COMMUNICATION SYSTEMS LAB 2 DATE-31/8/2021

NAME – JAGRIT LODHA

ID – 2019A3PS0165P

SECTION – P4

TASK 1 –

clc; clear all; close all;

% ID - 2019A3PS0165P

N=5;

B=N+5;

fm=B;

fs=15\*fm; % 15 is taken to show more accurate graph

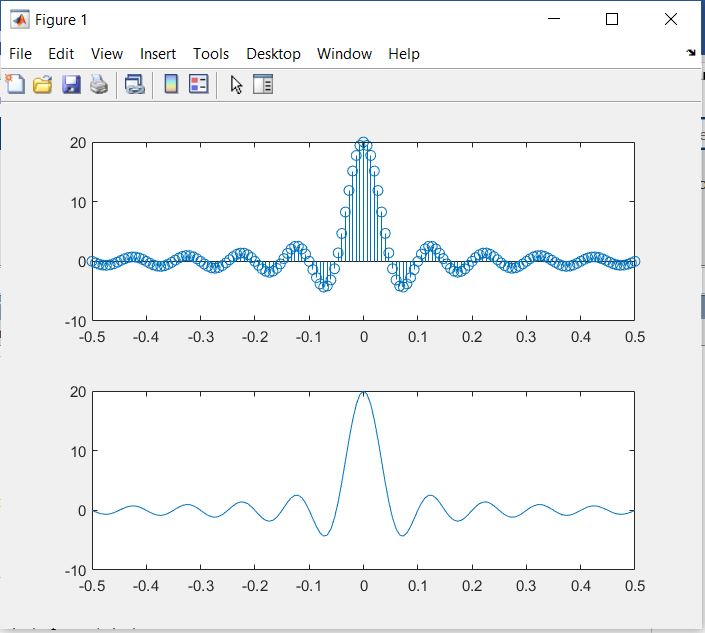
% Start time = -0.5 and Stop time = 0.5 to show symmetry

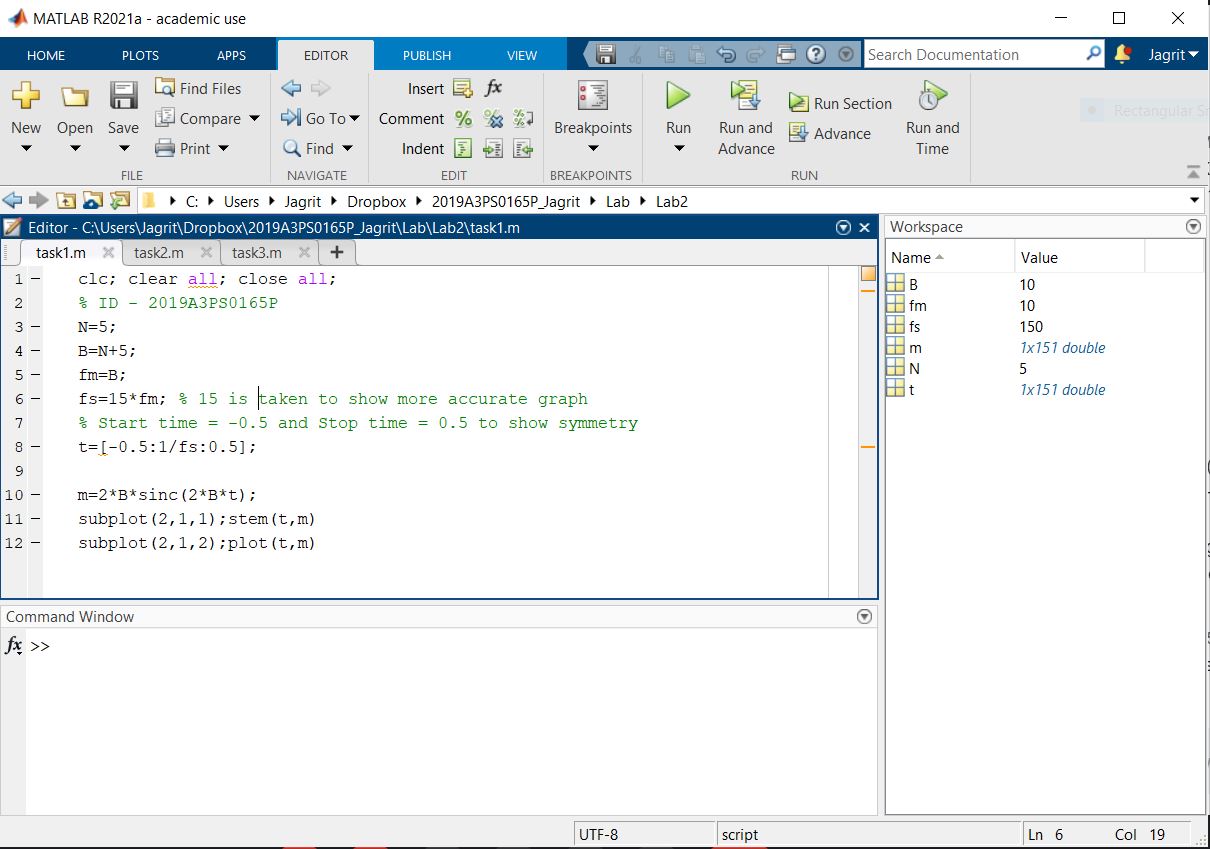
t=[-0.5:1/fs:0.5];

m=2\*B\*sinc(2\*B\*t);

subplot(2,1,1);stem(t,m)

subplot(2,1,2);plot(t,m)





We started off with fs = 5\*fm but there the sampling period was too small and so a continuous curved graph could not be obtained. So the value of fs was increased until accurate graphs were obtained.

Finally we settled at fs = 15\*fm as the stem plot obtained was also very satisfactory with close enough points for sampling.

TASK 2 –

clc; clear all; close all;

roc=280/(10^3);

ac=0.0969/(10^12);

Lo=587.3/(10^9);

Linf=426/(10^9);

b=1.385;

fm=745900;

Cinf=50/(10^12);

Co=0;

ce=1;

go=0;

ge=1;

f=4000; % Changed for different required values 4KHz, 4MHz and 4GHz

R=sqrt(sqrt((roc^4)+ac\*(f^2)));

L=(Lo+Linf\*((f/fm)^b))/(1+(f/fm)^b);

C=Cinf+Co\*(f^(-ce));

G=go\*f^ge;

w=2\*pi\*f;

gamma=sqrt((R+1j\*w\*L)\*(G+1j\*w\*C));

H\_channel=[];

l\_start=10;

l\_end=5000;

for d=l\_start:500:l\_end

H=(exp(1))^(-1\*gamma\*d);

H\_mod\_db=10\*log10(abs(H));

H\_channel=[H\_channel H\_mod\_db];

end

l\_axis=l\_start:500:l\_end;

H\_channel

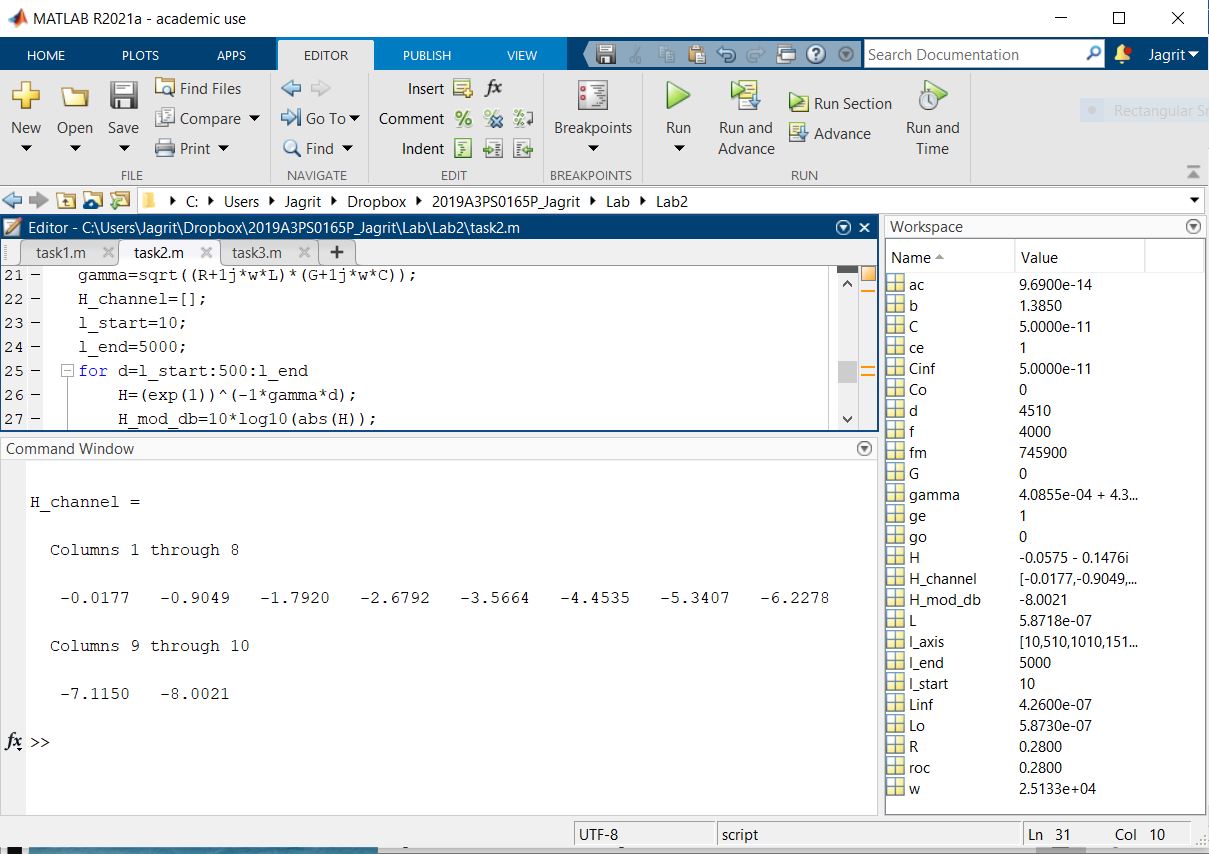
plot(l\_axis,H\_channel)

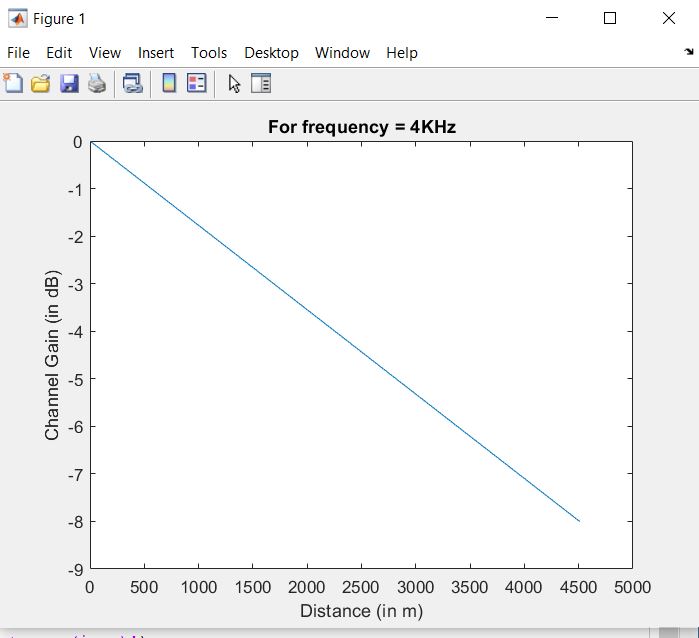
title('For frequency = 4KHz');

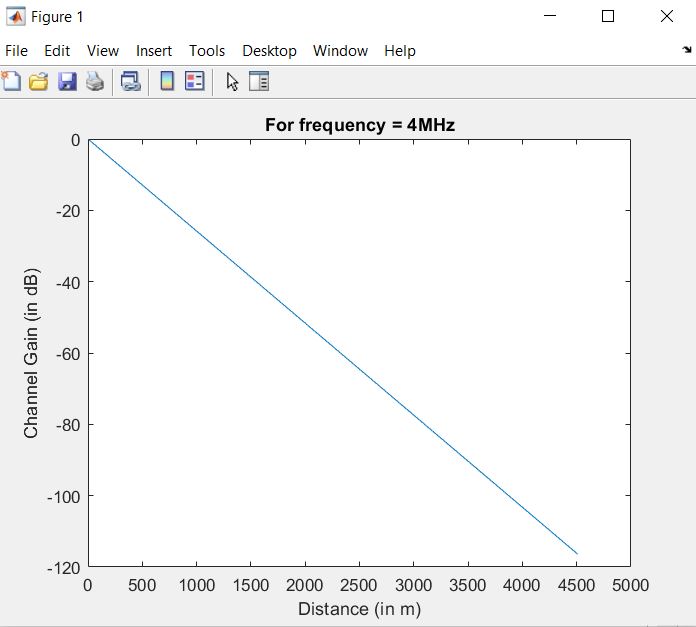
xlabel('Distance (in m)');

