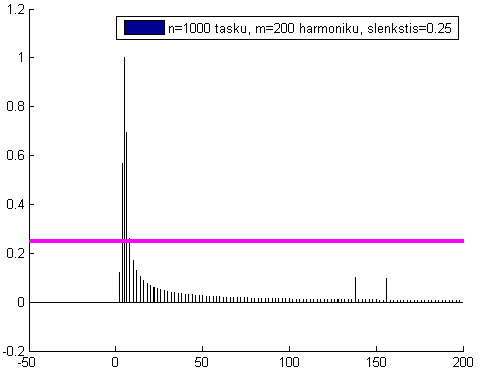


Pav. Duotos funkcijos grafikas

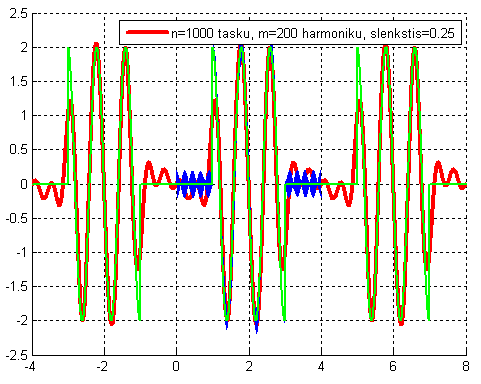


Pav. Funkcijos grafikas su triukšmais

Atliktas filtravimas atmetant harmonikas, kurių amplitudė mažesnė už nustatytą

slenksčio reikšmę 0, 25. Pateikti harmonikų ir aproksimuotos funkcijos grafikai. 

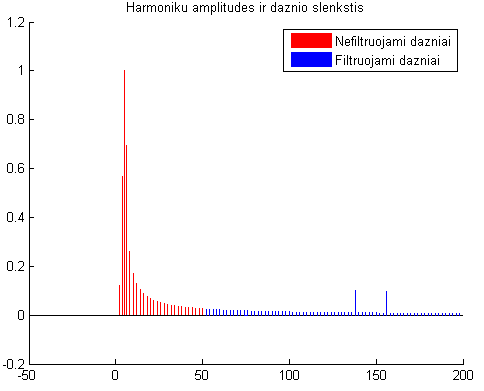
Pav. Harmonikų amplitudės ir amplitudinis slekstis



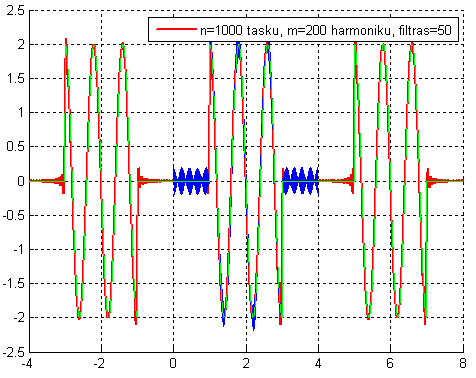
Pav. Aproksimuotos funkcijos grafikas

Atliktas filtravimas atmetant harmonikas, kurių dažnis didesnis už 50. Pateikti

harmonikų ir aproksimuotos funkcijos grafikai.



Pav. Harmonikų amplitudės ir dažnio slenkstis



Pav. Aproksimuotos funkcijos grafikas

**Programos kodas**

%

% Viename periode aprasytai funkcijai atliekama diskrecioji Furje

% transformacija

%

function main

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];

% disp('kontrole:'),disp(sum(fC(3,T,t).\*fC(0,T,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,'r.-');

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

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

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],'m--','LineWidth',3); % braizo 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];

% frequencies=[1:3,5,6,10,30,50];

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,'r-','LineWidth',3);plot(t,fff,'b-');plot(ttt,fff2,'g-','LineWidth',2);

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

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

Y = fft(fnk(T,t))

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

filt = 50; %daznis filtravimui

ind=Pyy<filt;

Y(ind)=0;

yt=ifft(Y);

figure(4),hold on,grid on,plot(t,yt,'b.-','MarkerSize',8);

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

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

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('Harmoniku amplitudes ir 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];

% frequencies=[1:3,5,6,10,30,50];

for i=frequencies

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

end

figure(6),hold on,grid on, plot(ttt,fffz1,'r','LineWidth',2);plot(t,fff,'b-','LineWidth',1);plot(ttt,fff2,'g-')

