Netiesinių lygčių sistemų sprendimas

Temoje aiškinama:

- **NLS matematinė formuluotė ir grafinis sprendimo būdas;**
- **Paprastųjų iteracijų algoritmas NLS sprendimui;**
- **Niutono metodas NLS sprendimui. Niutono-Rafsono metodas;**
- **Kvazi-Niutono metodai. Broideno metodas;**

NLS matematinė formuluotė ir grafinis sprendimo būdas

Netiesinių algebrinių lygčių sprendimas. Matematinė formuluotė

$$f(x) = 0$$

Vieno kintamojo lygtis (skaliarinė skaliarinio argumento funkcija)

$$\mathbf{f}(\mathbf{x}) = \begin{Bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}$$

„Bold“ šriftas reiškia vektorių arba matricą

Lygčių sistema su daugeliu kintamųjų (vektorinė vektorinio argumento funkcija)

$$\mathbf{f}\left(\mathbf{x}\right)=\begin{Bmatrix} f_1\left(x_1,x_2,\ldots,x_n\right) \\ f_2\left(x_1,x_2,\ldots,x_n\right) \\ \vdots \\ f_n\left(x_1,x_2,\ldots,x_n\right) \end{Bmatrix}=\begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}$$

pavyzdys

$$\mathbf{f}\left(\mathbf{x}\right)=\begin{Bmatrix} {x_1}^2+{x_2}^2-2 \\ {x_1}^2-{x_2}^2 \end{Bmatrix}=\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

Netiesinių algebrinių lygčių sistemų sprendimas. Pradinio artinio nustatymas

- **Daugiamatėje kintamųjų ir funkcijų erdvėje universalių metodų pradiniam artiniui rasti nėra;**
- **Atskiroms lygčių klasėms dažnai pavyksta parinkti neblogą pradinį priartėjimą remiantis išankstinėmis žiniomis apie nagrinėjamą objektą;**
- **Dviejų kintamųjų atveju lygčių sistemai ištirti galima panaudoti funkcijų grafinį vaizdavimą**

Grafinis metodas 2 lygčių sistemai MATLAB

```
x=[-5:0.5:5];
y=[-6:0.5:6];

for i=1:length(x), for j=1:length(y)
    Z(i,j,1:2)=f ([x(i),y(j)]);
end, end

figure(1), hold on
mesh(x,y,Z(:,:,1);
contour(x,y,Z(:,:,1)',[0 0]);

figure(2), hold on
mesh(x,y,Z(:,:,2)‘);
contour(x,y,Z(:,:,2)',[0 0]);

figure(3), hold on
contour(x,y,Z(:,:,1)’,[0 0]);
contour(x,y,Z(:,:,2)’,[0 0]);
end
```

Lygčių sistemos funkcijų reikšmės

1 funkcijos paviršius

1 funkcijos nulio reikšmių linija

2 funkcijos paviršius

2 funkcijos nulio reikšmių linija

Abiejų nulio reikšmių linijų susikirtimai yra sprendinio taškai

```
function fff=f(x)  %   Lygciu sistemos funkcija
       fff=[x(1)^2+x(2)^2-2;
            x(1)^2-x(2)^2];
    return
    end
```

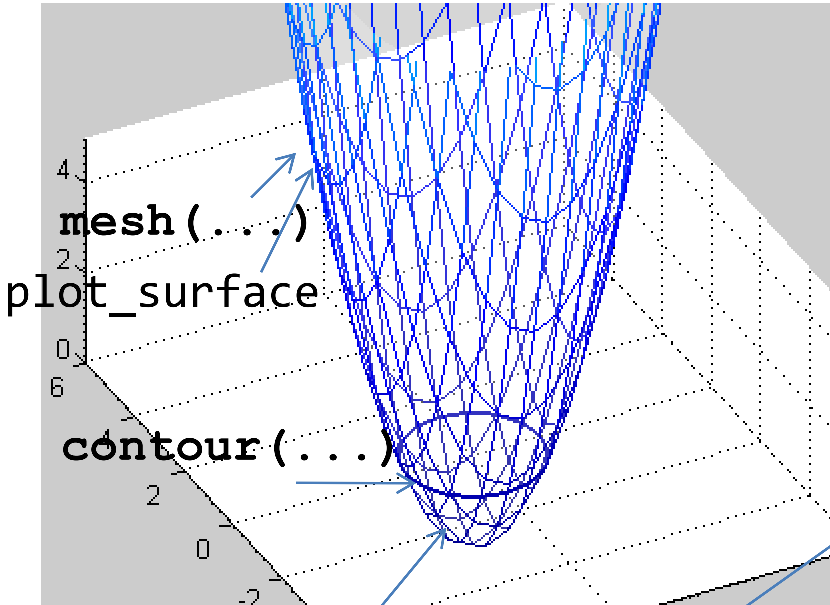
Grafinis metodas 2 lygčių sistemai Python

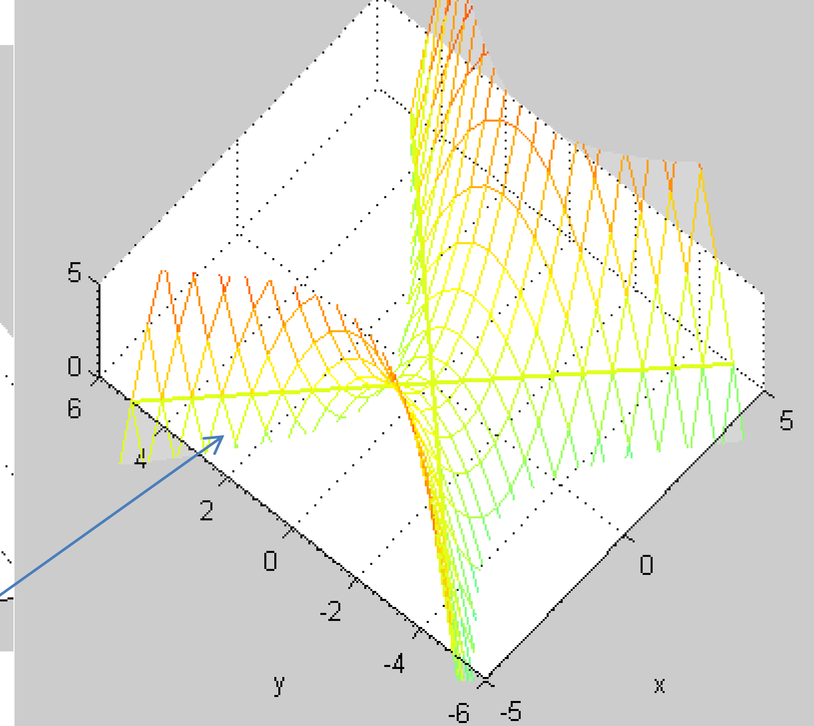
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import math

def LF(x):  #-------- Lygciu sistemos funkcija---------
    s=np.matrix( [[x[0]**2+x[1]**2-2], [x[0]**2-x[1]**2]])
    return s
#-----------------------------------
fig1=plt.figure(1,figsize=plt.figaspect(0.5));
ax1 = fig1.add_subplot(1, 2, 1, projection='3d'); ax2 = fig1.add_subplot(1, 2, 2, projection='3d');
plt.draw();

xx=np.linspace(-5,5,20);yy=np.linspace(-6,6,20);
X, Y = np.meshgrid(xx, yy);
Z=np.zeros(shape=(len(xx),len(yy),2))
for i in range (0,len(xx)):
    for j in range (0,len(yy)):  Z[i,j,:]=LF([X[i][j],Y[i][j]]).transpose();

surf1 = ax1.plot_surface(X, Y, Z[:,:,0], color='blue', alpha=0.4)
CS11 =  ax1.contour(X, Y, Z[:,:,0],[0],colors='b')

surf2 = ax1.plot_surface(X, Y, Z[:,:,1], color='purple',alpha=0.4)
CS12 =  ax1.contour(X, Y, Z[:,:,1],[0],colors='g')

CS1 =   ax2.contour(X, Y, Z[:,:,0],[0],colors='b')
CS2 =   ax2.contour(X, Y, Z[:,:,1],[0],colors='g')

