# Netiesinių lygčių sistemų sprendimas

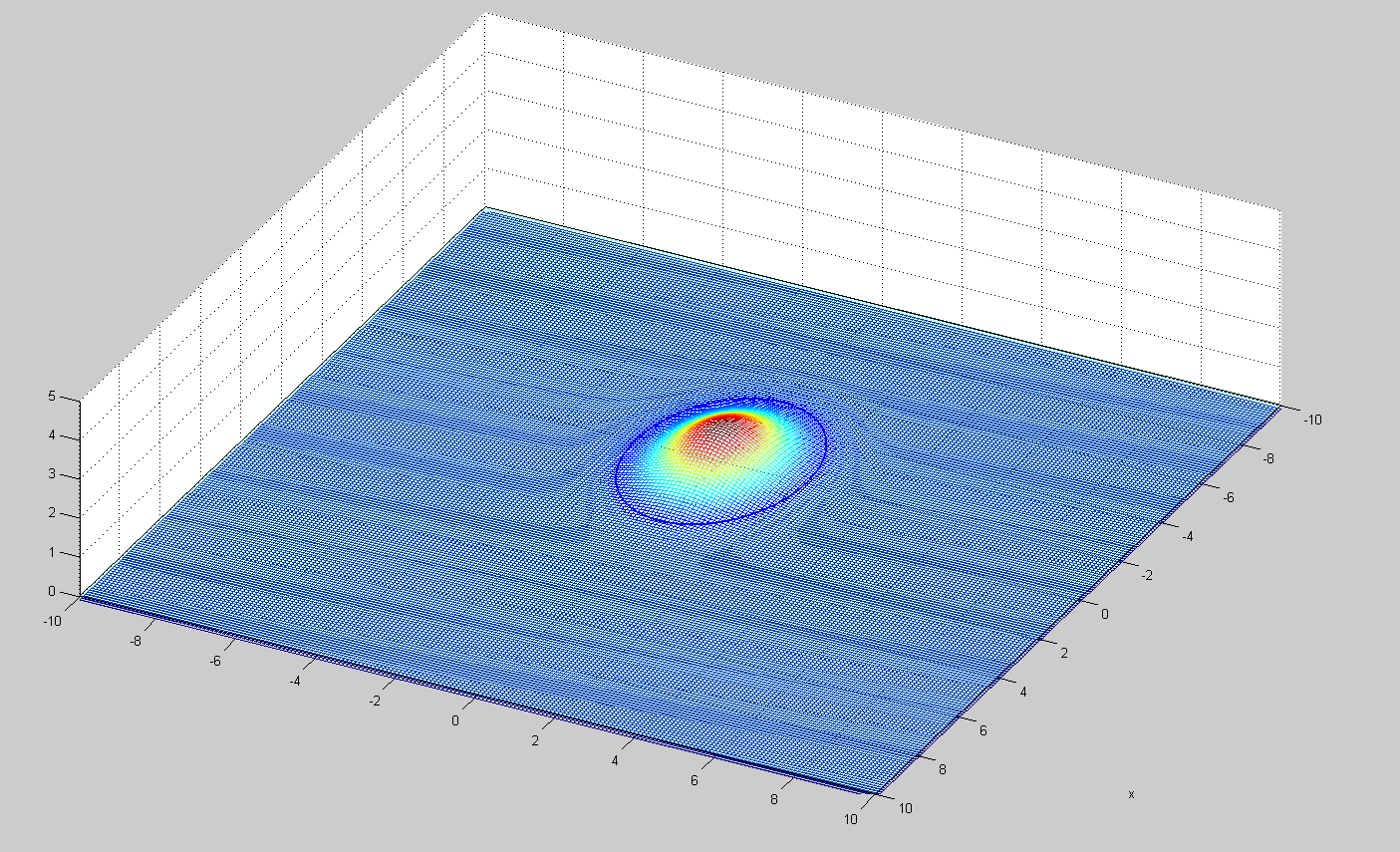
Duotos dvi netiesinės lygčių sistemos:

1 lentelė. Nagrinėjama užduotis

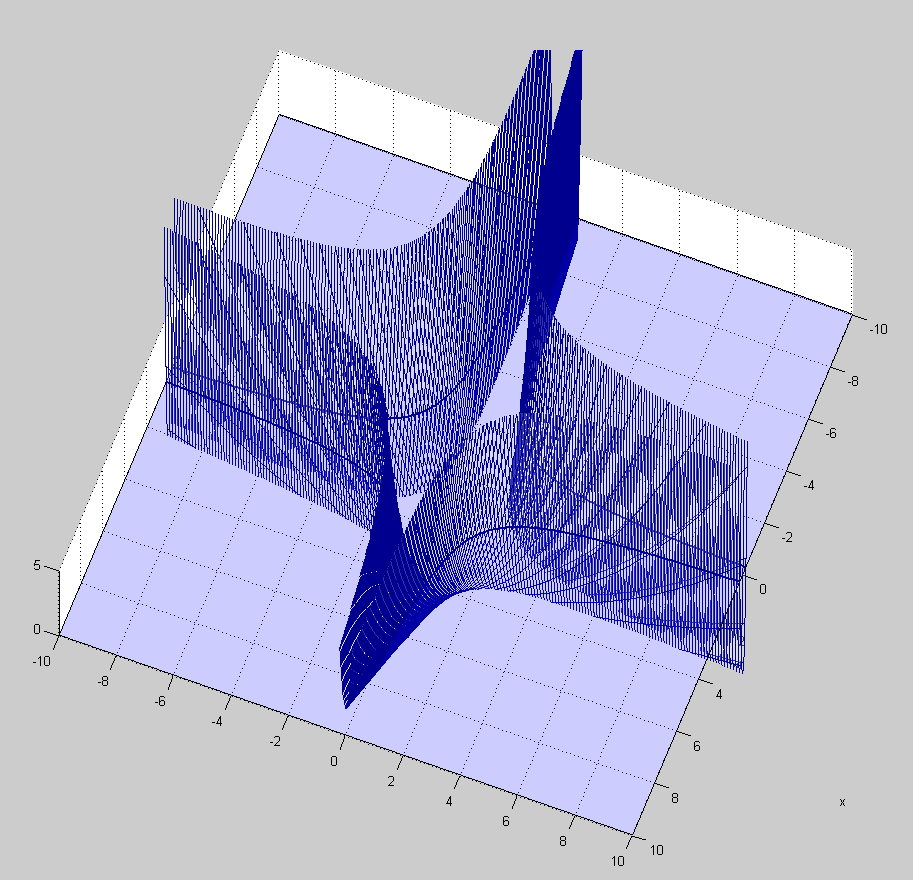
|  |  |  |  |
| --- | --- | --- | --- |
| Varianto Nr. | I lygčių sistema | II lygčių sistema | Metodas |
| 16 |  |  | Niutono |

## I-os netiesinių lygčių sistemos sprendimas

## Paviršių grafinis vaizdas

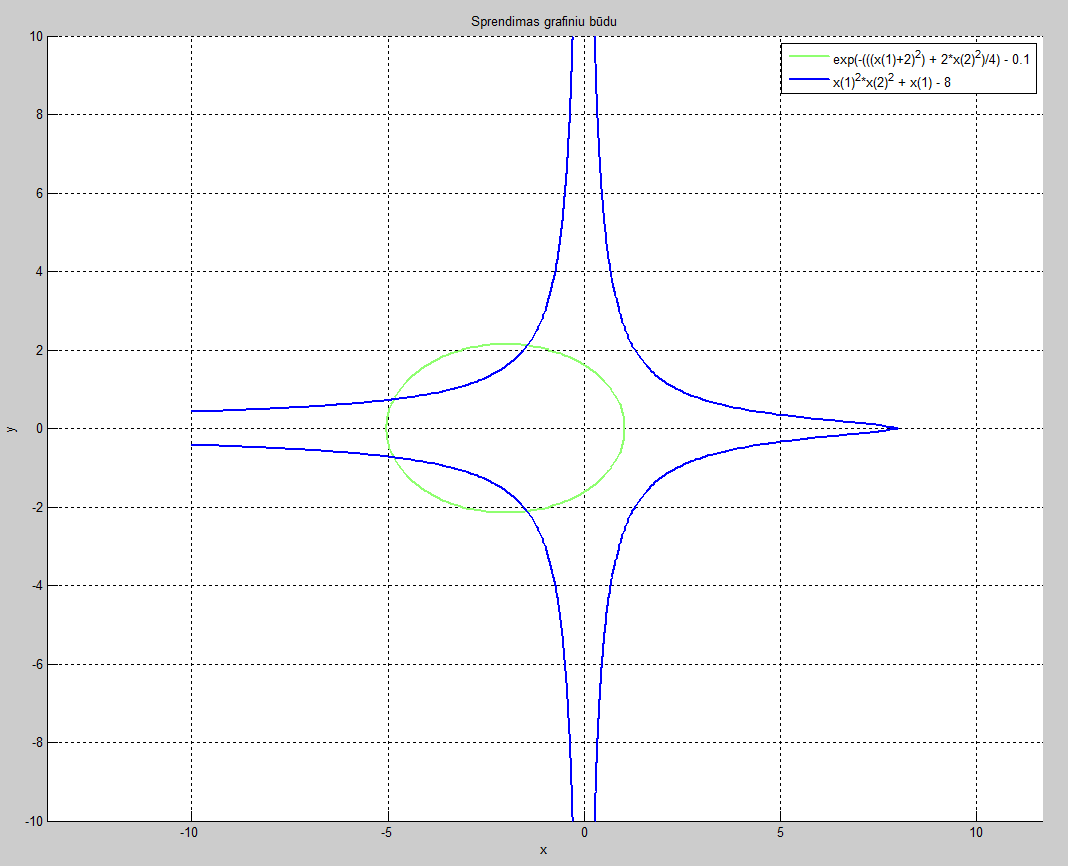


1 pav. Paviršiaus grafinis vaizdas



2 pav. Paviršiaus grafinis vaizdas

## Sprendimas grafiniu būdu



3 pav. Lygčių sistemos sprendiniai nustatomi grafikų susikirtimo taškuose

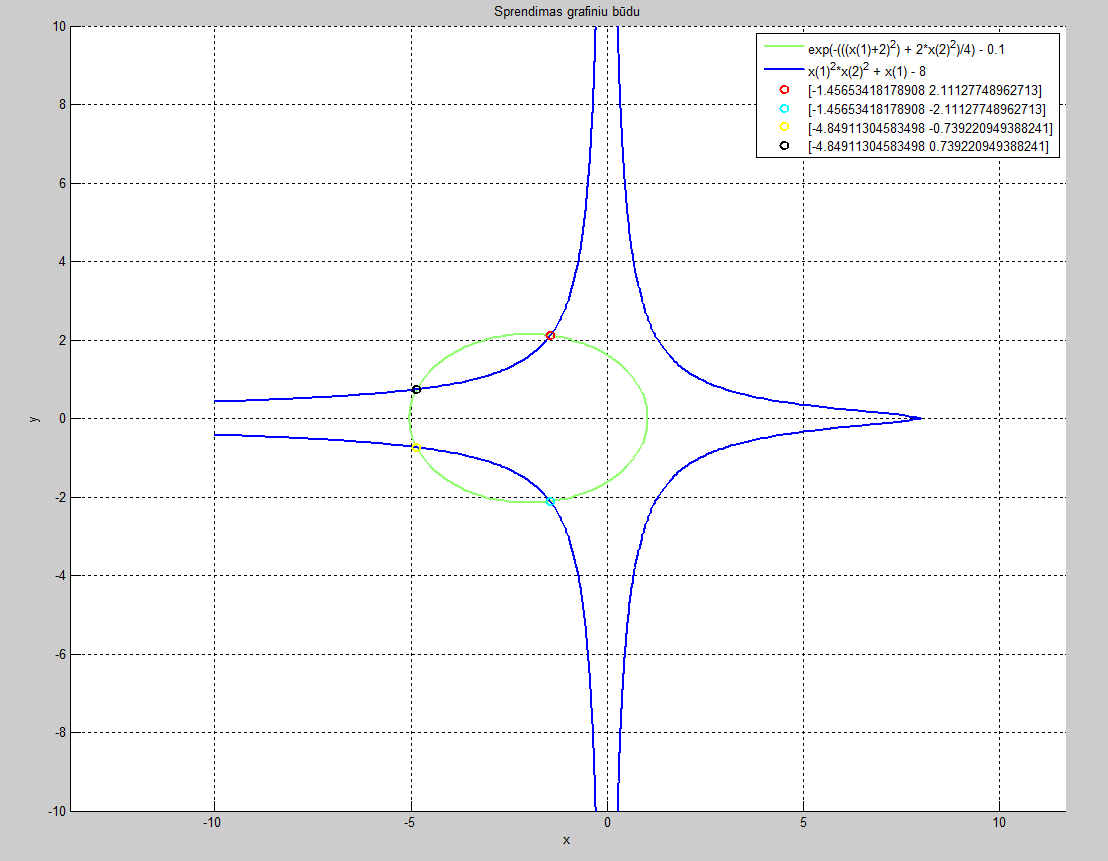
**Išvados:** Iš plokštumų grafinių vaizdų (1 pav., 2 pav.) bei lygčių sistemos sprendimo grafiniu būdu vaizdo (3 pav.) matome, kad plokštumų Z1 bei Z2 kreivės plokštumoje z = 0 susikerta 4 vietose, todėl iš šio vaizdo galima daryti prielaidą, kad ši lygčių sistema galimai turės 4 skirtingas šaknis.

## Sprendimas Niutono metodu

Tariama, kad xg yra šaknis (stabdomi skaičiavimai), jei |f(xg)| < 1e-9. Skaičiavimuose naudojamas šaknies tikslumo įvertis |f(xg)|. Pradinis metodo žingsnis: 0,5.