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),

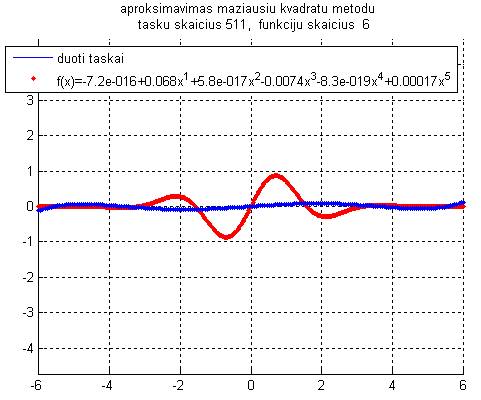
rez=(1-sign(cos(2\*pi\*t/T))).\*sin(2\*pi\*5\*t/T) + 0.09.\*sin(2\*pi\*156\*t/T) + 0.11.\*sin(2\*pi\*138\*t/T); return, end %su triuksmais

function rez=fnk2(T,t),

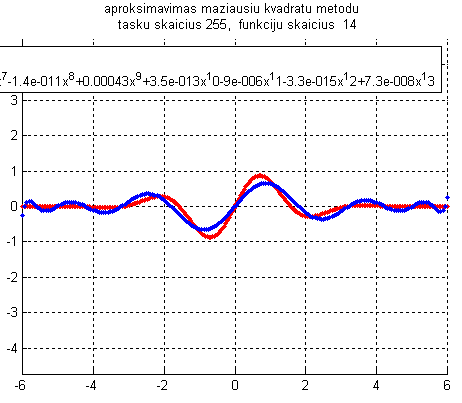
rez=(1-sign(cos(2\*pi\*t/T))).\*sin(2\*pi\*5\*t/T); return, end %be triuksmu

**Aproksimavimas daugianariais vienanarių bazėje**

**Funkcija**

****

Daugianaris: f(x)=-7.2e-016+0.068x^1+5.8e-017x^2-0.0074x^3-8.3e-019x^4+0.00017x^5

****

Daugianaris: f(x)=-3.5e-009+ 1.2x^1+6.8e-009x^2-0.68x^3-2.2e-009x^4+ 0.12x^5+2.6e-010x^6-0.01x^7-1.4e-011x^8+0.00043x^9+3.5e-013x^10-9e-006x^11-3.3e-015x^12+7.3e-008x^13

**Programos kodas:**

%

% Aproksimuojama vienanariu bazeje. Duomenys imami is failo

% Parodoma, kokia yra aproksimuojancios funkcijos elgsena

% uz duomenu intervalo ribu

function main

clc,close all,clear all

% Is failu ivedami duomenys:

npower=9

n=2^npower-1;

fclose all; fhx=fopen('carx.txt','r'); fhy=fopen('cary.txt','r');

figure(100); axis equal,hold on,grid on

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

ppp=0\*60, % sukuriamas "atstumas" tarp signalu %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if ppp~= 0 % sukuriamas vaizdas "ilgame periode"

SX(2:end+1)=SX(1:end);SX(1)=SX(2)-ppp;SX(end+1)=SX(end)+ppp;

SY(2:end+1)=SY(1:end);SY(1)=0;SY(end+1)=0;

end

plot(SX,SY);

a=min(SX);b=max(SX);t=[a:(b-a)/n:b];

fff=interp1(SX,SY,t); % perskaiciuojama i naujas abscises

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

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

pause

% Maziausiu kvadratu metodo lygciu sistema:

m= 6;

G=base(m,t)

c=(G'\*G)\(G'\*fff')

sss=sprintf('%5.2g',c(1));

for i=1:m-1

sss=[sss,sprintf('+%5.2gx^%1d',c(i+1),i)];

end

sss=strrep(sss,'+-','-');

% Aproksimuojanti funkcija:

nnn=200; %vaizdavimo tasku skaicius

tmin=min(t);tmax=max(t);

% tmin=-14;tmax=28;

ttt=[tmin:(tmax-tmin)/(nnn-1):tmax]; %vaizdavimo taskai

Gv=base(m,ttt);

fff1=Gv\*c;

plot(ttt,fff1,'b.');

legend({'duoti taskai',sprintf('f(x)=%s',sss)})

title(sprintf('aproksimavimas maziausiu kvadratu metodu \n tasku skaicius %d, funkciju skaicius %d',n,m));

end

function G=base(m,x)

for i=1:m, G(:,i)=x.^(i-1); end

return

end