plt.show()
```

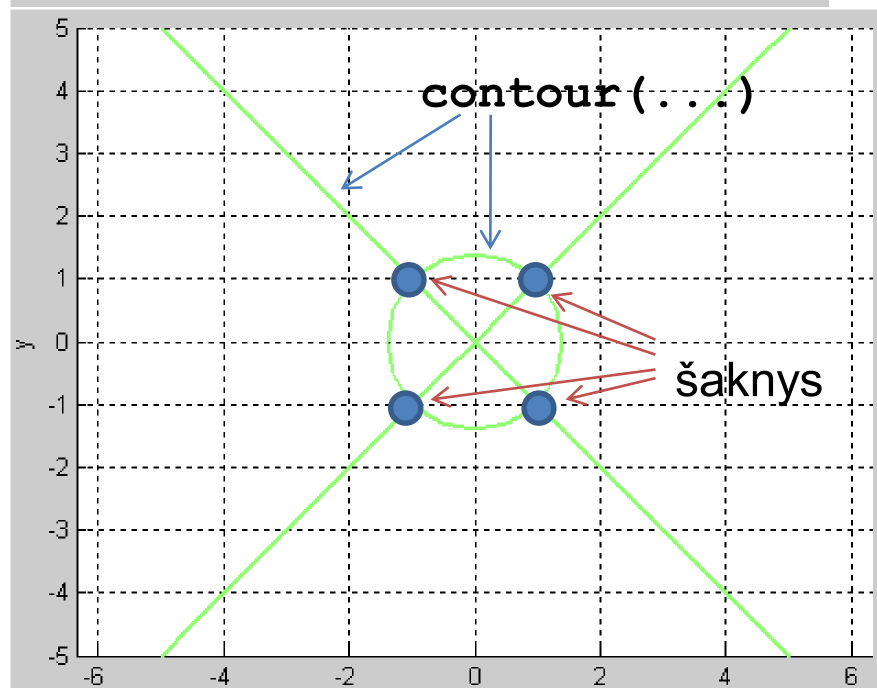
1 funkcijos paviršius ir nulio reikšmių linija

2 funkcijos paviršius ir nulio reikšmių linija

Abiejų nulio reikšmių linijų susikirtimai yra sprendinio taškai







$$\begin{cases} f_1(x_1, x_2) = x_1^{\;2} + x_2^{\;2} - 2 = 0; \\ f_2(x_1, x_2) = x_1^{\;2} - x_2^{\;2} = 0 \end{cases}$$

Pvz_5_1_Grafinis_metodas_lygciu_sistemai.m

Paprastųjų iteracijų algoritmas NLS sprendimui

Netiesinių algebrinių lygčių sprendimas.
Paprastųjų iteracijų metodas (1)

$$\mathbf{f}\left(\mathbf{x}\right)=0 \quad \Rightarrow \quad \mathbf{x}=\mathbf{x}+\left[\boldsymbol{\alpha}\right]^{-1}\mathbf{f}\left(\mathbf{x}\right)$$

$$\left[\boldsymbol{\alpha}\right]\mathbf{x}=\mathbf{f}(\mathbf{x})+\left[\boldsymbol{\alpha}\right]\mathbf{x}$$

Gali būti įstrižaininė matrica

$\mathbf{x}^0$ - pradinis artinys

$$\boxed{\mathbf{x}^{i+1}=\mathbf{x}^{(i)}+\left[\boldsymbol{\alpha}\right]^{-1}\mathbf{f}\left(\mathbf{x}^{(i)}\right), \quad i=0,1,2,3,\ldots}$$

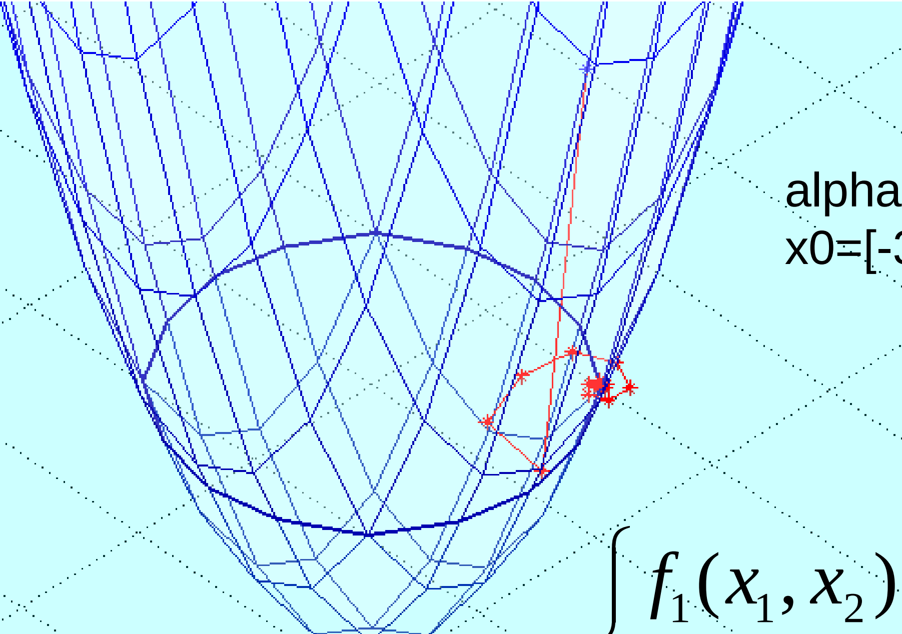
Netiesinių algebrinių lygčių sprendimas.
Paprastųjų iteracijų metodas (2)

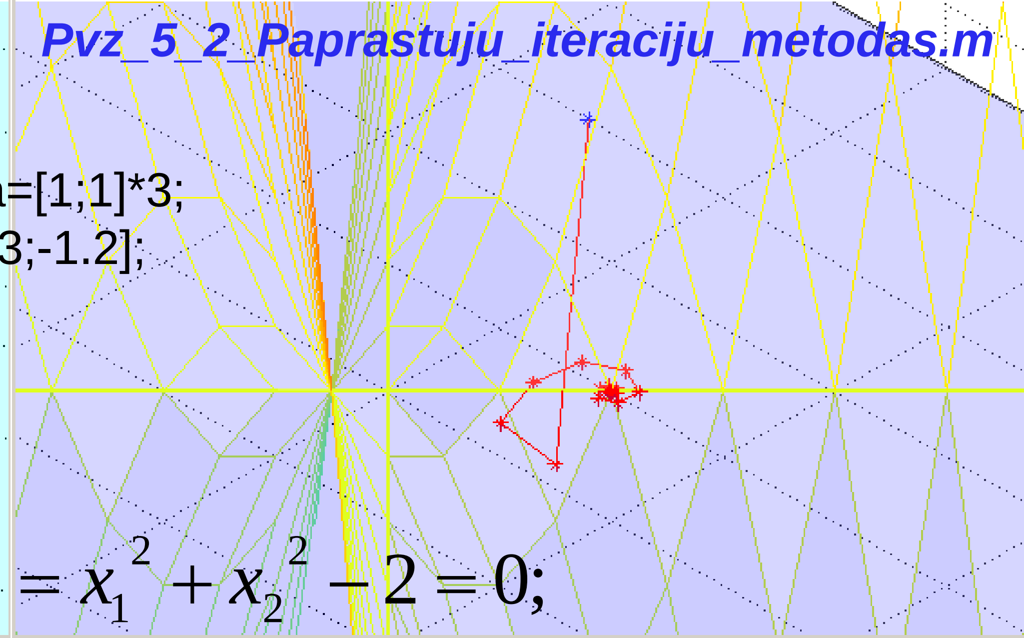
$$\mathbf{x}^{i+1} = \mathbf{x}^{(i)} + [\boldsymbol{\alpha}]^{-1}\mathbf{f}\left(\mathbf{x}^{(i)}\right), \quad i = 0,1,2,3,\ldots$$

$$[\boldsymbol{\alpha}] = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_n \end{bmatrix} \Rightarrow [\boldsymbol{\alpha}]^{-1} = \begin{bmatrix} 1/\alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & 1/\alpha_2 & 0 & \cdots & 0 \\ 0 & 0 & 1/\alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1/\alpha_n \end{bmatrix}$$

- Įstrižaininės matricos inversija gaunama, invertuojant įstrižainės elementus;
- Matricos alpha inversija ir daugyba iš matricos paprastųjų iteracijų formulės dešinėje pusėje reiškia, kad kiekvienas funkcijos vektoriaus elementas dalijamas iš atitinkamos alpha reikšmės

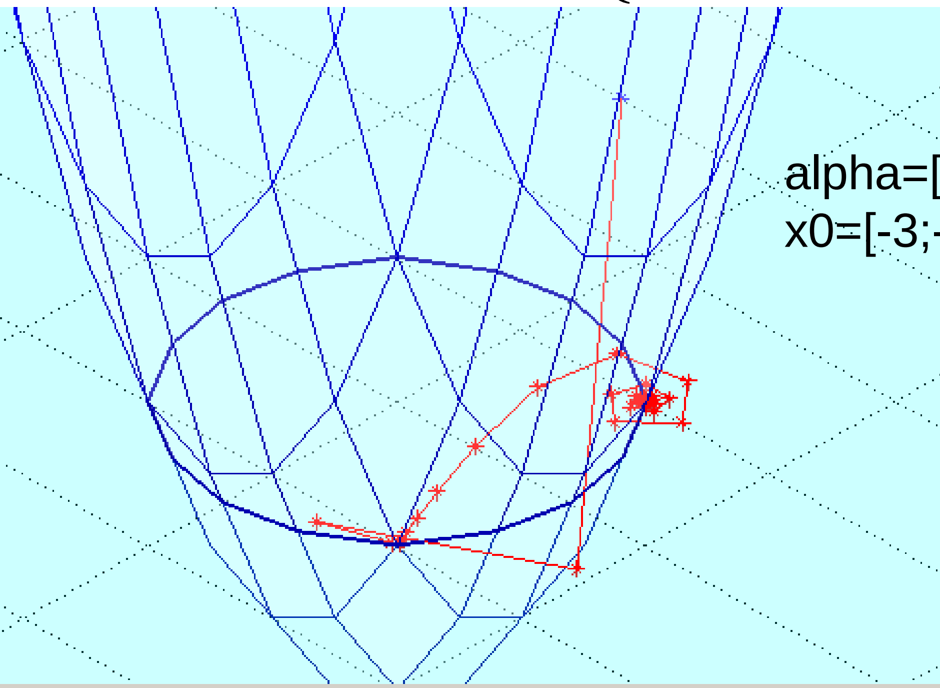
Pvz_5_2_Paprastuju_iteraciju_metodas.m

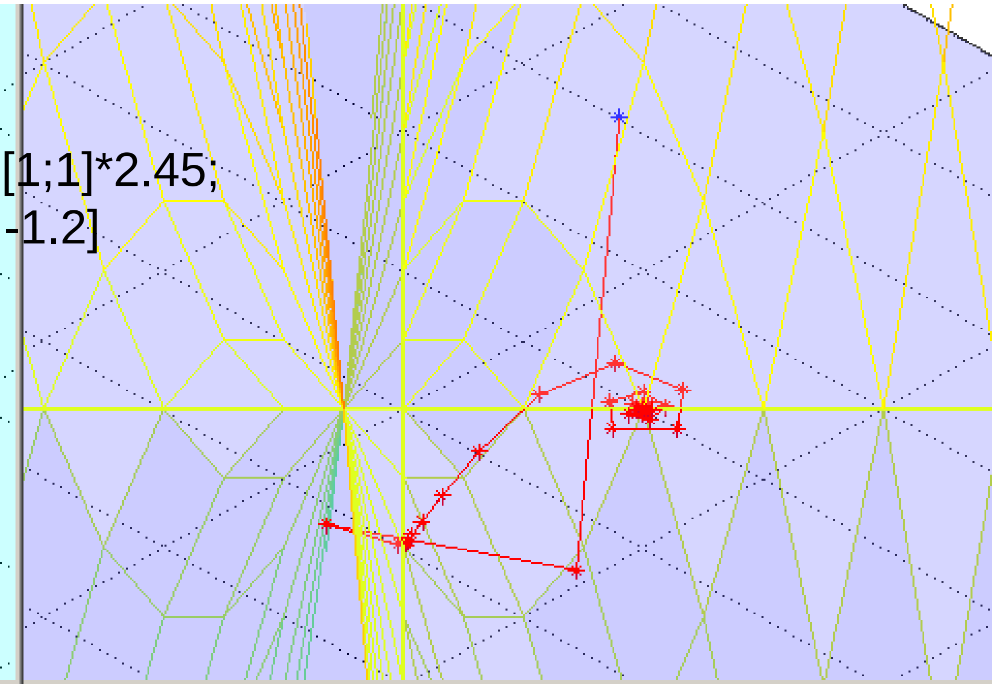




alpha=[1;1]*3;
x0=[-3;-1.2];

$$\begin{cases} f_1(x_1, x_2) = x_1^{\,2} + x_2^{\,2} - 2 = 0; \\ f_2(x_1, x_2) = x_1^{\,2} - x_2^{\,2} = 0 \end{cases}$$





alpha=[1;1]*2.45;
x0=[-3;-1.2]

Niutono metodas NLS sprendimui. Niutono-Rafsono metodas

Netiesinių algebrinių lygčių sprendimas.

Atkirstoji Teiloro eilutė vektorinio argumento vektorinės funkcijos atveju