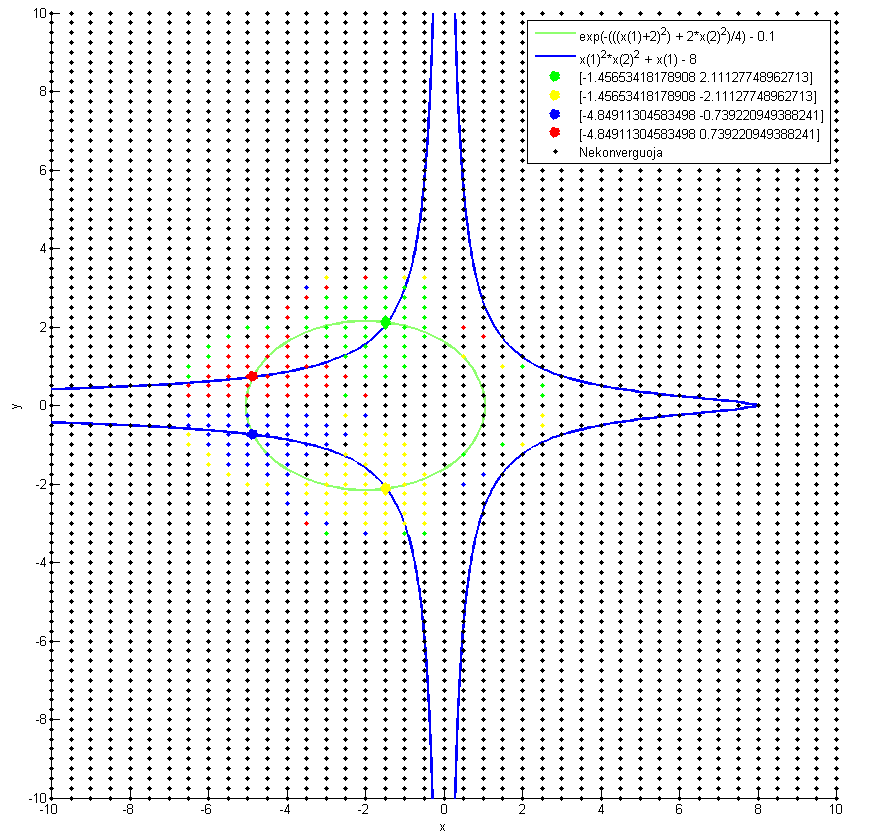
2 lentelė. Rezultatų lentelė

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pradinis artinys | Sprendinys Niutono metodu | Tikslumas | Iteracijų skaičius | Sprendinys MATLAB funkcija fsolve |
| [-2; 2] | [-1.456534181857481  2.111277489618907] | 7.46109e-10 | 34 | [-1.456534181789080; 2.111277489627133] |
| [-2; -2] | [-1.456534181857481  -2.111277489618907] | 7.46109e-10 | 34 | [-1.456534181789080;  -2.111277489627133] |
| [-5; 1] | [-4.849113045835488  0.739220949412258] | 8.36911e-10 | 35 | [-4.849113045834978; 0.739220949388241] |
| [-5; -1] | [-4.849113045835488  -0.739220949412258] | 8.36911e-10 | 35 | [-4.849113045834978;  -0.739220949388241] |
| [0; 0] | Nekonverguoja |  |  | Nerastas sprendinys |

