KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

SKAITINIAI METODAI IR ALGORITMAI(P170B115)

2 LABORATORINIS DARBAS

Varianto Nr. 5

Atliko:

IFF-1/8 gr. studentas

Matas Palujanskas

Priėmė:

Prof. Rimantas Barauskas

Doc. Andrius Kriščiūnas

KAUNAS

2023

Matas Palujanskas

1. Turinys

Matas Palujanskas

1. Pirma užduoties dalis

1.1 Užduoties sąlygos

1 Tiesinių lygčių sistemų sprendimas

a) Lentelėje 1 duotos tiesinės lygčių sistemos, 2 lentelėje nurodyti metodai ir lygčių sistemų numeriai (iš 1 lentelės). Reikia suprogramuoti nurodytus metodus ir jais išspręsti pateiktas lygčių sistemas.

b) Lentelėje 3 duotos tiesinės lygčių sistemos, laisvųjų narių vektoriai ir nurodytas skaidos metodas. Reikia suprogramuoti nurodytą metodą ir juo išspręsti pateiktas lygčių sistemas.

Sprendžiant lygčių sistemas (a ir b punktuose), turi būti:

a) Programoje turi būti įvertinti atvejai:
- kai lygčių sistema turi vieną sprendinį;
- kai lygčių sistema sprendinių neturi;
- kai lygčių sistema turi be gali daug sprendinių.

b) Patikrinkite gautus sprendinius ir skaidas, įrašydami juos į pradinę lygčių sistemą.

c) Gautą sprendinį patikrinkite naudodami išorinius išteklius (pvz., standartines Python funkcijas).

pav. 1 Antrojo laboratorinio pirmos dalies užduotys

Užduoties variantas: 5

2 lentelėje nurodyti metodai:

5	Gauso	2, 12, 17
	Paprastųjų iteracijų	2

pav. 2 Nurodyti metodai

2	$\begin{cases} 3x_1 + 10x_2 + x_3 + 5x_4 = 83 \\ -2x_1 + 6x_2 + 12x_3 + 14x_4 = 178 \\ 3x_1 + 12x_2 + 5x_3 + x_4 = 37 \\ -3x_1 - 9x_2 + 5x_3 = -26 \end{cases}$	12	$\begin{cases} 3x_1 + x_2 - x_3 + 5x_4 = 20 \\ -3x_1 + 4x_2 - 8x_3 - x_4 = -36 \\ x_1 - 3x_2 + 7x_3 + 6x_4 = 41 \\ 5x_2 - 9x_3 + 4x_4 = -16 \end{cases}$
17	$\begin{cases} 2x_1 + 5x_2 + x_3 + 2x_4 = -1 \\ -2x_1 + 3x_3 + 5x_4 = 7 \\ x_1 - x_3 + x_4 = 3 \\ 5x_2 + 4x_3 + 7x_4 = 4 \end{cases}$		

pav. 3 Gautos tiesinių lygčių sistemos

2, 12 ir 17 lygčių sistemas reikia išspręsti Gauso metodu, 2 taip pat ir paprastųjų iteracijų metodu.

5.	$\begin{cases} 4x_1 + 3x_2 - x_3 + x_4 = \cdots \\ 3x_1 + 9x_2 - 2x_3 - 2x_4 = \cdots \\ -x_1 - 2x_2 + 11x_3 - x_4 = \cdots \\ x_1 - 2x_2 - x_3 + 5x_4 = \cdots \end{cases}$	$\begin{cases} \ldots = 7 \\ \ldots = 8 \\ \ldots = 7 \\ \ldots = 3 \end{cases}$	$\begin{cases} \ldots = 20 \\ \ldots = 0 \\ \ldots = 18 \\ \ldots = 40 \end{cases}$	$\begin{cases} \ldots = 1 \\ \ldots = 4 \\ \ldots = -10 \\ \ldots = 3.5 \end{cases}$	QR

pav. 4 3 lentelės tiesinės lygčių sistemos su laisvaisiais nariais ir nurodytas sklaidos metodas

Matas Palujanskas

1.2 Tiesinių lygčių sprendimas Gauso metodu

1.2.1 Sprendimo kodas

Gauso.ipynb:

```
import numpy as np

# --------------------iseities duomnenys:
A=np.matrix([[3, 10, 1, 5],
             [-2, 6, 12, 14],
             [3, 12, 5, 1],
             [-3, -9, 5, 0]]).astype(float)       # koeficientu matrica
b=(np.matrix([83,178,37,-26])).transpose()  #laisvuju nariu vektorius-stulpelis
n=(np.shape(A))[0]   # lygciu skaicius nustatomas pagal ivesta matrica A
nb=(np.shape(b))[1]  # laisvuju nariu vektoriu skaicius nustatomas pagal ivesta
matrica b

A1=np.hstack((A,b))  #isplestoji matrica

print(A);print(b);print(n);print(nb);

n = A.shape[1]  # Tiesinių lygčių skaičius
rangas_A = np.linalg.matrix_rank(A)
rangas_AB = np.linalg.matrix_rank(np.hstack((A, b)))

# tiesioginis etapas:

for i in range (0,n-1):   # range pradeda 0 ir baigia n-2 (!)
    for j in range (i+1,n):  # range pradeda i+1 ir baigia n-1
        A1[j,i:n+nb]=A1[j,i:n+nb]-A1[i,i:n+nb]*A1[j,i]/A1[i,i];
        A1[j,i]=0;
    print(A1)

#  atvirkstinis etapas:
x=np.zeros(shape=(n,nb))
for i in range (n-1,-1,-1):    # range pradeda n-1 ir baigia 0 (trecias parametras
yra zingsnis)
    x[i,:]=(A1[i,n:n+nb]-A1[i,i+1:n]*x[i+1:n,:])/A1[i,i]
    print(x,"x")

if rangas_A == rangas_AB == n:
    # Tiesinių lygčių sistema turi vieną sprendinį
    print("Tiesinių lygčių sistema turi vieną sprendinį:")
    print(x)
    liekana=A.dot(x)-b;
    print("Bendra santykine paklaida:", np.linalg.norm(liekana)/ np.linalg.norm(x))
elif rangas_A == rangas_AB:
    # Tiesinių lygčių sistema turi daugybę sprendinių
    print("Tiesinių lygčių sistema turi daugybę sprendinių.")
elif rangas_A == rangas_AB < n:
    # Tiesinių lygčių sistema neturi sprendinių
    print("Tiesinių lygčių sistema neturi sprendinių.")
```

Matas Palujanskas

1.2.2 2 lygčių sistema

2	$\begin{cases} 3x_1 + 10x_2 + x_3 + 5x_4 = 83 \\ -2x_1 + 6x_2 + 12x_3 + 14x_4 = 178 \\ 3x_1 + 12x_2 + 5x_3 + x_4 = 37 \\ -3x_1 - 9x_2 + 5x_3 = -26 \end{cases}$

pav. 5 2 lygčių sistema

Galima įsistatyti į matricą.

3	10	1	5	83
-2	6	12	14	178
3	12	5	1	37
-3	-9	5	0	-26

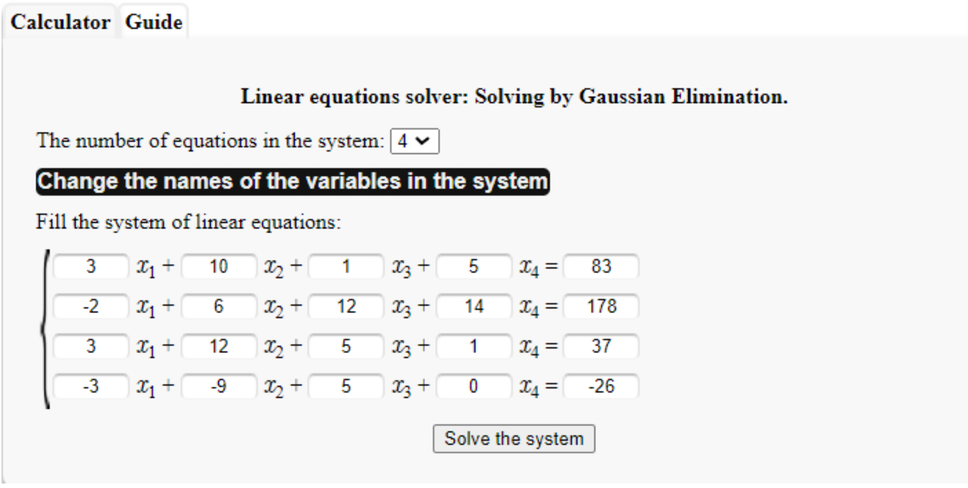
Programiškai gauti rezultatai:

```
[[  3.         10.          1.          5.          83.        ]
 [  0.         12.66666667  12.66666667  17.33333333 233.33333333]
 [  0.          2.          4.         -4.         -46.        ]
 [  0.          1.          6.          5.          57.        ]]
[[  3.         10.          1.          5.          83.        ]
 [  0.         12.66666667  12.66666667  17.33333333 233.33333333]
 [  0.          0.          2.         -6.73684211 -82.84210526]
 [  0.          0.          5.          3.63157895  38.57894737]]
[[  3.         10.          1.          5.          83.        ]
 [  0.         12.66666667  12.66666667  17.33333333 233.33333333]
 [  0.          0.          2.         -6.73684211 -82.84210526]
 [  0.          0.          0.         20.47368421 245.68421053]]
[[ 0.]
 [ 0.]
 [ 0.]
 [12.]] x
[[ 0.]
 [ 0.]
 [-1.]
 [12.]] x
[[ 0.]
 [ 3.]
 [-1.]
 [12.]] x
[[-2.]
 [ 3.]
 [-1.]
 [12.]] x
Tiesinių lygčių sistema turi vieną sprendinį:
[[-2.]
 [ 3.]
 [-1.]
 [12.]]
Bendra santykine paklaida: 6.319994012258868e-16
```

Gautas vienas sprendinys.

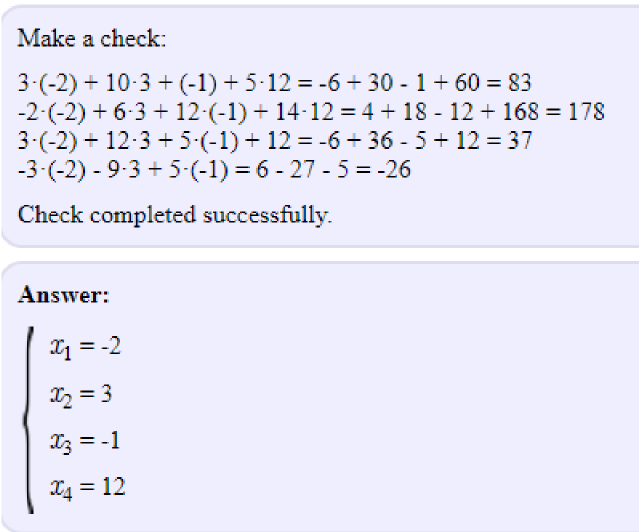
Tikriname ar rezultatai yra teisingi:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/



pav. 6 Tikrinimas naudojant išorinius šaltinius

Įstatome sprendinius į pradinę lygtį:



pav. 7 Gauti rezultatai

Išvada: gautas vienas sprendinys, naudojant išorinius šaltinius rezultatai sutapo.

1.2.3 12 lygčių sistema

12	$\begin{cases} 3x_1 + x_2 - x_3 + 5x_4 = 20 \\ -3x_1 + 4x_2 - 8\,x_3 - x_4 = -36 \\ x_1 - 3x_2 + 7\,x_3 + 6x_4 = 41 \\ 5x_2 - 9\,x_3 + 4x_4 = -16 \end{cases}$

pav. 8 12 lygčių sistema

Galima įsistatyti į matricą.

3	1	-1	5	20
-3	4	-8	-1	-36
1	-3	7	6	41
0	5	-9	4	-16

Programiškai gauti rezultatai:

```
[[  3.          1.         -1.          5.         20.        ]
 [  0.          5.         -9.          4.        -16.        ]
 [  0.         -3.33333333  7.33333333  4.33333333 34.33333333]
 [  0.          5.         -9.          4.        -16.        ]]
[[  3.          1.         -1.          5.         20.        ]
 [  0.          5.         -9.          4.        -16.        ]
 [  0.          0.          1.33333333  7.         23.66666667]
 [  0.          0.          0.          0.          0.        ]]
[[  3.          1.         -1.          5.         20.        ]
 [  0.          5.         -9.          4.        -16.        ]
 [  0.          0.          1.33333333  7.         23.66666667]
 [  0.          0.          0.          0.          0.        ]]
[[ 0.  ]
 [ 0.  ]
 [17.75]
 [ 0.  ]] x
[[ 0.  ]
 [28.75]
 [17.75]
 [ 0.  ]] x
[[ 3.  ]
 [28.75]
 [17.75]
 [ 0.  ]] x
Tiesinių lygčių sistema turi daugybę sprendinių.
Vienas is tu sprendiniu yra:
[[ 3.  ]
 [28.75]
 [17.75]
 [ 0.  ]]
```

pav. 9 Programiškai gauti rezultatai

Tikriname ar rezultatai yra teisingi:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/

Linear equations solver: Solving by Gaussian Elimination.

The number of equations in the system: 4

Change the names of the variables in the system

Fill the system of linear equations:

$$\begin{cases} 3x_1 + 1x_2 + (-1)x_3 + 5x_4 = 20 \\ -3x_1 + 4x_2 + (-8)x_3 + (-1)x_4 = -36 \\ 1x_1 + (-3)x_2 + 7x_3 + 6x_4 = 41 \\ 0x_1 + 5x_2 + (-9)x_3 + 4x_4 = -16 \end{cases}$$

Solve the system

pav. 10 Tikrinimas naudojant išorinius šaltinius

$R_1 - \frac{4}{15} R_3 \rightarrow R_1$ (multiply 3 row by $\frac{4}{15}$ and subtract it from 1 row); $R_2 + 1.8\ R_3 \rightarrow R_2$ (multiply 3 row by 1.8 and add it to 2 row)

$$\left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 10.25 & 28.75 \\ 0 & 0 & 1 & 5.25 & 17.75 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Answer:

The system of equations has a solution set:

$$\begin{cases} x_1 = 3 \\ x_2 + 10.25x_4 = 28.75 \\ x_3 + 5.25x_4 = 17.75 \end{cases}$$

pav. 11 Gauti rezultatai

Išvada: lygčių sistema turi be galo daug sprendinių, naudojant išorinius šaltinius rezultatai sutapo.

1.2.4 17 lygčių sistema

17	$\begin{cases} 2x_1 + 5x_2 + x_3 + 2x_4 = -1 \\ -2x_1 + 3x_3 + 5x_4 = 7 \\ x_1 - x_3 + x_4 = 3 \\ 5x_2 + 4x_3 + 7x_4 = 4 \end{cases}$

pav. 12 17 lygčių sistema

Galima įsistatyti į matricą.

2	5	1	2	-1
-2	0	3	5	7
1	0	-1	1	3
0	5	4	7	4

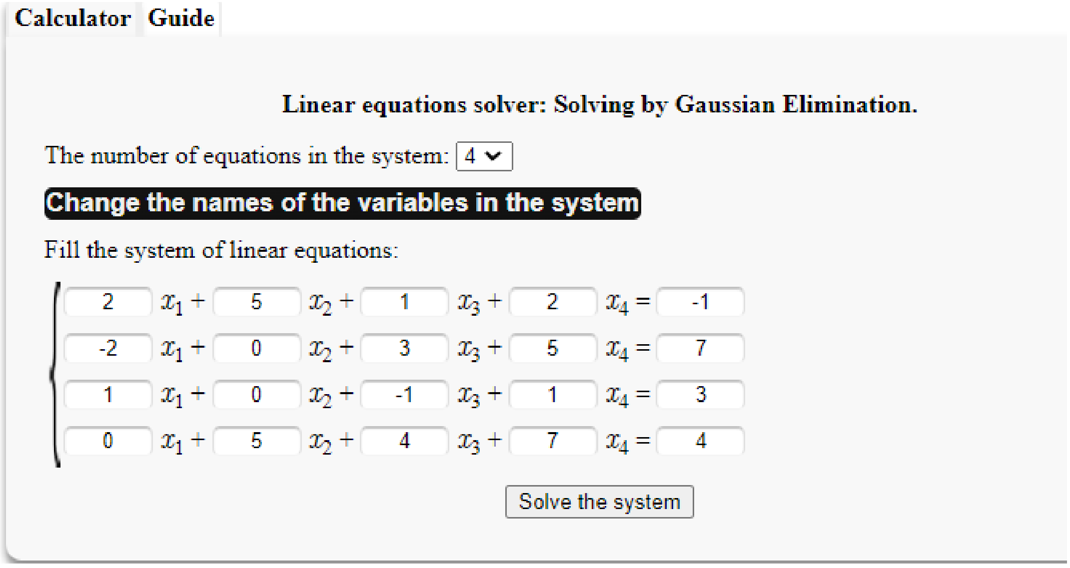
Programiškai gauti rezultatai:

```
[[ 2.   5.   1.   2.  -1. ]
 [ 0.   5.   4.   7.   6. ]
 [ 0.  -2.5 -1.5  0.   3.5]
 [ 0.   5.   4.   7.   4. ]]
[[ 2.   5.   1.   2.  -1. ]
 [ 0.   5.   4.   7.   6. ]
 [ 0.   0.   0.5  3.5  6.5]
 [ 0.   0.   0.   0.  -2. ]]
[[ 2.   5.   1.   2.  -1. ]
 [ 0.   5.   4.   7.   6. ]
 [ 0.   0.   0.5  3.5  6.5]
 [ 0.   0.   0.   0.  -2. ]]
[[  0.]
 [  0.]
 [  0.]
 [-inf]] x
[[  0.]
 [  0.]
 [ inf]
 [-inf]] x
[[  0.]
 [ nan]
 [ inf]
 [-inf]] x
[[ nan]
 [ nan]
 [ inf]
 [-inf]] x
Tiesinių lygčių sistema neturi sprendinių.
```

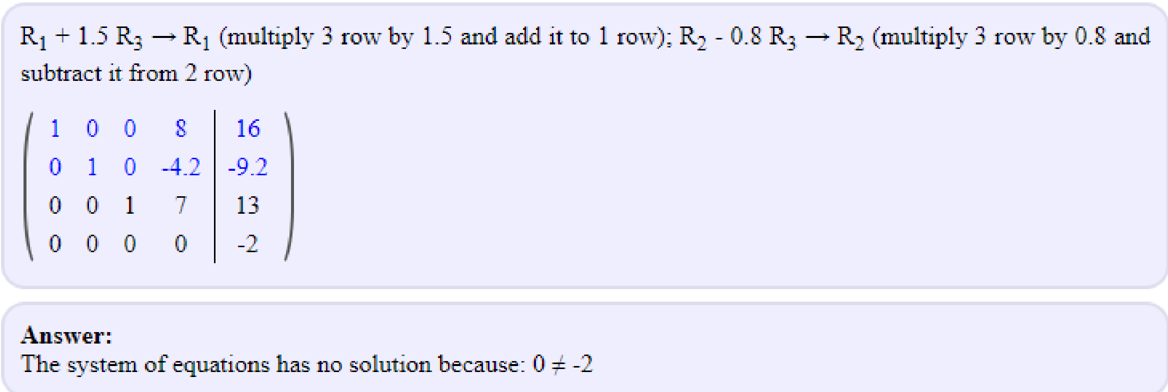
pav. 13 Programiškai gauti rezultatai

Tikriname ar rezultatai yra teisingi:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/



pav. 14 Tikrinimas naudojant išorinius šaltinius



pav. 15 Gauti rezultatai

Išvada: lygčių sistema neturi sprendinių, naudojant išorinius šaltinius rezultatai sutapo.

1.3 Tiesinės lygties sprendimas paprastųjų iteracijų metodu

1.3.1 **Sprendimo kodas**:

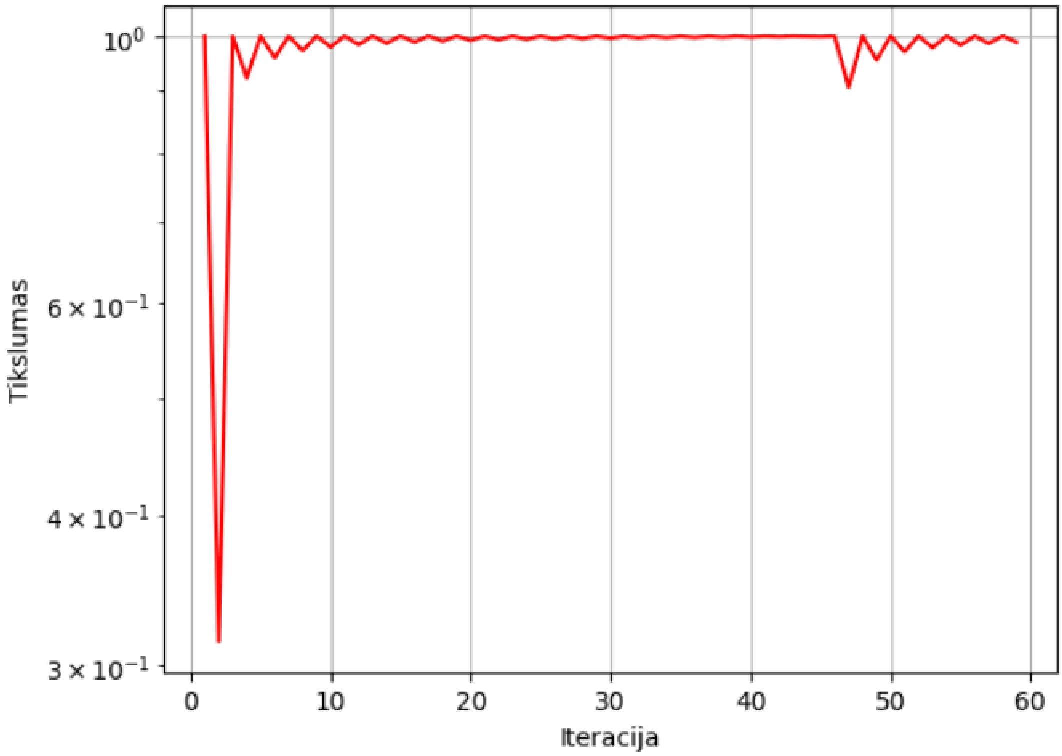
Paprastųjų iteracijų 2.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt

A = np.array([
    [3, 10, 1, 5],
    [-2, 6, 12, 14],
    [3, 12, 5, 1],
    [-3, -9, 5, 0]
], dtype=float)

b = np.array([83,178,37,-26], dtype=float)
epsilon = 1e-6
A = A + epsilon * np.eye(n)

n = A.shape[0]
Aprad = A.copy()

method = 'simple_iterations'
# method = 'Gauss-Seidel_iterations'
alpha = np.array([100, 20, 1, 1], dtype=float)  # Laisvai pasirenkami metodo
parametrai

Atld = np.diag(1. / np.diag(A)).dot(A) - np.diag(alpha)
btld = np.diag(1. / np.diag(A)).dot(b)

nitmax = 59
eps = 1e-12
x = np.zeros(n)
x1 = np.zeros(n)
prec = []

print('\nSprendimas iteracijomis:')
for it in range(nitmax):
    if method == 'Gauss-Seidel_iterations':
        for i in range(n):
            x1[i] = (btld[i] - np.dot(Atld[i, :], x1)) / alpha[i]
    elif method == 'simple_iterations':
        x1 = (btld - np.dot(Atld, x)) / alpha
    else:
        print('Neapibrėžtas metodas')
        break

    prec.append(np.linalg.norm(x1 - x) / (np.linalg.norm(x) + np.linalg.norm(x1)))
    print(f'Iteracija Nr. {it + 1}, Tikslumas: {prec[it]}')

    if prec[it] < eps:
        break

    x = x1

x_solution = x
print('Sprendinys:')
```

```
print(x_solution)
print('Patikrinimas:')
print(np.dot(Aprad, x_solution) - b)

plt.semilogy(range(1, len(prec) + 1), prec, 'r-')
plt.grid(True)
plt.xlabel('Iteracija')
plt.ylabel('Tikslumas')
plt.show()
```

1.3.2 **Programiškai gauti rezultatai**:

```
Iteracija Nr. 58, Tikslumas: 0.9999999971639698
Iteracija Nr. 59, Tikslumas: 0.9882687211780232
Sprendinys:
[-2.0000051   3.00000209 -1.00000169 11.99999961]
Patikrinimas:
[ 0.00000000e+00 -2.84217094e-14  0.00000000e+00 -1.06581410e-14]
```



pav. 16 Programiškai gauti rezultatai

Nustatyta, kad didžiausia iteracija yra 59.

1.3.3 **Sprendimo rezultatų tikrinimas**:

Tikriname ar rezultatai yra teisingi:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/

Make a check:

$3\cdot(-2) + 10\cdot3 + (-1) + 5\cdot12 = -6 + 30 - 1 + 60 = 83$
$-2\cdot(-2) + 6\cdot3 + 12\cdot(-1) + 14\cdot12 = 4 + 18 - 12 + 168 = 178$
$3\cdot(-2) + 12\cdot3 + 5\cdot(-1) + 12 = -6 + 36 - 5 + 12 = 37$
$-3\cdot(-2) - 9\cdot3 + 5\cdot(-1) = 6 - 27 - 5 = -26$

Check completed successfully.

Answer:

$$\begin{cases} x_1 = -2 \\ x_2 = 3 \\ x_3 = -1 \\ x_4 = 12 \end{cases}$$

pav. 17 Gauti rezultatai

Išvada: gauti tokie pat rezultatai kaip ir tikrinant kitais šaltiniais.

1.4 Tiesinių lygčių sprendimas sklaidos metodu

1.4.1 Sprendimo kodas

1b.ipynb:

```
import numpy as np

# -------------------iseities duomnenys:
A=np.matrix([[4 , 3,  -1,  1],
             [3, 9, -2,  -2],
             [-1,  -2, 11,  -1],
             [1,  -2,  -1, 5]]).astype(float)        # koeficientu matrica
Ap=A   # bus naudojama patikrinimui
b=(np.matrix([1,4,-10,3.5])).transpose().astype(float)   #laisvuju nariu vektorius-stulpelis
n=(np.shape(A))[0]   # lygciu skaicius nustatomas pagal ivesta matrica A
nb=(np.shape(b))[1]  # laisvuju nariu vektoriu skaicius nustatomas pagal ivesta matrica b

print(A,'A matrica');print(b,'b');print(n,'n');print(nb,'nb');