$$\mathbf{f}(\mathbf{x}) = 0 \ , \qquad \mathbf{x}^{(0)}$$ - pradinis artinys

$$f_k(\mathbf{x}+\Delta\mathbf{x}) \approx f_k(\mathbf{x}) + \Delta x_1 \left.\frac{\partial f_k}{\partial x_1}\right|_{\mathbf{x}} + \Delta x_2 \left.\frac{\partial f_k}{\partial x_2}\right|_{\mathbf{x}} + \ldots + \Delta x_n \left.\frac{\partial f_k}{\partial x_n}\right|_{\mathbf{x}}, \qquad k = 1:n$$

Atkirsta Teiloro eilutė

$$\begin{Bmatrix} f_1(\mathbf{x}+\Delta\mathbf{x}) \\ f_2(\mathbf{x}+\Delta\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}+\Delta\mathbf{x}) \end{Bmatrix} = \begin{Bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{Bmatrix} + \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}\right|_{\mathbf{x}} \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{Bmatrix}$$

Jakobio matricos eilutėse įrašyti vektorinės funkcijos komponenčių *gradientų vektoriai*

Netiesinių algebrinių lygčių sprendimas.
Niutono metodas

Sprendžiame prieaugiais:

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} + \Delta\mathbf{x} \Rightarrow \begin{Bmatrix} f_1(\mathbf{x}^{i+1}) \\ f_2(\mathbf{x}^{i+1}) \\ \vdots \\ f_n(\mathbf{x}^{i+1}) \end{Bmatrix} \approx \begin{Bmatrix} f_1(\mathbf{x}^{i}) \\ f_2(\mathbf{x}^{i}) \\ \vdots \\ f_n(\mathbf{x}^{i}) \end{Bmatrix} + \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}\right|_{\mathbf{x}^i} \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{Bmatrix}$$

Po argumentų prieaugio funkcijos reikšmė, apskaičiuota pagal pirmus Teiloro eilutės narius turi tapti lygia 0:

$$\mathbf{f}(\mathbf{x}^{i+1}) = 0 \Rightarrow \left[\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^i}\right]\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^i); \Rightarrow \Delta\mathbf{x} = -\left[\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^i}\right]^{-1}\mathbf{f}(\mathbf{x}^i)$$

$$\boxed{\mathbf{x}^{i+1} = \mathbf{x}^{i} - \left[\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^i}\right]^{-1}\mathbf{f}(\mathbf{x}^i)}$$

- **Užrašas simboliais toks pats, kaip ir Niutono metodo formulė vienai netiesinei lygčiai;**
- **Skiriasi tai, kad čia x ir f yra vektoriai, o vietoje funkcijos išvestinės įrašoma vektorinės funkcijos Jakobio matrica**

Netiesinių algebrinių lygčių sprendimas.
Niutono metodas

Niutono metodo iteracijų formulė:

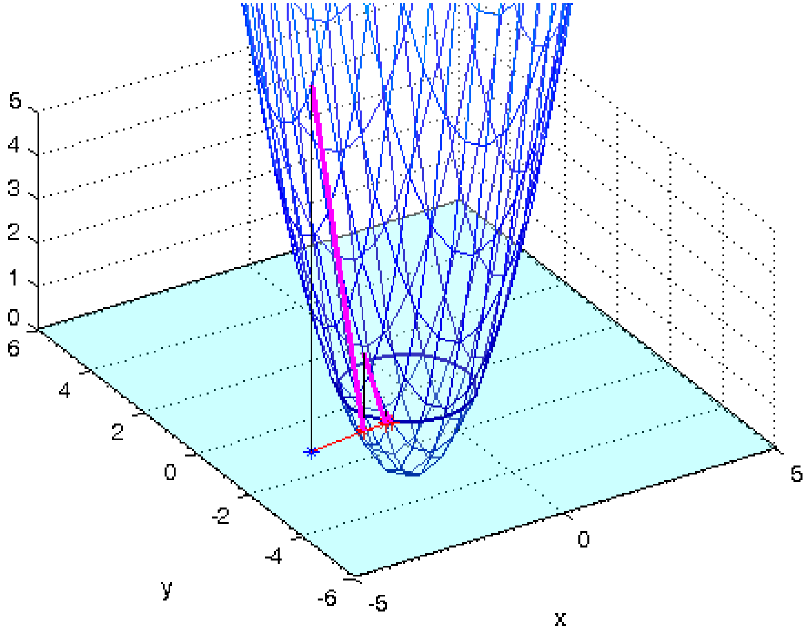
$$\mathbf{x}^{i+1} = \mathbf{x}^{i} - \left[\left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^i} \right]^{-1} \mathbf{f}(\mathbf{x}^i)$$