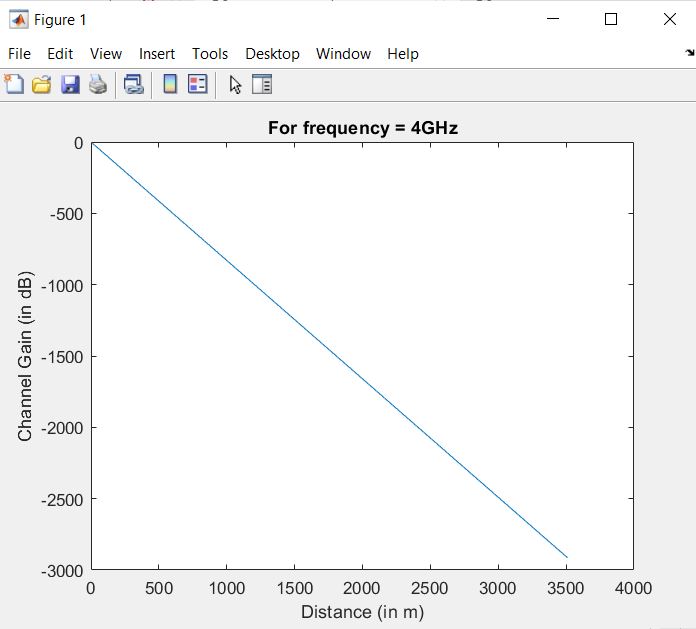
ylabel('Channel Gain (in dB)');

We run the loop and keep appending the channel gain values in the array H\_channel. The scrrenshot for the same has been shown for f=4KHz. The plots for channel gain have been shown for all three required frequencies, i.e., 4KHZ, 4MHz and 4GHz.









TASK 3 –

clc; clear all; close all;

% Name = JAGRIT LODHA

symbols={'A','D','G','H','I','J',' ','L','O','R','T'};

prob=[.167 .0833 .0833 .0833 .0833 .0833 .0833 .0833 .0833 .0833 .0833];

% J=1/12 A=2/12 G=1/12 R=1/12 I=1/12 T=1/12

% Blankspace=1/12

% L=1/12 O=1/12 D=1/12 H=1/12

[dict,avglen]=huffmandict(symbols,prob);

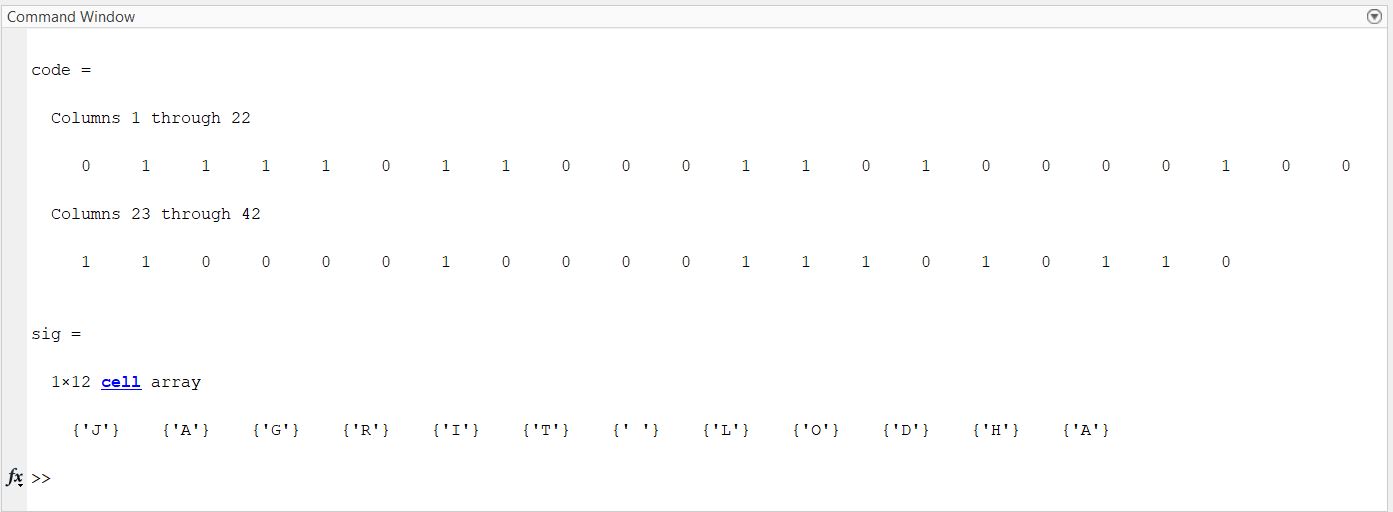
inpSig={'J','A','G','R','I','T',' ','L','O','D','H','A'};