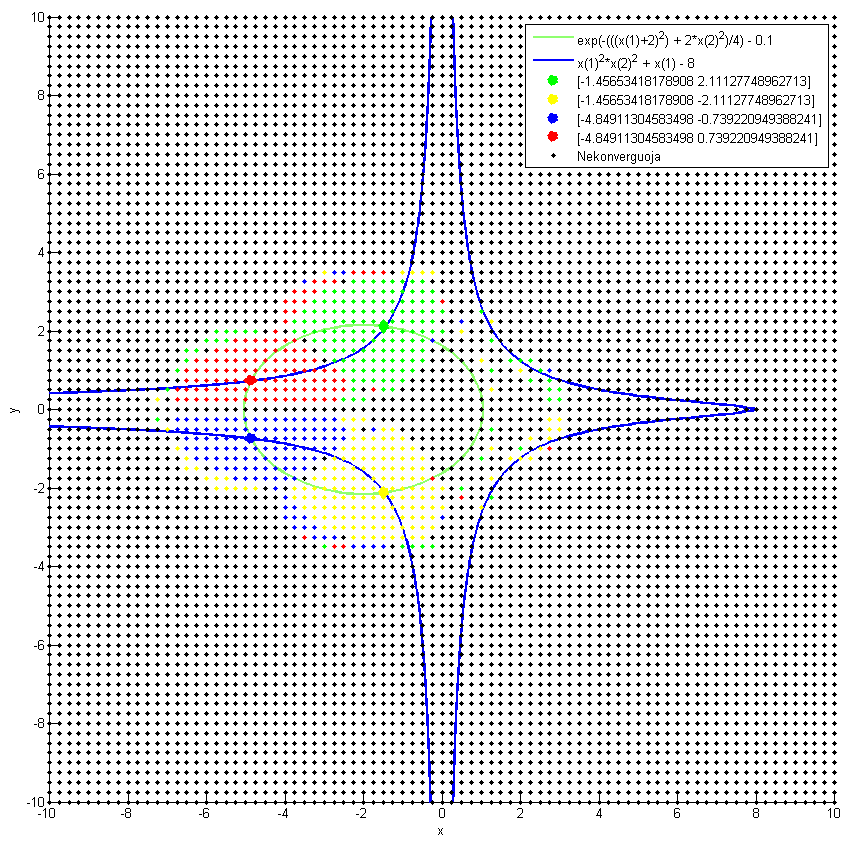


4 pav. Rasti sistemos sprendiniai

**Išvados:** Iš rezultatų lentelės (2 lentelė) matome, jog Niutono metodu rastos tokios šaknys, kokios buvo matomos ir grafiniame lygties sprendimo variante. Taip pat galime pastebėti, kad metodo rastų šaknų reikšmės yra labai panašios į tas, kurios buvo rastos naudojant MATLAB funkciją fsolve, todėl galime teigti, kad metodas veikia tikrai gerai ir teisingai. Taip pat iš rezultatų matome ir tai, jog su kai kuriais pradiniais artiniais jie nekonverguoja ir nerandama jokia šaknis.



