-APPENDIX-

EXPERIMENT 4.1 MLX DOCUMENT

```
clear;
M=7; K=3; N=2; L=2;

fs = 4000*K;
f_in = 300;
t = 0: 1/fs: 40e-3 - 1/fs;
input_signal = sin(2 * pi * f_in * t);

f_triangle = 300;
fs = 12000;
t = 0:1/fs:40e-3-1/fs;
triangular = sawtooth(2*pi*f_triangle*t,1/2);
```

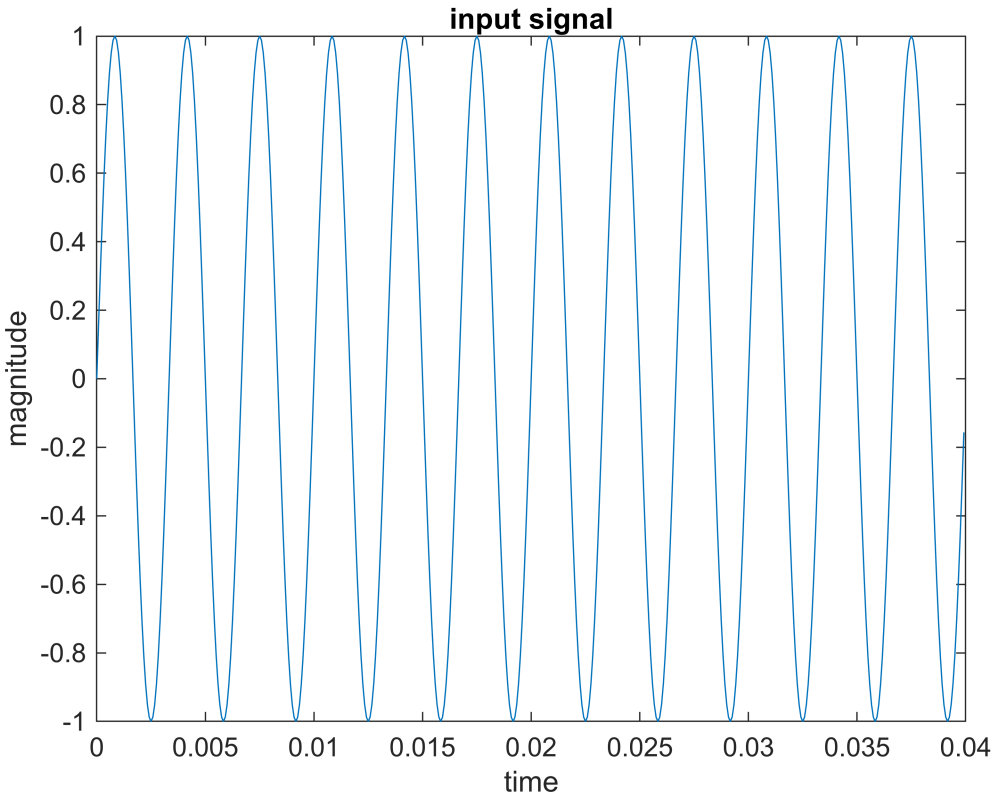
Part a)

EXPERIMENT 4.1 MLX DOCUMENT

```
M=7; K=3; N=2; L=2;
```

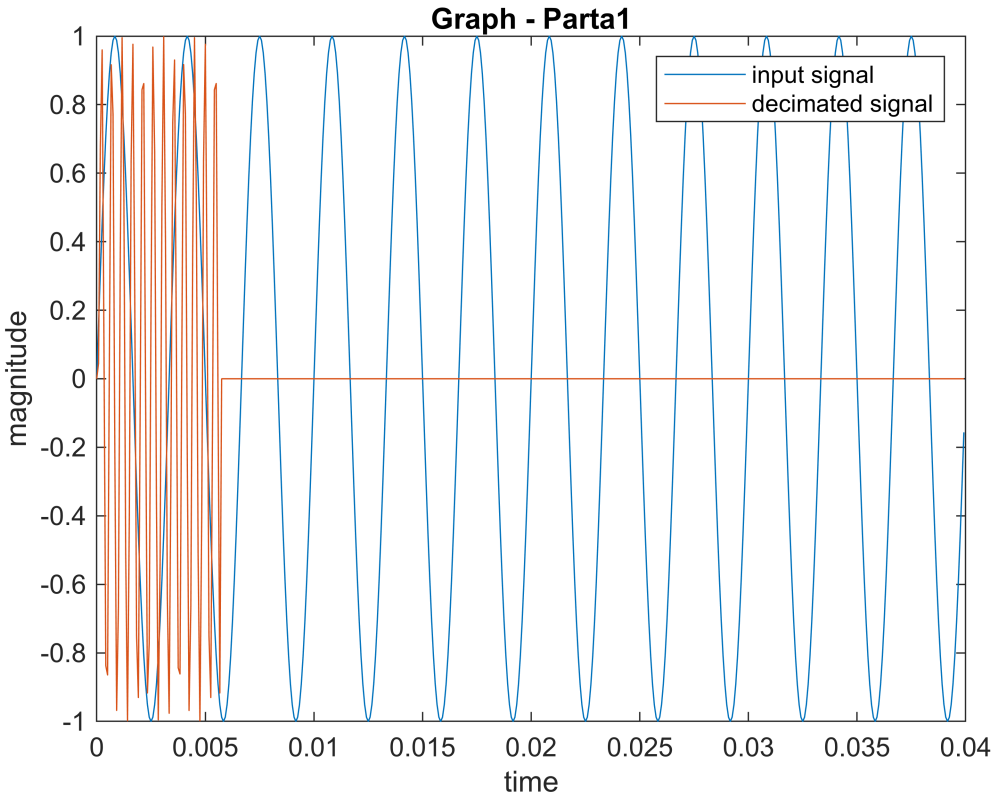
Part a)

```
fs = 4000*K;
f_in = 300;
t = 0: 1/fs: 40e-3 - 1/fs;
input_signal = sin(2 * pi * f_in * t);
figure;
plot(t, input_signal)
title("input signal")
xlabel("time");
ylabel("magnitude");
```



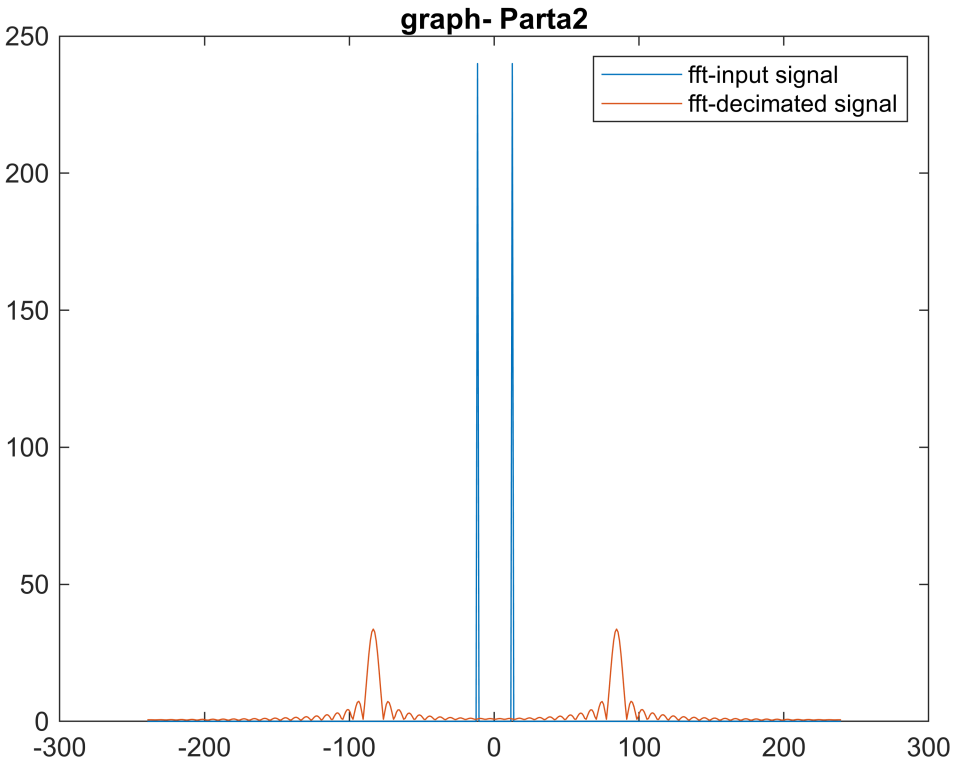
1)Decimation factor = 7

```
t = 0: 1/fs: 40e-3-1/fs;
decimated_signal = decimate(input_signal, 7);
decimated_signal2plot = cat(2, decimated_signal, zeros(1, length(input_signal)-length(decimated
figure;
plot(t, input_signal);
hold on;
plot(t, decimated_signal2plot);
title("Graph - Parta1")
xlabel("time");
ylabel("magnitude");
legend("input signal", "decimated signal");
```



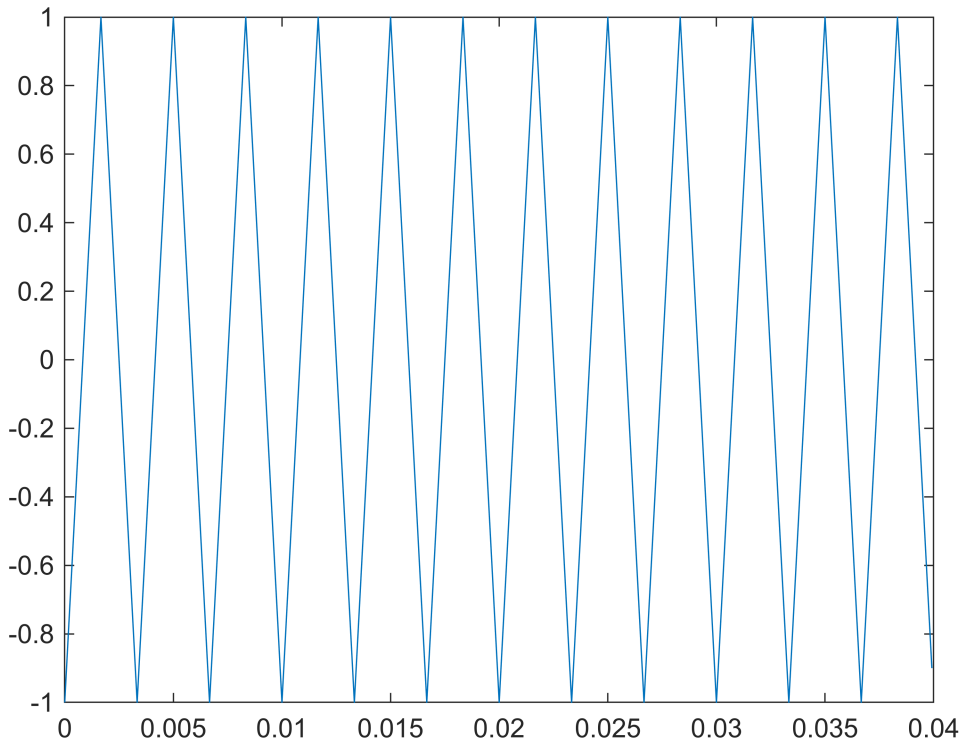
2)

```
f = -(length(input_signal)-1)/2:1:(length(input_signal)-1)/2;
fft_input = abs(fftshift(fft(input_signal)));
fft_decimated = abs(fftshift(fft(decimated_signal,length(input_signal))));
figure;
plot(f, fft_input);
hold on;
plot(f, fft_decimated);
legend("fft-input signal", "fft-decimated signal");
title("graph- Parta2")
```