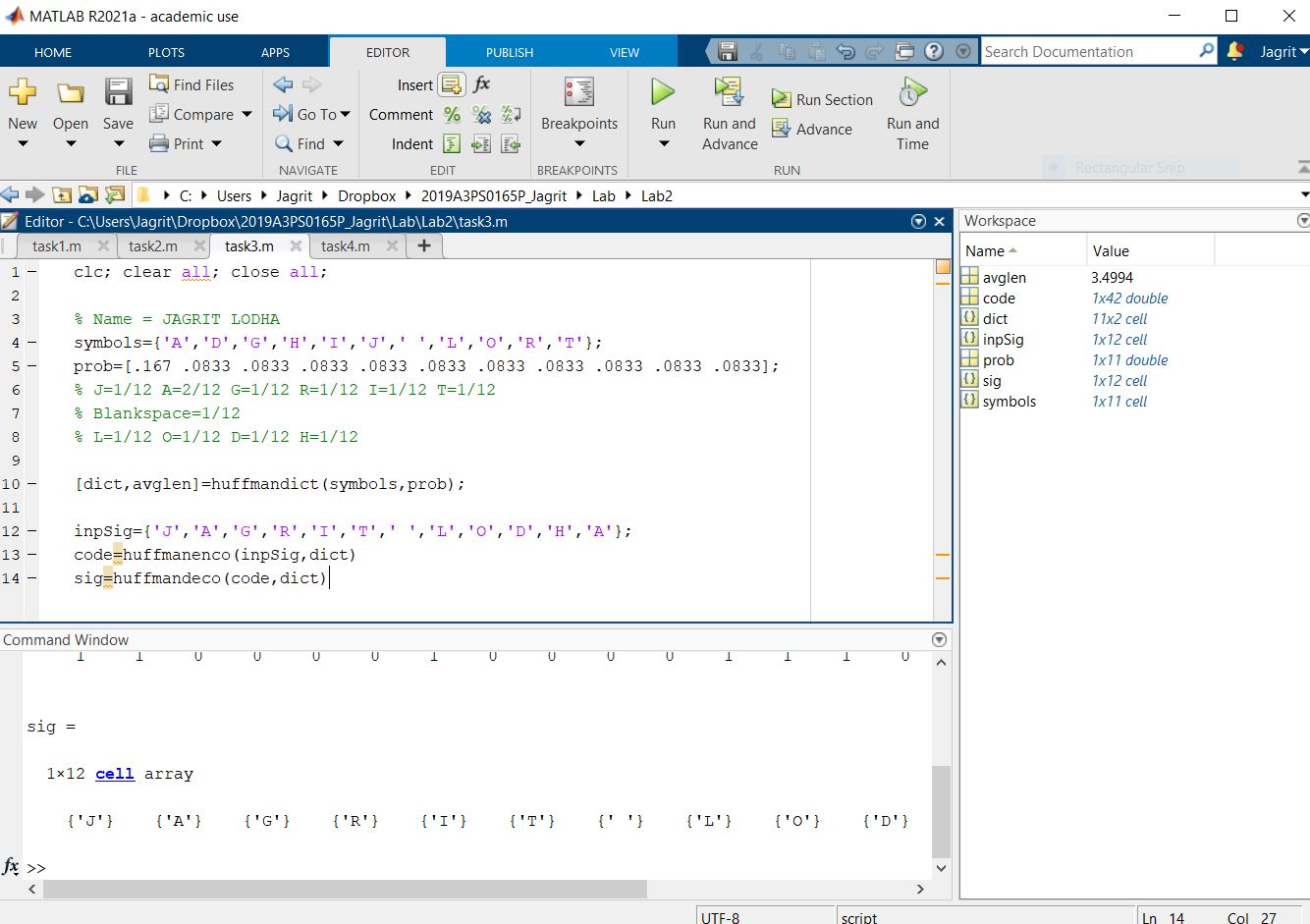
code=huffmanenco(inpSig,dict)