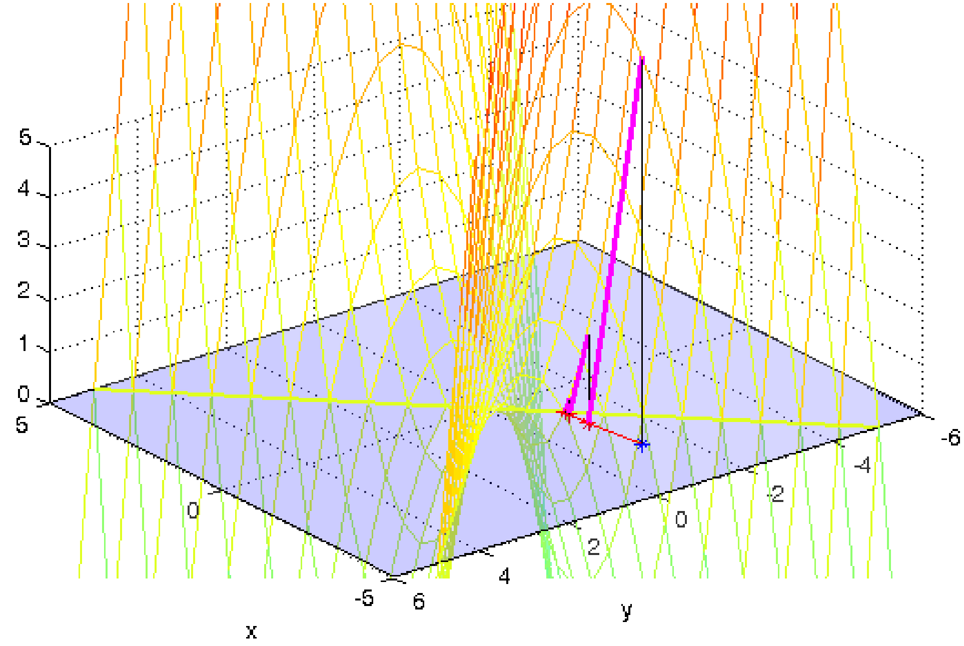
Vykdant algoritmą, neverta apskaičiuoti atvirkštinę matricą. Geriau kiekvienoje iteracijoje *spręsti tiesinių lygčių sistemą*:

$$\left[\left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^i} \right] \Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^i) \Rightarrow \Delta\mathbf{x} \Rightarrow \mathbf{x}^{i+1} = \mathbf{x}^{i} + \Delta\mathbf{x}$$

Jeigu Niutono metodas diverguoja, galima bandyti *sumažinti prieaugio dydį*:

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} + \beta\,\Delta\mathbf{x} \quad , \qquad 0 < \beta \le 1$$

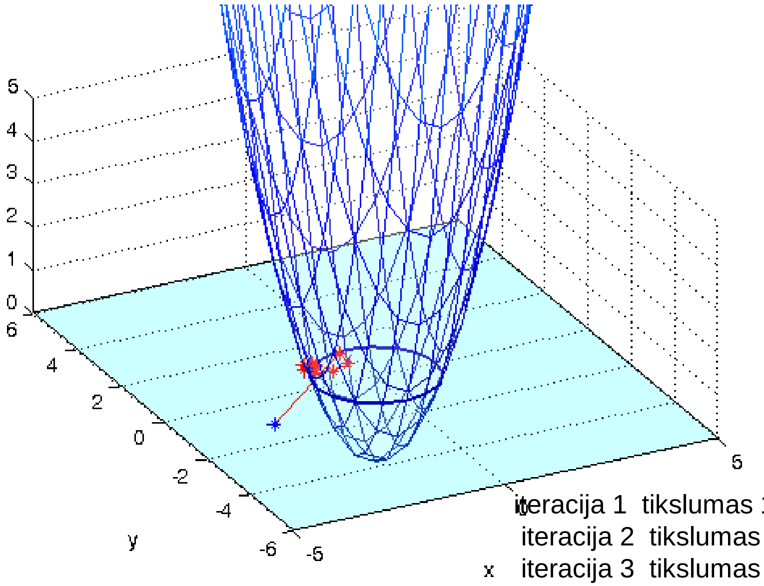


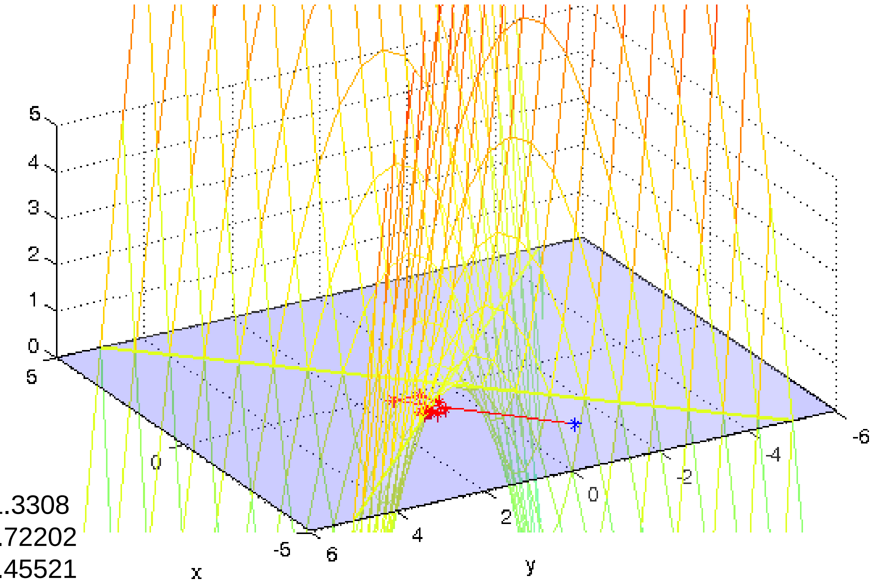


iteracija 1 tikslumas 0.294054
iteracija 2 tikslumas 0.214649
iteracija 3 tikslumas 0.0766578
iteracija 4 tikslumas 0.00547262
sprendinys x = -1.00784 -1

Pvz_SMA_5_3, 5_4, 5_5

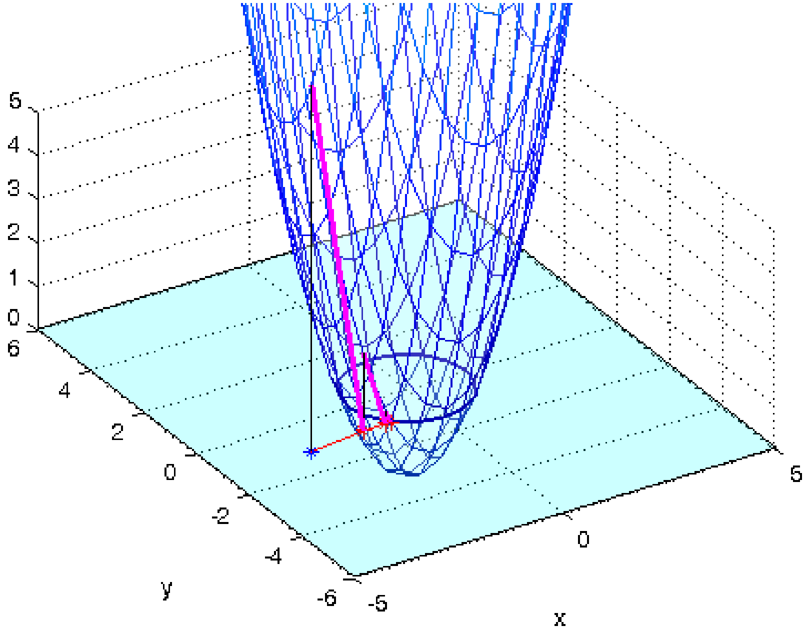
Palyginkime: ta pati lygčių sistema sprendžiama paprastųjų iteracijų metodu (alpha=[1;1]*3)

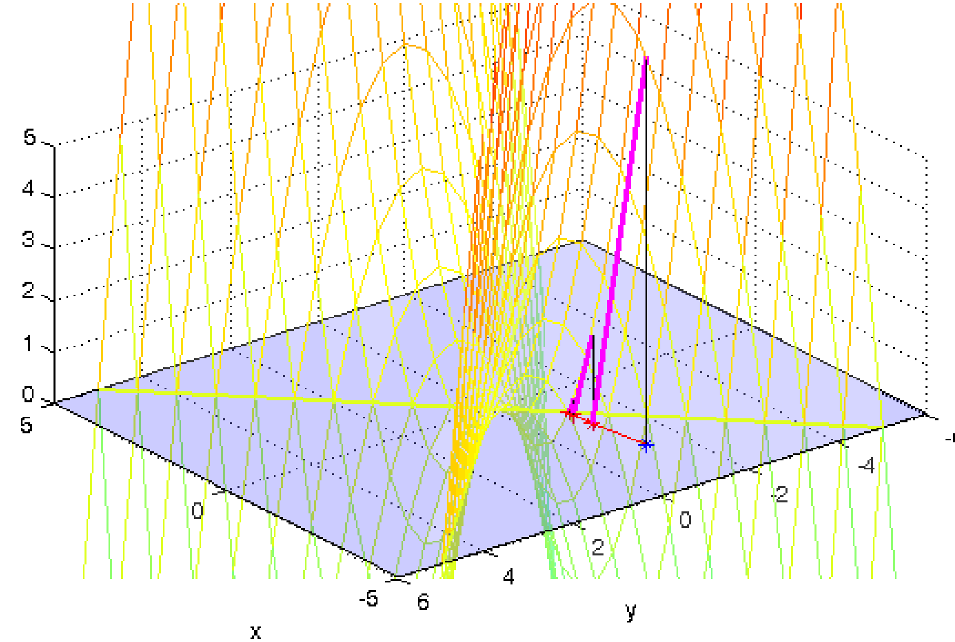




iteracija 1 tikslumas 11.3308
iteracija 2 tikslumas 1.72202
iteracija 3 tikslumas 1.45521
iteracija 4 tikslumas 1.15575
iteracija 5 tikslumas 0.879586
iteracija 6 tikslumas 0.751667
iteracija 7 tikslumas 0.533435
iteracija 8 tikslumas 0.399402
iteracija 9 tikslumas 0.286547
iteracija 10 tikslumas 0.214062
iteracija 11 tikslumas 0.161948
iteracija 12 tikslumas 0.121426
iteracija 13 tikslumas 0.0908646
iteracija 14 tikslumas 0.0669444
iteracija 15 tikslumas 0.0499964
iteracija 16 tikslumas 0.0371301
iteracija 17 tikslumas 0.027856
iteracija 18 tikslumas 0.0207214
iteracija 19 tikslumas 0.0154719
iteracija 20 tikslumas 0.0114985
iteracija 21 tikslumas 0.0085788
sprendinys **x = -0.997614 0.998122**

- PI metodu radome kitą sprendinį, negu Niutono metodu;
- Šis sprendinys nėra artimiausias pradiniam artiniui





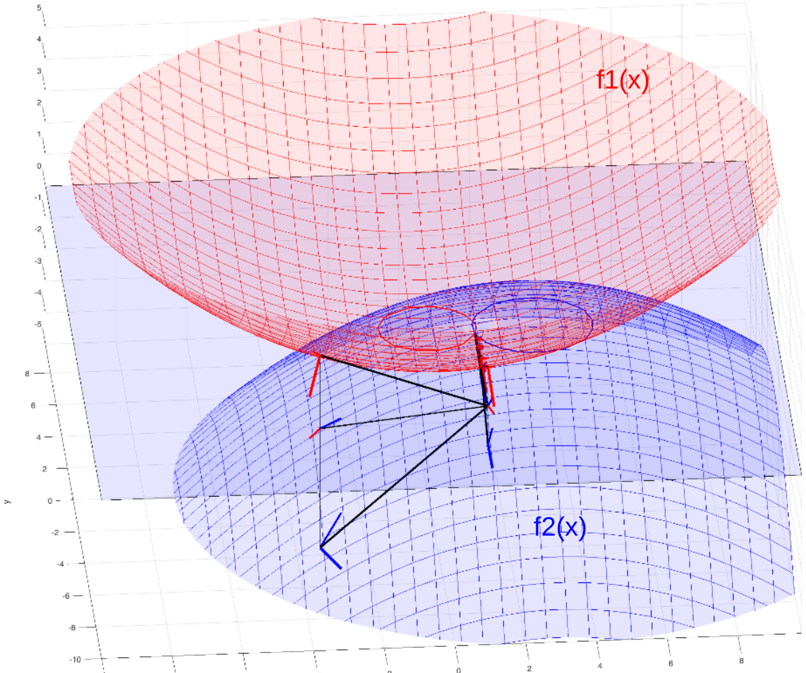
•Sprendžiant Niutono metodu, kiekvienos lygties funkcijai artinio taške brėžiama liestinė (hiper)plokštuma;

•Liestinės plokštumos kertasi su nuline plokštuma (hiper)tiesėmis;

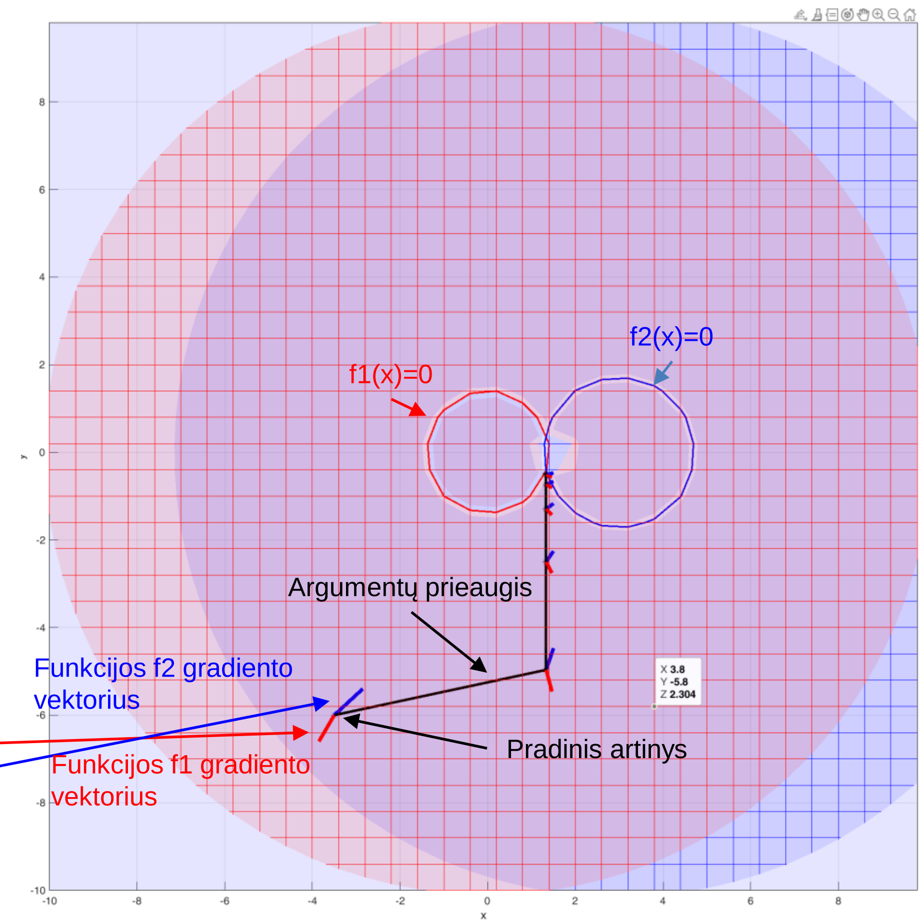
•Sekantis artinys parenkamas gautųjų (hiper)tiesių susikirtimo taške

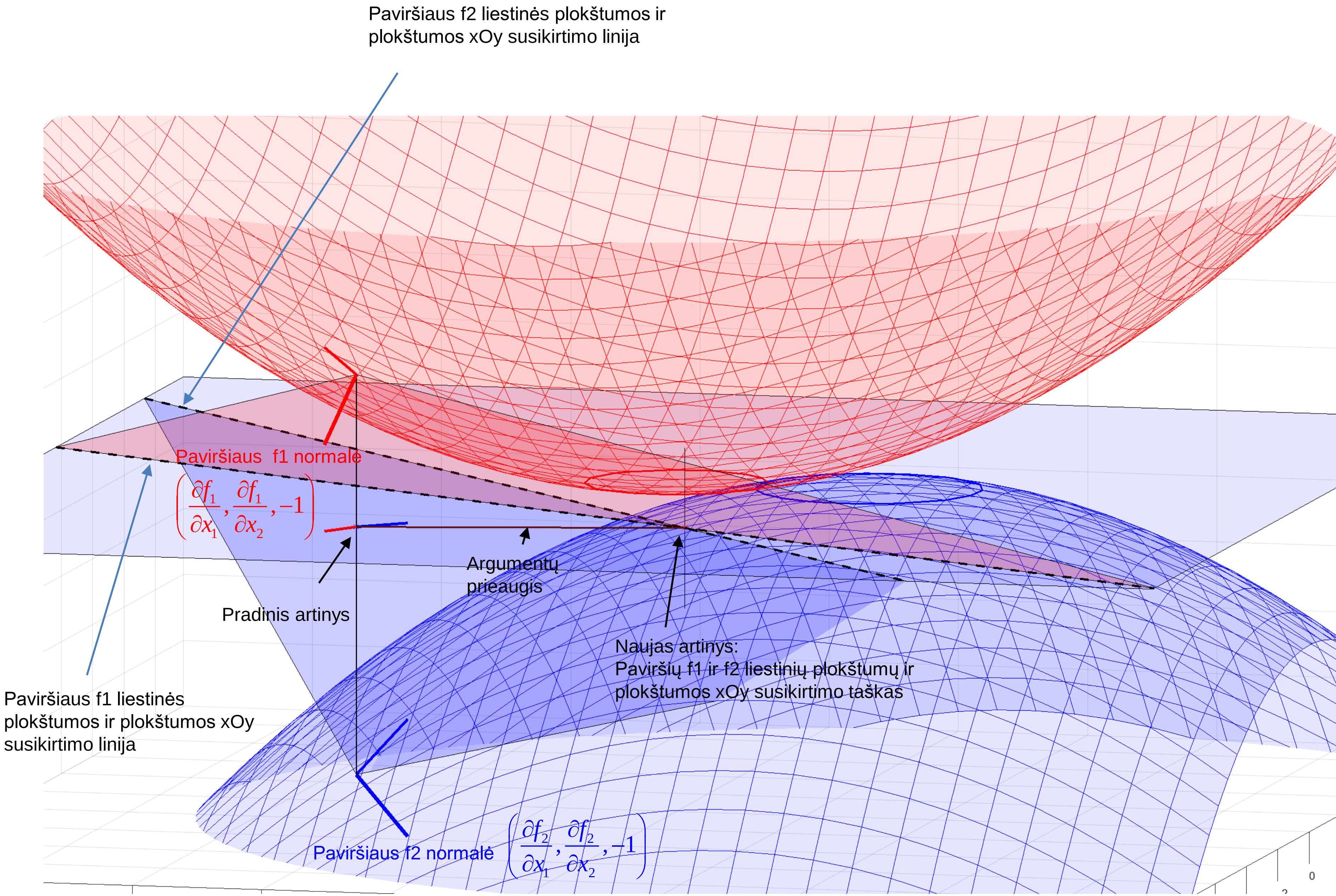
Niutono metodo grafinė interpretacija 2 lygčių sistemos atveju

Pvz_SMA_5_09_Niutono_metodas_2_LS_gradientai.m