# tiesioginis etapas(QR skaida):
Q=np.identity(n)
for i in range (0,n-1):
    z=A[i:n,i] #vektorius
    zp=np.zeros(np.shape(z)); zp[0]=np.linalg.norm(z)
    print(zp,'zp')
    omega=z-zp; omega=omega/np.linalg.norm(omega)
    Qi=np.identity(n-i)-2*omega*omega.transpose()
    A[i:n,:]=Qi.dot(A[i:n,:])
    print(A,'A matrica')
    Q[:,i:n]=Q[:,i:n].dot(Qi)
    print(Q,'Q reikšmė')
    print(A,'R reikšmė')

    # atgalinis etapas:
b1=Q.transpose().dot(b);
x=np.zeros(shape=(n,nb));
for i in range (n-1,-1,-1):    # range pradeda n-1 ir baigia 0 (trecias parametras yra zingsnis)
    x[i,:]=(b1[i,:]-A[i,i+1:n]*x[i+1:n,:])/A[i,i];

    print(x,"x")

print('X lygus:');
print(x);
print("Naudotos Q ir R reiksmes:")
print('Q -', Q)
print('R -', A)
print("----------- sprendinio patikrinimas ---------------");
print('liekana:')
liekana=Ap.dot(x)-b1;print(liekana);
print('bendra santykine paklaida:')
print(np.linalg.norm(liekana)/ np.linalg.norm(x))
```

1.4.1 B1 rezultatai

Reikšmės:

5.	$\begin{cases} 4x_1 + 3x_2 - x_3 + x_4 = \cdots \\ 3x_1 + 9x_2 - 2\,x_3 - 2\,x_4 = \cdots \\ -x_1 - 2x_2 + 11x_3 - x_4 = \cdots \\ x_1 - 2x_2 - x_3 + 5x_4 = \cdots \end{cases}$	$\begin{cases} \ldots = 7 \\ \ldots = 8 \\ \ldots = 7 \\ \ldots = 3 \end{cases}$

pav. 18 B1 reikšmės

Programos išvesti rezultatai:

```
sprendinys:
[[1.]
 [1.]
 [1.]
 [1.]] x
[[-1.00000000e+00 -1.06090208e-16  3.78919505e-17 -1.19823553e-16]
 [ 1.06090208e-16 -1.00000000e+00  2.68482355e-17  1.23191833e-16]
 [-3.78919505e-17 -2.68482355e-17 -1.00000000e+00  3.11088283e-16]
 [-1.19823553e-16  1.23191833e-16  3.11088283e-16  1.00000000e+00]] Q
[[ -5.19615242  -7.5055535    4.23390197  -0.76980036]
 [  0.          -6.45497224   1.42870052   4.45823416]
 [  0.           0.         -10.34567006   1.55719467]
 [  0.           0.           0.          -2.84722678]] R
------------ sprendinio patikrinimas ----------------
[[ 0.00000000e+00]
 [-1.11022302e-15]
 [ 1.77635684e-15]
 [-3.10862447e-15]] liekana
1.874272183692197e-15 bendra santykine paklaida:
```

pav. 19 Programiškai gauti rezultatai

Tikrinimas:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/

Įstatome sprendinius į pradinę lygtį:

Make a check:

$4\cdot1 + 3\cdot1 - 1 + 1 = 4 + 3 - 1 + 1 = 7$
$3\cdot1 + 9\cdot1 - 2\cdot1 - 2\cdot1 = 3 + 9 - 2 - 2 = 8$
$-1 - 2\cdot1 + 11\cdot1 - 1 = -1 - 2 + 11 - 1 = 7$
$1 - 2\cdot1 - 1 + 5\cdot1 = 1 - 2 - 1 + 5 = 3$

Check completed successfully.

Answer:

$$\begin{cases} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1 \\ x_4 = 1 \end{cases}$$

pav. 20 Tikrinimas naudojant išorinius šaltinius

Išvada: lygčių sistema turi vieną sprendinį, rezultatai sutampa ir yra teisingi.

1.4.2 B2 rezultatai

Reikšmės:

$$\begin{cases} \ldots = 20 \\ \ldots = 0 \\ \ldots = 18 \\ \ldots = 40 \end{cases}$$

pav. 21 B2 reikšmės

Programos išvesti rezultatai:

```
sprendinys:
[[2.]
 [2.]
 [3.]
 [9.]] x
[[-1.00000000e+00 -1.06090208e-16  3.78919505e-17 -1.19823553e-16]
 [ 1.06090208e-16 -1.00000000e+00  2.68482355e-17  1.23191833e-16]
 [-3.78919505e-17 -2.68482355e-17 -1.00000000e+00  3.11088283e-16]
 [-1.19823553e-16  1.23191833e-16  3.11088283e-16  1.00000000e+00]] Q
[[ -5.19615242  -7.5055535    4.23390197  -0.76980036]
 [  0.          -6.45497224   1.42870052   4.45823416]
 [  0.           0.         -10.34567006   1.55719467]
 [  0.           0.           0.          -2.84722678]] R
------------ sprendinio patikrinimas ----------------
[[ 0.00000000e+00]
 [ 0.00000000e+00]
 [ 3.55271368e-15]
 [-7.10542736e-15]] liekana
8.024762213889601e-16 bendra santykine paklaida:
```

pav. 22 Programiškai gauti rezultatai

Tikrinimas:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/

Įstatome sprendinius į pradinę lygtį:

Make a check:

4·2 + 3·2 - 3 + 9 = 8 + 6 - 3 + 9 = 20
3·2 + 9·2 - 2·3 - 2·9 = 6 + 18 - 6 - 18 = 0
-2 - 2·2 + 11·3 - 9 = -2 - 4 + 33 - 9 = 18
2 - 2·2 - 3 + 5·9 = 2 - 4 - 3 + 45 = 40

Check completed successfully.

Answer:

$$\begin{cases} x_1 = 2 \\ x_2 = 2 \\ x_3 = 3 \\ x_4 = 9 \end{cases}$$

pav. 23 Gauti rezultatai

Išvada: lygčių sistema turi vieną sprendinį, rezultatai sutampa ir yra teisingi.

1.4.3 B3 rezultatai

Reikšmės:

$$\begin{cases} ... = 1 \\ ... = 4 \\ ... = -10 \\ ... = 3.5 \end{cases}$$

pav. 24 B3 reikšmės

Programos išvesti rezultatai:

```
sprendinys:
[[-0.75]
 [ 0.75]
 [-0.75]
 [ 1.  ]] x
[[-1.00000000e+00 -1.06090208e-16  3.78919505e-17 -1.19823553e-16]
 [ 1.06090208e-16 -1.00000000e+00  2.68482355e-17  1.23191833e-16]
 [-3.78919505e-17 -2.68482355e-17 -1.00000000e+00  3.11088283e-16]
 [-1.19823553e-16  1.23191833e-16  3.11088283e-16  1.00000000e+00]] Q
[[ -5.19615242  -7.5055535    4.23390197  -0.76980036]
 [  0.          -6.45497224   1.42870052   4.45823416]
 [  0.           0.         -10.34567006   1.55719467]
 [  0.           0.           0.          -2.84722678]] R
------------ sprendinio patikrinimas ----------------
[[0.00000000e+00]
 [2.22044605e-16]
 [0.00000000e+00]
 [2.22044605e-15]] liekana
1.3612148423915415e-15 bendra santykine paklaida:
```

pav. 25 Programiškai gauti rezultatai

Tikrinimas:

Tikrinimui naudojame: https://onlinemschool.com/math/assistance/equation/gaus/

Įstatome sprendinius į pradinę lygtį:

Make a check:

4·(-0.75) + 3·0.75 - (-0.75) + 1 = -3 + 2.25 + 0.75 + 1 = 1
3·(-0.75) + 9·0.75 - 2·(-0.75) - 2·1 = -2.25 + 6.75 + 1.5 - 2 = 4
-(-0.75) - 2·0.75 + 11·(-0.75) - 1 = 0.75 - 1.5 - 8.25 - 1 = -10
(-0.75) - 2·0.75 - (-0.75) + 5·1 = -0.75 - 1.5 + 0.75 + 5 = 3.5

Check completed successfully.

Answer:

$$\begin{cases} x_1 = -0.75 \\ x_2 = 0.75 \\ x_3 = -0.75 \\ x_4 = 1 \end{cases}$$

pav. 26 Gauti rezultatai

Išvada: lygčių sistema turi vieną sprendinį, rezultatai sutampa ir yra teisingi.

Matas Palujanskas

2. Antra užduoties dalis

2.1. Užduoties sąlygos

2 Netiesinių lygčių sistemų sprendimas

Duota netiesinių lygčių sistema (4 lentelė):

$$\begin{cases} Z_1(x_1, x_2) = 0 \\ Z_2(x_1, x_2) = 0 \end{cases}$$

a. Skirtinguose grafikuose pavaizduokite paviršius $Z_1(x_1, x_2)$ ir $Z_2(x_1, x_2)$.
b. Užduotyje pateiktą netiesinių lygčių sistemą išspręskite grafiniu būdu.
c. Nagrinėjamoje srityje sudarykite stačiakampį tinklelį (x_1, x_2 poras). Naudodami užduotyje nurodytą metodą apskaičiuokite netiesinių lygčių sistemos sprendinius, kai pradinis artinys įgyja tinklelio koordinačių reikšmes. Tinklelyje vienodai pažymėkite taškus, kuriuos naudojant kaip pradinius artinius gaunamas tas pats sprendinys. Lentelėje pateikite apskaičiuotus skirtingus sistemos sprendinius ir bent po vieną jam atitinkantį pradinį artinį.
d. Gautus sprendinius patikrinkite naudodami išorinius išteklius (pvz., standartines Python funkcijas).

pav. 27 Antros dalies sąlygos

5 varianto netiesinės lygčių sistemos:

5	$\begin{cases} \dfrac{x_1^2}{(x_2 + \cos(x_1))^2 + 1} - 2 = 0 \\ \left(\dfrac{x_1}{3}\right)^2 + (x_2 + \cos(x_1))^2 - 5 = 0 \end{cases}$	Niutono

pav. 28 Netiesinės lygčių sistemos

2.2. Sprendimo kodas

Netiesiniu lygciu sprendimas.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.optimize import fsolve

# Niutono - Rafsono metodo naudojimas nelinearinių lygčių sistemoms spresti.
# Pagrindinė idėja: Jakobiano matrica * delta(x) = -f(x) => gauname delta(x) => x = x
+ delta(x)

# Lygčių sistema ir jos Jakobiano matricos apibrėžimas:

def LF_scipy(vars):
    x1, x2 = vars
    eq1 = x1**2 / ((x2 + math.cos(x1))**2 + 1) - 2
    eq2 = (x1/3)**2 + (x2 + math.cos(x1))**2 - 5
    return [eq1, eq2]

def LF(x):
    s = np.array([x[0]**2 / ((x[1] + math.cos(x[0]))**2 + 1) - 2, (x[0]/3)**2 + (x[1]
```

```
+ math.cos(x[0]))**2 - 5])
    return s


#jakobio matrica
def DLF(x):
    df1_dx0 = (2 * x[0] * (x[1] + math.cos(x[0]))**2) / (((x[1] + math.cos(x[0]))**2
+ 1)**2)
    df1_dx1 = -2 * x[0] * (x[1] + math.cos(x[0])) * math.sin(x[0]) / (((x[1] +
math.cos(x[0]))**2 + 1)**2)
    df2_dx0 = (2/9) * (x[0]**2 - 9 * (x[1] + math.cos(x[0]))**2)
    df2_dx1 = 2 * (x[1] + math.cos(x[0]))

    J = np.array([[df1_dx0, df1_dx1], [df2_dx0, df2_dx1]])
    return J

# Vizualizacijos funkcijos:

def braizom_viena(F, numb,fun_color, cont_color, title):
    fig1 = plt.figure(1, figsize=plt.figaspect(0.5))
    ax1 = fig1.add_subplot(1, 1, 1, projection='3d')
    plt.title(title)
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('z')
    xx = np.linspace(-10, 10, 100)
    yy = np.linspace(-10, 10, 100)
    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, :] = F([X[i][j], Y[i][j]]).transpose()
    surf1 = ax1.plot_surface(X, Y, Z[:, :, numb], color=fun_color, alpha=0.4)
    CS11 = ax1.contour(X, Y, Z[:, :, numb], [0], colors=cont_color)
    plt.show()

def braizom_sistema(F, title):
    fig1=plt.figure(1,figsize=plt.figaspect(0.5))
    ax1 = fig1.add_subplot(1, 2, 1, projection='3d')
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('z')
    ax1.set_title(title)
    ax2 = fig1.add_subplot(1, 2, 2, projection='3d')
    ax2.set_xlabel('x1')
    ax2.set_ylabel('x2')
    ax2.set_zlabel('z')
    #ax2.set_title(title)
    #ax2.set_title("Kontūrais, ties kuriais funkcijos kerta Z=0 plokštumą")
    plt.draw()
    xx = np.linspace(-10, 10, 100)
    yy = np.linspace(-10, 10, 100)
    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, :] = F([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')
```

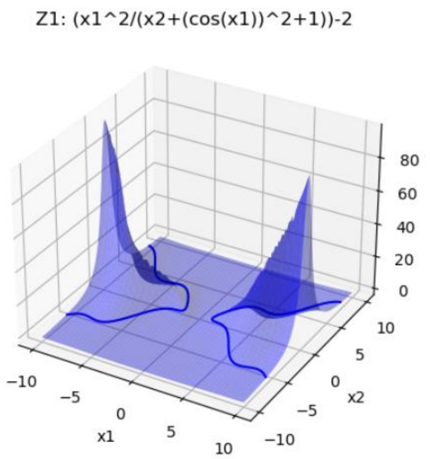
```
    CS1 = ax2.contour(X, Y, Z[:, :, 0], [0], colors='b')
    CS2 = ax2.contour(X, Y, Z[:, :, 1], [0], colors='g')
    plt.show()

# Niutono metodo funkcijos:
def Niutono(DF, F, iter_max, alpha, x0, eps, print_var):
    for i in range(iter_max):
        try:
            if np.any(np.abs(x0) > 1e50):
                return [math.inf, math.inf]
            ff = F(x0)
            dff = DF(x0)
            deltax = -np.linalg.solve(dff, ff)
            x1 = x0 + alpha * deltax
            precision = np.linalg.norm(deltax) / (np.linalg.norm(x0) +
np.linalg.norm(deltax))
            if precision < eps:
                if(print_var):
                    print(f"Konvergavo. sprendinys x = {x0}")
                return x0
            elif i == iter_max - 1:
                if (print_var):
                    print(f"Sprendinys pasiekė iteracijų limitą. paskutinis x =
{x0}")
                return [math.inf, math.inf]
            x0 = x1
        except:
            return [math.inf, math.inf]

# Tikrina, ar sprendiniai unikalūs:

def is_solution_unique(solution, solutions_list):
    for sol_dict in solutions_list:
        sol = sol_dict["solution"]
        if any(np.isclose(x, sol, atol=1e-5)):
            return False
    return True

# Inicializuoja pradinių artinių tinklelį vizualizacijai:

def init_paint(sprendiniai):
    fig1 = plt.figure(1, figsize=plt.figaspect(0.5))
    ax2 = fig1.add_subplot(1, 1, 1)
    ax2.set_xlabel('x1')
    ax2.set_ylabel('x2')
    ax2.set_title("Pradinių artinių tinklelis")
    plt.draw()
    xx = np.linspace(-10, 10, 100)
    yy = np.linspace(-10, 10, 100)
    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()
    CS1 = ax2.contour(X, Y, Z[:, :, 0], [0], colors='lime', linewidths=2, zorder=2)
    CS2 = ax2.contour(X, Y, Z[:, :, 1], [0], colors='lime', linewidths=2, zorder=2)
    for sol_dict in sprendiniai:
        sol = sol_dict["solution"]
        ax2.scatter(sol[0], sol[1], marker="P", color=sol_dict["color"],
linewidths=1, s=50, edgecolor='black',zorder=3)
    return ax2
```