sig=huffmandeco(code,dict)

We first assign the probabilities for each letter in my name and assign them to the Huffman dictionary. Finally the Input signal is entered and we encode the signal.

We can also see that on decoding, we get the original signal back.





TASK 4 –

clear all; close all; clc;

t = 0:1/1000:6;

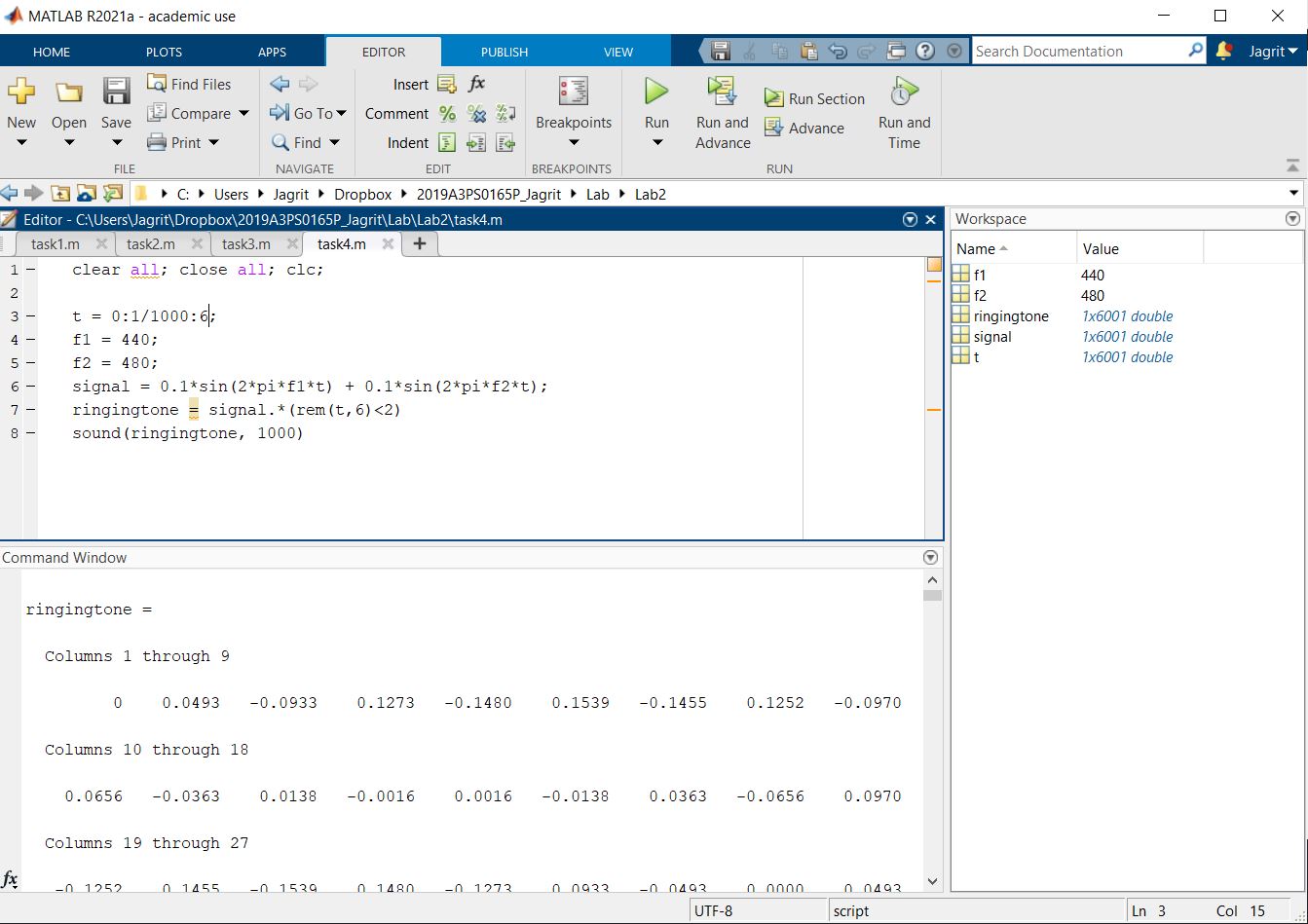
f1 = 440;

