DSP LAB TASK

LAB one

```
A=[1 2 3];
B=[1 2 3 ; 4 5 6; 7 8 9];
size A
```

```
ans = 1×2
     1     1
```

```
size B
```

```
ans = 1×2
     1     1
```

```
a=A*A'
```

```
a = 14
```

```
ones(1,4)
```

```
ans = 1×4
     1     1     1     1
```

```
zeros(2,2)
```

```
ans = 2×2
     0     0
     0     0
```

```
KK=0:2:10
```

```
KK = 1×6
     0     2     4     6     8    10
```

```
B(1,:)
```

```
ans = 1×3
     1     2     3
```

```
B(2,:)
```

```
ans = 1×3
     4     5     6
```

```
B(:,2)
```

```
ans = 3×1
     2
     5
     8
```

In MATLAB, both `linspace` and `logspace` are used to create vectors of equally spaced values, but they differ in terms of the spacing between the values they generate.

1. `linspace`:

The `linspace` function generates a vector of linearly spaced values between a specified starting point and ending point. It takes three arguments: the starting value, the ending value, and the number of points you want in the generated vector. The generated values have a constant difference between them.

Example:

```matlab
start_value = 1;
end_value = 10;
num_points = 5;

linear_vector = linspace(start_value, end_value, num_points);
disp(linear_vector);
```

Output:

```
1    3.25    5.5    7.75    10
```

2. `logspace`:

The `logspace` function generates a vector of values that are logarithmically spaced. It takes the same three arguments as `linspace`: the starting exponent, the ending exponent, and the number of points you want. The function generates values that are evenly spaced on a logarithmic scale.

Example:

```matlab
start_exponent = 1;
end_exponent = 5;
num_points = 4;

logarithmic_vector = logspace(start_exponent, end_exponent, num_points);
```

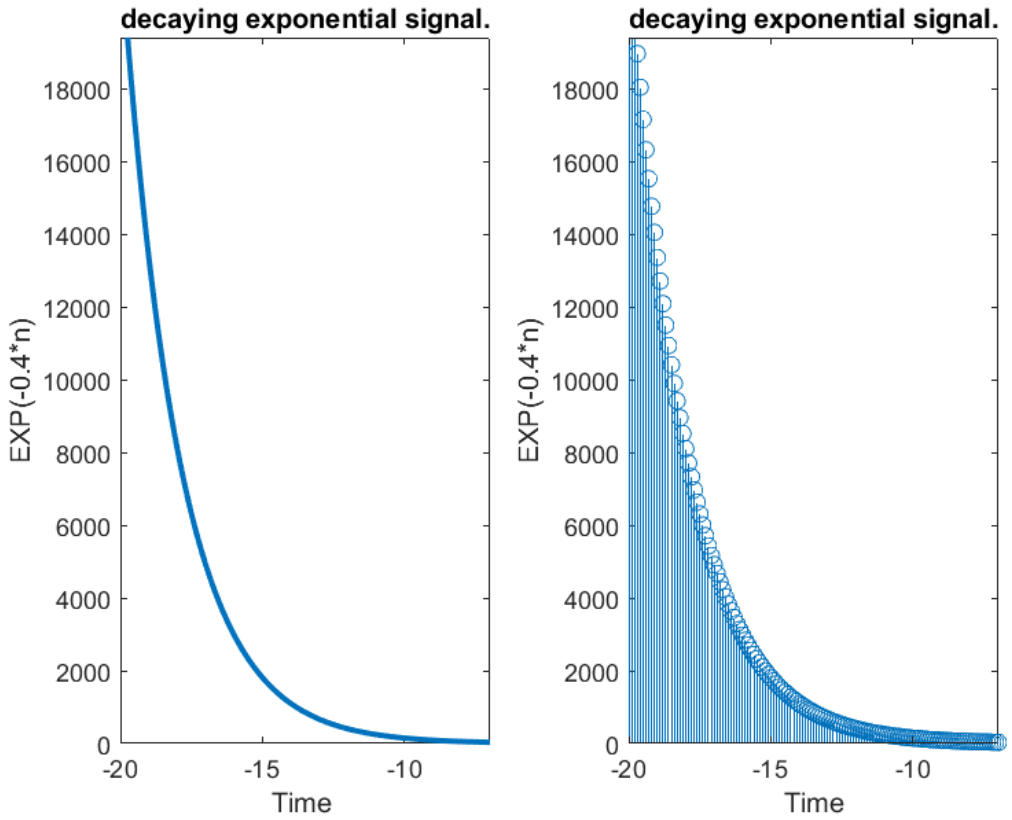
disp(logarithmic_vector);

```

Output:

```

10   100   1000   10000

```

In the `logspace` example, the generated values are 10^1, 10^2, 10^3, and 10^4, which are spaced evenly on a logarithmic scale.

In summary, `linspace` generates a vector with values spaced linearly, while `logspace` generates a vector with values spaced logarithmically. The choice between these functions depends on the specific application and the nature of the values you want to generat

```
F=input('Enter the Value In Fahrenheit temperature:')
```

```
F = 70
```

```
a=5/9;
c=(F-32)*a;
disp('Centigrade is') ,disp(c)
```

```
Centigrade is
   21.1111
```

ommand Window

```
Enter the Value In Fahrenheit temperature:70
>>
```
```

```
n=-20:0.1:20;
a=0.5;
y=exp(-a*n);
subplot(1,2,1)
plot(n,y,'LineWidth',2)
axis([-20  -7  0 19400  ])
subplot(1,2,2)
stem(n,y)
axis([-20  -7  0 19400  ])

subplot(1,2,1)
title('decaying exponential signal.')
xlabel('Time ')
ylabel('EXP(-0.4*n)')
subplot(1,2,2)
title('decaying exponential signal.')
xlabel('Time')
ylabel('EXP(-0.4*n)')
```



Implus signals

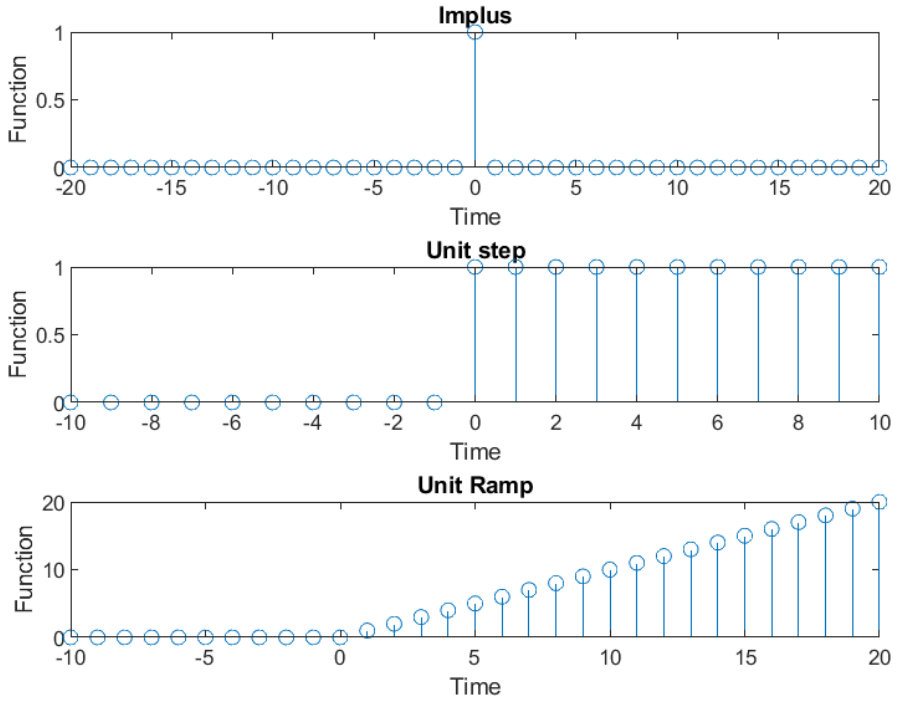
```
n=-20:20;
delta_n=n==0;
subplot(3,1,1,gca)
stem(n,delta_n)
```

Unit Step

```
n=-10:10;
x=n>=0;
subplot(3,1,2)
stem(n,x)
```

Unit Ramp

```
n=-10:20;
x=(n>=0).*n;
subplot(3,1,3)
stem(n,x)
```




```
p=0;
for a= 2:10
    p(a)=p(a-1)+2;
end
disp(p)
```

```
     0     2     4     6     8    10    12    14    16    18
```