

LAB 1

Introduction to Octave

1. Creating vectors

(a) Generate following vectors:

```
A = [1 0 4 5 3 9 0 2]
```

```
a = [4 5 0 2 0 0 7 1]
```

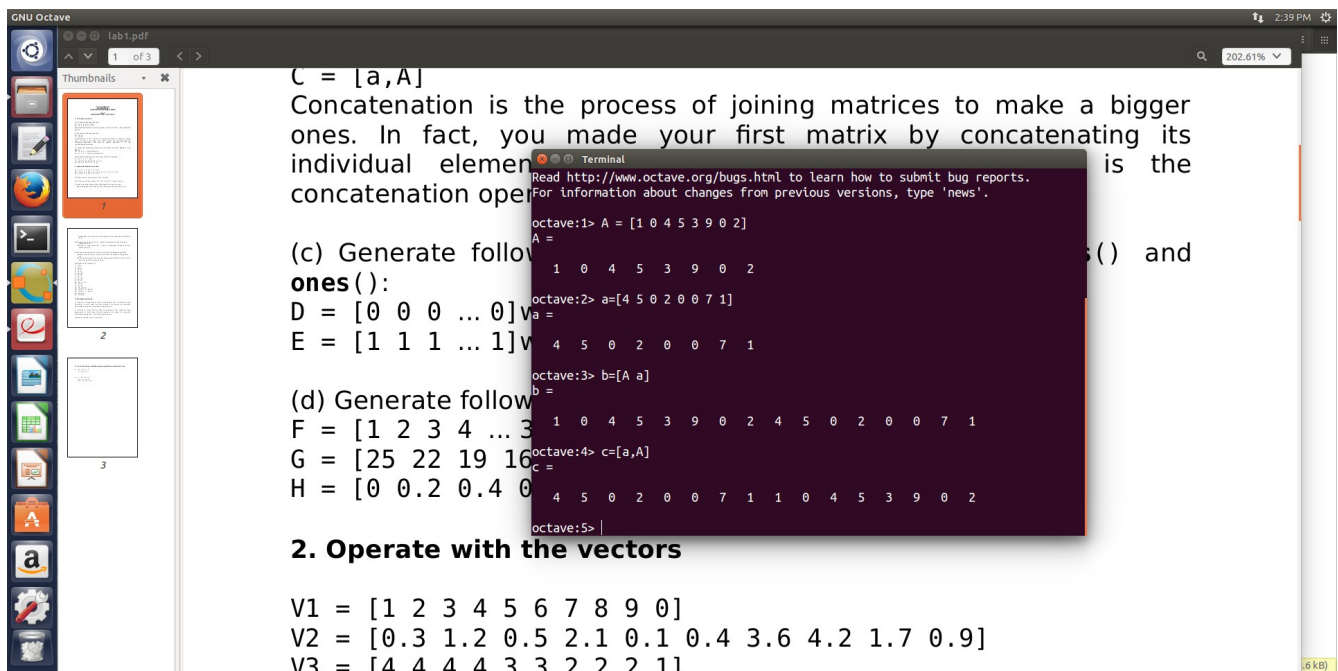
Be aware that Octave is case sensitive. Vector A and a have different values.

(b) Generate following vectors:

```
B = [A a]
```

```
C = [a,A]
```

Concatenation is the process of joining matrices to make a bigger ones. In fact, you made your first matrix by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.



The screenshot shows the GNU Octave environment. A terminal window is open, displaying the following commands and their outputs:

```
octave:1> A = [1 0 4 5 3 9 0 2]
A =
   1   0   4   5   3   9   0   2

octave:2> a = [4 5 0 2 0 0 7 1]
a =
   4   5   0   2   0   0   7   1

octave:3> b = [A a]
b =
   1   0   4   5   3   9   0   2   4   5   0   2   0   0   7   1

octave:4> c = [a,A]
c =
   4   5   0   2   0   0   7   1   1   0   4   5   3   9   0   2

octave:5> |
```

The background text in the image shows the continuation of the lab instructions:

```
C = [a,A]
Concatenation is the process of joining matrices to make a bigger
ones. In fact, you made your first matrix by concatenating its
individual elements. The pair of square brackets [] is the
concatenation operator.

(c) Generate following vectors using inbuilt functions zeros() and
ones():
D = [0 0 0 ... 0] with fifty 0's.
E = [1 1 1 ... 1] with fifty 1's.

(d) Generate following vectors:
F = [1 2 3 4 ... 30]
G = [25 22 19 16 ... 2]
H = [0 0.2 0.4 0.6 ... 0.9]
```

Below the terminal window, the text "2. Operate with the vectors" is visible, followed by the definitions of vectors V1, V2, and V3:

```
V1 = [1 2 3 4 5 6 7 8 9 0]
V2 = [0.3 1.2 0.5 2.1 0.1 0.4 3.6 4.2 1.7 0.9]
V3 = [4 4 4 4 3 3 2 2 2 1]
```

(c) Generate following vectors using inbuilt functions zeros() and ones():

D = [0 0 0 ... 0] with fifty 0's.

GNU Octave

lab1.pdf

1 of 3

Concatenation is a bigger operation than concatenating its individual elements. In fact, you can concatenate matrices into a bigger matrix using the `cat()` and `ones()` functions.

(c) Generate following matrices using `ones()`:

```
D = [0 0 0 ... 0]
E = [1 1 1 ... 1]
```

(d) Generate following matrices:

```
F = [1 2 3 4 ...]
G = [25 22 19 16 ...]
H = [0 0.2 0.4 0.6 ... 2.0]
```

2. Operate with the vectors

```
V1 = [1 2 3 4 5 6 7 8 9 0]
V2 = [0.3 1.2 0.5 2.1 0.1 0.4 3.6 4.2 1.7 0.9]
V3 = [4 4 4 4 3 3 2 2 2 1]
```

Terminal

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Columns 17 through 30:
17 18 19 20 21 22 23 24 25 26 27 28 29 30
octave:10> G=[25:-3:1]
G =
25 22 19 16 13 10 7 4 1
octave:11> H=[0:0.2:2.0]
H =
Columns 1 through 8:
0.00000 0.20000 0.40000 0.60000 0.80000 1.00000 1.20000 1.40000
Columns 9 through 11:
1.60000 1.80000 2.00000
octave:12> |
```

2. Operate with the vectors

```
V1 = [1 2 3 4 5 6 7 8 9 0]
V2 = [0.3 1.2 0.5 2.1 0.1 0.4 3.6 4.2 1.7 0.9]
V3 = [4 4 4 4 3 3 2 2 2 1]
```

- Find sum of the vectors $V1, V2$, and $V3$.
- Find sum of the vectors $V1^T, V2^T$ and $V3^T$ respectively.

Terminal

```
1.60000 1.80000 2.00000
octave:12> clc
octave:13> v1=[1 2 3 4 5 6 7 8 9 0]
v1 =
1 2 3 4 5 6 7 8 9 0
octave:14> v2=[0.3 1.2 0.5 2.1 0.1 0.4 3.6 4.2 1.7 0.9]
v2 =
Columns 1 through 6:
0.30000 1.20000 0.50000 2.10000 0.10000 0.40000
Columns 7 through 10:
3.60000 4.20000 1.70000 0.90000
octave:15> v3 = [4 4 4 4 3 3 2 2 2 1]
v3 =
4 4 4 4 3 3 2 2 2 1
octave:16> v1+v2+v3
ans =
Columns 1 through 7:
5.3000 10.8000 7.5000 10.1000 8.1000 9.4000 12.6000
Columns 8 through 10:
14.2000 12.7000 1.9000
octave:17> v1'+v2'+v3'
ans =
5.3000
10.8000
7.5000
10.1000
8.1000
9.4000
12.6000
14.2000
12.7000
1.9000
octave:18> |
```

(c) How to access value of the fifth element of each vector?

What happens if we execute the command `V1(0)` and `V1(11)`? Remember if a vector has N elements, their subscripts are from 1 to N .

(d) Generate a new vector `V4`, which is composed of the first five elements of `V2`.

Generate a new vector `V5`, which is composed of the last five elements of `V2`.

The screenshot shows a GNU Octave window titled 'lab1.pdf' with three thumbnails. The main window displays the following code:

```
V1 = [1 2 3 4 5 6 7 8 9 0]
V2 = [0.3 1.2 0.5 2.1 0.1 0.4 3.6 4.2 1.7 0.9]
V3 = [4 4 4 4 3 3 2 2 2 1]
```

Below the code are three questions:

- (a) Find sum of
- (b) Find sum of
- (c) How to access value of the fifth element of each vector? What happens if we execute the command `V1(0)` and `V1(11)`? Remember if a vector has N elements, their subscripts are from 1 to N .

A terminal window is overlaid on the Octave window, showing the following commands and outputs:

```
octave:19> v2(5)
ans = 0.10000
octave:20> v3(5)
ans = 3
octave:21> v1(0)
error: subscript indices must be either positive integers less than 2^31 or logicals
octave:21> v1(11)
error: A(I): index out of bounds; value 11 out of bound 10
octave:21> v4=v2(1:5)
v4 =
    0.30000    1.20000    0.50000    2.10000    0.10000
octave:22> v5=v2(end-5:end)
v5 =
    0.10000    0.40000    3.60000    4.20000    1.70000    0.90000
octave:23> v5=v2(end-4:end)
v5 =
    0.40000    3.60000    4.20000    1.70000    0.90000
octave:24> |
```

(e) Derive a new vector `V6` from `V2`, with its 6th element omitted.

Derive a new vector `V7` from `V2`, with its 7th element changed to 1.4.

Derive a new vector `V8` from `V2`, whose elements are the 1st, 3rd, 5th, 7th, and 9th elements of `V2`.

```
Terminal
octave:29> v6(6)=[]
v6 =

Columns 1 through 7:
    0.30000    102.00000    0.50000    2.10000    0.10000    3.60000    4.20000

Columns 8 and 9:
    1.70000    0.90000

octave:30> v7=v2()
v7 =

Columns 1 through 7:
    0.30000    102.00000    0.50000    2.10000    0.10000    0.40000    3.60000

Columns 8 through 10:
    4.20000    1.70000    0.90000

octave:31> v7(7)=v2
error: A(1) = X: X must have the same size as I
octave:31> v7=v2()
v7 =

Columns 1 through 7:
    0.30000    102.00000    0.50000    2.10000    0.10000    0.40000    3.60000

Columns 8 through 10:
    4.20000    1.70000    0.90000

octave:32> v7(7)=[1.4]
v7 =

Columns 1 through 7:
    0.30000    102.00000    0.50000    2.10000    0.10000    0.40000    1.40000

Columns 8 through 10:
    4.20000    1.70000    0.90000

octave:33> v8=v2(1:2:end)
v8 =

    0.30000    0.50000    0.10000    3.60000    1.70000
```

(f) What are the results of

1. $9 - V1$
2. $V1 * 5$
3. $V1 + V2$
4. $V1 - V3$
5. $V1 .* V2$
6. $V1 * V2$
7. $V1.^2$
8. $V1.^3$

```
Terminal
octave:34> 9 - v1
ans =

    8    7    6    5    4    3    2    1    0    9

octave:35> v1*5
ans =

    5   10   15   20   25   30   35   40   45    0

octave:36> v1+v2
ans =

Columns 1 through 8:
    1.30000    104.00000    3.50000    6.10000    5.10000    6.40000    10.60000    12.20000

Columns 9 and 10:
    10.70000    0.90000

octave:37> v1-v3
ans =

   -3   -2   -1    0    2    3    5    6    7   -1

octave:38> v1.*v2
ans =

Columns 1 through 8:
    0.30000    204.00000    1.50000    8.40000    0.50000    2.40000    25.20000    33.60000

Columns 9 and 10:
    15.30000    0.80000

octave:39> v1*v2
error: operator *: nonconformant arguments (op1 is 1x10, op2 is 1x10)
octave:39> v1.^2
ans =

    1    4    9   16   25   36   49   64   81    0

octave:40> v1.^v3
ans =

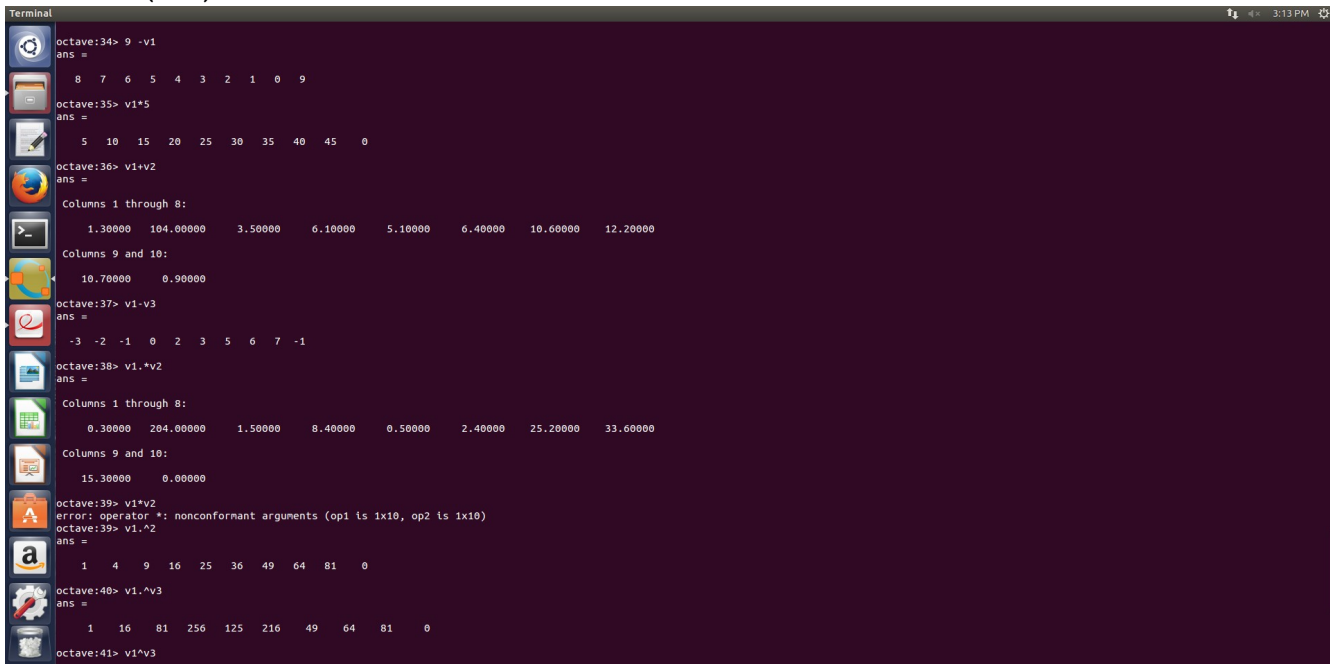
    1   16   81  256  125  216   49   64   81    0

octave:41> v1^v3
```

```

9. V1^V3
10. V1 == V3
11. V1>6
12. V1>V3
13. V3-(V1>2)
14. (V1>2) & (V1<6)
15. (V1>2) | (V1<6)
16. any(V1)
17. all(V1)

```



```

octave:34> 9 -v1
ans =
    8    7    6    5    4    3    2    1    0    9

octave:35> v1*5
ans =
    5   10   15   20   25   30   35   40   45    0

octave:36> v1+v2
ans =
Columns 1 through 8:
    1.30000   104.00000    3.50000    6.10000    5.10000    6.40000   10.60000   12.20000
Columns 9 and 10:
   10.70000    0.90000

octave:37> v1-v3
ans =
   -3   -2   -1    0    2    3    5    6    7   -1

octave:38> v1.*v2
ans =
Columns 1 through 8:
    0.30000   204.00000    1.50000    8.40000    0.50000    2.40000   25.20000   33.60000
Columns 9 and 10:
   15.30000    0.00000

octave:39> v1*v2
error: operator *: nonconformant arguments (op1 is 1x10, op2 is 1x10)
octave:39> v1.^2
ans =
    1    4    9   16   25   36   49   64   81    0

octave:40> v1.^v3
ans =
    1   16   81  256  125  216   49   64   81    0

octave:41> v1^v3

```

3. Plotting functions

- Create a script file(.m file) to generate the continuous time functions (i) unit step (ii) unit impulse (iii) ramp (iv) periodic sinusoidal sequences. Plot all the sequences.
- Create a script file(.m file) to generate the discrete time sequences (i) unit step (ii) unit impulse (iii) ramp (iv) periodic sinusoidal sequences. Plot all the sequences.
- Make a function for 3.a and 3.b.

-->UNITSTEP Function:-

```

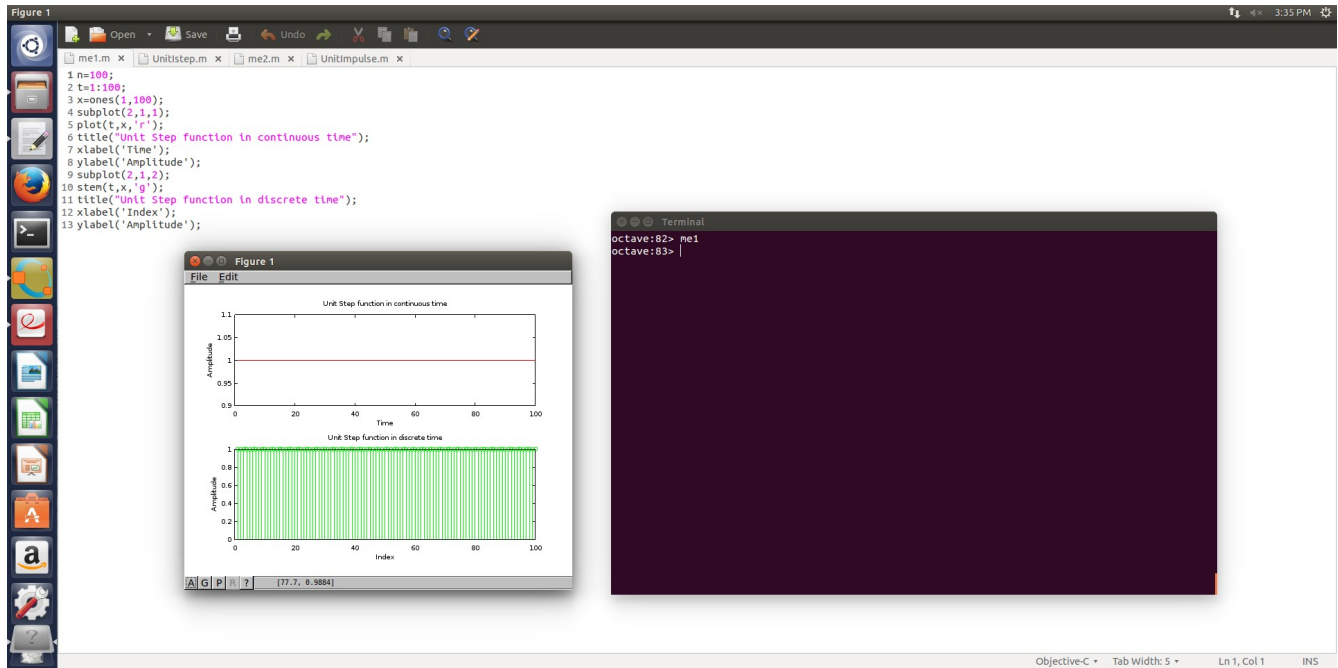
n=100;
t=1:100;
x=ones(1,100);
subplot(2,1,1);
plot(t,x,'r');
title("Unit Step function in continuous time");
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);

```

```

stem(t,x,'g');
title("Unit Step function in discrete time");
xlabel('Index');
ylabel('Amplitude');

```

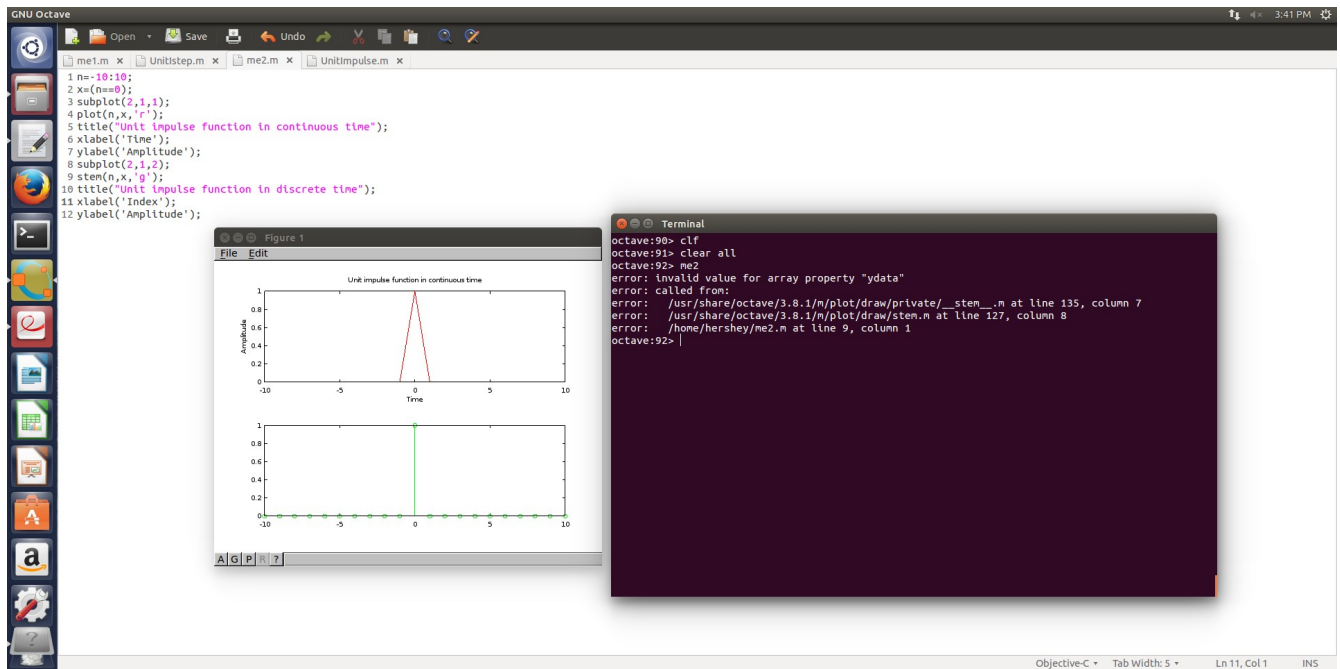


-->UNITIMPULSE FUNCTION :-

```

n=-10:10;
x=(n==0);
subplot(2,1,1);
plot(n,x,'r');
title("Unit impulse function in continuous time");
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);
stem(n,x,'g');
title("Unit impulse function in discrete time");
xlabel('Index');
ylabel('Amplitude');

```

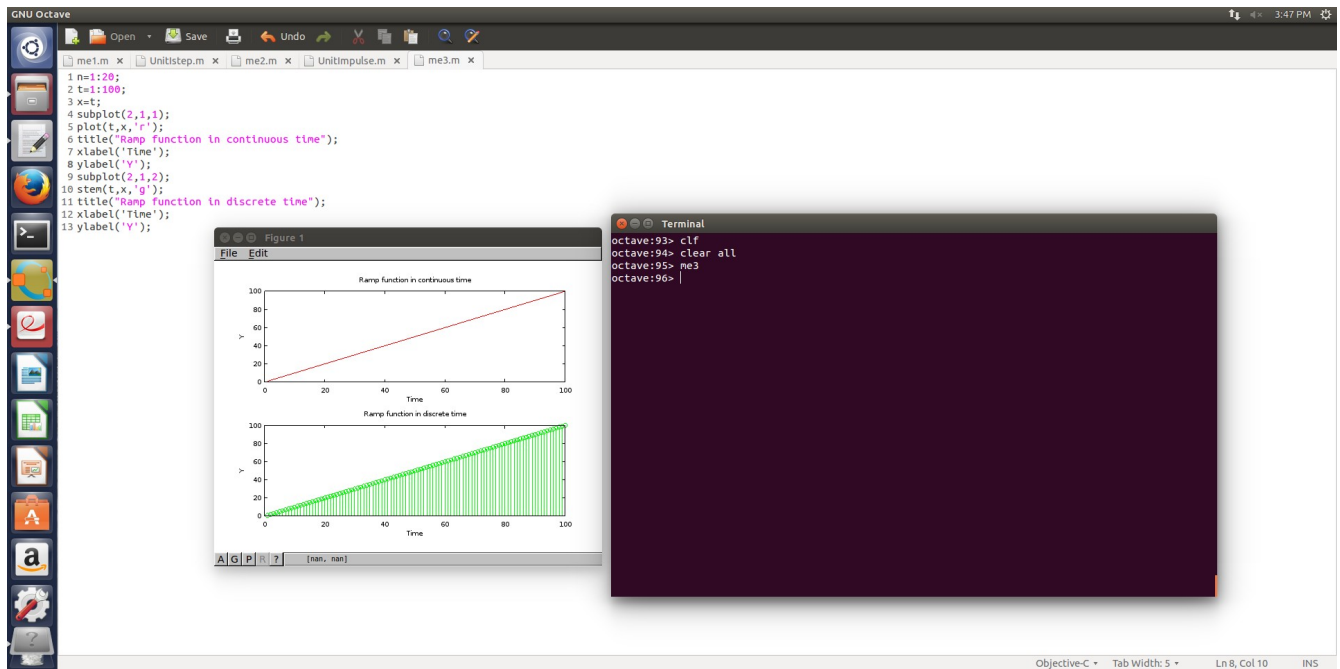


-->RAMP FUNCTION:-

```

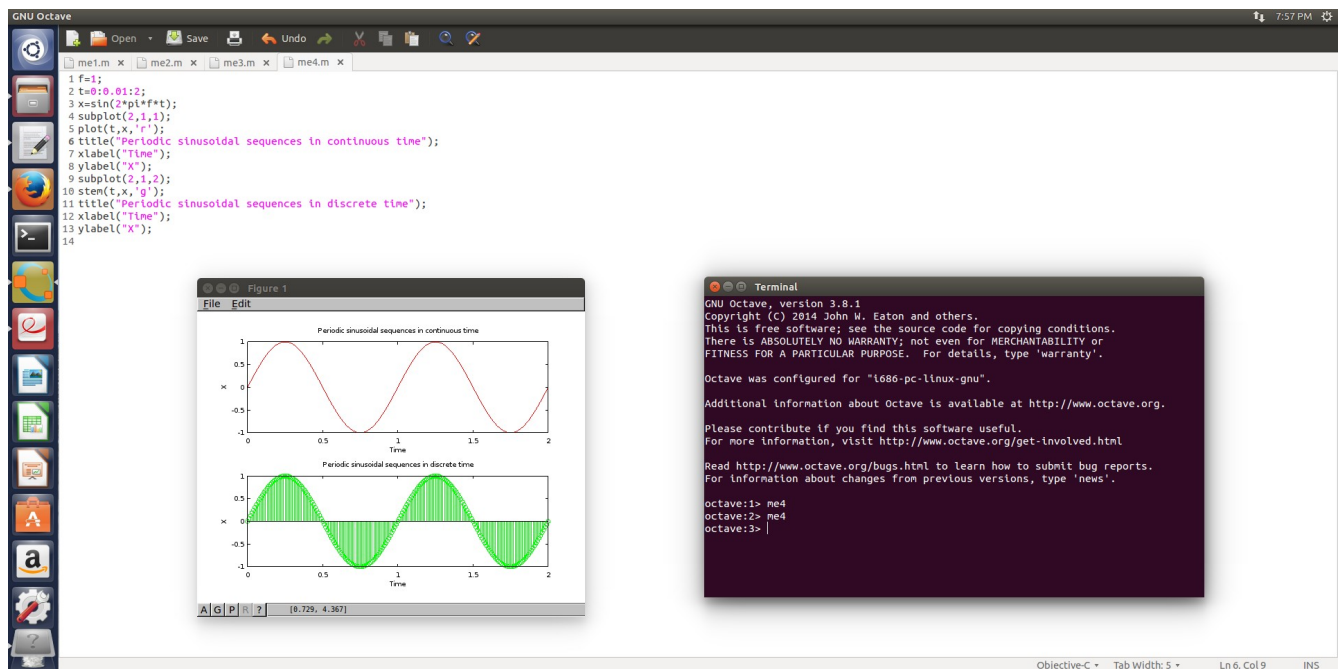
n=1:20;
t=1:100;
x=t;
subplot(2,1,1);
plot(t,x,'r');
title("Ramp function in continuous time");
xlabel('Time');
ylabel('Y');
subplot(2,1,2);
stem(t,x,'g');
title("Ramp function in discrete time");
xlabel('Time');
ylabel('Y');

```

-->PERIODIC SINUSOIDAL SEQUENCES:-

```
f=1;
t=0:0.01:2;
x=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,x,'r');
title("Periodic sinusoidal sequences in continuous time");
xlabel("Time");
ylabel("X");
subplot(2,1,2);
stem(t,x,'g');
title("Periodic sinusoidal sequences in discrete time");
xlabel("Time");
ylabel("X");
```



4. To solve linear simultaneous equations in matrix form

a. $3x + 2y = -2$

$x + 4y = 6$

a. $x - 2y + z = 3$

$2x + y - z = 5$

$3x - y + 2z = 12$