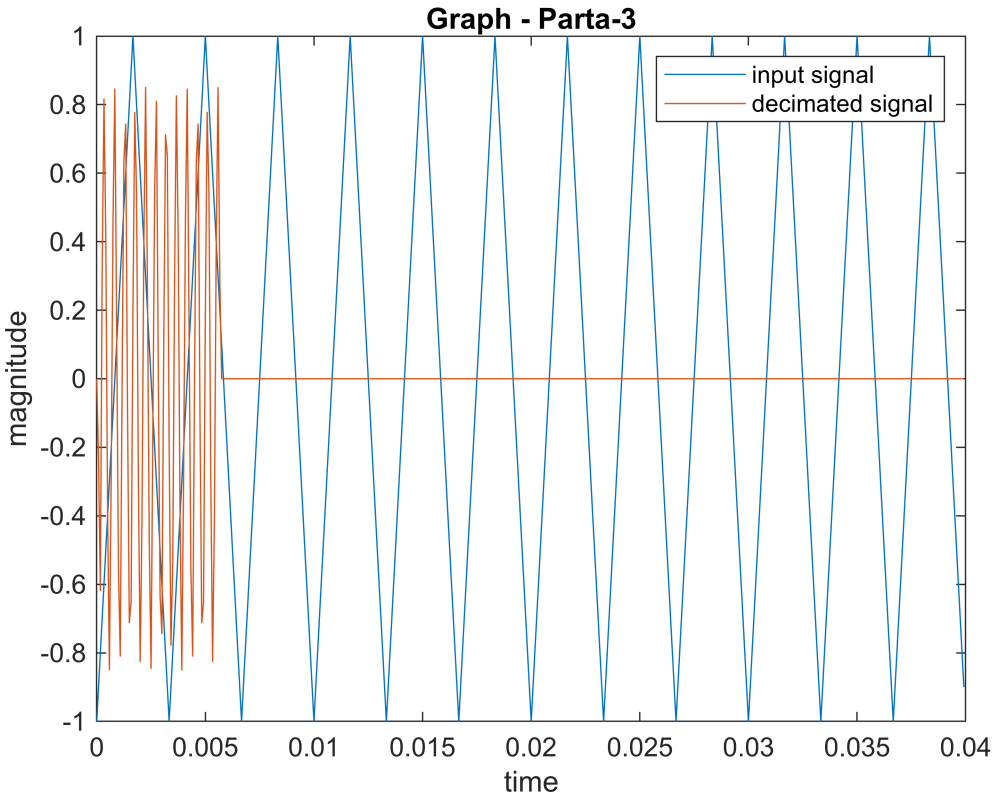


3)

```
f_triangle = 300;
fs = 12000;
t = 0:1/fs:40e-3-1/fs;
triangular = sawtooth(2*pi*f_triangle*t,1/2);
figure;
plot(t,triangular);
hold on;
```

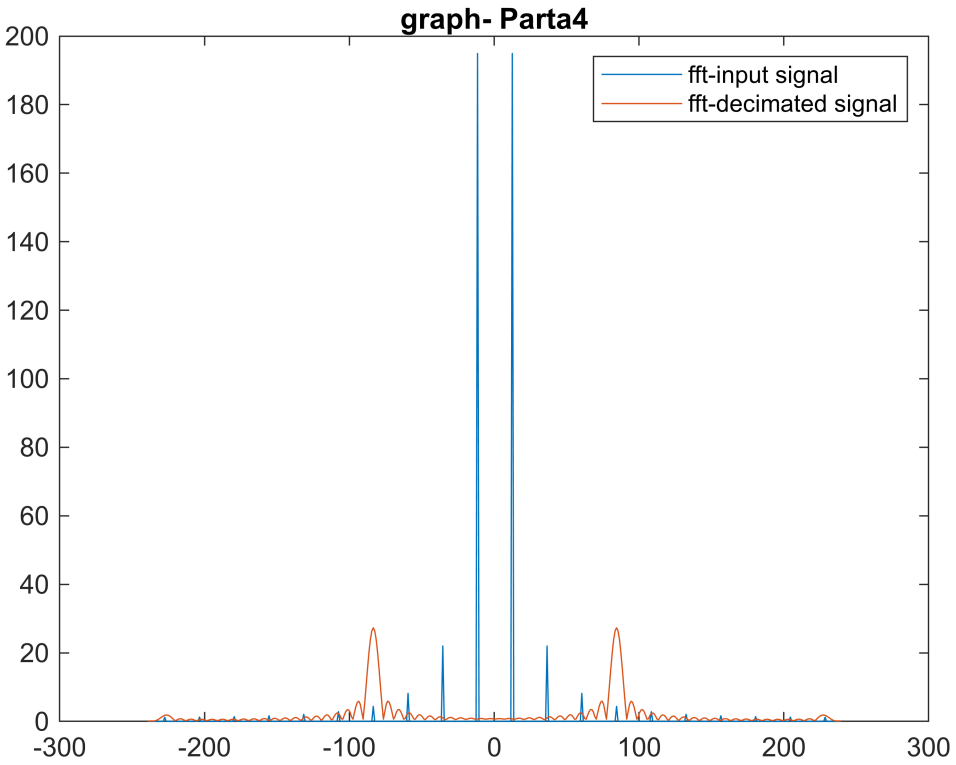



```
t = 0:1/fs:40e-3-1/fs;
decimated_signal = decimate(triangular, 7);
decimated_signal2plot = cat(2, decimated_signal, zeros(1, length(triangular)-length(decimated_s
figure;
plot(t, triangular);
hold on;
plot(t, decimated_signal2plot);
title("Graph - Parta-3")
xlabel("time");
ylabel("magnitude");
legend("input signal", "decimated signal");
```



4)

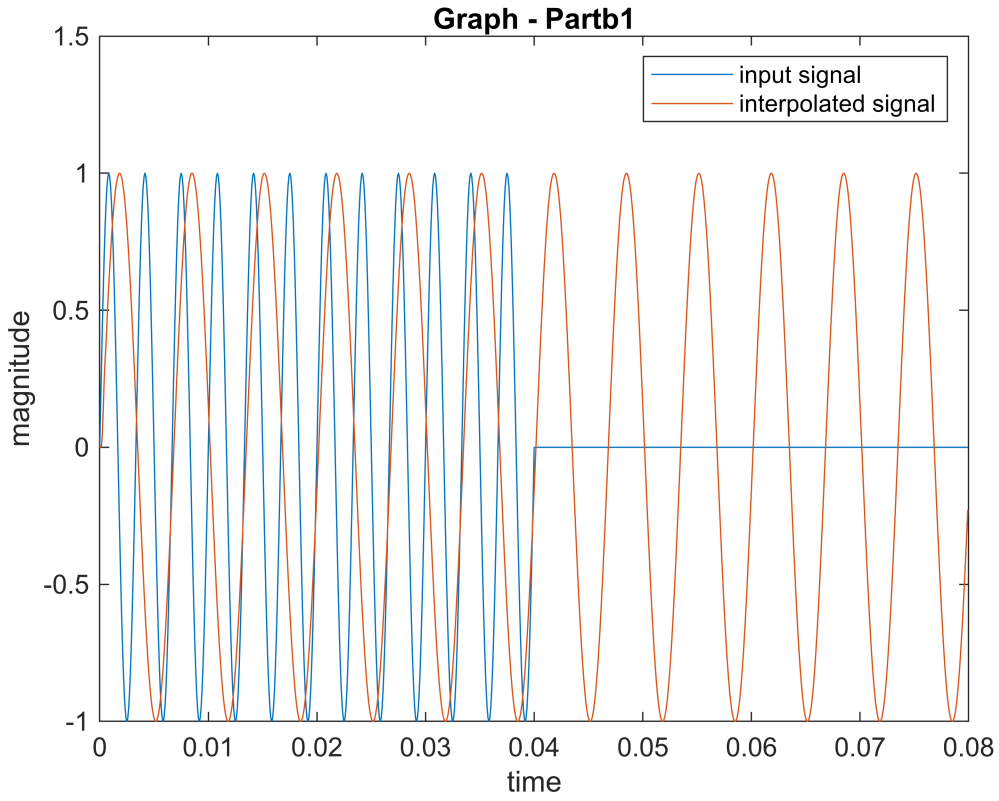
```
f = -(length(triangular)-1)/2:1:(length(triangular)-1)/2;
fft_input = abs(fftshift(fft(triangular)));
fft_decimated = abs(fftshift(fft(decimated_signal,length(triangular))));
figure;
plot(f, fft_input);
hold on;
plot(f, fft_decimated);
legend("fft-input signal", "fft-decimated signal");
title("graph- Parta4")
```



Part b)

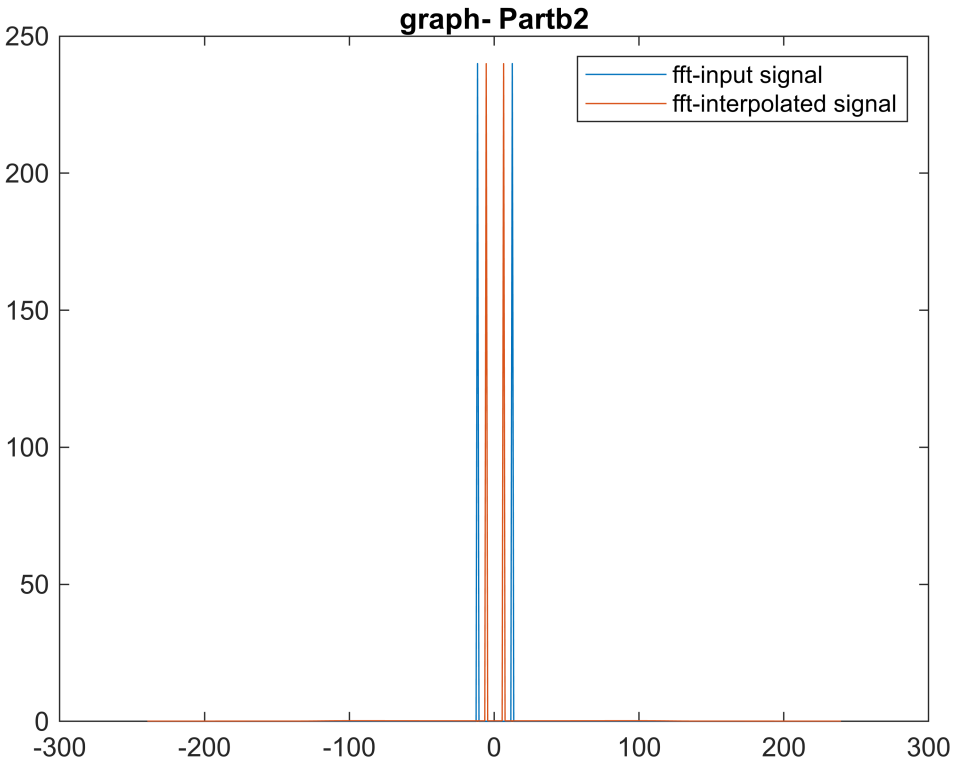
1) Interpol factor = 2

```
t = 0: 1/fs: 2*40e-3-1/fs;
interpolated_signal = interpol(input_signal, 2);
input_signal2plot = cat(2, input_signal, zeros(1, length(interpolated_signal)-length(input_sig
figure;
plot(t, input_signal2plot);
hold on;
plot(t, interpolated_signal);
title("Graph - Partb1")
xlabel("time");
ylabel("magnitude");
legend("input signal", "interpolated signal");
```



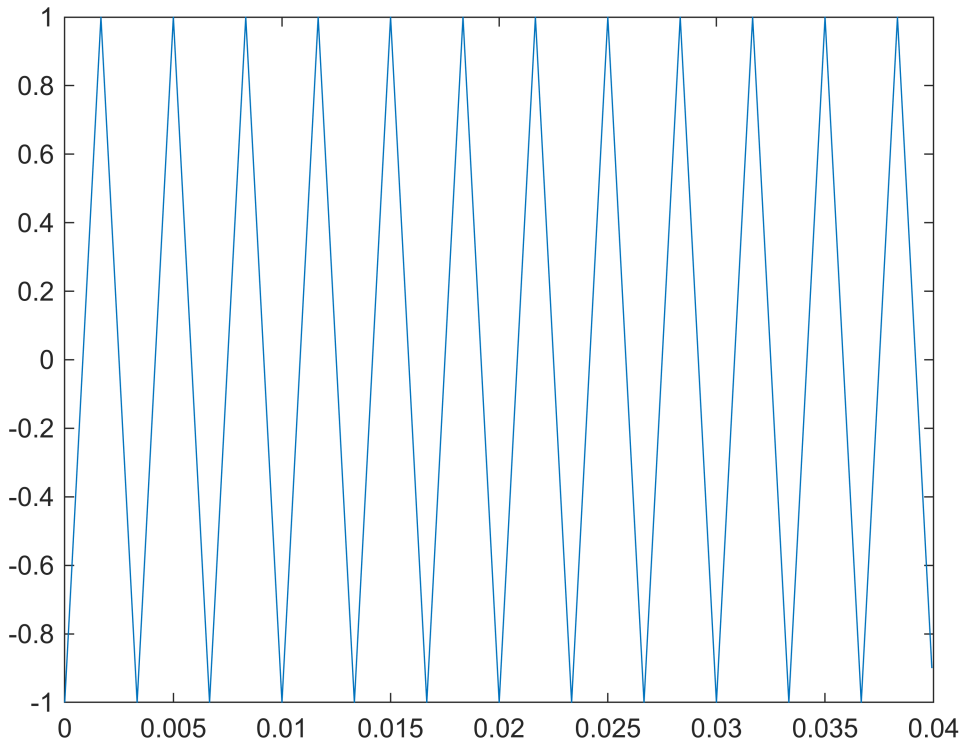
2)

```
f = -(length(input_signal)-1)/2:1:(length(input_signal)-1)/2;
fft_input = abs(fftshift(fft(input_signal)));
fft_interpolated = abs(fftshift(fft(interpolated_signal,length(input_signal))));
figure;
plot(f, fft_input);
hold on;
plot(f, fft_interpolated);
legend("fft-input signal", "fft-interpolated signal");
title("graph- Partb2")
```

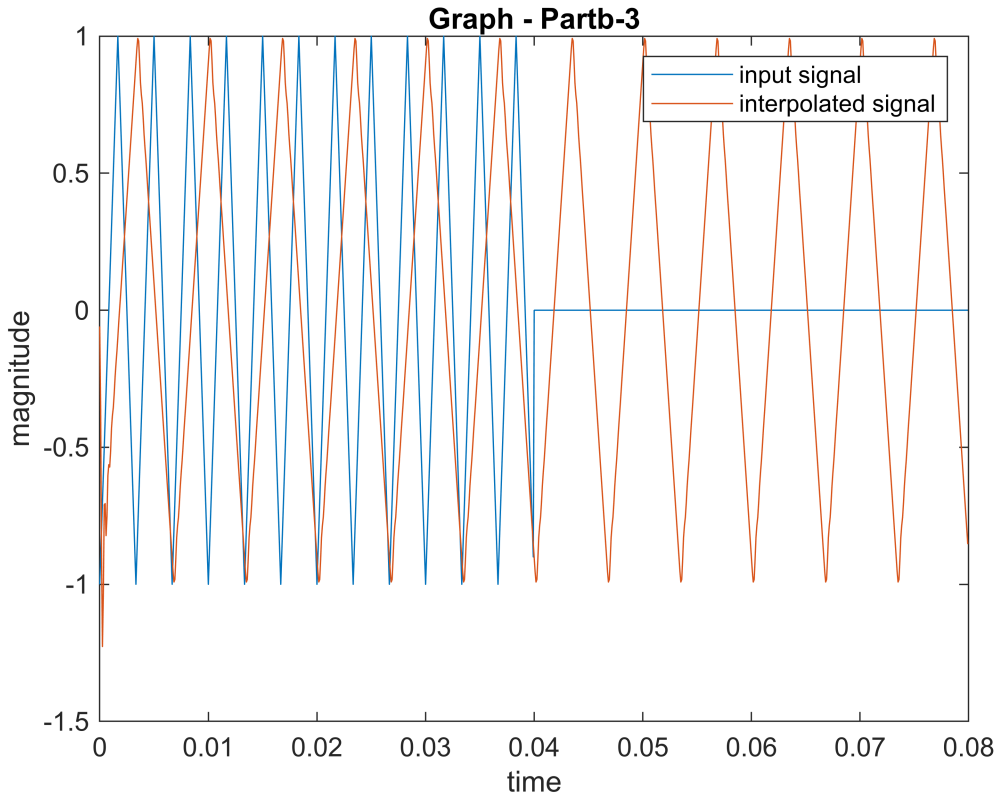


3)

```
f_triangle = 300;
fs = 12000;
t = 0:1/fs:40e-3-1/fs;
triangular = sawtooth(2*pi*f_triangle*t,1/2);
figure;
plot(t,triangular);
hold on;
```




```
t = 0:1/fs:2*40e-3-1/fs;
interpolated_signal = interpol(triangular, 2);
triangular_signal2plot = cat(2, triangular, zeros(1, length(interpolated_signal)-length(triangu
figure;
plot(t, triangular_signal2plot);
hold on;
plot(t, interpolated_signal);
title("Graph - Partb-3")
xlabel("time");
ylabel("magnitude");
legend("input signal", "interpolated signal");
```



4)