```
% Lygciu sistemos funkcija
function fff=f(x)
fff=0.05*[x(1)^2+x(2)^2-2;
-(x(1)-3)^2-x(2)^2+3];
return
end
% Jakobio matrica
function dff=df(x)
dff=0.05*[2*x(1),2*x(2);
-2*(x(1)-3),-2*x(2)];
return
end
```





Paviršiaus f2 liestinės plokštumos ir plokštumos xOy susikirtimo linija
Paviršiaus f1 normalė
$\left(\frac{\partial f_1}{\partial x_1}, \frac{\partial f_1}{\partial x_2}, -1\right)$
Argumentų prieaugis
Pradinis artinys
Naujas artinys:
Paviršių f1 ir f2 liestinių plokštumų ir plokštumos xOy susikirtimo taškas
Paviršiaus f1 liestinės plokštumos ir plokštumos xOy susikirtimo linija
Paviršiaus f2 normalė
$\left(\frac{\partial f_2}{\partial x_1}, \frac{\partial f_2}{\partial x_2}, -1\right)$

Netiesinių algebrinių lygčių sprendimas.
Niutono metodas – apibendrinimai (1)

- Niutono metodas visuomet konverguoja, kai pradedama skaičiuoti nuo gero pradinio artinio;
- Kiekvienos iteracijos metu reikia apskaičiuoti funkcijos ir Jakobio matricos reikšmes

Jakobio matricos reikšmę galima įvertinti skaitiškai, nenaudojant analitinių diferencijavimo formulių:

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} - \left[\left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^{i}} \right]^{-1} \mathbf{f}\left(\mathbf{x}^{i}\right)$$

$$\Downarrow$$

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(x_1, x_2, \ldots, x_j + h, \ldots, x_n) - f_i(x_1, x_2, \ldots, x_j, \ldots, x_n)}{h}$$

Lygčių sistemos atveju, Jakobio matricos radimui kiekvienos iteracijos metu tektų $2n^2$ kartų apskaičiuoti funkcijų reikšmes. Tai *neekonomiška* skaičiavimų imlumo požiūriu.

Jakobio matricos skaitinio įverčio apskaičiavimas Python:

```
import numpy as np

def numerical_jacobi(f,x,dx):
  # f -  vektorine funkcija, aprasyta kaip def
  # x -  vektorinis argumentas
  # dx - argumentu pokytis, skaitiskai ivertinant isvestines, skaliarinis dydis
  n=np.size(f(x));m=np.size(x);
  J=np.matrix(np.zeros((n,m),dtype=float))
  x1=np.matrix(x)
  for j in range (m): x1[j]=x[j]+dx; J[:,j]=(f(x1)-f(x))/dx; x1[j]=x[j];
  return J

def f1(x):
  fff=np.vstack((np.power(x[0],2)+np.power(x[1],2)-2,np.power(x[0],2)-np.power(x[1],2)))
  return fff

x=np.matrix([[1],[1]],dtype=float)
print(numerical_jacobi(f1,x,0.000001))
```

$$\mathbf{f}(\mathbf{x}) = \begin{Bmatrix} x_1^{\;2} + x_2^{\;2} - 2 \\ x_1^{\;2} - x_2^{\;2} \end{Bmatrix}$$

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 2x_1 & 2x_2 \\ 2x_1 & -2x_2 \end{bmatrix} \approx$$

```
[[ 2.000001 2.000001]
 [ 2.000001 -2.000001]]
```

Netiesinių algebrinių lygčių sprendimas.
Niutono metodas – apibendrinimai (2)

Rafsono modifikacija:

- **Jakobio matricos reikšmes galima atnaujinti ne kiekvienoje iteracijoje, tačiau *kas tam tikrą skaičių iteracijų;***
- **Konvergavimo sparta sumažėja, tačiau kiekvienoje iteracijoje sumažėja atliekamų veiksmų skaičius. Suminis sprendimo laikas dažniausiai gaunamas mažesnis;**
- **Toks sprendimo būdas vadinamas *Niutono-Rafsono metodu***

Kvazi-Niutono metodai.
Broideno metodas

Kvazi-Niutono metodai: Broideno metodas (1)

Niutono metodo iteracijų formulė:

kvazi-Niutono =
„lyg ir Niutono“,
„tartum Niutono“

$$\mathbf{x}^{i+1} = \mathbf{x}^{i} - \left[\left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^i} \right]^{-1} \mathbf{f}(\mathbf{x}^i)$$

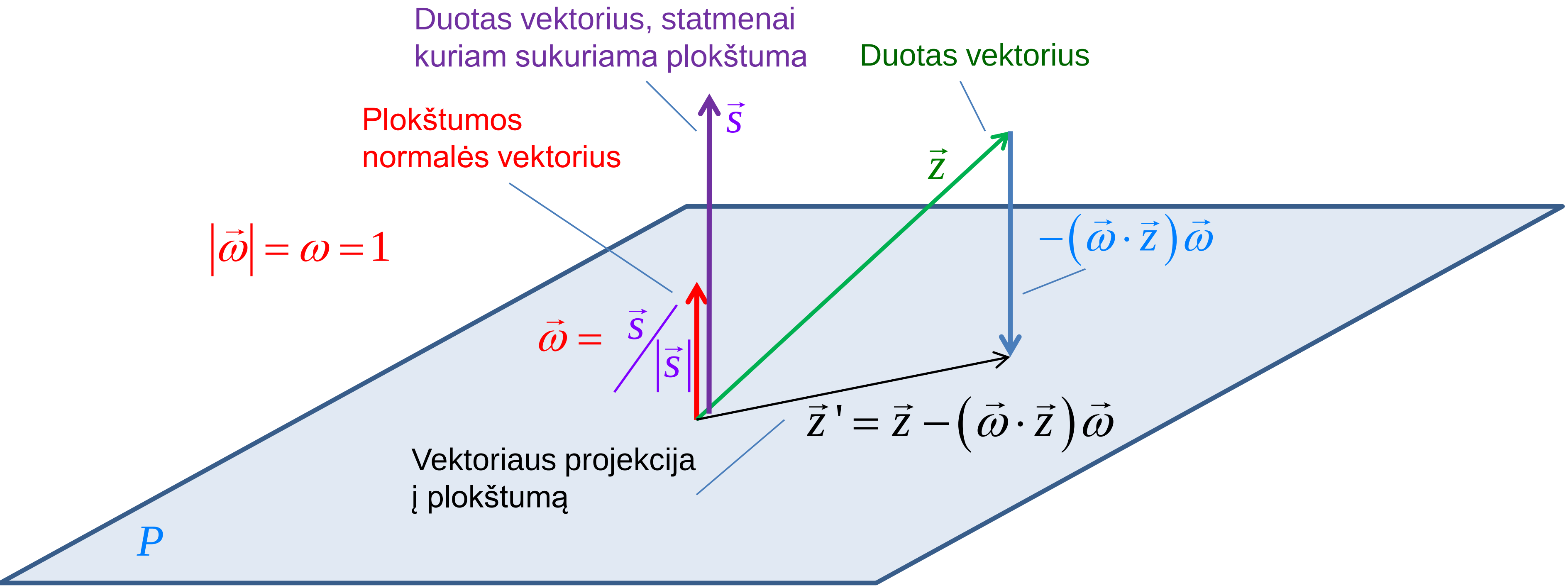
- **Siekiame apytiksliai apskaičiuoti Jakobio matricą;** $\rightarrow \left[\left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}^i} \right] \approx \mathbf{A}_i$
- **Apytikslė Jakobio matrica apskaičiuojama panašiai, kaip ir kirstinių metode , t.y. pagal dabartinį ir prieš tai apskaičiuotąjį artinius:**

$$\mathbf{A}_i \left(\mathbf{x}^i - \mathbf{x}^{i-1} \right) = \mathbf{f}\left(\mathbf{x}^i \right) - \mathbf{f}\left(\mathbf{x}^{i-1} \right)$$

$$\mathbf{A}_i \mathbf{s} = \mathbf{y}$$

- **Iš tokios lygčių sistemos matricos A elementų vienareikšmiškai apskaičiuoti nepavyktų;**
- **Sistema turi n lygčių ir n^2 nežinomųjų;**
- **Nežinomieji yra visi matricos A elementai, o vektoriai s ir y yra žinomi**

Broideno metodo idėja: vektoriaus projekcija į plokštumą 3D erdvėje



$$\vec{z}\,' = \vec{z} - \vec{\omega}(\vec{\omega}\cdot\vec{z}) = \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} - \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \left(\begin{Bmatrix} \omega_x & \omega_y & \omega_z \end{Bmatrix} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} \right) =$$

Projekcijos matrica

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} - \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \begin{Bmatrix} \omega_x & \omega_y & \omega_z \end{Bmatrix} \begin{Bmatrix} z_x \\ z_y \\ z_z \end{Bmatrix} = \mathbf{E}\mathbf{z} - \boldsymbol{\omega}\boldsymbol{\omega}^T\mathbf{z} = \left(\mathbf{E} - \frac{\mathbf{s}\mathbf{s}^T}{\|\mathbf{s}\|^2} \right)\mathbf{z} = \boxed{\left(\mathbf{E} - \frac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \right)}\mathbf{z}$$

Broideno metodas (2)

$$\mathbf{Z}^* = \left(\mathbf{E} - \frac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right)\mathbf{Z}$$

- Matricos **Z*** **stulpeliai** yra matricos **Z** **stulpelių**(t.y. vektorių) projekcijų į duotą plokštumą vektoriai;
- Projekcijos plokštuma yra statmena vektoriui **s**

$$\left(\mathbf{Z}^*\right)^T = \mathbf{Z}^T\left(\mathbf{E} - \frac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right)$$

- Matricos (**Z***)^T **eilutės** yra matricos (**Z**)^T **eilučių** (t.y. vektorių) projekcijų į duotą plokštumą vektoriai;
- Projekcijos plokštuma yra statmena vektoriui **s** ;
- Projekcijos matrica $\left(\mathbf{E} - \frac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right)$ yra simetrinė, todėl ji išlieka tokia pati abiem atvejais

Broideno metodas (3)

- Apytikslei Jakobio matricai apskaičiuoti priimame papildomą sąlygą, kad atnaujinta matrica sekančioje iteracijoje būtų kiek galima artimesnė prieš tai naudotai matricai;
- Pareikalaujame, kad nepakistų atnaujintos matricos A eilučių, kaip vektorių, projekcijos į hiperplokštumą, statmeną prieš tai atliktoje iteracijoje apskaičiuoto kintamųjų prieaugio vektoriui;
- Siekiame išlaikyti nepakitusius ***ne stulpelių, o būtent eilučių projekcijų vektorius***, kadangi kiekvienoje Jakobio matricos eilutėje yra įrašytas kiekvienos funkcijų vektoriaus komponentės gradiento vektorius

$$\begin{cases} \mathbf{A}_i \mathbf{s} = \mathbf{y}; \\ \mathbf{A}_i \left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \right) = \mathbf{A}_{i-1} \left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \right) \end{cases}$$

$$\mathbf{s} = \mathbf{x}^i - \mathbf{x}^{i-1}$$

$$\mathbf{y} = \mathbf{f}\left(\mathbf{x}^i\right) - \mathbf{f}\left(\mathbf{x}^{i-1}\right)$$

Broideno metodas (4)

$$\cdot\frac{\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}$$

Iš šio vektoriaus padauginame abi pirmos lygties puses

$$\begin{cases} \mathbf{A}_i\mathbf{s} = \mathbf{y}; \\ \mathbf{A}_i\left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right) = \mathbf{A}_{i-1}\left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right); \end{cases}$$

$$\begin{cases} \mathbf{A}_i\dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} = \dfrac{\mathbf{y}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \\ \mathbf{A}_i\left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right) = \mathbf{A}_{i-1}\left(\mathbf{E} - \dfrac{\mathbf{s}\,\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}\right); \end{cases}$$

$$\boxed{\mathbf{A}_i = \mathbf{A}_{i-1} + \frac{\left(\mathbf{y} - \mathbf{A}_{i-1}\mathbf{s}\right)\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}}$$

$$\mathbf{s} = \mathbf{x}^i - \mathbf{x}^{i-1}$$

$$\mathbf{y} = \mathbf{f}\left(\mathbf{x}^i\right) - \mathbf{f}\left(\mathbf{x}^{i-1}\right)$$

Broideno metodas (5)

$$\mathbf{x}^0, \mathbf{A}_0$$

Pradinis sprendinio artinys ir skaitiniu diferencijavimu gautas arba laisvai parinktas Jakobio matricos artinys

$$\mathbf{A}_{i-1}\mathbf{s}_i = -\mathbf{f}(\mathbf{x}^{i-1});$$

Argumentų prieaugiu apskaičiavimas

$$\mathbf{x}^i = \mathbf{x}^{i-1} + \mathbf{s}_i$$

Sekantis artinys

$$\mathbf{y}_i = \mathbf{f}(\mathbf{x}^i) - \mathbf{f}(\mathbf{x}^{i-1});$$

Funkcijos reikšmės pokytis

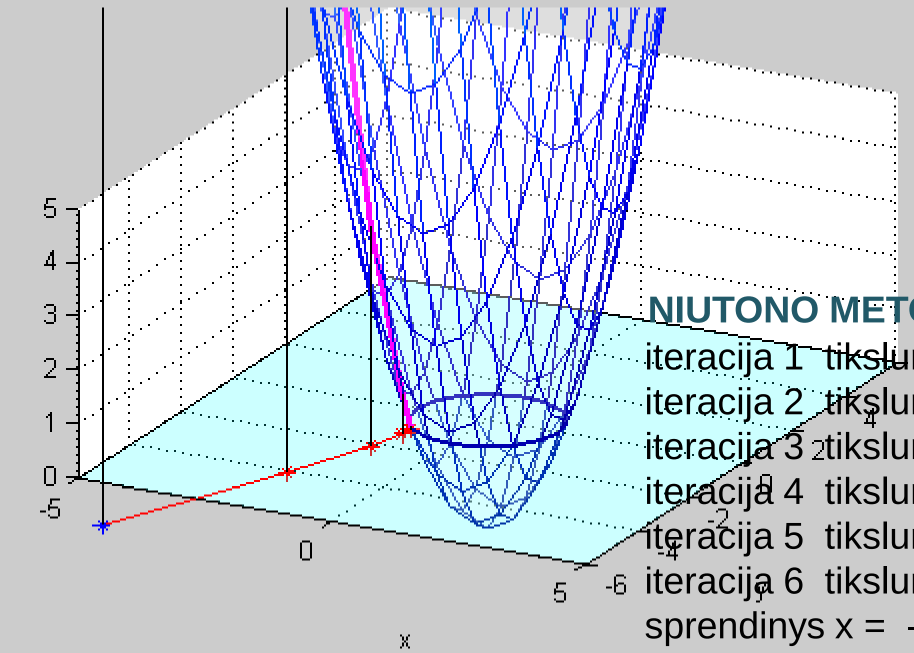
$$\mathbf{A}_i = \mathbf{A}_{i-1} + \frac{(\mathbf{y}_i - \mathbf{A}_{i-1}\mathbf{s}_i)\mathbf{s}_i^T}{\mathbf{s}_i^T \mathbf{s}_i};$$

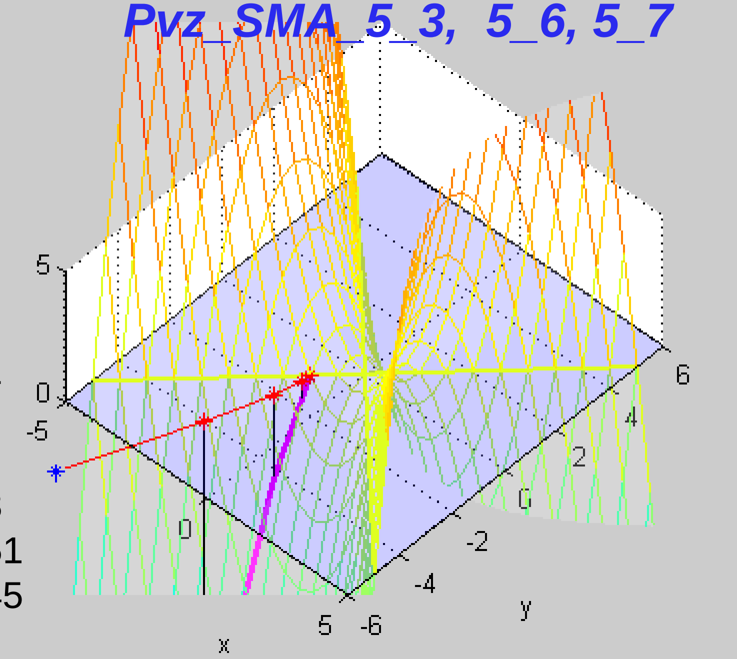
Jakobio matricos atnaujinimas pagal Broideno formulę

$$i = 1, 2, \ldots$$

- **Apskaičiavę naują artinį, pagal jo reikšmę atnaujiname ir Jakobio matricą;**
- **Atliekant pirmąją iteraciją, A_0 jau turi būti tam tikru būdu apskaičiuotas;**
- **A_0 galima apskaičiuoti, taikant skaitinio diferencijavimo veiksmą, arba parinkti tam tikrą įstrižaininę matricą**

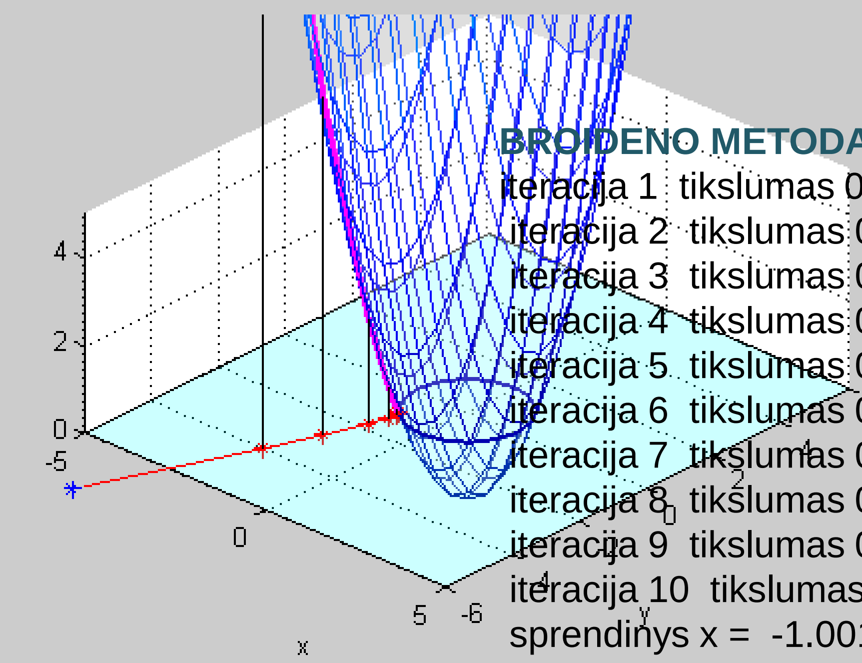
Pvz_SMA_5_3, 5_6, 5_7

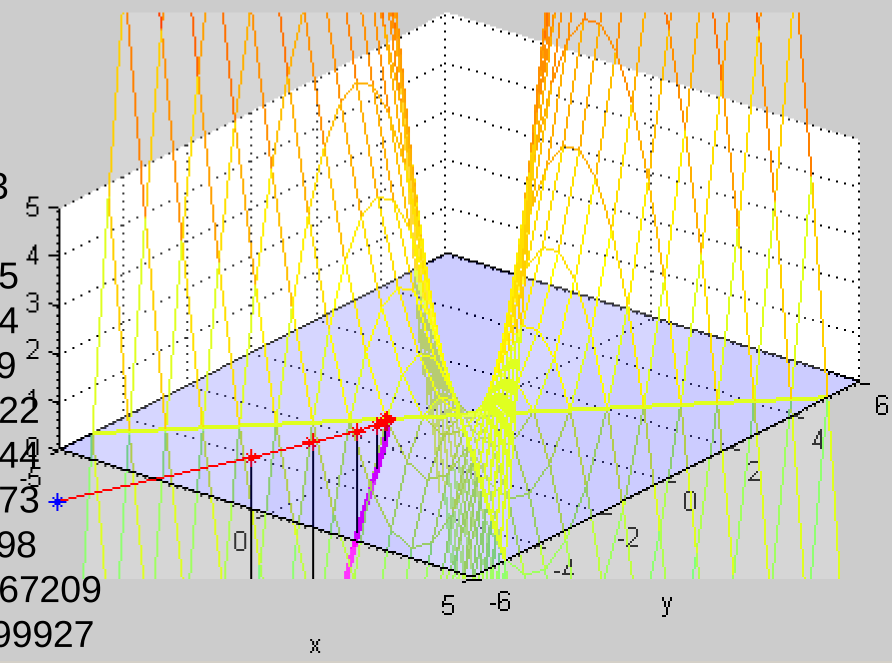




NIUTONO METODAS

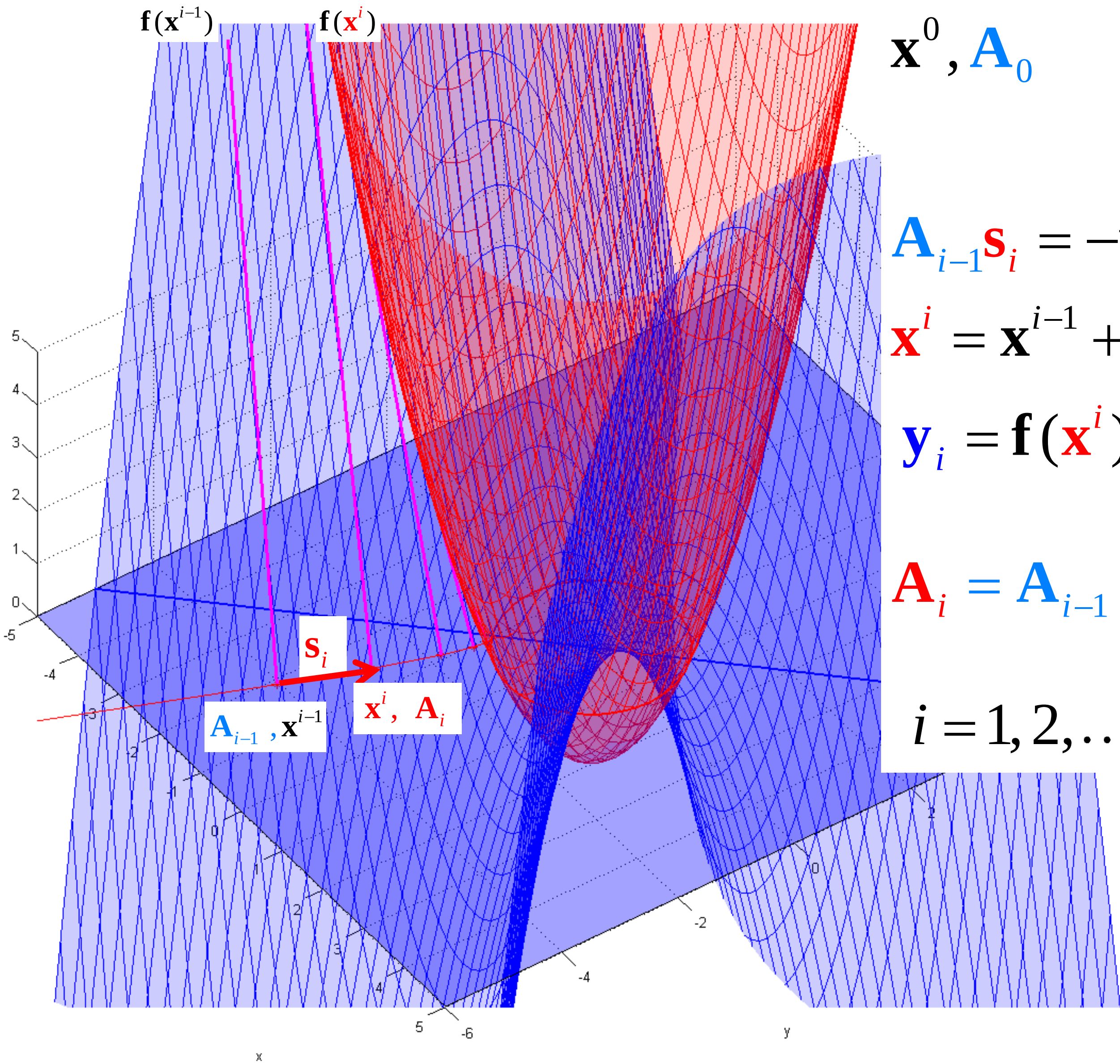
iteracija 1 tikslumas 0.327521
iteracija 2 tikslumas 0.311368
iteracija 3 tikslumas 0.25965
iteracija 4 tikslumas 0.141478
iteracija 5 tikslumas 0.0243051
iteracija 6 tikslumas 0.0004545
sprendinys x = -1 -1.00064





BROIDENO METODAS

iteracija 1 tikslumas 0.327523
iteracija 2 tikslumas 0.23425
iteracija 3 tikslumas 0.247065
iteracija 4 tikslumas 0.186294
iteracija 5 tikslumas 0.115689
iteracija 6 tikslumas 0.0479122
iteracija 7 tikslumas 0.0240844
iteracija 8 tikslumas 0.0236073
iteracija 9 tikslumas 0.0110998
iteracija 10 tikslumas 0.000467209
sprendinys x = -1.00121 -0.99927



$$\mathbf{x}^0, \mathbf{A}_0$$

$$\mathbf{A}_{i-1}\mathbf{s}_i = -\mathbf{f}(\mathbf{x}^{i-1});$$

$$\mathbf{x}^i = \mathbf{x}^{i-1} + \mathbf{s}_i$$

$$\mathbf{y}_i = \mathbf{f}(\mathbf{x}^i) - \mathbf{f}(\mathbf{x}^{i-1});$$

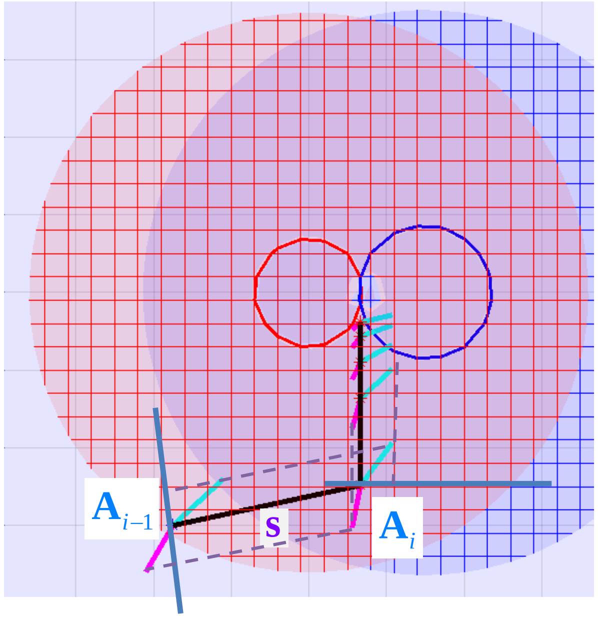
$$\mathbf{A}_i = \mathbf{A}_{i-1} + \frac{(\mathbf{y}_i - \mathbf{A}_{i-1}\mathbf{s}_i)\mathbf{s}_i^T}{\mathbf{s}_i^T\mathbf{s}_i};$$

$$i = 1, 2, \ldots$$

Broideno metodo grafinė interpretacija 2 lygčių sistemos atveju

Pvz_SMA_5_08_Broideno_metodas_2_LS_gradientai.m

Pvz_SMA_5_09_Niutono_metodas_2_LS_gradientai.m

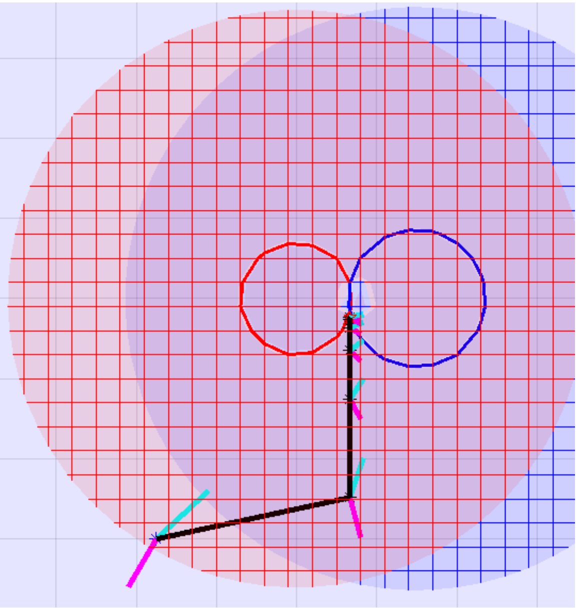


$$\begin{cases} \mathbf{A}_i \mathbf{s} = \mathbf{y}; \\ \mathbf{A}_i \left(\mathbf{E} - \dfrac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \right) = \mathbf{A}_{i-1} \left(\mathbf{E} - \dfrac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}} \right) \end{cases}$$

$$\mathbf{A}_i = \mathbf{A}_{i-1} + \frac{\left(\mathbf{y} - \mathbf{A}_{i-1}\mathbf{s}\right)\mathbf{s}^T}{\mathbf{s}^T\mathbf{s}}$$

$$\mathbf{s} = \mathbf{x}^i - \mathbf{x}^{i-1}; \ \mathbf{y} = \mathbf{f}\left(\mathbf{x}^i\right) - \mathbf{f}\left(\mathbf{x}^{i-1}\right)$$

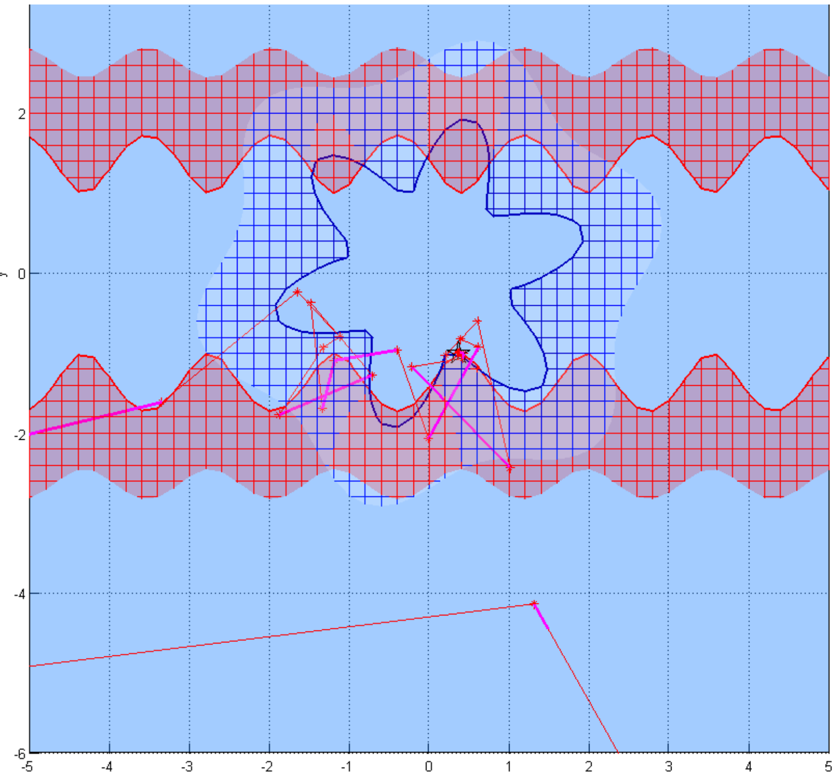
- Tik pradiniai gradientai yra tikslūs, kiek tą galima pasiekti skaitiniu diferencijavimu;
- Sprendžiant Broideno metodu, apytikslėje Jakobio matricoje gradientų projekcijos į paskutiniam argumentų prieaugiui statmeną plokštumą yra vienodos. Tačiau patys gradientai yra apytiksliai

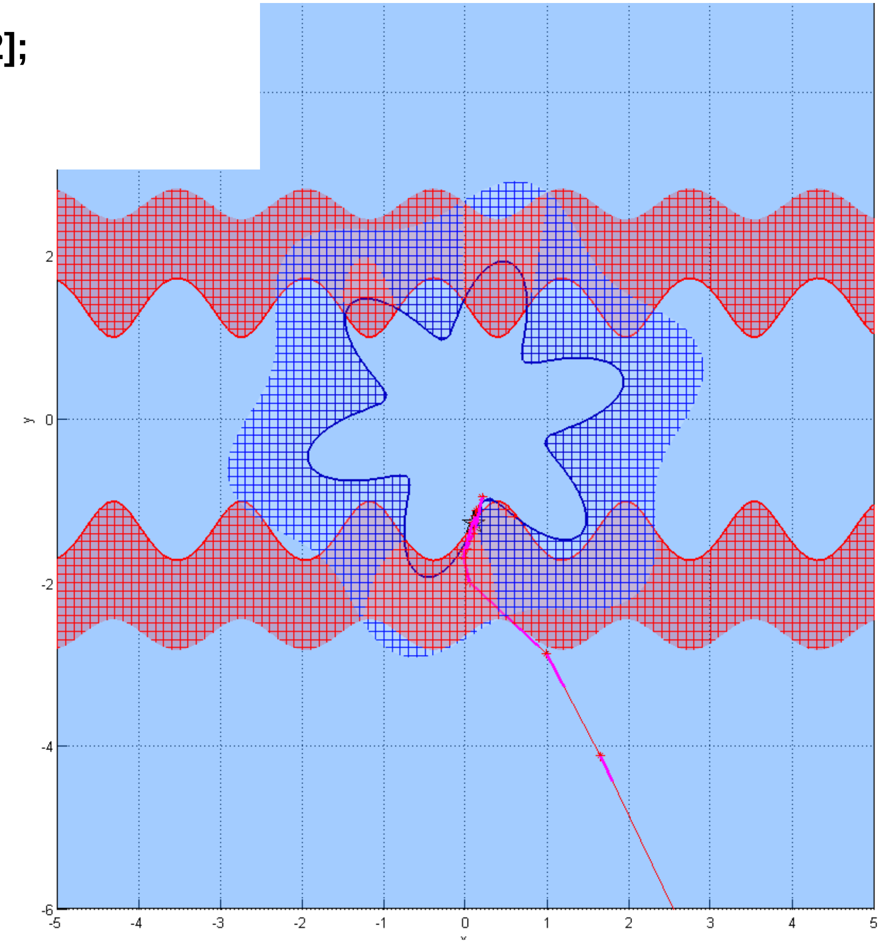


- Palyginkime: Niutono metodu gaunami gradientai

