KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

Laboratorinis darbas nr. 2

Variantas 12

Dėstytojai:
Lekt. Dalia Čalnerytė

Studentas:
Justas Milišiūnas IFF-7/2

KAUNAS, 2019

Turinys

1. Įvadas

Šio laboratorinio darbo esmė išmokti skaičiuoti lygčių sistemų nežinomuosius sprendinius pasinaudojant kompiuteriu. Jas sprendžiant grafiškai arba naudojant kokį nors metodą.

2. Užduotis

2.1. Tiesinių lygčių sistemu sprendimas

Užduotis:

Duota tiesinių lygčių sistema [A][X] = [B] ir jos sprendimui nurodytas metodas (1 lentelė).

1. Išspręskite tiesinių lygčių sistemą. Jeigu sprendinių be galo daug, raskite bent vieną iš jų. Jeigu sprendinių nėra, pagrįskite, kodėl taip yra.
   Jei metodas paremtas matricos pertvarkymu, pateikite matricų išraiškas kiekviename žingsnyje. Jei metodas iteracinis, grafiškai pavaizduokite, kaip atliekant iteracijas kinta santykinis sprendinio tikslumas esant kelioms skirtingoms konvergavimo daugiklio reikšmėms.
2. Patikrinkite gautus sprendinius ir skaidas, įrašydami juos į pradinę lygčių sistemą.
3. Gautą sprendinį patikrinkite naudodami išorinius išteklius (pvz., standartines MATLAB funkcijas).

pav. 1 Pirmoji užduotis

Mano varianto lygčių sistema:

12	$\begin{cases} 3x_1 + x_2 - x_3 + x_4 = 27 \\ x_1 - 2x_2 + 3x_3 + x_4 = 24 \\ 2x_1 - 9x_2 + 5x_3 + 2x_4 = 27 \\ x_1 - 7x_2 + 2x_3 + x_4 = 3 \end{cases}$	QR skaidos

pav. 2 12 varianto pirmos užduoties lygčių sistema

Šios matricos sprendimo rezultatas naudojant QR skaidos metod:
x =
[[-2.13310334]
 [3.96809588]
 [1.15952025]
 [30.59073423]]

Šio sprendinio patikrinimas:

$$\begin{cases} 3 * (-2.13310334) + 3.96809588 - 1.15952025 + 30.59073423 = 26.999 \\ -2.13310334 - 2 * 3.96809588 + 3 * 1.15952025 + 30.59073423 = 23.999 \\ 2 * -2.13310334 - 9 * 3.96809588 + 5 * 1.15952025 + 2 * 30.59073423 = 27 \\ -2.13310334 - 7 * 3.96809588 + 2 * 1.15952025 + 30.59073423 = 3 \end{cases}$$

Iš čio patikrinimo matome, kad sprendinys yra geras.

Šios sistemos sprendimas naudojant išorinius išteklius (python numpy):
x = [-28.52631579 6.26315789 -10.31578947 96.]
Gautas kitos x sprendinys, bet patikrinus jis irgi tinka.

2.2. Netiesnių lygčių sistemų sprendimas

Užduotis:

1. Duota netiesinių lygčių sistema (2 lentelė. I lygčių sistema):

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

   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. Užduotyje pateiktą netiesinių lygčių sistemą išspręskite naudodami užduotyje nurodytą metodą su laisvai pasirinktu pradiniu artiniu (išbandykite bent keturis pradinius artinius). Nurodykite iteracijų pabaigos sąlygas. Lentelėje pateikite pradinį artinį, tikslumą, iteracijų skaičių.
   d. Gautus sprendinius patikrinkite naudodami išorinius išteklius (pvz., standartines MATLAB funkcijas).

2. Duota netiesinių lygčių sistema (2 lentelė. II lygčių sistema):

$$\begin{cases} Z_1(x_1, x_2, x_3, x_4) = 0 \\ Z_2(x_1, x_2, x_3, x_4) = 0 \\ Z_3(x_1, x_2, x_3, x_4) = 0 \\ Z_4(x_1, x_2, x_3, x_4) = 0 \end{cases}$$

   a. Užduotyje nurodytu metodu išspręskite netiesinių lygčių sistemą su laisvai pasirinktu pradiniu artiniu.
   b. Gautą sprendinį patikrinkite naudodami išorinius išteklius (pvz., standartines MATLAB funkcijas).

pav. 3 2 užduotis

Mano varianto lygčių sistema:

Nr.	I lygčių sistema	II lygčių sistema	Metodas
12	$\begin{cases} x_1^2 + 10(\sin(x_1) + \cos(x_2))^2 - 10 = 0 \\ (x_2 - 3)^2 + x_1 - 8 = 0 \end{cases}$	$\begin{cases} 5x_1 + x_2 + x_3 + 4x_4 + 5 = 0 \\ -x_1^2 + x_3^2 + 5 = 0 \\ 4x_3^3 - x_4^2 - 3x_2x_4 - 28 = 0 \\ x_1 - 3x_2 + 4x_3 - x_4 - 3 = 0 \end{cases}$	Niutono

pav. 4 12 varianto 2 uždtuoties lygčių sistemos

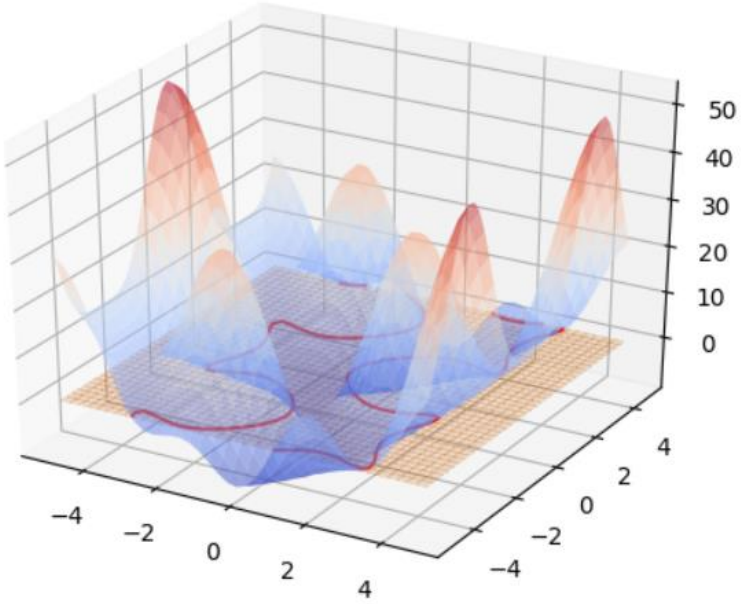
QR skaidos metodo kodas:

```
def qr_decomposition(a):
    # Get dimension of matrix.
    n = len(a)
    # Create a copy of the matrix.
    cp = a.copy()

    # Initialize 2 matrices Q and R.
    Q = np.zeros(shape=(n, n))
    R = np.zeros(shape=(n, n))

    for i in range(n):
        u = a[:, i]
        u = u.astype('float32')
        w = u
        # Apply GS method.
        for k in range(i):
            u -= projection(w, Q[:, k])
        Q[:, i] = normalize(u)
        # Fill R at correct position.
        for j in range(i + 1):
            R[j, i] = np.vdot(Q[:, j], cp[:, i])

    # Return results.
    return Q, R
```

pav. 5 QR skaidos metodo kodas

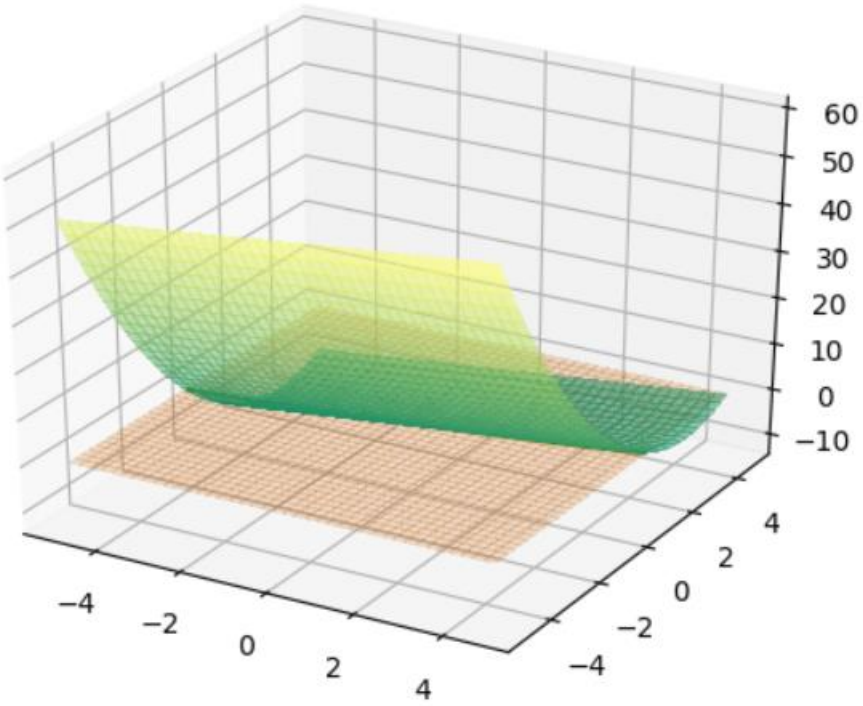
2.2.1. Pirmosios lygčių sistemos sprendimas