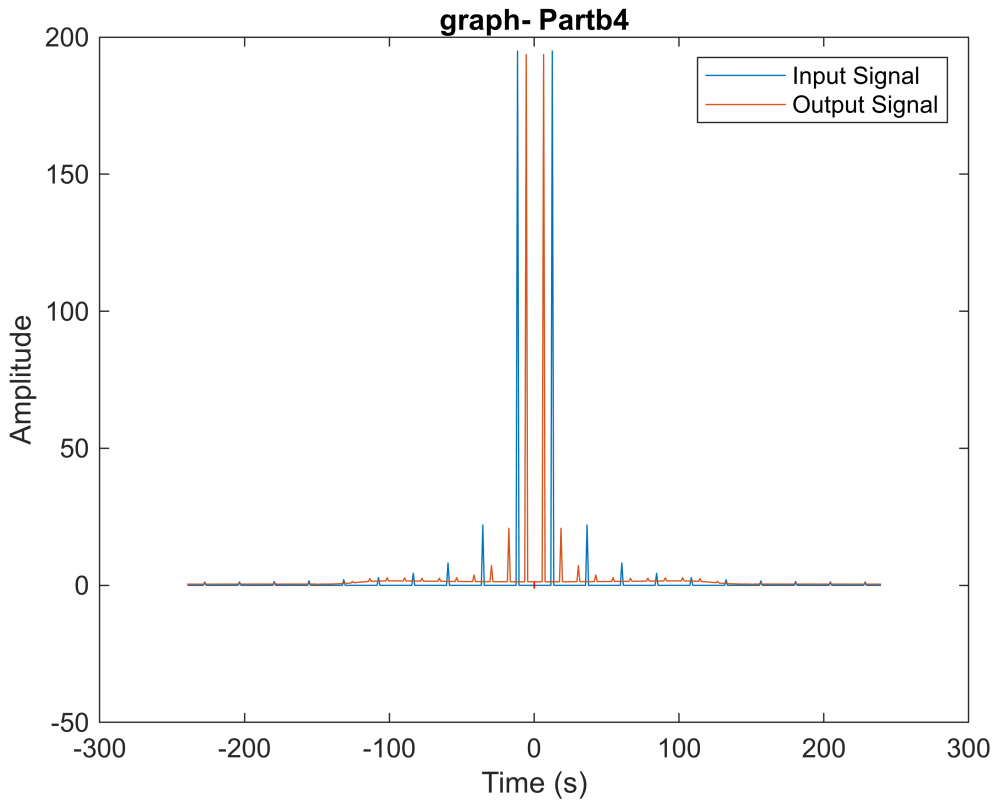
```
f = -(length(triangular)-1)/2:1:(length(triangular)-1)/2;
fft_input = abs(fftshift(fft(triangular)));
fft_interpolated = abs(fftshift(fft(interpolated_signal,length(input_signal))));
figure;
plot(f, fft_input);
hold on;
plot(f, fft_interpolated);
legend("fft-input signal", "fft-interpolated signal");
title("graph- Partb4")
```

Part c)

c.1)

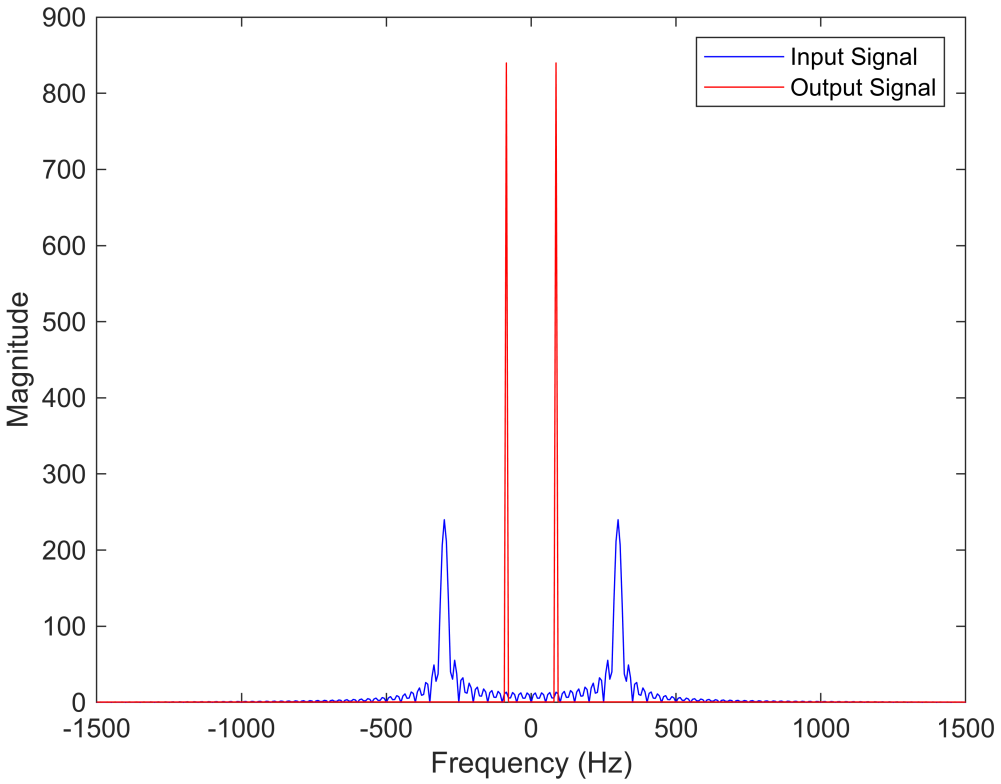
```
sr_changed_signal = rate_change(input_signal,(M/L));

t_out = 0: 1/fs: (40e-3*M/L - 1/fs);
plot(t_out, [input_signal, zeros(1,(length(t_out)-length(input_signal)))], 'b');
hold on
plot(t_out, sr_changed_signal, 'r');
xlabel('Time (s)');
```

```
ylabel('Amplitude');
legend('Input Signal', 'Output Signal');
hold off;
```