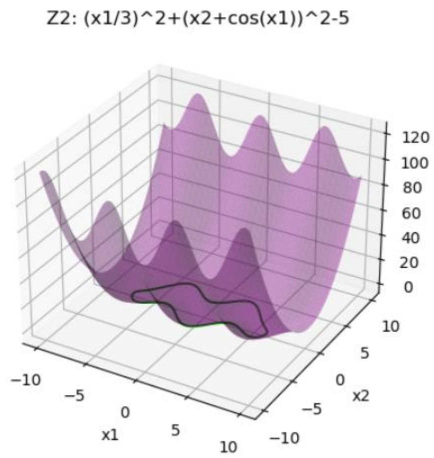
```
if __name__ == '__main__':
    # Vizualizacija: braižomi funkcijų grafikai ir kontūrai.
    braizom_viena(LF,0,'blue','b', "Z1: (x1^2/(x2+(cos(x1))^2+1))-2")
    braizom_viena(LF,1,'purple','g', "Z2: (x1/3)^2+(x2+cos(x1))^2-5")
    print("Kontūrais, ties kuriais funkcijos kerta Z=0 plokštumą:")
    braizom_sistema(LF,"(x1^2/(x2+(cos(x1))^2+1)-2 ir (x1/3)^2+(x2+cos(x1))^2-5")

    # Parametrai Niutono metodui:
    eps = 1e-10
    alpha = 1
    itmax = 4

    #netiesinė lygciu sistema grafiškai išsprendus turi 4 sprendinius. ju artinius
apsirašome:
    print("Netiesinių lygčių sistemos sprendiniai")
    Artiniai = [[-3, 3], [-3, -1], [3, 3], [3, -1]]
    #Artiniai = [[0, 0]]
    Sprendiniai = []
    spalvos = ['b', 'g', 'r', 'c', 'm', 'y']
    ind=0
    for art in Artiniai:
        x = Niutono(DLF, LF, itmax, alpha, art, eps, True)
        if x[0] != math.inf and x[1] != math.inf and is_solution_unique(x,
Sprendiniai):
            Sprendiniai.append({"solution": x, "color": spalvos[ind]})
            ind=ind+1

    if(len(Artiniai) == len(Sprendiniai)):
        print("visi Sprendiniai yra unikalūs")
    else:
        print("visi Sprendiniai yra unikalūs")
        #print("Yra vienodų sprendinių")

    #braizome tinkleli
    ax = init_paint(Sprendiniai)

    for i in range(-10, 11):
        for j in range(-10, 11):
            x = Niutono(DLF, LF, itmax, alpha, np.array([i, j]), eps, False)
            if x[0] != math.inf and x[1] != math.inf:
                for sol_dict in Sprendiniai:
                    sol = sol_dict["solution"]
                    if any(np.isclose(x, sol, atol=1e-5)):
                        ax.scatter(i, j, marker="o", c=sol_dict["color"], zorder=1,
s=100, edgecolor='black', linewidths=2)
            elif i == 0:  # Tikriname, ar taškas yra per centrą ir yra vertikalias
                ax.scatter(i, j, marker="o", c="black", zorder=1, s=100,
edgecolor='black', linewidths=2)
            else:
                ax.scatter(i, j, marker="o", c="purple", zorder=1, s=100,
edgecolor='black', linewidths=2)

    plt.show()

    #Tikrinimas su scipy
    scipy_sprendiniai = []
    A = Sprendiniai
    for art in Artiniai:
        x1, x2 = fsolve(LF_scipy, (art[0], art[1]))
```

```
        #if is_solution_unique_scipy([x1, x2], scipy_sprendiniai):
        scipy_sprendiniai.append([x1, x2])

    sprend = []
    for sp in Sprendiniai:
        sprend.append([sp["solution"][0], sp["solution"][1]])

    if len(sprend) == len(scipy_sprendiniai):
        sprend.sort(key=lambda x: (x[1], x[0]))
        scipy_sprendiniai.sort(key=lambda x: (x[1], x[0]))
        visi_vienodi = True
        for i in range(len(scipy_sprendiniai)):
            print(f"niutono sprendinys - {sprend[i]} ir scipy sprendinys -
{scipy_sprendiniai[i]}")
            if not (np.isclose(sprend[i][0], scipy_sprendiniai[i][0], atol=1e-5) and
np.isclose(sprend[i][1], scipy_sprendiniai[i][1], atol=1e-5)):
                visi_vienodi = False
        print(f"Ar visi vienodi? - {visi_vienodi}")
    else:
        print("Sąrašų ilgiai skiriasi. Negalima lyginti.")
```

2.3. Gauti rezultatai

Atvaizduoti Z1 ir Z2 paviršiai skirtinguose grafikuose:





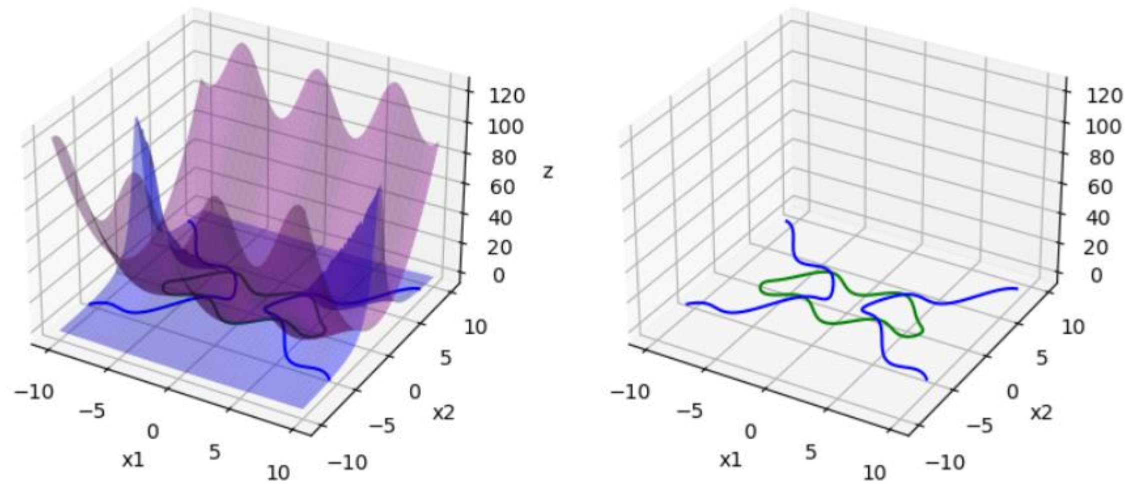
pav. 29 Z1 ir Z2

Matas Palujanskas

Abu paviršiai viename grafike:

Kontūrais, ties kuriais funkcijos kerta Z=0 plokštumą:

(x1^2/(x2+(cos(x1))^2+1)-2 ir (x1/3)^2+(x2+cos(x1))^2-5



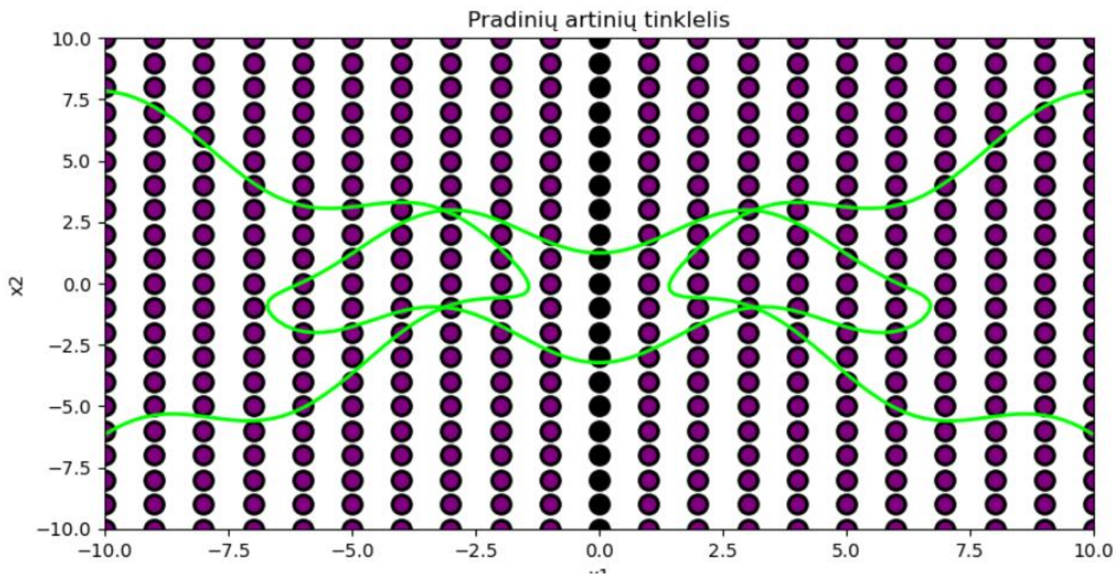
pav. 30 Z1 ir Z2 viename grafike

Gauti rezultatai, patikrinta su scipy:

```
Netiesinių lygčių sistemos sprendiniai
Sprendinys pasiekė iteracijų limitą. paskutinis x = [-4.00401947 -0.31460859]
Sprendinys pasiekė iteracijų limitą. paskutinis x = [-3.64035779  0.65629797]
Sprendinys pasiekė iteracijų limitą. paskutinis x = [4.50582209 4.90949658]
Sprendinys pasiekė iteracijų limitą. paskutinis x = [ 4.08090508 -2.01865963]
visi Sprendiniai yra unikalūs
```

pav. 31 Rezultatai

Grafinis sprendimas:



pav. 32 Grafinis sprendimas

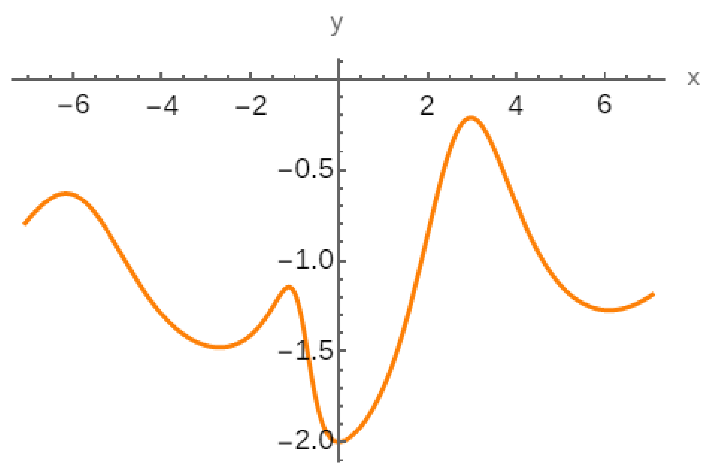
2.4. Tikrinimas

Tikrinimui naudota Wolfram Alpha aplinka.

Input:

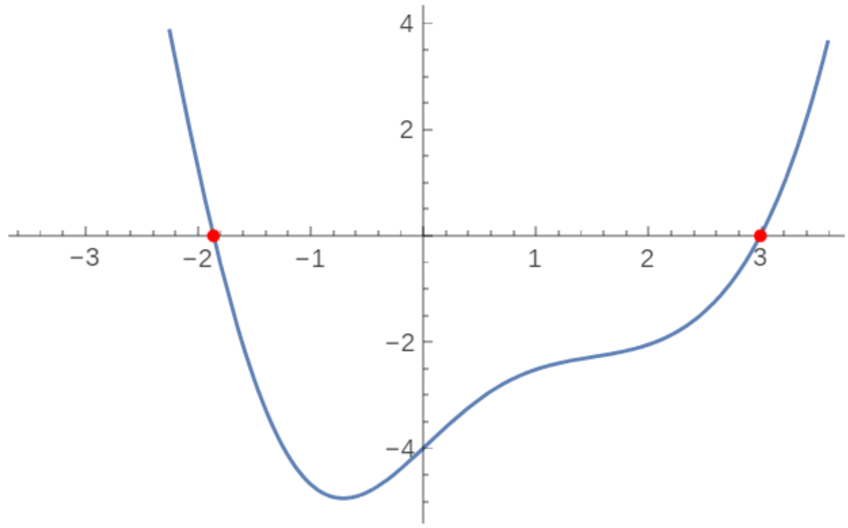
$$\frac{x^2}{(x+\cos(x))^2+1}-2$$

Plots:



pav. 33 Z2 lygtis

Root plot:



pav. 34 Z1 lygtis

Išvada: Gauti grafikai sutampa.

3. Trečia dalis

3.1 Užduoties sąlyga

3 Optimizavimas

Pagal pateiktą uždavinio sąlygą (5 lentelė) sudarykite tikslo funkciją ir išspręskite jį vienu iš gradientinių metodų (gradientiniu, greičiausio nusileidimo). Gautą taškų konfigūraciją pavaizduokite programoje, skirtingais ženklais pavaizduokite duotus ir pridėtus (jei sąlygoje tokių yra) taškus. Ataskaitoje pateikite pradinę ir gautą taškų konfigūracijas, taikytos tikslo funkcijos aprašymą, taikyto metodo pavadinimą ir parametrus, iteracijų skaičių, iteracijų pabaigos sąlygas ir tikslo funkcijos priklausomybės nuo iteracijų skaičiaus grafiką.

pav. 35 Trečios dalies sąlyga

Mano variantas yra 5, todėl gautas šis uždavinys:

Uždavinys 4-6 variantams

Miestas išsidėstęs kvadrate, kurio koordinatės ($-10 \le \boldsymbol{x} \le 10,\ -10 \le \boldsymbol{y} \le 10$). Mieste yra **n** ($\boldsymbol{n} \ge \mathbf{3}$) vieno tinklo parduotuvių, kurių koordinatės yra žinomos (*Koordinatės gali būti generuojamos atsitiktinai, negali būti kelios parduotuvės toje pačioje vietoje*). Planuojama pastatyti dar **m** ($\boldsymbol{m} \ge \mathbf{3}$) šio tinklo parduotuvių. Parduotuvės pastatymo kaina (vietos netinkamumas) vertinama pagal atstumus iki kitų parduotuvių ir miesto ribos. Reikia parinkti naujų parduotuvių vietas (koordinates) taip, kad parduotuvių pastatymo kainų suma būtų kuo mažesnė.

Atstumo tarp dviejų parduotuvių, kurių koordinatės $(\boldsymbol{x_1}, \boldsymbol{y_1})$ ir $(\boldsymbol{x_2}, \boldsymbol{y_2})$, kaina apskaičiuojama pagal formulę:

$$C(x_1, y_1, x_2, y_2) = \exp(-0.1 \cdot ((x_1 - x_2)^2 + (y_1 - y_2)^2))$$

Atstumo tarp parduotuvės, kurios koordinatės $(\boldsymbol{x_1}, \boldsymbol{y_1})$, ir artimiausio miesto ribos taško, kurio koordinatės $(\boldsymbol{x_r}, \boldsymbol{y_r})$, kaina apskaičiuojama pagal formulę:

$$C^R(x_1, y_1, x_r, y_r) = \begin{cases} 0, \mathit{jeigu\ parduotuvę\ planuojama\ statyti\ miesto\ ribose} \\ 0.5 \cdot ((x_1 - x_r)^2 + (y_1 - y_r)^2), \mathit{kitais\ atvejais} \end{cases}$$

pav. 36 Gautas uždavinys

3.2 Sprendimo kodas

3 uzduotis.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt

# Pradinės sąlygos
n = 5  # Parduotuvių skaičius
m = 3  # Planuojamų parduotuvių skaičius
miesto_ribos = 10  # Miesto ribos dydis

# Sugeneruojame pradines parduotuvių koordinates
pradines_parduotuves = np.random.rand(n, 2) * 20 - 10  # Koordinatės nuo -10 iki 10
```

```
# Sugeneruojame pradines naujų parduotuvių koordinates
naujos_parduotuves = np.random.rand(m, 2) * 20 - 10  # Koordinatės nuo -10 iki 10

# Apskaičiuojame pastatytų parduotuvių vietos netinkamumo kainą
def pastatytu_parduotuviu_kaina(parduotuves):
    kaina = 0
    for i in range(len(parduotuves)):
        for j in range(len(parduotuves)):
            if i != j:
                kaina += np.exp(-0.1 * ((parduotuves[i][0] - parduotuves[j][0])**2 +
(parduotuves[i][1] - parduotuves[j][1])**2))
        kaina += 0.5 * ((miesto_ribos - parduotuves[i][0])**2 + (miesto_ribos -
parduotuves[i][1])**2)
    return kaina

# Gradientinio nusileidimo metodas
def gradientinis_nusileidimas(parduotuves, m, miesto_ribos, mokymo_zingsnis,
iteraciju_skaicius):
    kainos = []
    for _ in range(iteraciju_skaicius):
        gradientas = np.zeros((m, 2))
        for i in range(m):
            for j in range(m):
                if i != j:
                    gradientas[i] += 0.2 * (parduotuves[i] - parduotuves[j]) *
np.exp(-0.1 * ((parduotuves[i][0] - parduotuves[j][0])**2 + (parduotuves[i][1] -
parduotuves[j][1])**2))
            gradientas[i] += 0.1 * (parduotuves[i] - miesto_ribos) * np.exp(-0.1 *
((parduotuves[i][0] - miesto_ribos)**2 + (parduotuves[i][1] - miesto_ribos)**2))
        parduotuves -= mokymo_zingsnis * gradientas
        kainos.append(pastatytu_parduotuviu_kaina(parduotuves))
    return parduotuves, kainos

# Nustatome optimizavimo parametrus
mokymo_zingsnis = 0.5
iteraciju_skaicius = 100

# Vykdome gradientinio nusileidimo metodą
parduotuves, kainos = gradientinis_nusileidimas(naujos_parduotuves, m, miesto_ribos,
mokymo_zingsnis, iteraciju_skaicius)

# Spausdiname pradinių parduotuvių koordinates ir jų pastatymo kainas
print('Pradinės parduotuves:')
for i, (x, y) in enumerate(pradines_parduotuves):
    kaina = pastatytu_parduotuviu_kaina(pradines_parduotuves)
    print(f"Parduotuvė {i + 1}: Koordinatės ({x:.2f}, {y:.2f}), Pastatymo kaina:
{kaina:.2f}")

# Spausdiname naujų parduotuvių koordinates ir jų pastatymo kainas
print('Naujos parduotuves:')
for i, (x, y) in enumerate(parduotuves):
    kaina = pastatytu_parduotuviu_kaina(parduotuves)
    print(f"Parduotuvė {i + 1}: Koordinatės ({x:.2f}, {y:.2f}), Pastatymo kaina:
{kaina:.2f}")

# Pavaizduojame rezultatus
plt.scatter(pradines_parduotuves[:, 0], pradines_parduotuves[:, 1], c='r',
marker='x', label='Pradinės parduotuvės')
plt.scatter(parduotuves[:, 0], parduotuves[:, 1], c='g', marker='s', label='Naujos
optimizuotos parduotuvės')
```

```
plt.xlim(-11, 11)
plt.ylim(-11, 11)
plt.legend()
plt.title("Parduotuvių vietos")
plt.show()

plt.plot(kainos)
plt.xlabel('Iteracijos')
plt.ylabel('Kaina')
plt.title('Tikslo funkcijos priklausomybė nuo iteracijų skaičiaus')
plt.show()

#Tikriname iteracijų pabaigos sąlygas ir spausdiname rezultatą
if len(kainos) >= 2 and abs(kainos[-1] - kainos[-2]) < 0.01:
    print("Optimizacija pasieka stabilumą.")
elif len(kainos) >= iteraciju_skaicius:
    print("Pasiektas maksimalus iteracijų skaičius.")
else:
    print("Optimizacija baigėsi dėl kitos priežasties.")
```

3.3 Aprašymas

1) Pradinės sąlygos:
   - Parduotuvių skaičius (n): 5
   - Planuojamų parduotuvių skaičius (m): 3
   - Miesto ribos dydis: 10

2) Pradinės taškų konfigūracijos:
   - Sugeneruotos pradinės parduotuvės: 5 atsitiktinai sugeneruotos parduotuvės su koordinatėmis nuo -10 iki 10.
   - Sugeneruotos naujos parduotuvės: 3 atsitiktinai sugeneruotos parduotuvės su koordinatėmis nuo -10 iki 10.

3) Tikslo funkcijos aprašymas:

Programoje taikoma tikslo funkcija, kuri apskaičiuoja kainą, susijusią su pastatytomis parduotuvėmis. Ši funkcija apima du komponentus:

   - Komponentas, apibūdinantis atstumų tarp parduotuvių įtaką kainai. Jis pagrįstas eksponentinio sumažinimo funkcija, kurioje atstumas tarp parduotuvių mažinant apie 0, taip pat užima parduotuvės vietą.
   - Komponentas, apibūdinantis atstumą nuo parduotuvės iki miesto ribos ir užimantį parduotuvės vietą. Jis taip pat pagrįstas eksponentinio sumažinimo funkcija.

4) Taikytas metodas:

Naudojamas gradientinio nusileidimo metodas parduotuvių vietos optimizavimui.

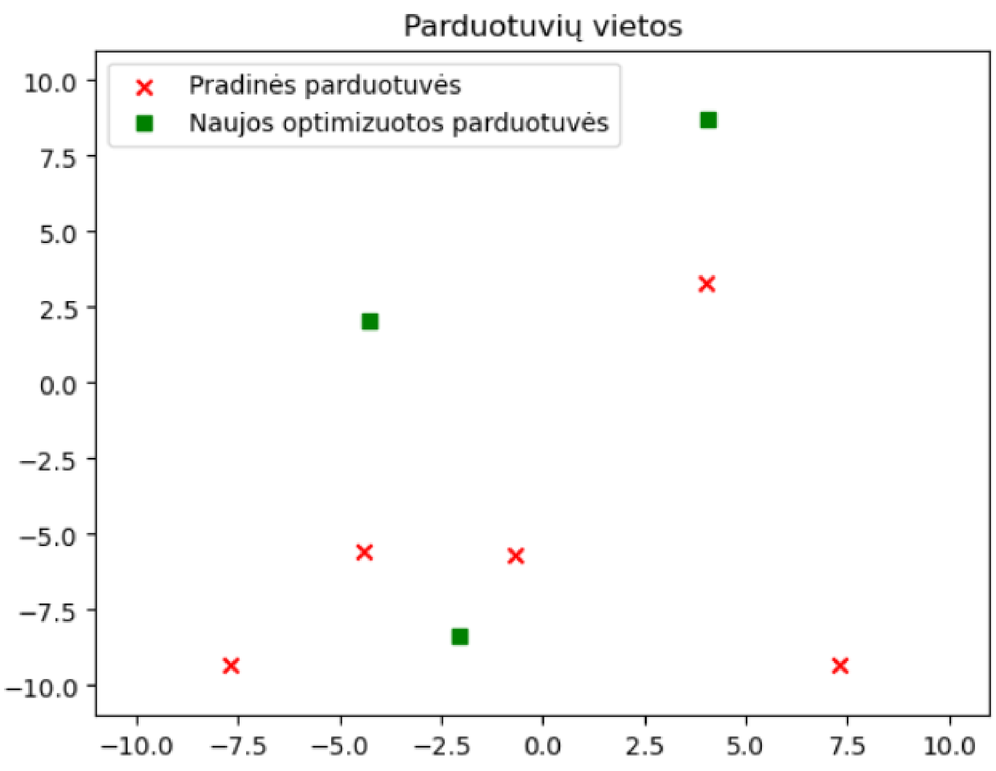
5) Optimizacijos parametrai:
    - Mokymo žingsnis: 0.01
    - Iteracijų skaičius: 1000

6) Iteracijos pabaigos sąlygos:
    - Programa vykdo 1000 iteracijų

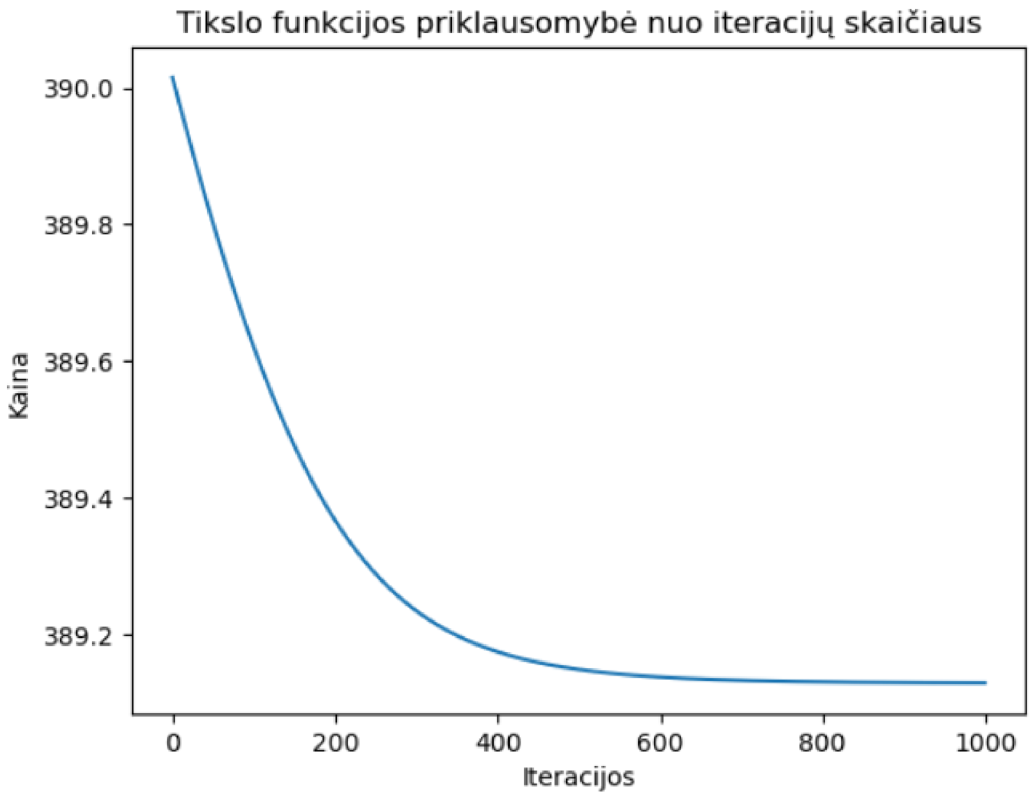
3.4 Gauti rezultatai

```
Pradinės parduotuves:
Parduotuvė 1: Koordinatės (-4.40, -5.60), Pastatymo kaina: 980.45
Parduotuvė 2: Koordinatės (-7.68, -9.34), Pastatymo kaina: 980.45
Parduotuvė 3: Koordinatės (-0.67, -5.70), Pastatymo kaina: 980.45
Parduotuvė 4: Koordinatės (4.01, 3.31), Pastatymo kaina: 980.45
Parduotuvė 5: Koordinatės (7.32, -9.34), Pastatymo kaina: 980.45
Naujos parduotuves:
Parduotuvė 1: Koordinatės (-2.07, -8.38), Pastatymo kaina: 393.75
Parduotuvė 2: Koordinatės (-4.27, 2.03), Pastatymo kaina: 393.75
Parduotuvė 3: Koordinatės (4.06, 8.69), Pastatymo kaina: 393.75
```

pav. 37 Programiškai gauti rezultatai



pav. 38 Grafiškai atvaizduotos pradinės ir naujos parduotuvės




```
Optimizacija pasieka stabilumą.
```

pav. 39 Tikslo funkcijos priklausomybės nuo iteracijų skaičiaus grafikas

Išvada: gautos naujų optimizuotų parduotuvių vietos ir kainos, keičiant optimizavimo parametrus skyrėsi ir gauti rezultatai, rezultatai buvo skirtingi ir todėl, nes kiekvieną kartą buvo generuojama su atsitiktinai pasirinktomis pradinėmis koordinatėmis. Geriausi rezultatai buvo pasiekti su 1000 iteracijų skaičiumi.

4. Literatūros sąrašas

1. „Skaitiniai metodai ir algoritmai“ „Moodle“ aplinkoje HTTPS://MOODLE.KTU.EDU/COURSE/VIEW.PHP?ID=7639