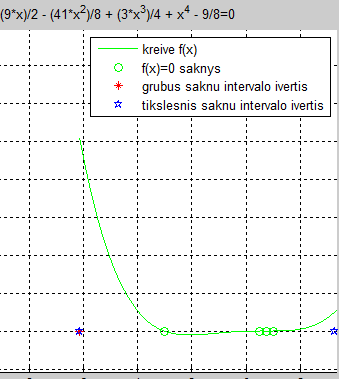
# **1 Netiesinių lygčių sprendimas**

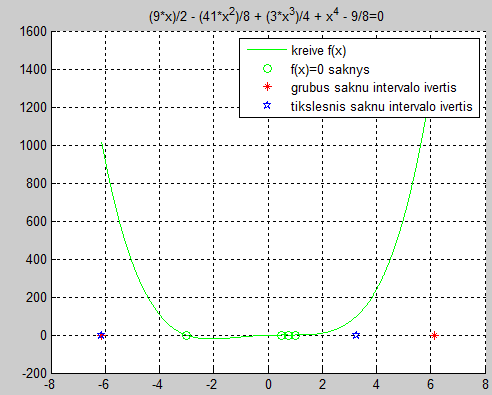
Duotos dvi netiesinės lygtys: daugianaris f(x) = 0 ir transcendentinė funkcija g(x)=0.

|  |  |  |
| --- | --- | --- |
| Nr. | Daugianaris f(x) | Funkcija g(x) |
| 21 |  |  |
| Sprendimo metodai: skenavimo , stygų, kirstinių | | |

## **Lygties f(x) = 0 (f(x) – daugianaris) sprendimas**

* ***Daugianario šaknų intervalo įverčiai***





a b

1. pav Daugianario šaknų intervalo įverčiai (a) ir grafinis funkcijos vaizdas tikslesniame šaknų intervale (b)

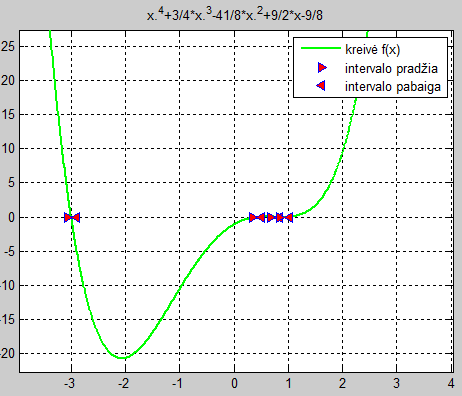
**1 lentelė**. Šaknų intervalo įverčiai

|  |  |
| --- | --- |
| Grubus lygties 𝑓(𝑥) = 0 šaknų intervalo įvertis | [-6.125;6.125] |
| Tikslesnis lygties 𝑓(𝑥) = 0 šaknų intervalo įvertis | [ -6.125;3.26385] |

Iš grafiko matome, kad grubaus ir tikslesnio šaknų intervalo kairieji rėžiai sutampa, tuo tarpu, dešinysis rėžis naudojant tikslesnio lygties f(x) intervalo įverčius yra beveik perpus sumažinamas t.y nuo 6.125 jis atsitraukia iki 3.26385.

* ***Šaknų atskyrimas skenavimo metodu***

Skenavimas atliekamas intervale [ -6.125;3.26385], skenavimo žingsnis lygus 0.17.



2 pav. Daugianario šaknų atskyrimo intervalai

**2 lentelė** Šaknies atskyrimo intervalai

|  |  |
| --- | --- |
| Intervalo Nr. | Intervalas |
| 1 | [-3.065000000000001; -2.895000000000001] |
| 2 | [0.334999999999998 ; 0.504999999999998] |
| 3 | [0.674999999999998 ; 0.844999999999998] |
| 4 | [0.844999999999998 ; 1.014999999999998] |

Skenavimo žingsnis parenkamas atsižvelgiant į tai kaip arti viena kitos išsidėsčiusios šaknys, nes parinkus per didelį žingsnį galima peršokti arti esančia šaknį. Šiuo atveju pirmoji šaknis yra nutolus nuo likusių, tačiau likusios trys šaknys yra intervale nuo 0.3349999 iki 1.014999, jos išsidėsčiusios viena šalia kitos tad reikia objektyviai parinkti gan maža žingsnį. Parinkus žingsnį 0.17 visos šaknys yra teisingai randamos.

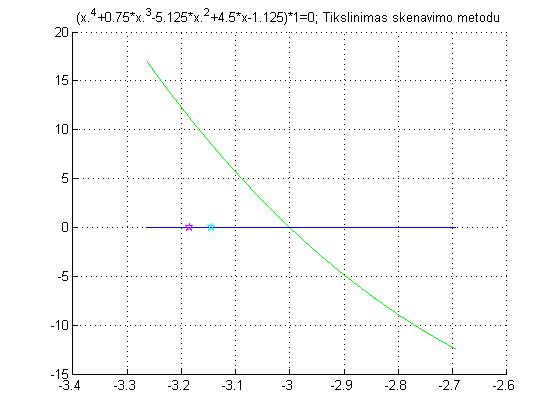
• ***Šaknų tikslinimas skenavimo, stygų ir kirstinių metodais***

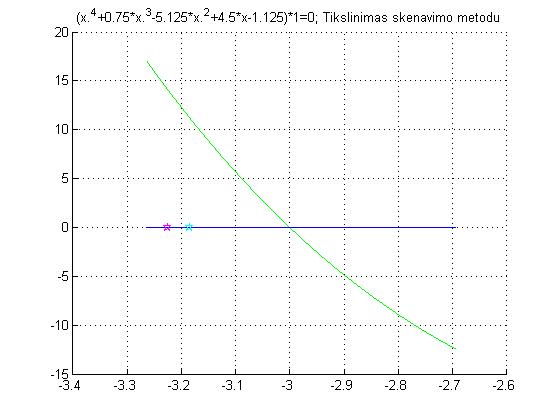
Tariama, kad 𝑥mid yra šaknis (stabdomi skaičiavimai), jei |𝑓(𝑥mid)| < 1𝑒 − 9. Skaičiavimuose naudojamas šaknies tikslumo įvertis |𝑓(𝑥mid)|.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Skenavimo metodas | Pradinis intervalas | Šaknis | Tikslumas | | Iteracijų skaičius | | |
| [-3.065000000000001;  -2.895000000000001] | -2,999999999096609 | 0.000000000903391 | | 109 | | |
| [0.334999999999998; 0.504999999999998] | 0,499999999775999 | 0.000000000224001 | | 24 | | |
| [0.674999999999998; 0.844999999999998] | 0,74999999916 | 0.00000000084 | | 111 | | |
| [0.844999999999998; 1.014999999999998] | 0,999999999743999 | 0.000000000256001 | | 35 | | |
| Stygų metodas | Pradinis intervalas | Šaknis | Tikslumas | | Iteracijų skaičius | | |
| [-3.065000000000001;  -2.895000000000001] | -2,99999999975722 | 0.00000000024278 | | 8 | | |
| [0.334999999999998; 0.504999999999998] | 0,499999999191698 | 0.000000000808302 | | 23 | | |
| [0.674999999999998; 0.844999999999998] | 0,7499999999993952 | 0.00000000000060485 | | 4 | | |
| [0.844999999999998; 1.014999999999998] | 0,999999999167884 | 0.000000000832116 | | 8 | | |
| Kirstinių metodas | Pradinis artinys | Antras pradinis artinys | Šaknis | Tikslumas | | Iteracijų skaičius |
| 3.065000000000001 | 3.265000000000001 | -2.99999999999999 | 0.0000000000000710543 | | 5 |
| 0.334999999999998 | 0.534999999999998 | 0.499999999570992 | 0.000000000187691 | | 6 |
| 0.674999999999998 | 0.874999999999998 | 0.750000000176986 | 0.0000000000414815 | | 4 |
| 0.844999999999998 | 1.044999999999998 | 0.999999999998594 | 0.000000000000703659 | | 9 |
| MATLAB funkcijos | Pradinis intervalas | Šaknys(fzero) | Tikslumas | | Iteracijų skaičius | | |
| 3.065000000000001 | -3.000000000000000 |  | |  | | |
| 0.334999999999998 | 0.500000000000000 |  | |  | | |
| 0.674999999999998 | 0.750000000000000 |  | |  | | |
| 0.844999999999998 | 1.000000000000001 |  | |  | | |