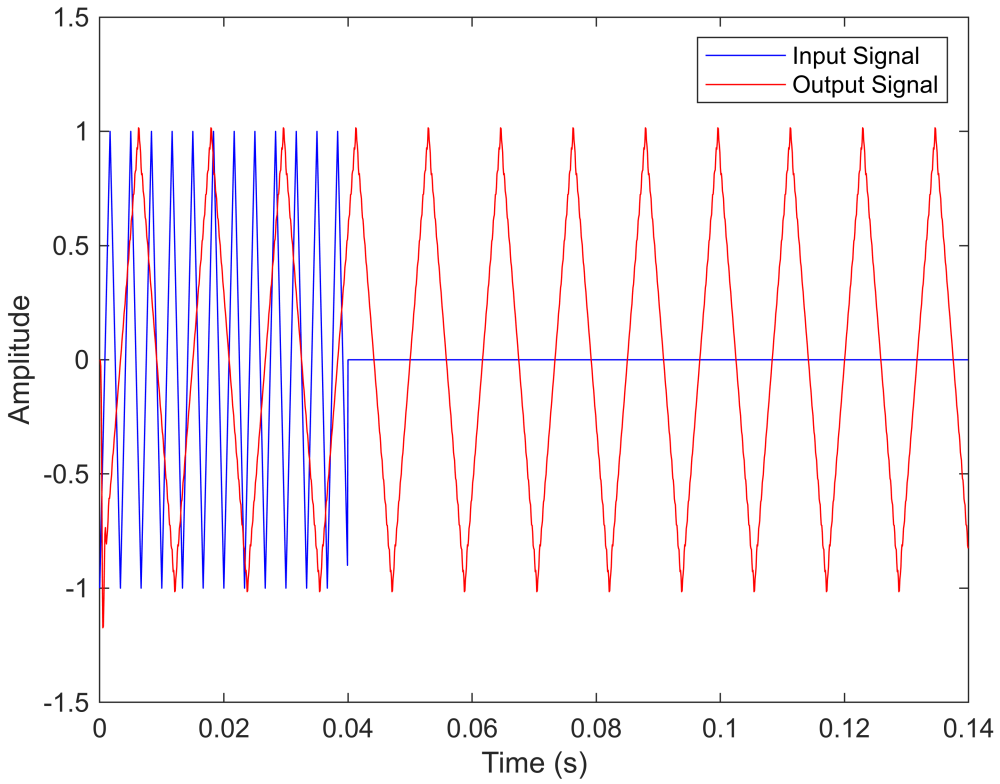
c.2)

```
NFFT = length(sr_changed_signal);  % length of FFT
Y_in = fft(input_signal, NFFT);
Y_out = fft(sr_changed_signal, NFFT);

f = fs*(-NFFT/2:NFFT/2-1)/NFFT;

figure;
plot(f, fftshift(abs(Y_in)), 'b', f, fftshift(abs(Y_out)), 'r');
xlabel('Frequency (Hz)');
xlim ([-1500 1500]);
ylabel('Magnitude');
legend('Input Signal', 'Output Signal');
```



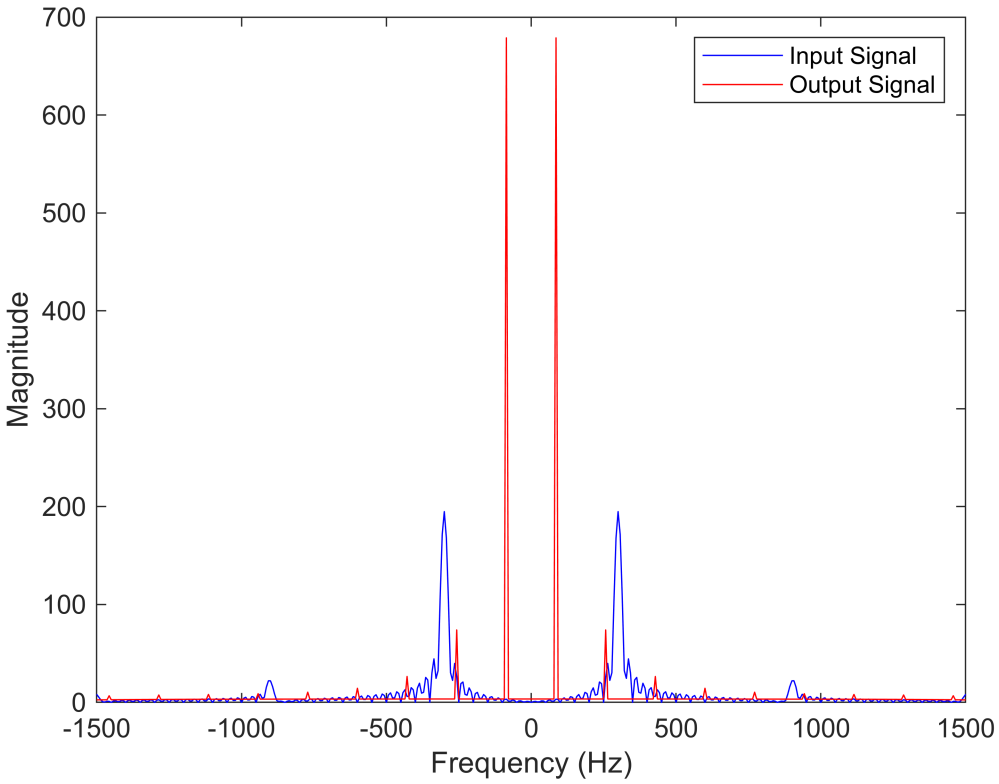
c.3)

```
tr_changed_signal = rate_change(triangular,(M/L));

t_out = 0: 1/fs: (40e-3*M/L - 1/fs);
plot(t_out, [triangular, zeros(1,(length(t_out)-length(triangular)))], 'b');
hold on
plot(t_out, tr_changed_signal, 'r');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Input Signal', 'Output Signal');
hold off;
```



c.4)

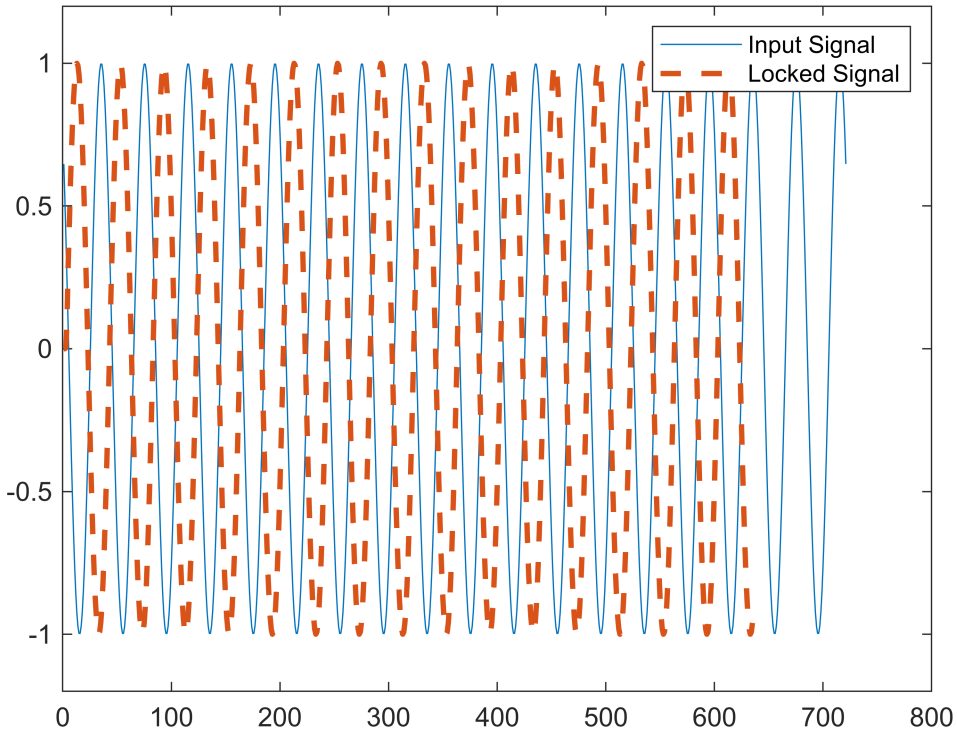
```
NFFT = length(tr_changed_signal);  % length of FFT
Y_in = fft(triangular, NFFT);
Y_out = fft(tr_changed_signal, NFFT);

f = fs*(-NFFT/2:NFFT/2-1)/NFFT;

figure;
plot(f, fftshift(abs(Y_in)), 'b', f, fftshift(abs(Y_out)), 'r');
xlabel('Frequency (Hz)');
xlim ([-1500 1500]);
ylabel('Magnitude');
legend('Input Signal', 'Output Signal');
```