Lygties x1^2 + 10 * (sin(x1) + cos(x2))^2 – 10 = 0 grafikas:



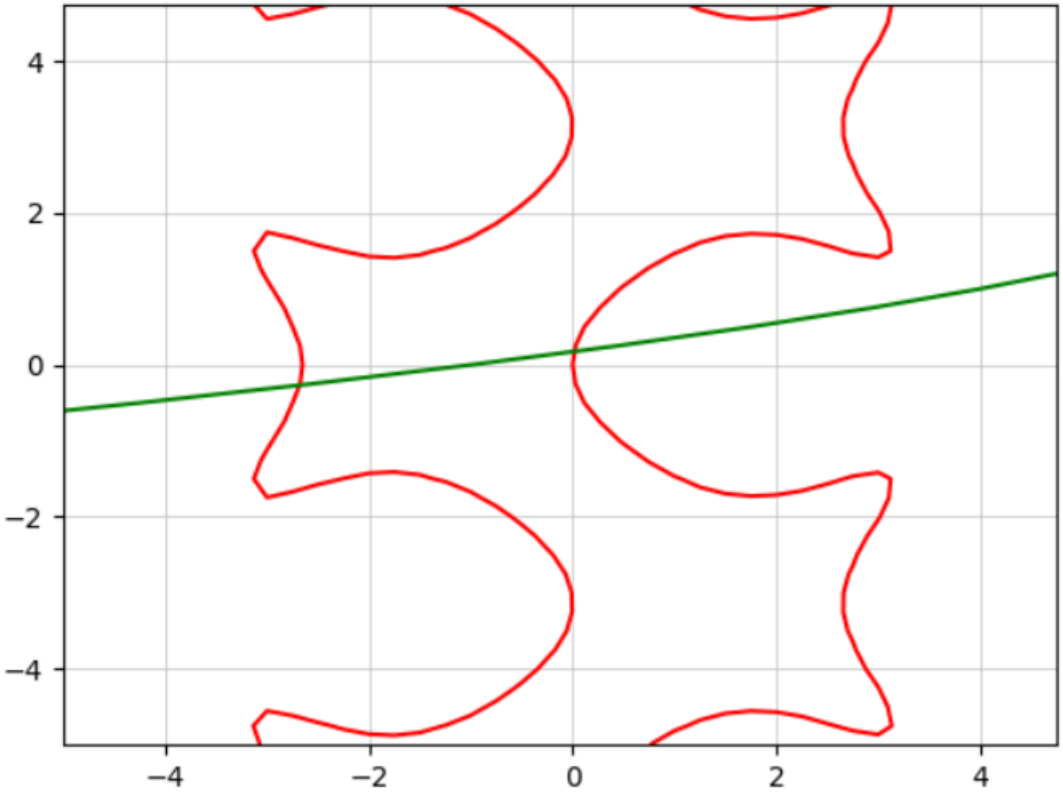
pav. 6 Pirmosios lygties grafikas

Lygties (x2 - 3)^2 + x1 – 8 = 0 grafikas:



pav. 7 Antrosios lygties grafikas

Sistemos grafinis sprendinys:



pav. 8 Sistemos grafinis sprendinys

Iš šio grafiko gauti sprendiniai:
x = [-2.68051, -0.26745]
x = [0.0198026, 0.177446]

Šios sistemos sprendimai naudojant niutono metodą:
Artinys: [1, 1]
Iteration: 0 Precision: 0.6667739291088707
Iteration: 1 Precision: 0.6235252863910974
Iteration: 2 Precision: 0.23226861023967485
Iteration: 3 Precision: 0.03363760584774807
Iteration: 4 Precision: 0.002484517295836433
Iteration: 5 Precision: 1.215415044979624e-05
Iteration: 6 Precision: 2.8805629860205556e-10
Iteration: 7 Precision: 5.267627446762938e-17
Solution: x = [-2.68871391 -0.26935986]
Įstačius į sistemą: [0.0000000e+00, -8.8817842e-16]

Artinys: [2, 2]
Iteration: 0 Precision: 0.5936829084425157
Iteration: 1 Precision: 0.31193623427472805
Iteration: 2 Precision: 0.17015623969465507
Iteration: 3 Precision: 0.1245961190081576
Iteration: 4 Precision: 0.4927602295314902
Iteration: 5 Precision: 0.2308731473300684

Iteration: 6 Precision: 0.2324036009884189
Iteration: 7 Precision: 0.1726052244909754
Iteration: 8 Precision: 0.12248695317279906
Iteration: 9 Precision: 0.5046398510801887
Iteration: 10 Precision: 0.23648139955660516
Iteration: 11 Precision: 0.24442284250091584
Iteration: 12 Precision: 0.14482443161688072
Iteration: 13 Precision: 0.12485776353936778
Iteration: 14 Precision: 0.1501753194919908
Iteration: 15 Precision: 0.12197432578652075
Iteration: 16 Precision: 0.16877362712837715
Iteration: 17 Precision: 0.12094638585319818
Iteration: 18 Precision: 0.37334506622574043
Iteration: 19 Precision: 0.3775700420320316
Iteration: 20 Precision: 0.3669427382332869
Iteration: 21 Precision: 0.26126341395832503
Iteration: 22 Precision: 0.19600190096640263
Iteration: 23 Precision: 0.8996534645827423
Iteration: 24 Precision: 0.2930720388328951
Iteration: 25 Precision: 0.32210583758297845
Iteration: 26 Precision: 0.25734991354826187
Iteration: 27 Precision: 0.19781802319778102
Iteration: 28 Precision: 0.3742147321215397
Iteration: 29 Precision: 0.3626257699397778
Iteration: 30 Precision: 0.2429068013491741
Iteration: 31 Precision: 0.17422277485654586
Iteration: 32 Precision: 0.37513524053090785
Iteration: 33 Precision: 0.7538241068032131
Iteration: 34 Precision: 0.08880676517734451
Iteration: 35 Precision: 0.01250741604620087
Iteration: 36 Precision: 0.0003020338338783022
Iteration: 37 Precision: 1.7720075480681914e-07
Iteration: 38 Precision: 6.122114553415676e-14
Solution: x = [-2.68871391 -0.26935986]
Įstačius į sistemą: [2.35189646e-12, 7.10542736e-15]

Artinys: [-1, -1]
Iteration: 0 Precision: 0.6101263444985333
Iteration: 1 Precision: 0.1066868974856024
Iteration: 2 Precision: 0.026392080590530474
Iteration: 3 Precision: 0.0015176029197673405
Iteration: 4 Precision: 4.521040026757387e-06
Iteration: 5 Precision: 3.9860166064880225e-11
Solution: x = [-2.68871391 -0.26935986]
Įstačius į sistemą: [1.53204027e-09, 3.64508423e-12]

Artinys: [3, 3]
Iteration: 0 Precision: 0.9224085108199245

Iteration: 1 Precision: 0.3457046838327203
Iteration: 2 Precision: 0.40216509862956934
Iteration: 3 Precision: 0.3612869378950808
Iteration: 4 Precision: 0.4659414882831723
Iteration: 5 Precision: 0.18664327368558184
Iteration: 6 Precision: 0.3873402801705707
Iteration: 7 Precision: 0.3017087904364839
Iteration: 8 Precision: 0.5942521017144653
Iteration: 9 Precision: 0.29701653167550185
Iteration: 10 Precision: 0.24869636338448334
Iteration: 11 Precision: 0.9085871496371221
Iteration: 12 Precision: 0.31193769009486527
Iteration: 13 Precision: 0.402897398199075
Iteration: 14 Precision: 0.37725687888650233
Iteration: 15 Precision: 0.2956641486247761
Iteration: 16 Precision: 0.21571701946553642
Iteration: 17 Precision: 0.1457136778848266
Iteration: 18 Precision: 0.12403952101402839
Iteration: 19 Precision: 0.15423953623177958
Iteration: 20 Precision: 0.12050781643126643
Iteration: 21 Precision: 0.19350914650397968
Iteration: 22 Precision: 0.131352520105231
Iteration: 23 Precision: 0.4712388811157742
Iteration: 24 Precision: 0.447411683373473
Iteration: 25 Precision: 0.16323922906856267
Iteration: 26 Precision: 0.0037474902213544136
Iteration: 27 Precision: 1.2575627315902499e-06
Iteration: 28 Precision: 1.4068574725031086e-13
Solution: x = [0.01513214 0.17424915]
Įstačius į sistemą: [4.72510919e-13, 1.77635684e-15]