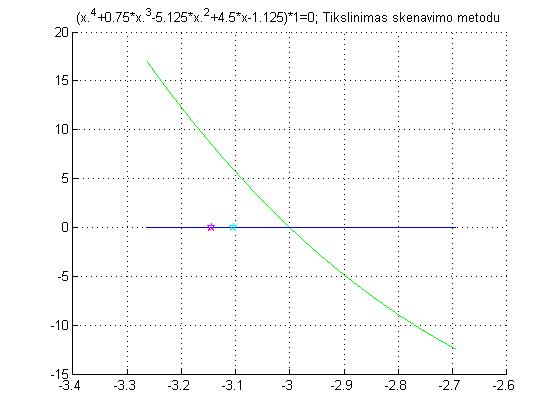
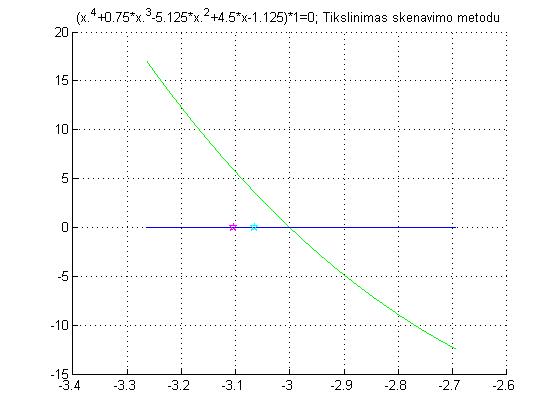
***Metodų palyginimas:***

Žiūrint pagal metodų iteracijų skaičių greičiausiai veikė kirstinių metodas, kurių iteracijų skaičius svyravo nuo 5 iki 9 iteracijų, tuo tarpu, stygų metodas buvo šiek tiek lėtesnis ir jam prireikė nuo 4 iki 23 iteracijų. Pats lėčiausias metodas buvo skenavimo – jo iteracijų skaičius svyravo nuo 24 iki 111 iteracijų.

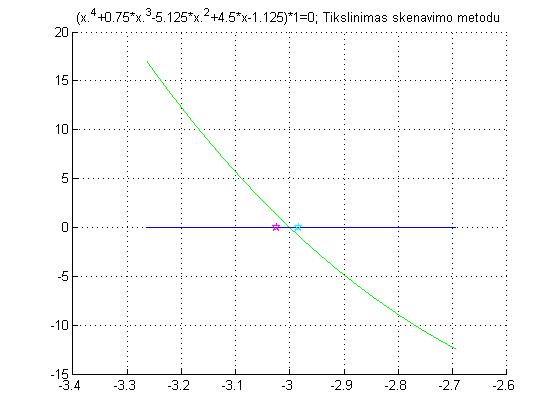


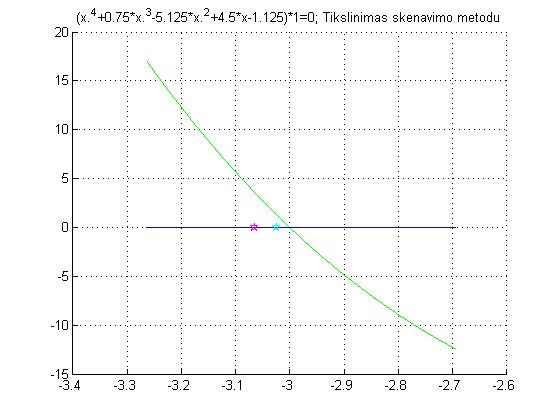


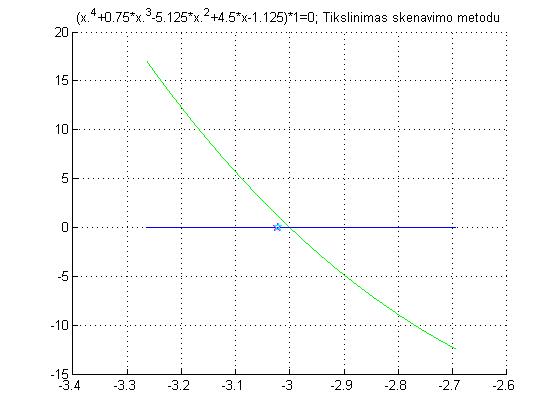
1 iteracija 2 iteracija



3 iteracija 4 iteracija



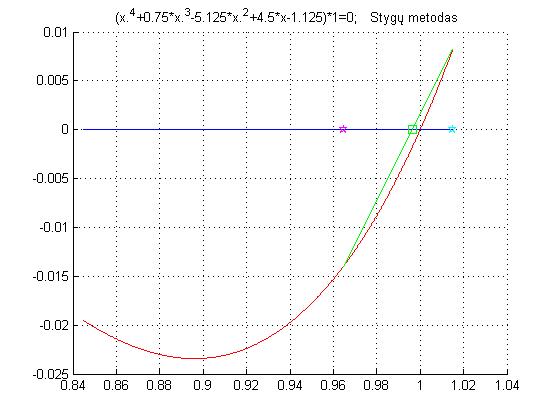
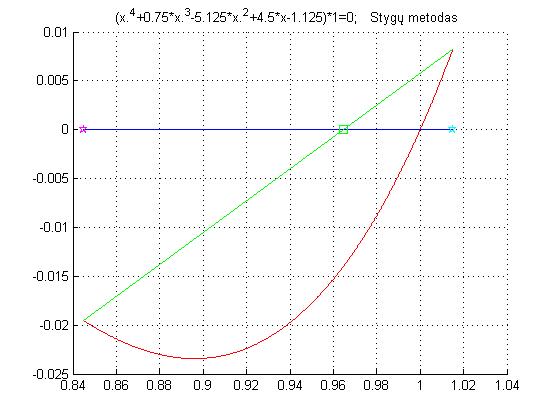


5 iteracija 6 iteracija

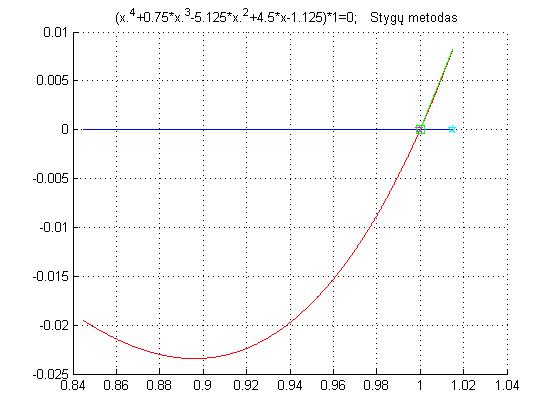
7 iteracija

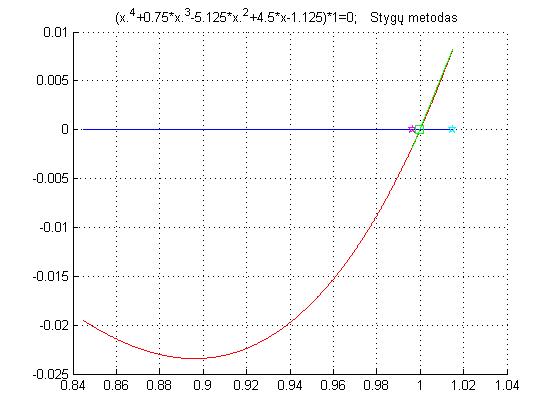
3 pav. Šaknies 𝒙mid = −3 tikslinimo skenavimo metodu vizualizacija. Mėlyna linija brėžiama funkcija, raudonu ir žydru taškais žymimi iteracijoje nagrinėjamo intervalo galai.

Skenavimas vykdomas žingsniu 0.17 tad norint surasti reikšmę, kurios tikslumas mažesnis už 1𝑒 – 9 prireikia net 109 iteracijos.

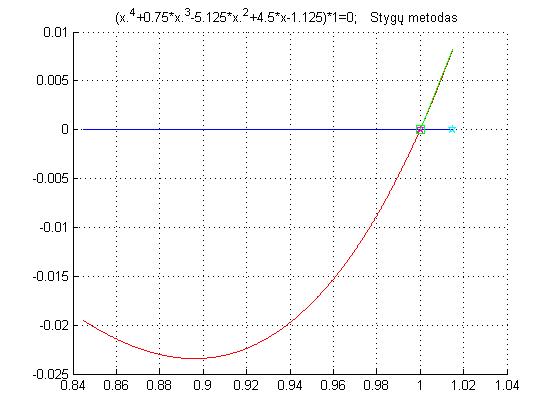


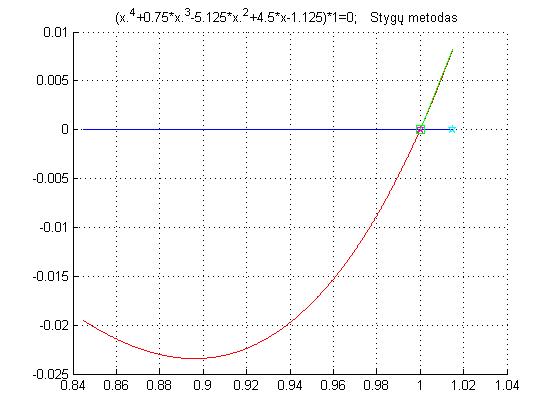
1 iteracija 2 iteracija





3 iteracija 4 iteracija

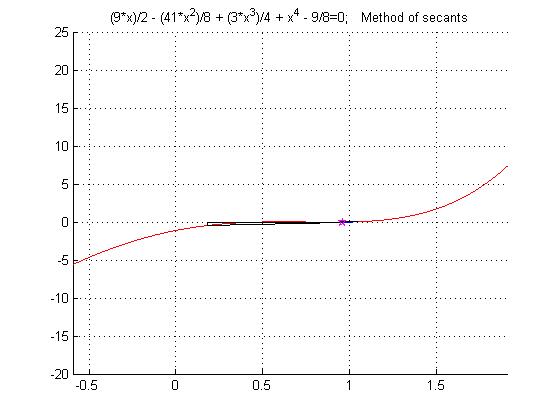


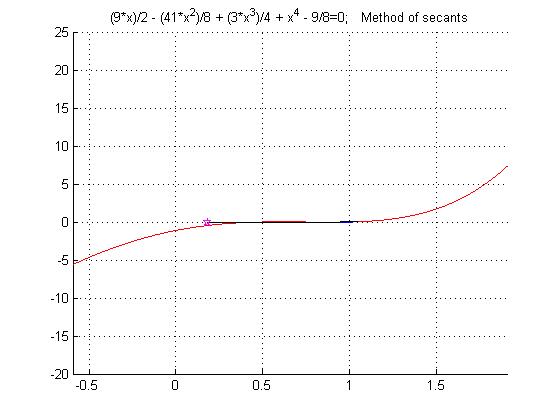


5 iteracija 6 iteracija

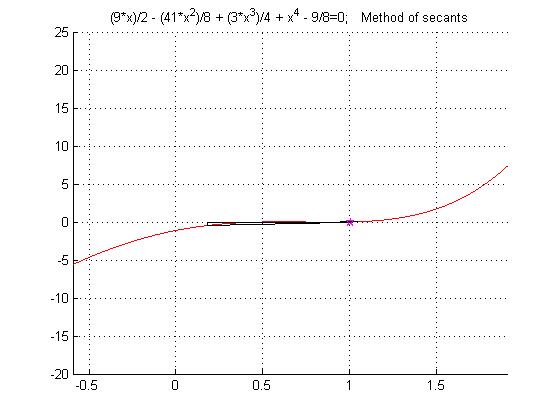
4.pav – Šaknies xmid = 1 tikslinimas **stygų** metodu. Violetinė linija brėžiama funkcija, žalia linija styga, žalias kvadratas - taškas, kur styga kerta x ašį. Violetinis taškas stygos kairiojo galo reikšmė x ašyje.

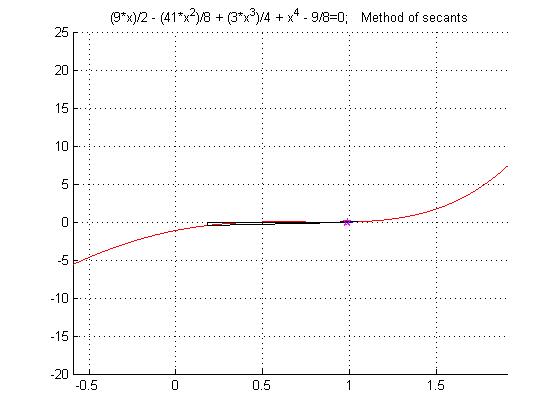
Stygų metodui prireikia kur kas mažiau įteracijų nei skenavimo metodui, šiuo atveju prireikia tik 8 iteracijų, kai naudojant skenavimo metodą reikėjo 35 iteracijų.

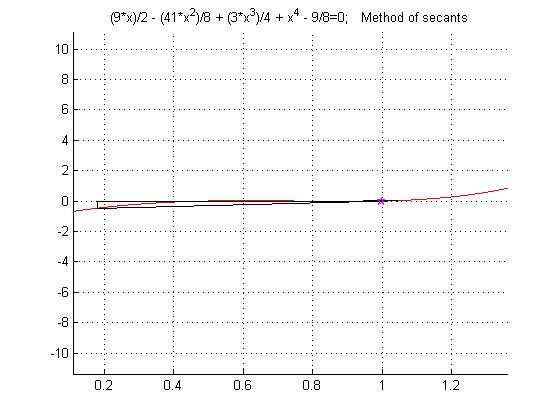




1 iteracija 2 iteracija





3 iteracija 4 iteracija****

5 iteracija

5 pav. Šaknies 𝒙mid = 1 tikslinimo kirstinių metodu vizualizacija. Raudona linija brėžiama funkcija, juoda – kirstinės linijos.

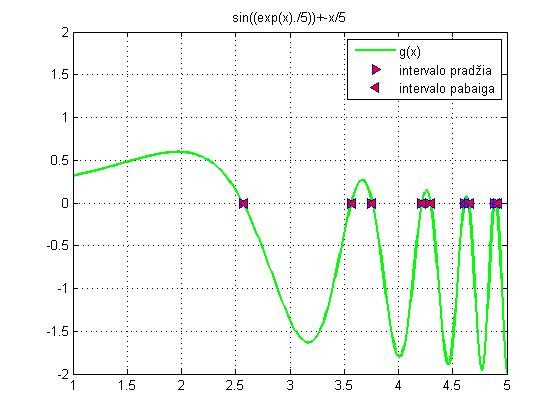
Kirstinių metodas efektyviausia tikslina šaknis, jam prireikia pačio mažiausio iteracijų skaičiaus, šis metodas išsiskiria tuo, kad yra parenkami du pradiniai artiniai, pagal kuriuos yra pradedamas vykdyti tikslinimas

# **1.2 Lygties 𝒈(𝒙) = 𝟎 (𝒈(𝒙) – transcendentinė funkcija) sprendimas**

* Šaknų atskyrimas skenavimo metodu

**4 lentelė** Šaknies atskyrimo intervalai

|  |  |
| --- | --- |
| Intervalo Nr. | Intervalas |
| 1 | [2.560000000000001 ; 2.580000000000001] |
| 2 | [3.560000000000002; 3.580000000000002] |
| 3 | [3.740000000000002 ; 3.760000000000003] |
| 4 | [4.199999999999998 ; 4.219999999999998] |
| 5 | [4.279999999999997 ; 4.299999999999996] |
| 6 | [4.599999999999990 ; 4.619999999999989] |
| 7 | [4.639999999999989 ; 4.659999999999989] |
| 8 | [4.879999999999984 ; 4.899999999999984] |
| 9 | [4.899999999999984 ; 4.919999999999983] |



6 Pav Funkcijos šaknų atskyrimo intervalai

* **Šaknų tikslinimas skenavimo, stygų ir kirstinių metodais**

Tariama, kad 𝑥mid yra šaknis (stabdomi skaičiavimai), jei |𝑓(𝑥mid)| < 1𝑒 − 9. Skaičiavimuose naudojamas šaknies tikslumo įvertis |𝑓(𝑥mid)|.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Skenavimo metodas | Pradinis intervalas | | Šaknis | | Tikslumas | Iteracijų skaičius |
| [2.560000000000001 ; 2.580000000000001] | | 2.56598 | | 0.000000000860587 | 103 |
| [3.560000000000002 ; 3.580000000000002] | | 3.56634 | | 0.000000000139909 | 65 |
| [3.740000000000002 ; 3.760000000000003] | | 3.7582 | | 0.000000000795581 | 143 |
| [4.199999999999998 ; 4.219999999999998] | | 4.2173 | | 0.000000000941724 | 94 |
| [4.279999999999997 ; 4.299999999999996] | | 4.29554 | | 0.000000000656891 | 85 |
| [4.599999999999990 ; 4.619999999999989] | | 4.60621 | | 0.000000000409686 | 102 |
| [4.639999999999989; 4.659999999999989] | | 4.64438 | | 0.000000000717995 | 111 |
| [4.879999999999984 ; 4.899999999999984] | | 4.8862 | | 0.00000000086753 | 109 |
| [4.899999999999984 ; 4.919999999999983] | | 4.90165 | | 0.000000000791088 | 110 |
| Stygų metodas | Pradinis intervalas | | Šaknis | | Tikslumas | Iteracijų skaičius |
| 2.560000000000001 ; 2.580000000000001] | | 2.56598 | | 0.000000000996376 | 4 |
| [3.560000000000002 ; 3.580000000000002] | | 3.56634 | | 0.00000000020717 | 5 |
| [3.740000000000002 ; 3.760000000000003] | | 3.7582 | | 0.000000000885035 | 4 |
| [4.199999999999998 ; 4.219999999999998] | | 4.2173 | | 0.000000000511267 | 9 |
| [4.279999999999997 ; 4.299999999999996] | | 4.29554 | | 0.000000000163267 | 7 |
| [4.599999999999990 ; 4.619999999999989] | | 4.60621 | | 0.000000000169774 | 10 |
| [4.639999999999989; 4.659999999999989] | | 4.64438 | | 0.000000000738486 | 14 |
| [4.879999999999984 ; 4.899999999999984] | | 4.8862 | | 0.000000000815626 | 16 |
| [4.899999999999984 ; 4.919999999999983] | | 4.90165 | | 0.000000000859369 | 26 |
| Kirstinių metodas | Pradinis artinys | Antrasis pradinis artinys | | Šaknis | Tikslumas | Iteracijų skaičius |
| 2.560000000000001 | 2.570000000000001 | | 2.565982530425093 | 0.0000000000161378 | 3 |
| 3.560000000000002 | 3.570000000000002 | | 3.566335164091894 | 0.00000000000000843 | 4 |
| 3.740000000000002 | 3.750000000000002 | | 3.758199466932750 | 0.00000000000000244 | 5 |
| 4.199999999999998 | 4.219999999999998 | | 4.217304107427022 | 0.00000000000018829 | 5 |
| 4.279999999999997 | 4.299999999999997 | | 4.295544920883287 | 0.0000000000207935 | 5 |
| 4.599999999999990 | 4.619999999999990 | | 4.606210523882144 | 0.00000000000020894 | 5 |
| 4.639999999999989 | 4.659999999999989 | | 4.644380407903426 | 0.00000000000074074 | 5 |
| 4.879999999999984 | 4.899999999999984 | | 4.886196665199048 | 0.00000000000048683 | 6 |
| 4.899999999999984 | 4.919999999999984 | | 4.901646385986794 | 0.000000000018591 | 5 |
| MAT  LAB funkcijos | Pradinis artinys | | Šaknis(fzero) | | Tikslumas | Iteracijų skaičius |
| 2.560000000000001 | | ***2.565982530431723*** | |  |  |
| 3.560000000000002 | | 3.566335164091893 | |  |  |
| 3.740000000000002 | | 3.758199466932751 | |  |  |
| 4.199999999999998 | | 4.217304107457049 | |  |  |
| 4.279999999999997 | | 4.295544920885984 | |  |  |
| 4.599999999999990 | | 4.606210523882116 | |  |  |
| 4.639999999999989 | | 4.644380407903426 | |  |  |
| 4.879999999999984 | | 4.886196665199138 | |  |  |
| 4.899999999999984 | | 4.901646385986794 | |  |  |

***Metodų palyginimas***

Žiūrint pagal metodų iteracijų skaičių greičiausiai veikė kirstinių metodas, kurių iteracijų skaičius svyravo nuo 3 iki 6 iteracijų, tuo tarpu, stygų metodas buvo šiek tiek lėtesnis ir jam prireikė nuo 4 iki 26 iteracijų. Pats lėčiausias metodas buvo skenavimo – jo iteracijų skaičius svyravo nuo 65 iki 143 iteracijų.

## **Išvados**

* Tiek su daugianariu f(x)=0, tiek su transcendentine funkcija g(x) =0 efektyviausiai veikė kirstinių metodas, kurio iteracijų skaičius buvo mažiausias.
* Kirstinių metodui nei karto neprireikė daugiau nei 10 iteracijų šaknies tikslinimui.
* Daugiausiai iteracijų prireikė skenavimo metodui.
* Neretai skenavimo metodui prireikdavo daugiau nei 100 iteracijų.

## **Programų tekstai**

* ***Daugianario šaknų intervalo įverčių nustatymas***

function [minimalus, maximalus] = Daugianario\_saknu\_reziu\_iverciai(ff,saknys)

clc, close all

syms f x

% daugianario sudarymas: ------

option=1; % 1 - daugianaris, sugeneruojamas pagal duotas realias saknis;

% 2 - argumento x funkcija, užrašoma simboline išraiška

switch option

case 1 % daugianaris

f=1;

for i=1:numel(saknys), f=f\*(x-saknys(i)); end

f=expand(f); % panariui išskleistas daugianaris

case 2 % simboline išraiška, argumentas x

f=ff; % grubus įvertis geresnis uz tikslesnį (kadangi B/a0 <1)

end

f=expand(f); % daugianario skleistinė

fneig=expand(subs(f,x,-x)); % daugianario skleistinė pakeitus x-> -x

[CF1,orders]=coeffs(f,x); % daugianario f koeficientai ir juos atitinkantys x laipsniai

auksciausias\_x\_laipsnis=char(orders(1));

nnn=strfind(auksciausias\_x\_laipsnis,'^');

n=str2num(auksciausias\_x\_laipsnis(nnn+1:end)); % aukščiausias x laipsnio rodiklis daugianaryje (daugianario eile)

[CF1\_neig,orders\_neig]=coeffs(fneig,x); % daugianario fneig koeficientai ir juos atitinkantys x laipsniai

% suformuojama visu x laipsnių eilė:

for i=1:n+1, orders\_full(i)=x^(n-i+1); end

orders;

% koeficientu eilė papildoma nuliniais nariais:

for i=1:n+1

j=find(orders == orders\_full(i));

if j>0, CF(i)=CF1(j);

CF\_neig(i)=CF1\_neig(j);

else, CF(i)=0;

CF\_neig(i)=0;

end

end

% koeficientas prie aukščiausio x laipsnio turi būti teigiamas:

CF=CF/CF(1); % f(x) koeficientai

CF\_neig=CF\_neig/CF\_neig(1); % f(-x) koeficientai

CF;

CF\_neig;

% Šaknų intervalo įverčiai:

% ------------- Grubus ivertis:

CF\_value=eval(CF); % f(x) koeficientų simboliai paverčiami skaičiais

R=max(abs(CF\_value(2:end)))/CF\_value(1)+1; % taikoma grubaus įverčio formulė

% grafinis funkcijos, šaknų ir grubaus įverčio intervalo pavaizdavimas:

t=-R:R/500:R;

figure(1);grid on;hold on

plot(t,fnk(CF\_value,t),'g-');

if option == 1, plot(saknys,0\*saknys,'go');end

plot([-R,R],[0 0],'r\*')

% ------------ Tikslesnis ivertis:

% teigiamoms šaknims:

neig\_ind=find(CF\_value(2:end) < 0);

if ~isempty(neig\_ind);

B=max(abs(CF\_value(neig\_ind+1)));

k=neig\_ind(1);

Rteig=1+(B/CF\_value(1))^(1/k);

else

Rteig=0;

end

plot(min(R,Rteig),0,'bp'); % pavaizduojamas teigiamų šaknų viršutinės ribos įvertis

% neigiamoms šaknims:

CF\_value\_neig=eval(CF\_neig); % f(-x) koeficientu simboliai paverčiami skaičiais

neig\_ind1=find(CF\_value\_neig(2:end) < 0);

if ~isempty(neig\_ind1)

B=max(abs(CF\_value\_neig(neig\_ind1+1)));

k=neig\_ind1(1);

Rneig=1+(B/CF\_value\_neig(1))^(1/k);

else

Rneig=0

end

maximalus=min(R,Rteig);

minimalus=-min(R,Rneig);

plot(-min(R,Rneig),0,'bp')

switch option, case 1, legend('kreive f(x)','f(x)=0 saknys','grubus saknu intervalo ivertis','tikslesnis saknu intervalo ivertis');

case 2, legend('kreive f(x)','grubus saknu intervalo ivertis','tikslesnis saknu intervalo ivertis');

end

title([char(f),'=0'])

end

function p=fnk(CF,x)

% Apskaičiuoja daugianario reikšmes, kai argumentas yra x

% Kai x yra reikšmių vektorius, p taip pat yra atitinkamu funkcijos reikšmių vektorius

p=0; n=length(CF)-1;

for i=1:length(CF), p=p+CF(i)\*x.^(n-i+1); end % veiksmas < .^ > reiškia, kad laipsniu keliami visi vektoriaus x elementai

return

end

* ***Skenavimo metodas***

function skenavimo\_metodas(f,a,b,nr)

range=[a-0.2; b+0.2]; % parenkame saknu atskyrimo intervala

plot(range(1), range(2));

eps=1e-9; % parenkame tikslumo reikšme

nitmax=150;% parenkame didžiausią leistiną iteracijų skaičių

step = 1; % žingsnis

konst = 25;

% Grafiko braižymas

npoints=1000; x=range(1): (range(2)-range(1))/(npoints-1) :range(2);

figure(nr); grid on; hold on;

str=[f,'=0; Tikslinimas skenavimo metodu']; title(str);

plot(x,eval(f),'g-');

plot(range,[0 0],'b-');

xn=range(1);

xn1=range(2);

i = 0;

while xn < range(2)

nit=0;

prec = 1;

while prec > eps && xn < range(2)

nit=nit+1;

if nit > nitmax, fprintf('Viršitas leistinas iteraciju skaičius');break;end

plot(xn,0,'mp'); %vaizduojamas kairys taškas

h = findobj(gca,'Type','line');

h1=h(1); % paskutinio grafinio objekto valdiklis įrašomas handle masyvo priekyje

plot(xn1,0,'cp'); %dešinio taško vaizdavimas

h = findobj(gca,'Type','line');

h2=h(1);

xn1 = xn + step;

x = xn1;

fxn1 = eval(f);

x = xn;

fxn = eval(f);

if sign(fxn) == sign(fxn1) ~= 0

xn = xn1;

xn1 = xn + step;

else

step = step / konst;

xn = xn + step;

xn1 = xn + step;

end

x = xn;

fx = eval(f);

prec = abs(fx);

pause(0.2)

delete(h1);delete(h2);

fprintf(1,'iteracija %d tikslumas= %g \n',nit,prec);

end

if (prec < eps)

xmid = xn;

plot(xn,0,'k\*');plot(xn,0,'ko');

fprintf(1,'\n tikslumas pasiektas, šaknis xmid= %g -> tikslumas= %g\n',xmid, prec);

xn = xn1 + step;

step = 1;

i=i+1;

end

end

function skenavimo\_metodas2(f,a,b,nr)

range=[a; b]; % parenkame saknu atskyrimo intervala

plot(range(1), range(2));

eps=1e-9; % parenkame tikslumo reikšme

nitmax=150;% parenkame didžiausią leistiną iteracijų skaičių

step = 0.01; % žingsnis

konst = 40;

% Grafiko braižymas

npoints=1000; x=range(1): (range(2)-range(1))/(npoints-1) :range(2);

figure(nr); grid on; hold on;

str=[f,'=0; Tikslinimas skenavimo metodu']; title(str);

plot(x,eval(f),'g-');

plot(range,[0 0],'b-');

xn=range(1);

xn1=range(2);

i = 0;

while xn < range(2)

nit=0;

prec = 1;

while prec > eps && xn < range(2)

nit=nit+1;

if nit > nitmax, fprintf('Viršitas leistinas iteraciju skaičius');break;end

plot(xn,0,'mp'); %vaizduojamas kairys taškas

h = findobj(gca,'Type','line');

h1=h(1); % paskutinio grafinio objekto valdiklis įrašomas handle masyvo priekyje

plot(xn1,0,'cp'); %dešinio taško vaizdavimas

h = findobj(gca,'Type','line');

h2=h(1);

xn1 = xn + step;

x = xn1;

fxn1 = eval(f);

x = xn;

fxn = eval(f);

if sign(fxn) == sign(fxn1) ~= 0

xn = xn1;

xn1 = xn + step;

else

step = step / konst;

xn = xn + step;

xn1 = xn + step;

end

x = xn;

fx = eval(f);

prec = abs(fx);

pause(0.1)

delete(h1);delete(h2);

fprintf(1,'iteracija %d tikslumas= %g \n',nit,prec);

end

if (prec < eps)

xmid = xn;

plot(xn,0,'k\*');plot(xn,0,'ko');

fprintf(1,'\n tikslumas pasiektas, šaknis xmid= %g -> tikslumas= %g\n',xmid, prec);

xn = xn1 + step;

step = 1;

i=i+1;

end

end

* ***Stygų metodas***

% Stygu metodas

function stygu\_metodas(f,a,b,sk)

range=[a, b]; % Intervalas tarp vienos šaknies

eps=1e-9; % parenkame tikslumo reikšmę

nitmax=100;% parenkame didžiausią leistiną iteracijų skaičių

% Grafiko braižymas

npoints=1000; x=range(1): (range(2)-range(1))/(npoints-1) :range(2);

figure(sk); grid on; hold on;

str=[f,'=0; Stygų metodas ']; title(str);

plot(x,eval(f),'r-');

plot(range,[0 0],'b-');

xn=range(1);xn1=range(2);prec=1;

nit=0;

while prec > eps

nit=nit+1;

if nit > nitmax, fprintf('Viršytas leistinas iteracijų skaičius');break;end

plot(xn,0,'mp');h = findobj(gca,'Type','line');h1=h(1); % paskutinio grafinio objekto valdiklis įrašomas handle masyvo priekyje

plot(xn1,0,'cp');h = findobj(gca,'Type','line');h2=h(1);

x=xn;fxn=eval(f);x=xn1;fxn1=eval(f);

k=abs(fxn/fxn1);xmid=(xn+k\*xn1)/(1+k);

plot(xmid,0,'gs');plot([xn,xn1],[fxn,fxn1],'g-');h = findobj(gca,'Type','line');h3=h(1:2);

x=xmid;fxmid=eval(f);

% jeigu pradžioje tikriname kairį tašką

x=xn;fxn=eval(f);

if sign(fxmid) == sign(fxn), xn=xmid;

else, xn1=xmid;

end

pause(0.2)

delete(h1);delete(h2);delete(h3);

prec=abs(fxmid);

fprintf(1,'iteracija %d x= %g prec= %g \n',nit,xn,prec);

end

plot(xmid,0,'k\*');plot(xmid,0,'ko');

fprintf(1,'\n tikslumas pasiektas, šaknis xmid= %g\n\n',xmid);

end

* ***Kirstinių metodas***

%

% Kirstinių metodas

%

function kirstiniu\_metodas(a,b,sk)

syms f x

f=(-1.125+x.^4+0.75\*x.^3-5.125\*x.^2+4.5\*x.^1)\*1

x0=a; % parenkame pradini artini

x01=a+0.2; % kirstiniu metodui parenkame antra pradini artini

deltax=0.1; % parenkame pradine zingsnio reiksme (reikalinga tik kirstiniu metodui)

nitmax=140; % parenkame didziausia leistina iteraciju skaiciu

if x0 ~= 0, range=[a,b]; % parenkame intervala vaizdavimui

else, range=[a,b];

end

eps=1e-9; % Parenkame tiksluma

method='secants';

x=x01;fxn1=eval(f);x=x0;fxn=eval(f);

dfxn=(fxn1-fxn)/(x01-x0); % Taikant kirstiniu metoda, reiks apskaiciuoti ,

% pradines kirstines krypti pagal du pradinius artinius

% braizomas funkcijos grafikas:

npoints=1000; xrange=range(1)-2: (range(2)-range(1))/(npoints-1) :range(2)+2;

figure(sk); grid on; hold on; str=[char(f),'=0; Method of ',method];title(str);

x=xrange; % simbolinis x keiciamas reiksmemis is parinkto funkcijos vaizdavimo intervalo

plot(x,eval(f),'r-');

plot(range,[0 0],'b-');

plot(x0,0,'mp');

h = findobj(gca,'Type','line');

h1=h(1);

figure(sk);

%Sprendimas

xn=x0;

prec=1;

nit=0;

xn1=x01;

plot([xn,xn,xn1,xn1],[0,fxn,fxn1,0],'k-');

% antras pradinis artinys

while prec > eps % iteracijos

nit=nit+1;

if nit > nitmax, fprintf('Virsytas leistinas iteraciju skaicius');break;end

xn1=xn-fxn/dfxn;

plot([xn,xn,xn1],[0,fxn,0],'k-');

pause(0.3);

delete(h1);plot(xn1,0,'mp');h = findobj(gca,'Type','line');h1=h(1);

x=xn1;fxn1=eval(f);dfxn=(fxn1-fxn)/(xn1-xn);

xn=xn1;

fxn=fxn1;

input('Press Enter')

% pause(1)

figure(sk); % skaiciavimas stabdomas iki bus paspaustas Enter klavisas

x=xn;fxn=eval(f);prec=abs(fxn);

fprintf(1,'iteracija %d x= %g prec= %g \n',nit,xn,prec);

end

plot(xn,fxn,'k\*');plot(xn,fxn,'ko');

xn

nit

end

%

% Kirstinių metodas

%

function kirstiniu\_metodas2(a,b,sk)

syms f x

f=sin((exp(x)./5))+-x/5;

x0=a; % parenkame pradini artini

x01=a+0.01; % kirstiniu metodui parenkame antra pradini artini

deltax=0.01; % parenkame pradine zingsnio reiksme (reikalinga tik kirstiniu metodui)

nitmax=150; % parenkame didziausia leistina iteraciju skaiciu

if x0 ~= 0, range=[a-0.2,b+0.2]; % parenkame intervala vaizdavimui

else, range=[a-0.2,b+0.2];

end

eps=1e-9; % Parenkame tiksluma

x=x01;fxn1=eval(f);x=x0;fxn=eval(f);

dfxn=(fxn1-fxn)/(x01-x0); % Taikant kirstiniu metoda, reiks apskaiciuoti ,

% pradines kirstines krypti pagal du pradinius artinius

% braizomas funkcijos grafikas:

npoints=1000; xrange=range(1): (range(2)-range(1))/(npoints-1) :range(2);

figure(sk); grid on; hold on; str=[char(f),'=0; Kirstiniu metodas'];title(str);

x=xrange; % simbolinis x keiciamas reiksmemis is parinkto funkcijos vaizdavimo intervalo

plot(x,eval(f),'r-');

plot(range,[0 0],'b-');

plot(x0,0,'mp');

h = findobj(gca,'Type','line');

h1=h(1);

figure(sk); % skaiciavimas stabdomas iki bus paspaustas Enter klavisas

%Sprendimas

xn=x0;

prec=1;

nit=0;

xn1=x01;

plot([xn,xn,xn1,xn1],[0,fxn,fxn1,0],'k-');

