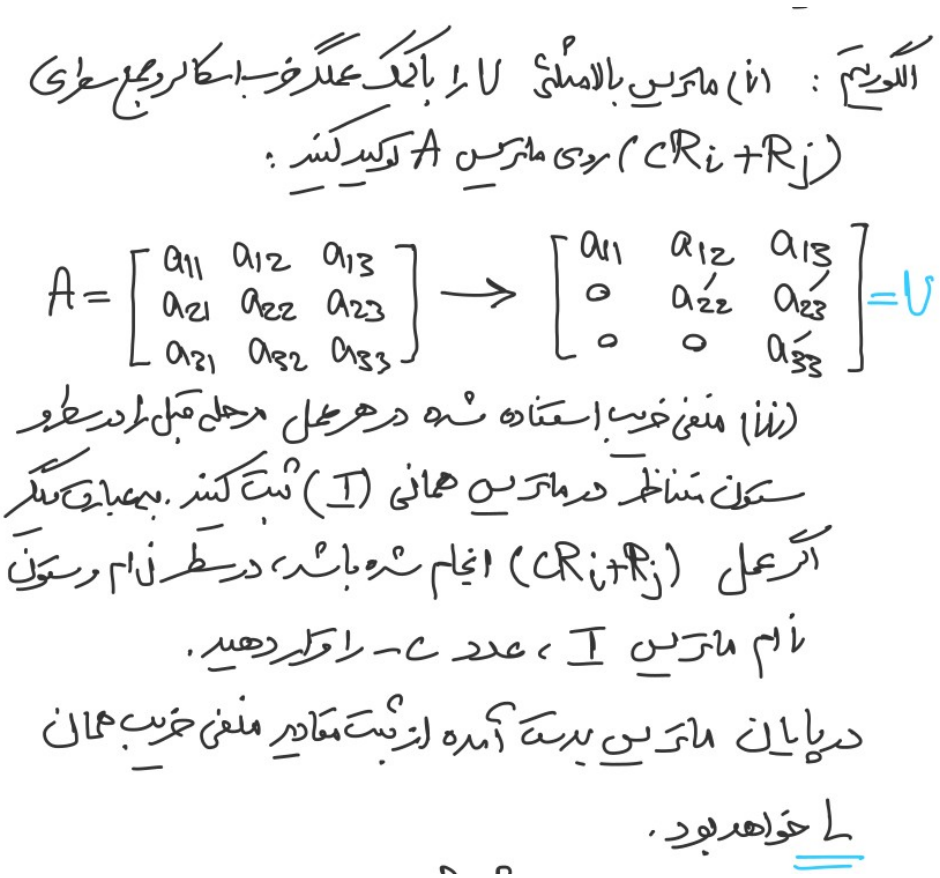
تمرین ۴ کامپیوتری جبر خطی

۱ – باید تجزیه‌ی LU رو پیاده می‌کردیم که طبق روشی که داخل جزوه گفته شده بود پیش رفتم و کد رو زدم:



ورودی گرفتن:

```python
import numpy as np


n = int(input("Enter the number of variables: "))
coefficients = []
constants = []

for i in range(2 * n):
    if i % 2 == 0:
        nums = map(int, input("Enter the coefficients of equation " + str(i // 2 + 1) + ": ").split())
        coefficients.append(list(nums))
    else:
        constants.append(int(input("Enter the constant of equation " + str(i // 2 + 1) + ": ")))
```

تجزیه‌ی LU: (برای تجزیه اول مقدار ثابتی (c) که متعلق به خونه‌ی [i][j] ماتریس L هست رو به دست میاریم و بعد با توجه به اون مقدار ثابت، ماتریس U رو تغییر می‌دیم تا بالامثلثی بشه، مشابه روش جزوه)

```python
def lu_decomposition(matrix, n):
    # initialize L and U
    l = np.zeros((n, n))
    u = np.array(matrix, dtype = float)

    # elements on the diagonal of L are 1
    for i in range(n):
        l[i][i] = 1

    for i in range(1, n):
        for j in range(i):
            l[i][j] = u[i][j] / u[j][j]
            # print(l)
            c = l[i][j]
            for k in range(n):
                u[i][k] -= c * u[j][k]

    return l, u
```

بعد از به دست‌آوردن L U از فرمول‌های زیر استفاده می‌کنیم تا جواب نهایی رو به دست بیاریم:

$$AX=B \Rightarrow (LU)X=B$$

اگر $A=LU$

$$\Rightarrow L(UX)=B$$

UX را Y در نظر می‌گیریم:

$LY=B$ (۱) را حل کرده و Y را به دست می‌آوریم.