Matome, kad su visais artiniais gauti sprendiniai yra teisingi, nes juos įstačius į sistemą gauname [0, 0].

Niutono metodo kodas 2 lygčių sistemai:

```
def newton():
    x = np.array([3.5, -8.])
    ff = f(x)
    dff = df(x)
    alpha = 1
    max_iterations = 200
    eps = 1e-10
    for i in range(max_iterations):
        dff = df(x)
        delta_x, a, b, c = np.linalg.lstsq(-dff, ff)

        x1 = x.reshape(2, 1) + alpha * delta_x
        ff1 = f([x1[0, 0], x1[1, 0]])

        precision = np.linalg.norm(delta_x) / (np.linalg.norm(x) + np.linalg.norm(delta_x))
        print(f"Iteration: {i} Precision: {precision}")

        if precision < eps:
            print(f"Solution: {x}")
            return x
        elif i == max_iterations:
            print(f"Set precision not reached. Last x = {x}")
            return

        x = np.array([x1[0, 0], x1[1, 0]])
        ff = ff1
```

pav. 9 Niutono metodo programos kodas 2 lygčių sistemai

2.2.2. Antrosios lygčių sistemos sprendimas

Antrosios lygties sprendimo rezultatas:
Iteration: 0 Precision: 0.5036572353193731
Iteration: 1 Precision: 0.14348989592897798
Iteration: 2 Precision: 0.021162333314487324
Iteration: 3 Precision: 0.0003138918347748688
Iteration: 4 Precision: 6.81413477147687e-08
Iteration: 5 Precision: 3.2316200872258726e-15
Solution: x = [2.50812353 3.17150142 1.13608258 -5.46205041]

Įstačius šį sprendinį į lygčių sistema gauname:
[0, 0, 0, 0]

Niutono metodo kodas 4 lygčių sistemos sprendimui:

```
def newton(x):
    for i in range(max_iterations):
        delta_x, a, b, c = np.linalg.lstsq(-df(x), f(x))
        x = x + np.array(delta_x).reshape(1, 4)
        x = np.array([x[0, 0], x[0, 1], x[0, 2], x[0, 3]])

        precision = np.linalg.norm(delta_x) / (np.linalg.norm(x) + np.linalg.norm(delta_x))
        print(f"Iteration: {i} Precision: {precision}")

        if precision < eps:
            print(f"Solution: {x}")
            print(f"f(x) = {f(x)}")
            return x
        elif i == max_iterations:
            print(f"Set precision not reached. Last x = {x}")
            return
```

pav. 10 Niutno metodo kodas 4 lygčių sistemos sprendimui

Iš to matome, kad šios lygties sprendinys yra teisingas.

3. Išvados

- Išmokau spręsti tiesinių lygčių sistema naudojant QR skaidos metoda
- Išmokau spręsti netiesinių lygčių sistema su 2 ir 4 lygtimis naudojant Niutono metodą
- Iš 2.1 užduoties sprendimo matome, kad sprendimo greitis labai priklauso nuo pasirinkto artinio. Su artiniu [-1, -1] sistema išspręsta per 5 iteracijas, o su artiniu [2, 2] net per 38 iteracijas.
- Norint gauti kelis sprendinius netiesinių lygčių sistemoje reikia spręsti kelius kartus pasirenkant vis kitus artinius
- Sprendžiant netiesiniu lygčių sistema grafiniu budu galime nesunkiai gauti daug sprendinių. Jie tik nėra tokie tikslūs. 2.1 užduotyje sprendžiai grafiškai, kai x = [-2.68051, -0.26745] įstačius į sistema tikslumas = [-0.11575228, -0.0042805], kai x = [0,0198026, 0.177446] tikslumas = [0.08254078, -0.01338632]. O sprendžiant su niutono metodu tikslumai atitinkamai gaunas: [1e-09, 1e-12] ir [1e-13, 1e-15]