% antras pradinis artinys

while prec > eps % iteracijos

nit=nit+1;

if nit > nitmax, fprintf('Virsytas leistinas iteraciju skaicius');break;end

xn1=xn-fxn/dfxn;

plot([xn,xn,xn1],[0,fxn,0],'k-');

pause(0.3);

delete(h1);plot(xn1,0,'mp');h = findobj(gca,'Type','line');h1=h(1);

x=xn1;fxn1=eval(f);dfxn=(fxn1-fxn)/(xn1-xn);

xn=xn1;

fxn=fxn1;

figure(sk);

x=xn;fxn=eval(f);prec=abs(fxn);

fprintf(1,'iteracija %d x= %g prec= %g \n',nit,xn,prec);

end

plot(xn,fxn,'k\*');plot(xn,fxn,'ko');

xn

nit

end

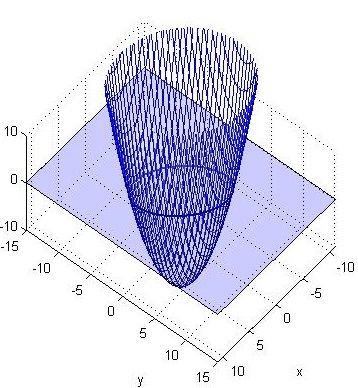
# **Netiesinių lygčių sistemų sprendimas**

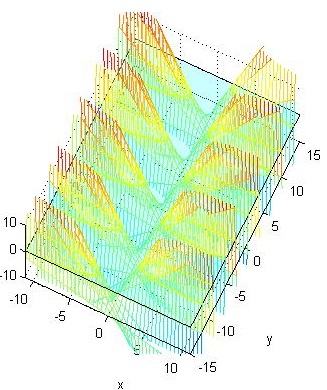
|  |  |  |  |
| --- | --- | --- | --- |
| Varianto Nr. | 1 lygčių sistema | 2 lygčių sistema | Metodas |
| 21 |  |  | Broideno |

# **2.1 1-os netiesinių lygčių sistemos sprendimas**

* ***Paviršių grafinis vaizdas***

***Z1*(x1,x2)= *Z1*(x1,x2=**

****

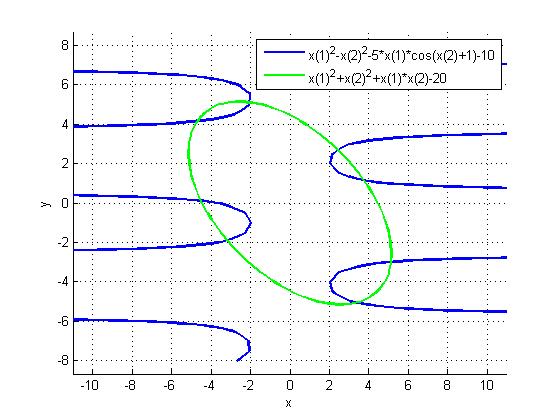


*a b*

*7pav. Paviršių grafinis vaizdas.*

Atvaizdavus 1 funkciją plokštumoje, galima įžvelgti, kad ji yra periodine, nes funkcijoje yra naudojama trigonometrinė cos funkcija. 2 Funkcija, priešingai, nėra periodinė.

* ***Sprendimas grafiniu būdu***



8 pav. Lygčių sistemos sprendiniai nustatomi grafikų susikirtimo taškuose.

Žiūrint į šį grafiką, galime pastebėti, kad yra 8 taškai, kuriuose kertasi grafikai, jų pozicija x ašyje svyruoja nuo beveik -6 iki 6, y ašyje taip pat nuo -6 iki 6.

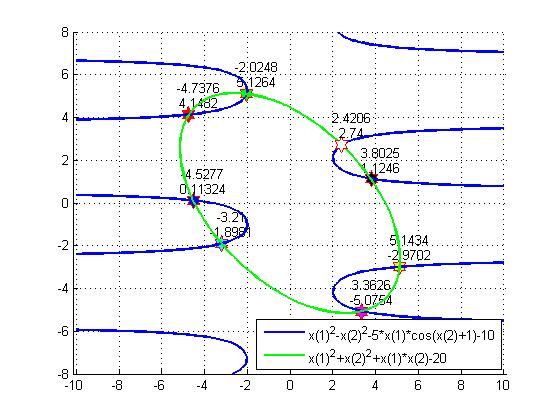
* ***Sprendimas Broideno metodu***

Broideno metodui naudojamas tikslumas 1e-5, tuomet ,kai jis yra pasiekiamas, stabdomi skaičiavimai ir išvedamas sprendinys x

**Lentelė 6** rezultatų lentelė

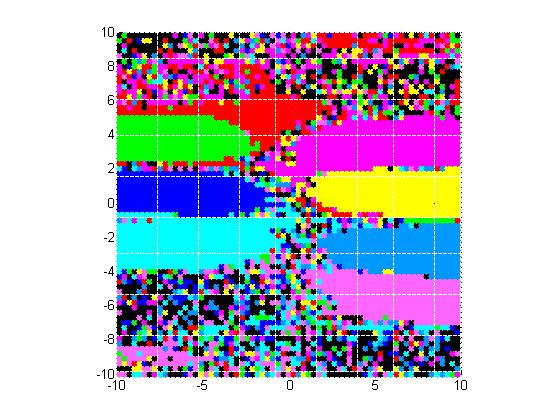
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pradinis artinys | Sprendinys Broideno metodu | Tikslumas | Iteracijų skaičius | Sprendinys MATLAB funkcija **fsolve** |
| [-4;4] | [-4.73758; 4.14823] | 0.00000558614 | 5 | [-4.7376 ;4.1482] |
| [-2;5] | [-2.02482; 5.12642] | 0.00000245641 | 4 | [-2.0248;5.1264] |
| [-4;0] | [-4.5277 ; 0.113245] | 0.00000662001 | 5 | -4.5277;0.1133] |
| [-2;-2] | [-3.20999 ; -1.89814] | 0.00000321482 | 6 | [-3.2100;-1.8982] |
| [4;-5] | [3.36256 ; -5.07538] | 0.00000454169 | 6 | 3.3625;-5.0754] |
| [5;-3] | [5.14344 ; -2.97017] | 0.000000121914 | 4 | [5.1434;-2.9702] |
| [4;1] | [3.80247; 1.12464] | 0.00000463276 | 4 | [3.8025;1.1247] |
| [2;3] | 2.42063 2.74004 | 0.00000890802 | 4 | [2.4207;2.7400] |

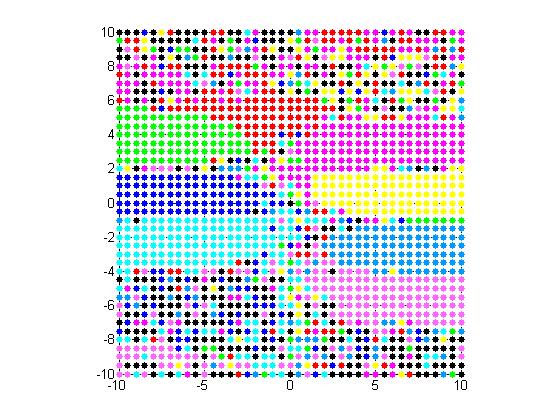
Iš lentelės užpildytos gautais duomenimis įvykdžius matlabo programą, galima pastebėti, kad parinkti teisingi pradiniai artiniai, nes iteracijų skaičius yra itin mažas, jis neviršija 6 iteracijų.



9 pav. Broideno metodo sprendiniai, pateikti 6 lentelėje (b).

Broideno metodu gautos šaknys pavaizduotus 9 paveikslėlyje, Broideno metodas leidžia surasti pakankamai tikslias šaknis, kurias buvo galima įžvelgti naudojant grafinį metodą.





a b

10 pav. Pradinių artinių tinklelis, kai broideno metodas vykdomas pažingsniu 0.5(a) ir 0.3(b)

Pradinių artinių tinklelis parodo, kokios šaknys gaunamos pagal skirtingose pozicijose esančius taškus, pagal sudarytus tinklelius matos, kad renkantis žingsnį rezultatai išlieka tokie pat, t.y. matomos 8 sritys, kurių taškai randa atitinkamai arti esančias šaknis, tuo tarpu, toliau esantys taškai veikia nenuspėjamai.