```
function fff=f(x)     %  Lygciu sistemos funkcija
    fff=[sin(4*x(1))+x(2)^2-2;
         x(1)^2+x(2)^2-(0.5*sin(6*atan(x(2)/x(1)))+1.5)^2];
return
end
```

Pvz_SMA_5_3x, 5_6x





NIUTONO METODAS

iteracija 1 tikslumas 0.33704
iteracija 2 tikslumas 0.766926
.
.
.
iteracija 29 tikslumas 1.87904e-05
iteracija 30 tikslumas 3.61564e-06
sprendinys x = 0.365414 -1.00297

BROIDENO METODAS

iteracija 1 tikslumas 0.329564
iteracija 2 tikslumas 0.241526
.
.
.
iteracija 18 tikslumas 6.96362e-05
iteracija 19 tikslumas 1.58517e-06
sprendinys x = 0.0997217 -1.26949

SMA_05_Klausimai savikontrolei(1):

1. Kaip parenkamas netiesinių lygčių sistemos (NLS) sprendinio pradinis artinys;
2. Kaip parenkamas netiesinių lygčių sistemos (NLS) sprendinio pradinis artinys;
   Paaiškinkite, kaip būtų galima grafiškai rasti dviejų NLS sprendinį;
3. Užrašykite Paprastųjų iteracijų metodo formulę bendrojo pavidalo NLS sprendimui. Kuo ji panaši ir kuo skiriasi nuo vienos netiesinės lygties sprendimo iteracinės formulės;
4. Kokį vaidmenį atlieka koeficientai "alpha" Paprastųjų iteracijų metode;
5. Kuo skiriasi Teiloro eilutės formulė vienmatėje ir daugiamatėje kintamųjų erdvėje;

SMA_05_ Klausimai savikontrolei(2):

6. Kokia yra Teiloro eilutės pirmųjų dviejų narių formulė, užrašyta matricomis. Kas yra Jakobio matrica;
7. Kaip gaunama Niutono metodo formulė daugiamatėje kintamųjų erdvėje;
8. Paaiškinkite du Niutono metodo formulės pavidalus, kai naudojama atvirkštinė matrica ir kai kiekviename žingsnyje sprendžiama lygčių sistema. Kuris pavidalas pranašesnis. Ar skiriasi sprendinio artinių seka, gauta taikant kiekvieną iš šių pavidalų;
9. Kokie yra svarbiausi Niutono metodo privalumai ir trūkumai;
10. Kas yra Niutono metodo Rafsono modifikacija (t.y.Niutono-Rafsono metodas). Ko ja siekiama;
11. Ar galima taikyti Niutono metodą, jeigu nėra galimybės analitiškai apskaičiuoti funkcijų išvestinių (Jakobio matricos). Ką reiškia terminas "kvazi-Niutono metodas";

SMA_05_Klausimai savikontrolei(3):

12. Paaiškinkite, kodėl daugiamatėje kintamųjų erdvėje nepavyksta tiesiogiai taikyti dviejų nuosekliai gautų sprendinio artinių Jakobio matricos įverčiui apskaičiuoti;
13. Koks metodas yra Broideno metodo analogas, sprendžiant vieną netiesinę lygtį;
14. Paaiškinkite Broideno metodo idėją;
15. Naudodamiesi skaidrėmis, paaiškinkite veiksmų seką, atliekamą vykdant vieną Broideno metodo iteraciją;