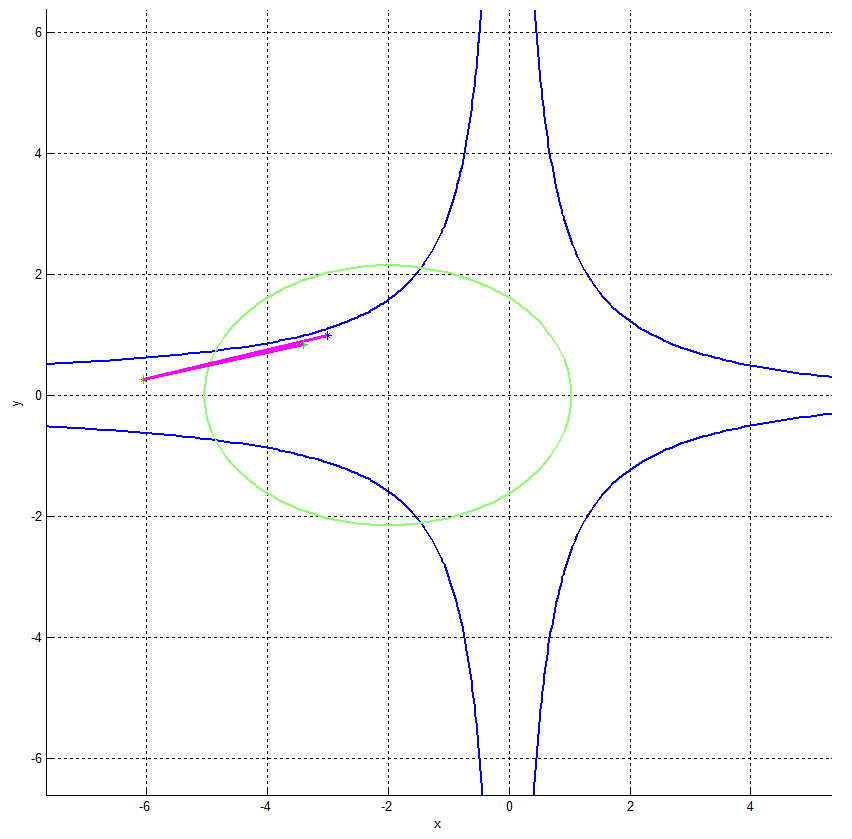
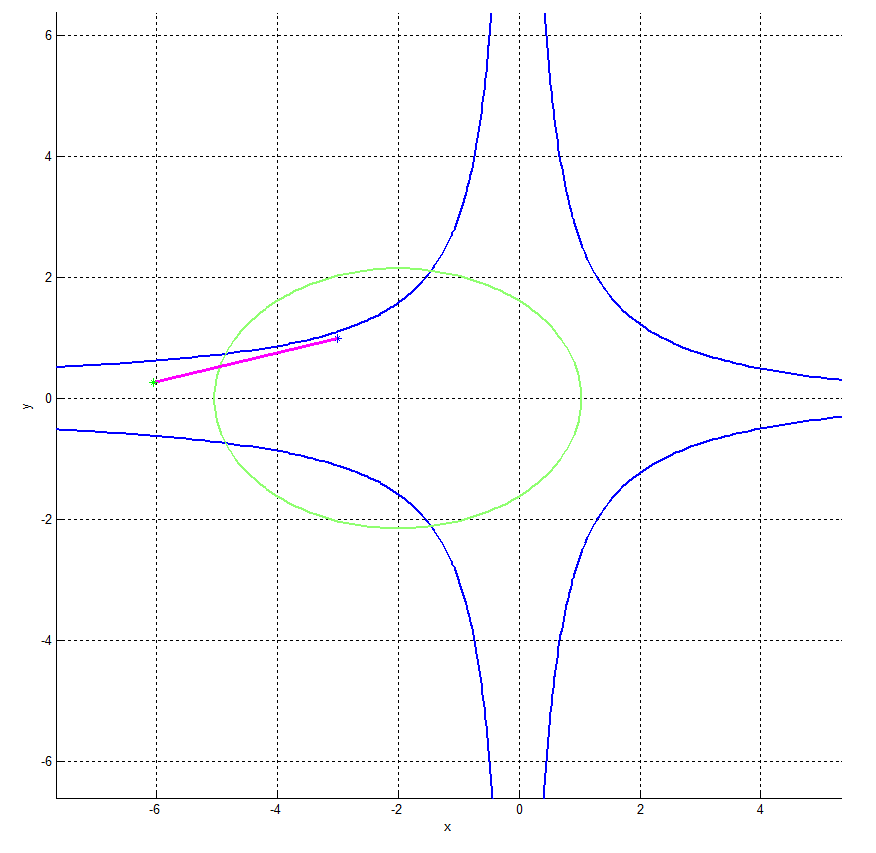
5 pav. Pradinių artinių tinklelis, kai Niutono metodo žingsnis 0,5. Ta pačia spalva žymimi pradiniai artiniai, nuo kurių pradėjus skaičiuoti gaunami artimi sprendiniai



6 pav. Pradinių artinių tinklelis, kai Niutono metodo žingsnis 0,25. Ta pačia spalva žymimi pradiniai artiniai, nuo kurių pradėjus skaičiuoti gaunami artimi sprendiniai

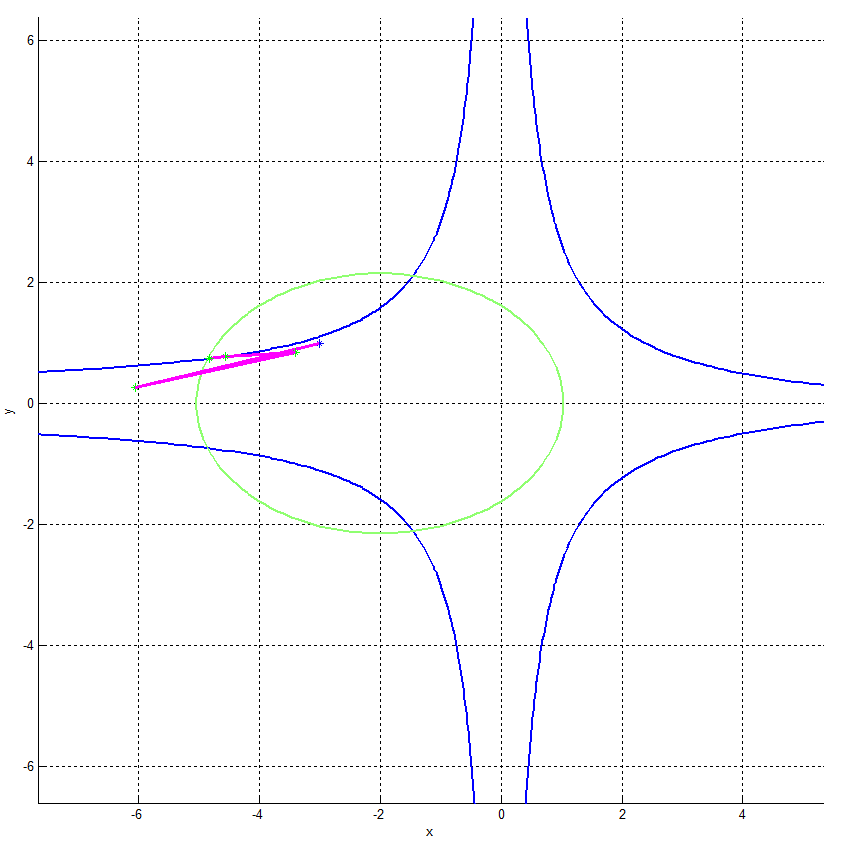
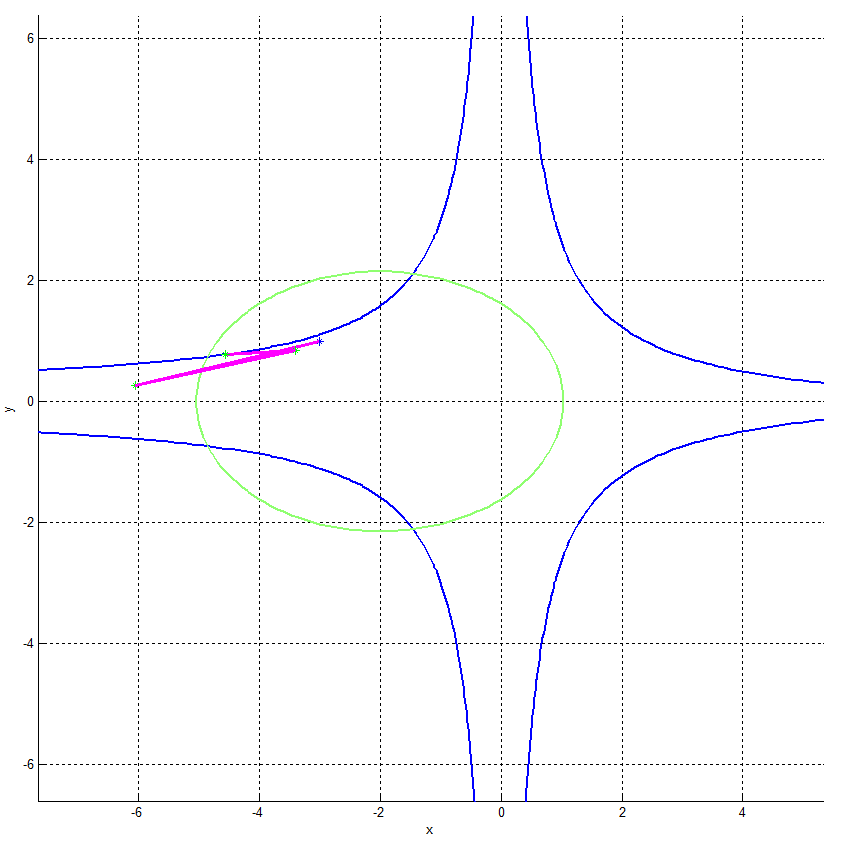
2

1

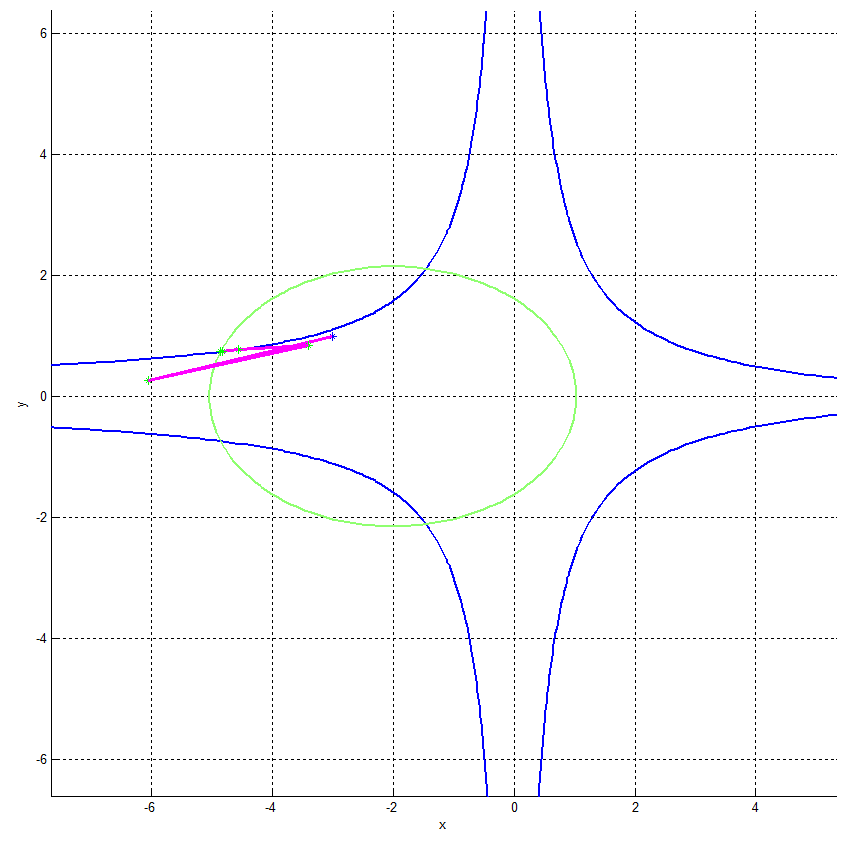


4

3



5



7 pav. Niutono metodo vizualizacija nuo pradinio artinio [-3; 1]

**Išvados:** Iš artinių vizualizacijų matome, kad Niutono metodo veikimas labai priklauso nuo pasirinkto pradinio artinio bei žingsnio dydžiu (5 pav., 6 pav.). Iš metodo veikimo vizualizacijos (7 pav.) galima pastebėti, jog parinkus teisingą ir gerą pradinį artinį, jis prie tikrosios šaknies reikšmės priartėja gan sparčiai tiksliai.

## II-os netiesinių lygčių sistemos sprendimas

Tariama, kad xg yra šaknis (stabdomi skaičiavimai), jei |f(xg)| < 1e-10. Skaičiavimuose naudojamas šaknies tikslumo įvertis |f(xg)|. Pradini metodo žingsnis: 0,5.

3 lentelė. Rezultatų lentelė

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pradinis artinys | Sprendinys Niutono metodu | Tikslumas | Iteracijų skaičius | Sprendinys MATLAB funkcija fsolve |
| [-10, -8, -8, -5] | [3.999999999999785 0.999999999999785 -3.000000000000175 0.999999999999955] | 5.073454258585653e-12 | 26 | [4.000000000017332  1.000000000017208  -2.999999999949488  1.000000000028291] |

**Išvados:** Kaip matome, Niutono metodas gali išspręsti lygčių sistemą, turinčią ir daugiau nei 2 netiesines lygtis. Taip pat galime pastebėti, kad ir šįkart metodo tikslumas nenuvylė – apskaičiuotos šaknų reikšmės (sprendiniai) beveik sutampa su MATLAB funkcijos fsolve apskaičiuotais sprendiniais. Tačiau metodui reikia parinkti tinkamą pradinį artinį, kitaip reikšmės surastos nebus.

## Išvados

Išnagrinėjus Niutono metodą netiesinių lygčių sistemoms galima teigti, kad metodas veikia teisingai, nepriklausomai nuo lygčių skaičiaus sistemoje. Rezultatai bei Niutono metodu gautų sprendinių palyginimas su MATLAB funkcijos fsolve gautais sprendiniais parodė, kad metodas veikia tikrai tiksliai, sprendinių reikšmės yra labai panašios, besiskiriančios labai maža skaičiaus dalimi. Tačiau reikia pabrėžti tai, kad metodo veikimas priklauso nuo parinkto žingsnio bei nuo pradinių artinių, kuriuos parinkus blogai, metodo tikslinama artinio reikšmė nekonverguos.

## Programų tekstai

Grafinis sprendimas:

function Grafinis\_sprendimas

clc, close all, clear all;

x=[-10:0.1:10];

y=[-10:0.1:10];

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

figure(1),hold on,grid on,axis equal,axis([min(x) max(x) min(y) max(y) 0 5]);view([1 1 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(2),hold on,grid on,axis equal,axis([min(x) max(x) min(y) max(y) 0 5]);view([1 1 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(3),hold on,grid on,axis equal

title('Sprendimas grafiniu būdu');

contour(x,y,Z(:,:,1)',[0 0],'LineWidth',1.5, 'DisplayName', 'exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1')

contour(x,y,Z(:,:,2)',[0 0],'b-', 'LineWidth',1.5, 'DisplayName', 'x(1)^2\*x(2)^2 + x(1) - 8')

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

% Šaknų paieška naudojant fsolve

artiniai=[-2 2; -2 -2; -5 -1; -5 1];

zymekliai=['ro'; 'co'; 'yo'; 'ko'];

options=optimset('Jacobian', 'on');

[rows, columns] = size(artiniai);

for i=1:rows

[root]=fsolve(@fjx, [artiniai(i,1) artiniai(i, 2)], options)

plot(root(1), root(2), zymekliai(i, :), 'LineWidth', 2, 'DisplayName',mat2str(root));

end

legend('show');

return

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1;

x(1)^2\*x(2)^2 + x(1) - 8];

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

% Funkcija, skirta šaknų suradimui su fsolve

function [F, J]=fjx(x)

F=[exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1;...

x(1)^2\*x(2)^2 + x(1) - 8];

if nargout > 1

J=[-exp(-(x(1)+ 2)^2/4-x(2)^2/2)\*(x(1)/2+1), ...

-x(2)\*exp(-(x(1)+2)^2/4-x(2)^2/2);...

2\*x(1)\*x(2)^2+1, 2\*x(1)^2\*x(2) ];

end

end

Netiesinių lygčių sistemos sprendimas (su vizualizacija):

function Niutonas\_netiesine\_sistema

clc,close all;

format long;

scrsz = get(0,'ScreenSize');

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

% Pačiame sprendime nanaudojami dalykai, bet reikalinga, norint

% susidaryti Jakobio matricą, kai ją apskaičiuoja MATLAB

syms x y

f1 = exp(-(((x+2)^2) + 2\*y^2)/4) - 0.1;

f2 = x^2\*y^2 + x - 8;

Jakobio\_Matrica=[diff(f1, 'x'), diff(f1, 'y'); diff(f2, 'x'), diff(f2, 'y')]

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

x=[-10:0.1:10];y=[-10:0.1:10];

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

% [left, bottom, width, height]

fig1=figure(1);set(fig1,'Position',[50 scrsz(4)/1.8 scrsz(3)/3 scrsz(4)/3],'Color','w');

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);

contour(x,y,Z(:,:,1)',[0 0],'LineWidth',1.5, 'DisplayName', 'exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1')

contour(x,y,Z(:,:,2)',[0 0],'b-', 'LineWidth',1.5, 'DisplayName', 'x(1)^2\*x(2)^2 + x(1) - 8')

eps=1e-9;

itmax=100;

x=[-2; 2] % pradinis artinys

ff=f(x); dff=df(x);

figure(1);plot3(x(1),x(2),0,'b\*');line([x(1),x(1)],[x(2),x(2)],[0,ff(1)],'Color','black');

alpha=0.5 %0.9 %0.8; % 0.5 % zingsnio sumazinimo koeficientas

for iii=1:itmax

dff=df(x);

deltax=-dff\ff;

x1=x+alpha\*deltax;

ff1=f(x1);

figure(1);plot3(x1(1),x1(2),0,'g\*');

line([x(1),x1(1)],[x(2),x1(2)],[0,0],'Color','magenta');

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

line([x1(1),x1(1)],[x1(2),x1(2)],[0,ff1(1)],'Color','black');

tikslumas=norm(f(x));

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,'gp'); break;

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

end

x=x1;

ff=ff1;

end

x

fprintf(1,'\n');

return

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1;

x(1)^2\*x(2)^2 + x(1) - 8];

return

end

% Jakobio matrica

function dfff=df(x)

dfff=[-exp(- (x(1) + 2)^2/4 - x(2)^2/2)\*(x(1)/2 + 1), ...

-x(2)\*exp(- (x(1) + 2)^2/4 - x(2)^2/2);

2\*x(1)\*x(2)^2 + 1, 2\*x(1)^2\*x(2)];

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

Artinių vizualizacija:

function Niutonas\_artiniai\_tinklelis

clc,close all

scrsz = get(0,'ScreenSize');

itmax=100;

alpha=0.25; %0.9 %0.8; % 0.5 % zingsnio sumazinimo koeficientas

warning off;

artinys=[-10:0.1:10];y=[-10:0.1:10];

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

fig1=figure(1);set(fig1,'Position',[50 scrsz(4)/1.8 scrsz(3)/3 scrsz(4)/3],'Color','w');

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

contour(artinys,y,Z(:,:,1)',[0 0],'LineWidth',1.5, 'DisplayName', 'exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1')

contour(artinys,y,Z(:,:,2)',[0 0],'b-', 'LineWidth',1.5, 'DisplayName', 'x(1)^2\*x(2)^2 + x(1) - 8')

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

% Šaknų paieška naudojant fsolve (reikalinga legendai sudaryti)

artiniai=[-2 2; -2 -2; -5 -1; -5 1];

zymekliai=['gp'; 'yp'; 'bp'; 'rp'];

options=optimset('Jacobian', 'on');

[rows, columns] = size(artiniai);

for i=1:rows

[root]=fsolve(@fjx, [artiniai(i,1) artiniai(i, 2)], options);

plot(root(1), root(2), zymekliai(i, :), 'LineWidth', 4, 'DisplayName',mat2str(root));

end

% Piešiama už plokštumos atvaizdavimo ribų, kad nesimatytų

plot(-15,-15,'k.', 'DisplayName', 'Nekonverguoja');

legend('show');

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

eps=1e-9; %tikslumas

for x=-10:0.25:10

for y=-10:0.25:10

artinys=[x;y];

ff=f(artinys);

dff=df(artinys);

plot(x,y,'k.', 'MarkerSize',10)

for iii=1:itmax

dff=df(artinys);

deltax=-dff\ff;

x1=artinys+alpha\*deltax;

ff1=f(x1);

tikslumas = norm(f(artinys));

if (tikslumas < eps)

if (abs(artinys(1)) < 2)

if (artinys(2) > 0)

plot(x,y,'g.', 'MarkerSize',10)

else

plot(x,y,'y.', 'MarkerSize',10)

end

else

if (artinys(2) > 0)

plot(x,y,'r.', 'MarkerSize',10)

else

plot(x,y,'b.', 'MarkerSize',10)

end

end

break;

end

artinys=x1;

ff=ff1;

end

end

end

end

% Lygciu sistemos funkcija

function fff=f(x)

fff=[exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1;

x(1)^2\*x(2)^2 + x(1) - 8];

return

end

% Jakobio matrica

function dfff=df(x)

dfff=[-exp(- (x(1) + 2)^2/4 - x(2)^2/2)\*(x(1)/2 + 1), ...

-x(2)\*exp(- (x(1) + 2)^2/4 - x(2)^2/2);

2\*x(1)\*x(2)^2 + 1, 2\*x(1)^2\*x(2)];

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

% Funkcija, skirta šaknų suradimui su fsolve

function [F, J]=fjx(x)

F=[exp(-(((x(1)+2)^2) + 2\*x(2)^2)/4) - 0.1;...

x(1)^2\*x(2)^2 + x(1) - 8];

if nargout > 1

J=[-exp(-(x(1)+ 2)^2/4-x(2)^2/2)\*(x(1)/2+1), ...

-x(2)\*exp(-(x(1)+2)^2/4-x(2)^2/2);...

2\*x(1)\*x(2)^2+1, 2\*x(1)^2\*x(2) ];

end

end

Netiesinių lygčių sistemos su 4 lygtimis sprendimas:

% Niutono metodas su simboliniu diferencijavimu

function Niutonas\_4\_lygtys

clc,close all

syms x1 x2 x3 x4

format long;

X=[x1; x2; x3; x4];

F(1)=X(2)-2\*X(3)+3\*X(4)-10;

F(2)=3\*X(1)\*X(3)-X(1)+40;

F(3)=2\*X(2)^3-X(2)^2-4\*X(3)^2+35;

F(4)=3\*X(1)-3\*X(2)-9;

eps=1e-10

itmax=100

x=[-10;-8;-8;-5];

fun = @f;

fsolved = fsolve(fun,x)

F=F(:);

DF=jacobian(F,X);

for iii=1:itmax

deltax=-eval(subs(DF,X,x))\eval(subs(F,X,x));

x=x+deltax;

tikslumas = norm(f(x));

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

if tikslumas < eps

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

fprintf(1,' %g',eval(subs(F,X,x)));

iii

tikslumas

break

elseif iii == itmax

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

% fprintf(1,'\n funkcijos reiksmes f ='); fprintf(1,' %g',eval(subs(F,X,x)));

break

end

end

x

fprintf(1,'\n');

return

end

function Ff=f(X)

Ff(1)=X(2)-2\*X(3)+3\*X(4)-10;

Ff(2)=3\*X(1)\*X(3)-X(1)+40;

Ff(3)=2\*X(2)^3-X(2)^2-4\*X(3)^2+35;

Ff(4)=3\*X(1)-3\*X(2)-9;

return

end