**Legenda**

Raudona \* -2.0248 ; 5.1264

Žalia \* -4.7376 ; 4.1482

Tamsiai mėlyna \* -4.5277 ; 0.1133

Žydra \* -3.2100 ; -1.8982

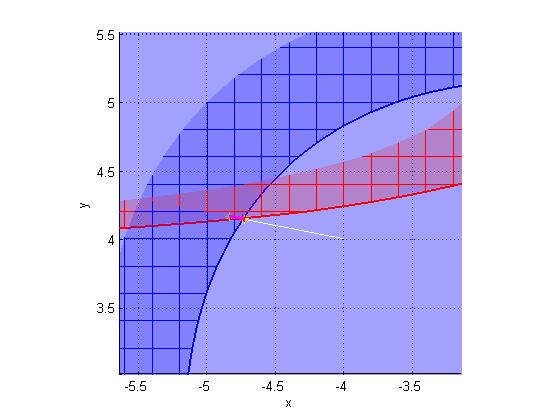
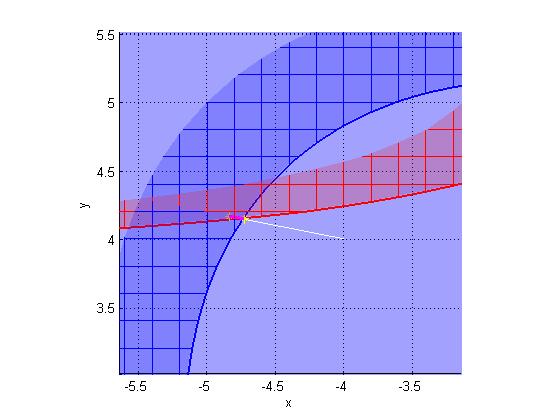
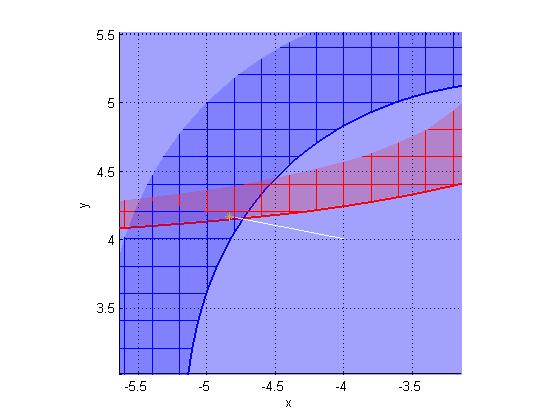
Šviesiai violetinė \* 3.3625 ; -5.0754

Tamsesnė mėlyna \* 5.1434 ; -2.9702

Geltona \* 3.8025 ; 1.1247

Ryškiai violetinė \* 2.4207 ; 2.7400

Juoda \* šaknis nerasta



11pav. Broideno metodo vizualizacija nuo pradinio artinio [-4 ; 4]

Kadangi parinktas tinkamas pradinis artinys (-4;4) dėl to Broideno metodui tereikia 5 įteracijų šaknies suradimui.

# 2.2 II-os netiesinių lygčių sistemos sprendimas

Tariama, kad 𝑥 yra sprendinys (stabdomi skaičiavimai), jei tikslumas < 1𝑒 – 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pradinis artinys | Sprendinys Broideno metodu | Tikslumas | Iteracijų skaičius | Sprendinys MATLAB funkcija fsolve |
| x=[-5 2 -9 3] | 4 5 3 -1 | 0.00000000000000963042 | 78 | 4.0000  5.0000  3.0000  -1.0000 |

Iš lentelės matome, kad Broideno metodas keturių funkcijų sistemai grąžina tokias pat šaknis kaip ir MATLAB metodas fsolve, šiam metodui prireikia daugiau iteracijų nei naudojant 2 funkcijų sistemoms ir šiuo atveju iteracijų skaičius pakyla iki beveik 80 iteracijų

Iš lentelės matome, kad pasirinktas pradinis artinys leido gauti tikslias šaknis, jam prireikė 78 iteracijų, reikšmės nesiskiria nuo gautų su fsolve funkcija

* 1. Išvados

Gautas artinių tinklelio išsidėstymas itin panašus į tai kaip išsidėsčiusios periodinės funkcijos

Broideno metodas 2 funkcijų sistemai veikė kur kas greičiau nei metodas 4 funkcijų sistemai.

Broideno metodui 2 funkcijų sistemai prireikė ne daugiau nei 6 iteracijų šaknims rasti

Pagal pradinių artinių tinklelyje išsidėsčiusių juodų taškų kiekį, matome, kad nemažai atveju funkcija konvergavo, neatsirado šaknų, arba tiesiog viršijo maksimalų iteracijų skaičių.

* 1. Programų tekstai

%Grafinis metodas lygciu sistemai

function grafinis\_metodas(sk)

x=[-11:0.5:11];y=[-8:0.5:8];

Z=pavirsius(@f,x,y);

figure(sk),hold on,grid on,axis equal,axis([min(x) max(x) min(y) max(y) -10 10]);view([1 1 1]);

sk=sk+1;

xlabel('x'),ylabel('y');

mesh(x,y,Z(:,:,1)','FaceAlpha',0.2);contour(x,y,Z(:,:,1)',[0,0 ],'LineWidth',1.5);

xx=axis;

fill([xx(1),xx(1),xx(2),xx(2)],[xx(3),xx(4),xx(4),xx(3)],'c','FaceAlpha',0.2);

figure(sk),hold on,grid on,axis equal,axis([min(x) max(x) min(y) max(y) -10 10]);view([1 1 1]);

sk=sk+1;

xlabel('x'),ylabel('y')

mesh(x,y,Z(:,:,2)','FaceAlpha',0.2);contour(x,y,Z(:,:,2)',[0 0],'LineWidth',1.5)

xx=axis;

fill([xx(1),xx(1),xx(2),xx(2)],[xx(3),xx(4),xx(4),xx(3)],'b','FaceAlpha',0.2);

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

sk=sk+1;

contour(x,y,Z(:,:,1)',[0 0],'LineWidth',1.5,'Linecolor',[0 0 1])

contour(x,y,Z(:,:,2)',[0 0],'LineWidth',1.5,'Linecolor',[0 1 0])

% contour(x,y,Z(:,:,1)')

% contour(x,y,Z(:,:,2)')

xlabel('x'),ylabel('y')

legend('x(1)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10','x(1)^2+x(2)^2+x(1)\*x(2)-20');

return

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[x(2)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10;

x(1)^2+x(2)^2+x(1)\*x(2)-20];

return

end

function Z=pavirsius(funk,x,y)

for i=1:length(x)

for j=1:length(y)

Z(i,j,1:2)=funk([x(i),y(j)]);

end

end

return

end

% Broideno metodas

function Broideno\_metodas(sk,ss,ss2,index)

x=[-10:0.2:10];y=[-10:0.2:10];

Z=pavirsius(@f,x,y);

fig1=figure(sk);

hold on,grid on,axis equal,axis([min(x) max(x) min(y) max(y) 0 5]);view([0 0 1]);xlabel('x'),ylabel('y');

mesh(x,y,Z(:,:,1)','FaceAlpha',0.2,'FaceColor','r','EdgeColor','r');contour(x,y,Z(:,:,1)',[0 0],'LineWidth',1.5,'LineColor','r');

mesh(x,y,Z(:,:,2)','FaceAlpha',0.2,'FaceColor','b','EdgeColor','b');contour(x,y,Z(:,:,2)',[0 0],'LineWidth',1.5,'LineColor','b');

xx=axis; fill([xx(1),xx(1),xx(2),xx(2)],[xx(3),xx(4),xx(4),xx(3)],'b','FaceAlpha',0.2);

xx=axis; fill([xx(1),xx(1),xx(2),xx(2)],[xx(3),xx(4),xx(4),xx(3)],'b','FaceAlpha',0.2);

eps=1e-5;itmax=100;

x=[ss;ss2];

n=length(x);

% Pradinio Jakobio matricos artinio apskaiciavimas: \*\*\*\*\*\*\*\*\*\*

if 1 % pradinis A artinys skaiciuojamas pagal skaitinio diferencijavimo formule;

dx=sum(abs(x))\*1e-5; f0=f(x);

for i=1:n

x1=x;x1(i)=x1(i)+dx;

f1=f(x1);

A(:,i)=(f1-f0)/dx;

end

else, % pradinis A artinys yra parinkta istrizainine matrica

A=eye(n)\*10 %\*15 \*8)

end

% Broideno metodo iteracijos:

ff=f(x); % pradine funkcijos reiksme

for iii=1:itmax

deltax=-A\ff;

x1=x+deltax;

ff1=f(x1);

A=A+(ff1-ff-A\*deltax)\*deltax'/(deltax'\*deltax);

figure(sk);plot3(x1(1),x1(2),0,'y\*');line([x(1),x1(1)],[x(2),x1(2)],[0,0],'Color','white');

line([x(1),x1(1)],[x(2),x1(2)],[ff(1),0\*ff1(1)],'Color','magenta','LineWidth',2.5);

pause(0.05);

tikslumas=norm(deltax)/(norm(x)+norm(deltax));

fprintf(1,'\n iteracija %d tikslumas %g',iii,tikslumas);

if tikslumas < eps, fprintf(1,'\n sprendinys x =');fprintf(1,' %g',x); plot3(x(1),x(2),0,'rp'); a=x(1); b=x(2); break;

elseif iii == itmax,fprintf(1,'\n \*\*\*\*tikslumas nepasiektas. Paskutinis artinys x =');fprintf(1,' %g',x); plot3(x(1),x(2),0,'gp'); break;

end

ff=ff1;x=x1;

end

figure(12),hold on,grid on,axis equal

x=[-10:0.2:10];y=[-8:0.2:8];

Z=pavirsius(@f,x,y);

contour(x,y,Z(:,:,1)',[0 0],'LineWidth',1.5,'Linecolor',[0 0 1])

contour(x,y,Z(:,:,2)',[0 0],'LineWidth',1.5,'Linecolor',[0 1 0])

colours='rgbcmykw';

plot(a,b,'rh','markers',12,'MarkerFaceColor',colours(index));

text(a+0.5,b+1.2,num2str(a),'HorizontalAlignment','center')

text(a+0.5,b+0.5,num2str(b),'HorizontalAlignment','center')

if(sk==11)

legend('Location','southeast','x(1)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10','x(1)^2+x(2)^2+x(1)\*x(2)-20');

end

return

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[x(2)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10;

x(1)^2+x(2)^2+x(1)\*x(2)-20];

return

end

function Z=pavirsius(funk,x,y)

for i=1:length(x), for j=1:length(y)

Z(i,j,1:2)=funk([x(i),y(j)]);

end, end

return

end

% Broideno metodas 2 lygciu sistemai su vaizdavimu

function output = Broiden\_metodas\_tinkleliui(ss,ss2)

x=[ss;ss2];

n=length(x);

itmax=150;

% Pradinio Jakobio matricos artinio apskaiciavimas: \*\*\*\*\*\*\*\*\*\*

if 1 % pradinis A artinys skaiciuojamas pagal skaitinio diferencijavimo formule;

dx=sum(abs(x))\*1e-5; f0=f(x);

for i=1:n

x1=x;x1(i)=x1(i)+dx;

f1=f(x1);

A(:,i)=(f1-f0)/dx;

end

else, % pradinis A artinys yra parinkta istrizainine matrica

A=eye(n)\*10 %\*15 \*8)

end

% Broideno metodo iteracijos:

ff=f(x); % pradine funkcijos reiksme

for iii=1:itmax

deltax=-A\ff;

x1=x+deltax;

ff1=f(x1);

A=A+(ff1-ff-A\*deltax)\*deltax'/(deltax'\*deltax);

tikslumas=norm(deltax)/(norm(x)+norm(deltax));

if tikslumas < eps, output=[x(1),x(2),1]; break;

elseif iii == itmax, output =[x(1),x(2),0]; break;

end

ff=ff1;x=x1;

end

return

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[x(2)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10;

x(1)^2+x(2)^2+x(1)\*x(2)-20];

return

end

function Z=pavirsius(funk,x,y)

for i=1:length(x), for j=1:length(y)

Z(i,j,1:2)=funk([x(i),y(j)]);

end, end

return

end

function Tinklelis

figure(13),hold on,grid on

x=[-10:0.5:10];y=[-10:0.5:10];

Z=pavirsius(@f,x,y);

hold on,grid on,axis equal,axis([-10 10 -10 10])

for i=-10:0.3:10

for j=-10:0.3:10

xx= Broiden\_metodas\_tinkleliui(i,j);

plot(i,j,'.','markers',16,'MarkerEdgeColor','r');

h = zeros(9,9);

if(xx(3)==1)

if(xx(1)<-4.6 && xx(1)>-4.9 && xx(2)<4.3 && xx(2)>4.05)

plot(i,j,'.','markers',16,'MarkerEdgeColor','g');

end

if(xx(1)<-1.9 && xx(1)>-2.2 && xx(2)<5.3 && xx(2)>5)

plot(i,j,'.','markers',16,'MarkerEdgeColor','r');

end

if(xx(1)<-4.4 && xx(1)>-4.7 && xx(2)<0.4 && xx(2)>0)

plot(i,j,'.','markers',16,'MarkerEdgeColor','b');

end

if(xx(1)<-3 && xx(1)>-3.35 && xx(2)<-1.7 && xx(2)>-2)

plot(i,j,'.','markers',16,'MarkerEdgeColor','c');

end

if(xx(1)<2.6 && xx(1)>2.3 && xx(2)<2.9 && xx(2)>2.6)

plot(i,j,'.','markers',16,'MarkerEdgeColor','m');

end

if(xx(1)<3.9 && xx(1)>3.6 && xx(2)<1.3 && xx(2)>1)

plot(i,j,'.','markers',16,'MarkerEdgeColor','y');

end

if(xx(1)<5.3 && xx(1)>5 && xx(2)<-2.7 && xx(2)>-3.2)

plot(i,j,'.','markers',16,'MarkerEdgeColor',[0 0.6 1]);

end

if(xx(1)<3.5 && xx(1)>3 && xx(2)<-4.8 && xx(2)>-5.2)

plot(i,j,'.','markers',16,'MarkerEdgeColor',[1 0.4 1]);

end

else plot(i,j,'.','markers',16,'MarkerEdgeColor','k');

end

end

end

end

function Z=pavirsius(funk,x,y)

for i=1:length(x), for j=1:length(y)

Z(i,j,1:2)=funk([x(i),y(j)]);

end, end

return

end

function fff=f(x)

fff=[x(2)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10;

x(1)^2+x(2)^2+x(1)\*x(2)-20];

return

end

function xsolvex

function fff=f(x)

fff=[x(2)^2-x(2)^2-5\*x(1)\*cos(x(2)+1)-10;

x(1)^2+x(2)^2+x(1)\*x(2)-20];

return

end

fun = @f;

xx=[-4 ;4 ];

xsolve=fsolve(fun,xx);

xx=[-2 ;5 ];

xsolve2=fsolve(fun,xx);

xx=[-4 ;0 ];

xsolve3=fsolve(fun,xx);

xx=[-2 ;-2 ];

xsolve4=fsolve(fun,xx);

xx=[4 ;-5 ];

xsolve5=fsolve(fun,xx);

xx=[5 ;-3 ];

xsolve6=fsolve(fun,xx);

xx=[4 ;1 ];

xsolve7=fsolve(fun,xx);

xx=[2 ;3 ];

xsolve8=fsolve(fun,xx);

xsolve

xsolve2

xsolve3

xsolve4

xsolve5

xsolve6

xsolve7

xsolve8

end

% Broideno metodas

function Broiden\_metodas2

clc,close all

eps=1e-15

itmax=100000

x=[-5;2;-9;3];

n=length(x);

% Pradines Jakobio matricos reiksmes apskaiciavimas:

dx=sum(abs(x))\*1e-5;

f0=f(x);

for i=1:n, x1=x; x1(i)=x1(i)+dx; f1=f(x1); A(:,i)=(f1-f0)/dx; end

% A=-eye(n)\*10 %\*10 \*(-10)

% Broideno metodo iteracijos:

fi=f(x); % pradine funkcijos reiksme

for iii=1:itmax

deltax=-A\f(x); x=x+deltax; fi1=f(x); A=A+(fi1-fi-A\*deltax)\*deltax'/(deltax'\*deltax);

tikslumas=norm(deltax)/(norm(x)+norm(deltax));

fprintf(1,'\n iteracija %d tikslumas %g',iii,tikslumas);

if tikslumas < eps

x

fprintf(1,'\n sprendinys x ='); fprintf(1,' %g',x);

fprintf(1,'\n funkcijos reiksme f ='); fprintf(1,' %g',f(x));

xx=[-5;2;-9;3];

fun = @f;

xsolve=fsolve(fun,xx);

xsolve

break

elseif iii == itmax

fprintf(1,'\n \*\*\*\*tikslumas nepasiektas. Paskutinis artinys x ='); fprintf(1,' %g',x);

fprintf(1,'\n funkcijos reiksme f ='); fprintf(1,' %g',f(x));

break

end

fi=fi1;

end

return

end

% Lygciu sistemos funkcija

function F=f(X)

F(1)=4\*X(2)+3\*X(3)+3\*X(4)-26;

F(2)=3\*X(2)+4\*X(2)\*X(3)-75;

F(3)=X(3)^3-2\*X(4)^2-25;

F(4)=5\*X(1)-12\*X(2)+40;

F=F(:);

return

end

% Vilius Turenko IFF-4/2

% 21 užduoties variantas

clc; close all; clear all;

%lygtis = '1 sistema';

lygtis = '2 sistema';

if strcmp(lygtis,'1 sistema')

%Skirtinguose grafikuose pavaizduokite paviršius Z1(x1,x2) ir Z2(x1,x2)

sk=1;

grafinis\_metodas(sk)

%Broideno metodu išsprendžiama sistema

sk=4;

xx=[-4 4;-2 5;-4 0;-2 -2;4 -5; 5 -3; 4 1;2 3]

for i=1:8

Broideno\_metodas(sk,xx(i,1),xx(i,2),i)

sk=sk+1;

end;

Tinklelis;

xsolve;

elseif strcmp(lygtis,'2 sistema')

Broiden\_metodas2;

end