$UX=Y$ (۲) را حل کرده و X را به دست می‌آوریم.

پس مرحله‌ی بعدی جایگذاری L در تساوی Ly = b و به دست‌اوردن y هست:

```
L, U = lu_decomposition(coefficients, n)

# solve for y in Ly = b
def forward_substitution(L, n, b):
    y = np.zeros(n)
    for i in range(n):
        if L[i][i] == 0:
            print("Matrix is singular!")
            exit()
        y[i] = b[i]
        for j in range(i):
            y[i] -= L[i][j] * y[j]
        y[i] /= L[i][i]
    return y

y = forward_substitution(L, n, constants)
```

همونطور که مشخصه برای چک‌کردن تکین بودن ماتریس از درایه‌های روی قطر اصلی استفاده کردم که اگر یکیشون صفر باشه، به معنی دترمینان صفره.

برای حل معادله هم می‌شد از np.solve استفاده کرد اما برای اینکه از کتابخونه استفاده نکنیم از روش forward substitution که برای ماتریس‌های پایین مثلثی هست رفتم.

در آخر هم جایگذاری U در Ux = y و به دست‌اوردن x:

```
# solve for x in Ux = y
def backward_substitution(U, n, y):
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        if U[i][i] == 0:
            print("Matrix is singular!")
            exit()
        x[i] = y[i]
        for j in range(i + 1, n):
            x[i] -= U[i][j] * x[j]
        x[i] /= U[i][i]
    return x

x = backward_substitution(U, n, y)

print("The solution is: ", x)
```

برای U چون بالامثلثیه از backward substitution استفاده می‌کنیم و همچنین تکین بودن رو به روشی که گفته شد چک می‌کنیم.

تست:

خروجی:

```
The solution is:  [-1. -2.  3.]
```

$$X_3 = 3,\quad X_2 = -2,\quad X_1 = -1$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 1 & 2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 4 \end{bmatrix}$$

۲ – این سؤال هم نکته‌ی خاصی نداشت و باید از ماتریس‌های چرخشی که از پیش تعریف شدن استفاده کنیم و در بردارهای داده شده ضرب کنیم.

ماتریس‌های چرخشی:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ورودی گرفتن:

```python
axis = input("Enter the axis of rotation (x, y or z): ")
angle = int(input("Enter the angle of rotation in degrees: "))
n = int(input("How many 3D points will you input? "))

points = []
for i in range(n):
    points.append(list(map(int, input("Enter the coordinates of point " + str(i + 1) + ": ").split(','))))

print(points)
```

تعریف ماتریس‌های چرخشی:

```python
def rotate(axis, angle, points):
    angle = np.radians(angle)
    if axis == 'x':
        rotation_matrix = np.array([[1, 0, 0],
                                    [0, np.cos(angle), -np.sin(angle)],
                                    [0, np.sin(angle), np.cos(angle)]])
    elif axis == 'y':
        rotation_matrix = np.array([[np.cos(angle), 0, np.sin(angle)],
                                    [0, 1, 0],
                                    [-np.sin(angle), 0, np.cos(angle)]])
    elif axis == 'z':
        rotation_matrix = np.array([[np.cos(angle), -np.sin(angle), 0],
                                    [np.sin(angle), np.cos(angle), 0],
                                    [0, 0, 1]])

    else:
        print("Invalid axis of rotation!")
        exit()
```

چک‌کردن متعامدبودن ماتریس حاصل و ضرب نقاط داده‌شده در ماتریس چرخشی:

```python
    # check the orthogonality of the matrix
    for i in range(3):
        for j in range(3):
            if i == j:
                assert np.isclose(np.dot(rotation_matrix[i], rotation_matrix[j]), 1)
            else:
                assert np.isclose(np.dot(rotation_matrix[i], rotation_matrix[j]), 0)

    return np.dot(points, rotation_matrix)
```

در آخر هم نمایش نقاط ورودی و خروجی به کمک matplotlib:

```python
import matplotlib.pyplot as plt

rotated_points = rotate(axis, angle, points)
print(rotated_points)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter([point[0] for point in points], [point[1] for point in points], [point[2] for point in points], c='r', marker='o')
ax.scatter([point[0] for point in rotated_points], [point[1] for point in rotated_points], [point[2] for point in rotated_points],
 c='b', marker='^')

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```

تست:

```
[[1, 1, 1], [0, 1, 2]]
```

حول محور ایکس‌ها به میزان ۳۰ درجه

خروجی: