

Course:
EEF 281

Homework 1

with GITHUB repository available

İTÜ



MATLAB / PYTHON / C

Javad Ibrahimli
040210932

Electronics & Communication Engineering
Istanbul Technical University, Turkey

Academic Year
2021-2022

You can get pure source codes of all 3 programming languages for this homework from my GITHUB repository.



Press this text in order to go my GITHUB Repository for this task

Links will be available after the due date of this task

TABLE OF CONTENTS

1. Task Analysis	
2. Before coding	
3. Solution with C	
4. Solution with Python	
5. Solution with MATLAB	

Task Analysis

1. Screenshot of the Homework pdf

İstanbul Teknik Üniversitesi
Elektrik Elektronik Fakültesi
Tüm Bölümler



EEF281- Lineer Cebir ve Uyg.
Dr. Mehmet Onur GÜLBAHÇE
Ödev-09.05.2022

ÖDEV-1

1. Üç gözlü bir elektrik devresindeki akınlar aşağıdaki lineer denklem sistemi ile ifade edilmiştir.

$$\underbrace{\begin{bmatrix} 1 & 1 & -1 \\ 3 & 5 & 1 \\ 2 & 1 & 4 \end{bmatrix}}_A \underbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}_x = \underbrace{\begin{bmatrix} -1 \\ 5 \\ 2 \end{bmatrix}}_b$$

Matlab, C veya Phyton dilinde Gauss-Jordan metodunun algoritmasını yazınız ve $Ax=b$ sistemini yazmış olduğunuz algoritma yardımı ile çözünüz.

ÖDEV HAZIRLARKEN ☺

- Ödevinizi yazım kurallarına uygun bir şekilde hazırlayınız ve mutlaka kapak sayfası ekleyiniz.
- MS Word veya Latex üzerinde hazırlayabilirsiniz.
- Yazdığınız kod parçalarını mutlaka veriniz.
- Ödevinizi sayfa sıralamasına göre zımbalayınız. İlave poşet dosyaya koymadan teslim ediniz.
- Ödevinizin son teslim tarihi 26 Mayıs 2022 perşembe günü saat 15:30'dur.

According to this task, students have to do following things:

- *Write an algorithm in order to calculate linear equation system with Gauss-Jordan method using Python / MATLAB / C*
- *Find unknown variables in the given linear equation with the help of that algorithm. (I1,I2,I3)*

Homework 1

Before coding

In order to solve this programming task, the user must first create his own algorithm. Here's how I set up my algorithm for this task.

1. Code Starts
2. Computer Gets Number of Unknowns from user input: n
3. Computer gets elements of augmented Matrix (A) of n by n+1 Size
4. Transform Augmented Matrix (A) to Diagonal Matrix by Row Operations.
5. Computer Obtains Solution by Making All Diagonal Elements 1.
6. Computer Displays Result.
7. Code Stops

I programmed the Gauss and Jordan method with 3 different programming languages:

PROGRAMMING LANGUAGE	Features
C	<ul style="list-style-type: none">• 35-40 lines of code• Matrix indexes start from 1
Python	<ul style="list-style-type: none">• 35-40 lines of code• Matrix indexes start from 0
MATLAB	<ul style="list-style-type: none">• 190-200 lines of code• User enters the whole matrix, rather than matrix elements one by one• This code can show the solution method step by step.

Note:

Be aware of that there are code line numbers, which will block user to copy code from PDF. You can get the pure code from my Github page. I will change that repository from Private to Public after the due date of this homework.

[Press this text in order to see my repository for this homework](#)

Solution with C

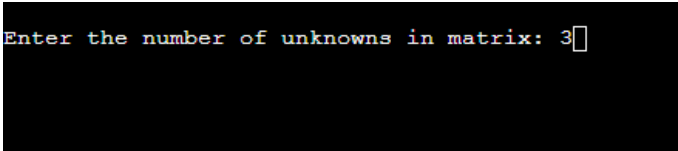
```

1. #include<stdio.h>
2.
3. int main()    //main function starts
4. {
5.     int i,j,k,n;           //defined integer variables for operations
6.     float A[20][20],c,x[10];
7.     printf("\nEnter the number of unknowns: ");
8.     scanf("%d",&n);        //With scanf code gets input from user
9.     printf("\nEnter the elements of augmented matrix row-wise:\n");
10.    for(i=1; i<=n; i++)    //user gives row-wise elements of matrix.
11.    {
12.        for(j=1; j<=(n+1); j++)
13.        {
14.            printf(" A[%d][%d]:", i,j);
15.            scanf("%f",&A[i][j]);
16.        }
17.    }
18.
19.    for(j=1; j<=n; j++)    //Now finding the elements of diagonal matrix
20.    {
21.        for(i=1; i<=n; i++)
22.        {
23.            if(i!=j)
24.            {
25.                c=A[i][j]/A[j][j];
26.                for(k=1; k<=n+1; k++)
27.                {
28.                    A[i][k]=A[i][k]-c*A[j][k];
29.                }
30.            }
31.        }
32.    }
33.    printf("\nThe solution is:\n"); //This line gives the text part of the solution
34.    for(i=1; i<=n; i++)
35.    {
36.        x[i]=A[i][n+1]/A[i][i];
37.        printf("\n x[%d]=%f\n",i,x[i]); // solution
38.    }
39.    return(0);
40. }
41.
42.

```

Code Analysis:

1. When the user runs the code, the computer prompts the user to enter the number of unknowns in the linear equation.



```
Enter the number of unknowns in matrix: 3
```

lines 7 and 8

Homework 1

2. After the user enters the number of unknowns to the computer, code asks user to enter elements of matrix one by one.

```
Enter the elements of augmented matrix row-wise:  
A[1][1]:1  
A[1][2]:1  
A[1][3]:-1  
A[1][4]:-1  
A[2][1]:3  
A[2][2]:5  
A[2][3]:1  
A[2][4]:5  
A[3][1]:2  
A[3][2]:1  
A[3][3]:4  
A[3][4]:2
```

lines between 10 and 17

3. After user gives all elements of matrix, computer calculates the matrix with Gauss and Jordan method. This shows that the correct answer combination for this equation is [-2, 2, 1].

```
The solution is:  
  
I1=-2.000000  
  
I2=2.000000  
  
I3=1.000000
```

lines between 19 and 40

NOTE: All Matrix indexes start from 1.

Solution with Python

```
1. # Importing NumPy Library
2. import numpy as np
3. import sys
4.
5. # Reading number of unknowns
6. n = int(input('Enter number of unknowns: '))
7.
8. # Making numpy array of n x n+1 size and initializing
9. # to zero for storing augmented matrix
10. a = np.zeros((n,n+1))
11.
12. # Making numpy array of n size and initializing
13. # to zero for storing solution vector
14. x = np.zeros(n)
15.
16. # Reading augmented matrix coefficients
17. print('Enter Augmented Matrix Coefficients:')
18. for i in range(n):
19.     for j in range(n+1):
20.         a[i][j] = float(input( 'a['+str(i)+']['+ str(j)+']='))
21.
22. # Applying Gauss Jordan Elimination
23. for i in range(n):
24.     if a[i][i] == 0.0:
25.         sys.exit('Divide by zero detected!')
26.
27.     for j in range(n):
28.         if i != j:
29.             ratio = a[j][i]/a[i][i]
30.
31.             for k in range(n+1):
32.                 a[j][k] = a[j][k] - ratio * a[i][k]
33.
34. # Obtaining Solution
35.
36. for i in range(n):
37.     x[i] = a[i][n]/a[i][i]
38.
39. # Displaying solution
40. print('\nRequired solution is: ')
41. for i in range(n):
42.     print('I%d = %0.2f' %(i,x[i]), end = '\t')
```

Code Analysis:

1. *In order to work with arrays and different functions numpy and sys libraries have to be imported.*

Homework 1

2. After importing libraries, computer asks user to enter the number of unknowns in linear equations.

```
In [4]: runcell(0, 'C:/Users/Cavad/AppData/Roaming/Microsoft/Windows/Network Shortcuts/temp.py')  
Enter number of unknowns: 3
```

3. After creating a numpy array according to entered value n , - computer asks user to enter the elements of matrix one by one.

```
Enter Augmented Matrix Coefficients:  
a[0][0]=1  
a[0][1]=1  
a[0][2]=-1  
a[0][3]=-1  
a[1][0]=3  
a[1][1]=5  
a[1][2]=1  
a[1][3]=5  
a[2][0]=2  
a[2][1]=1  
a[2][2]=4  
a[2][3]=2
```

4. In the next part of the code (lines between 22-33) computer applies Gauss and Jordan Elimination. *It can also detect "divide by 0" error.*
5. In the last part of the code user gets the correct combination of answer, which is $[-2, 2, 1]$.

```
Required solution is:  
I0 = -2.00 I1 = 2.00 I2 = 1.00
```


Solution with MATLAB

```
1. clear all;
2. clc;
3.
4. format rat
5.
6. disp('Gauss Jordan Method')
7. disp('In this method we convert the Augmented matrix in reduced echenlon form')
8.
9.
10. A=input('Enter the augmented matrix\n = ');
11.
12.
13. r=length(A(:,1));
14. c=length(A(1,:));
15.
16. disp('Solution')
17.
18. for i=1:r
19.
20.
21.
22.     f=0;
23.
24.     l=i;
25.
26.     if A(i,i) ~= 1
27.
28.         for m=i+1:r
29.
30.             if A(m,i) == 1
31.
32.                 B=A(i,:);
33.
34.                 A(i,:)=A(m,:);
35.
36.                 fprintf ('Swaping R%.f with R%.f.\n',m,i)
37.
38.                 A(m,:)=B;
39.
40.                 disp(A)
41.
42.                 break;
43.
44.             end
45.
46.         end
47.
48.     end
49.
50.
51.
52.     if A(i,i)== 0
53.
54.
55.
56.         for k=i+1:r
```

Homework 1

```
57.
58.         if A(k,i)~= 0
59.
60.             B=A(i,:);
61.
62.             A(i,:)=A(k,:);
63.
64.             fprintf ('Swaping R%.f with R%.f.\n',k,i)
65.
66.             A(k,:)= B;
67.
68.             diso(A);
69.
70.             d=A(i,i);
71.
72.             f=1;
73.
74.             break;
75.
76.         end
77.
78.     end
79.
80.
81.
82.     if f == 0
83.
84.         for l=i+1:c-1
85.
86.             if A(i,l) ~= 0
87.
88.                 d=A(i,l);
89.
90.                 f=1;
91.
92.                 break;
93.
94.             end
95.
96.         end
97.
98.     end
99.
100.
101.
102.     else
103.
104.         d=A(i,i);
105.
106.         f=1;
107.
108.     end
109.
110.
111.
112.     if f ~= 0
113.
114.         if d~=1
115.
116.             fprintf('R%.f / (%s) \n',i,rats(d))
117.
```

Homework 1

```
118.         A(i,:)=A(i,+)/d ;
119.
120.         disp(A);
121.
122.     end
123.
124.     for j=i+1:r
125.
126.         if A(j,1)== 0
127.
128.             continue;
129.
130.         end
131.
132.         fprintf('R%.f- (%s)*R%.f \n',j, rats(A(j,1)),i)
133.
134.         A(j,:)=A(j,:) - A(j,1)*A(i,:);
135.
136.         disp(A)
137.
138.     end
139.
140. end
141.
142. end
143.
144. disp('')
145.
146. disp('Reverse phase')
147.
148. fg=0;
149.
150. for i=r:-1:1
151.
152.     f=0;
153.
154.     l=i;
155.
156.     if A(i,i)== 0
157.
158.         for l=i+1:c-1
159.
160.             if A(i,l) ~= 0
161.
162.                 d=A(i,l);
163.
164.                 f=1;
165.
166.                 break;
167.
168.             end
169.
170.         end
171.
172.
173.     else
174.
175.
176.         d=A(i,i);
177.
178.         f=1;
```

Homework 1

```
179.
180.     end
181.
182.
183.
184.     if f ~= 0
185.
186.         for j=i-1:-1:1
187.
188.             if A(j,i) == 0
189.
190.                 continue;
191.
192.             end
193.
194.             fprintf('R%.f- (%s)*R%.f \n',j,rats(A(j,i)),i)
195.
196.             A(j,:)=A(j,:) - A(j,i)*A(i,:) ;
197.
198.             disp(A);
199.
200.             fg=1;
201.
202.         end
203.
204.     end
205.
206.     if ( (f==0 && A(i,c)) == 1 || fg==0 && i==1)
207.
208.         disp('No solution')
209.
210.         break;
211.     end
212.
213.     A(:, end)
214. end
215.
```

1. When user runs this code, it prints this statement.

Command Window

Gauss Jordan Method

In this method we convert the Augmented matrix in reduced echenlon form

2. Then, it asks user to enter the augmented matrix

```
Enter the augmented matrix
= [1 1 -1 -1
3 5 1 5
2 1 4 2]
```

3. In the next part of the code, computer gives the solution technique of the linear equation.

Homework 1

Solution

R2- (3) *R1

1	1	-1	-1
0	2	4	8
2	1	4	2

R3- (2) *R1

1	1	-1	-1
0	2	4	8
0	-1	6	4

R2 / (2)

1	1	-1	-1
0	1	2	4
0	-1	6	4

R3- (-1) *R2

1	1	-1	-1
0	1	2	4
0	0	8	8

R3 / (8)

1	1	-1	-1
0	1	2	4
0	0	1	1

Reverse phase

R2- (2) *R3

1	1	-1	-1
0	1	0	2
0	0	1	1

R1- (-1) *R3

1	1	0	0
0	1	0	2
0	0	1	1

R1- (1) *R2

1	0	0	-2
0	1	0	2
0	0	1	1

ans =

-2
2
1

Answer is correct.