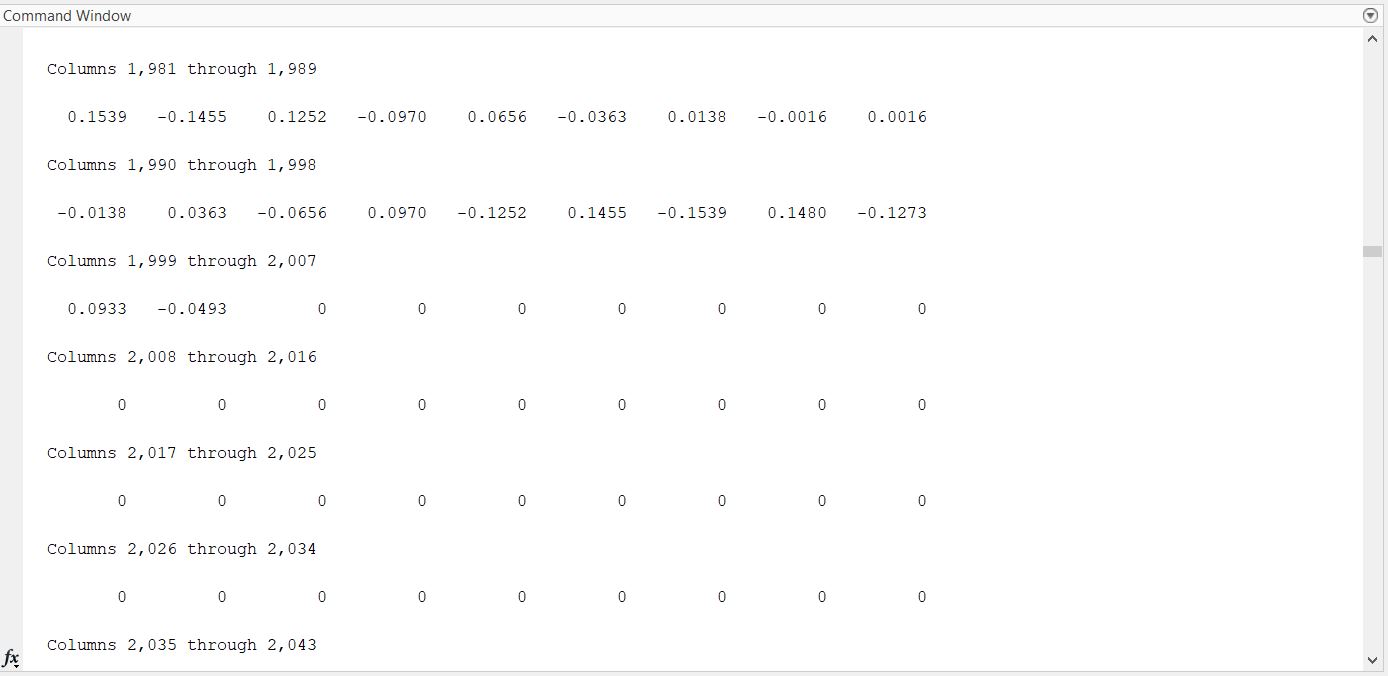
f2 = 480;

signal = 0.1\*sin(2\*pi\*f1\*t) + 0.1\*sin(2\*pi\*f2\*t);

ringingtone = signal.\*(rem(t,6)<2)

sound(ringingtone, 1000)





We see that the values go to 0 after 2000 columns (roughly 2 secs out of a total 6 secs.)