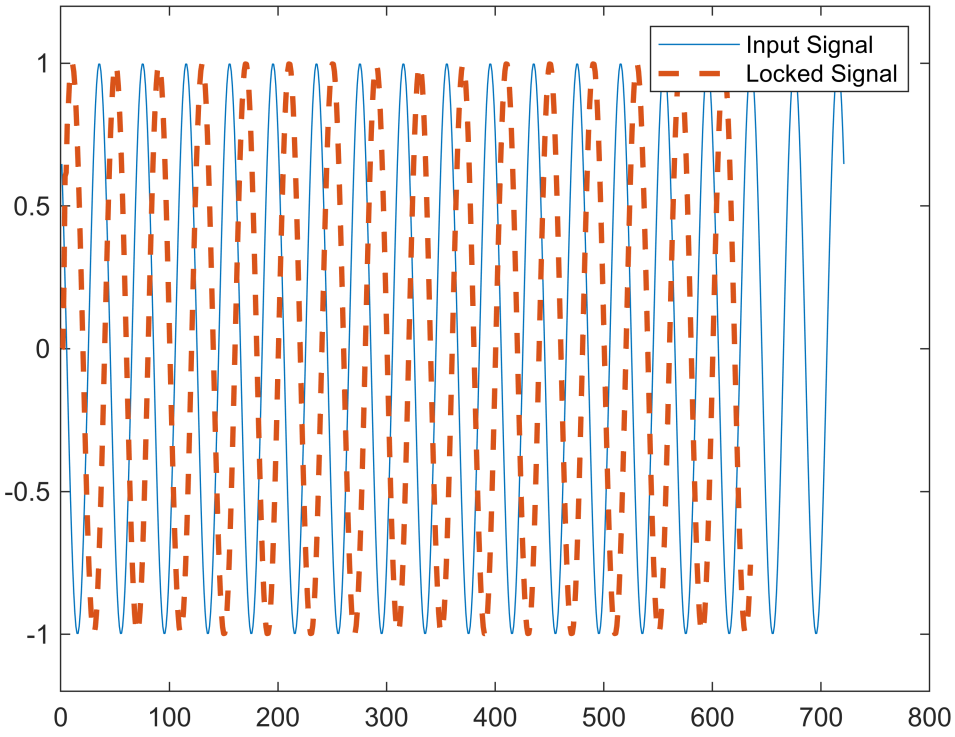
Part d)

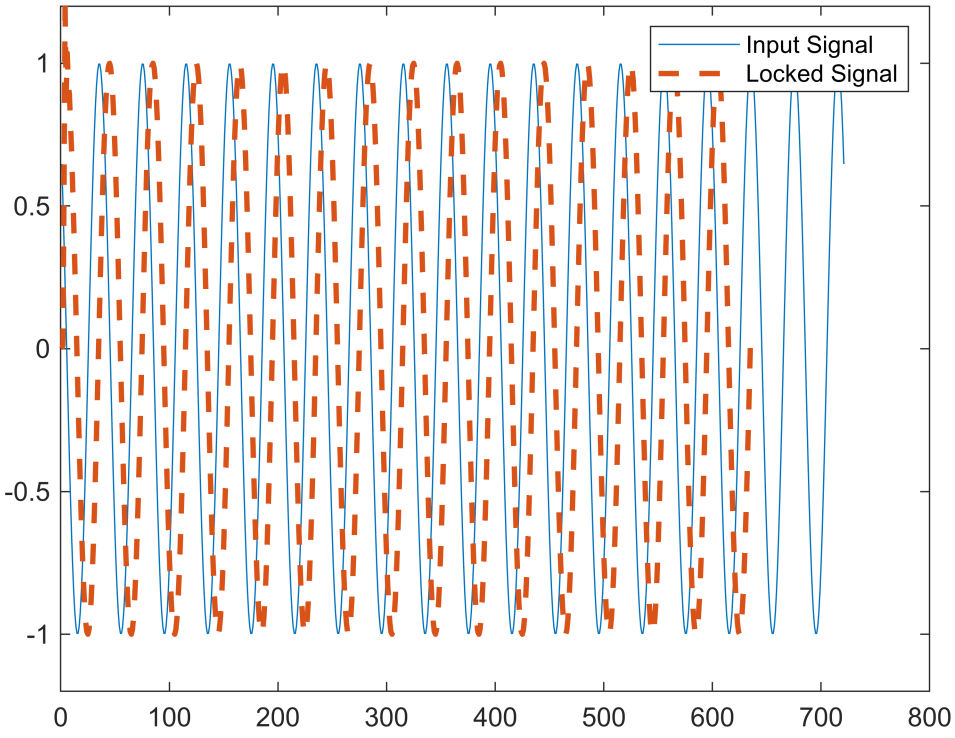
d.1)

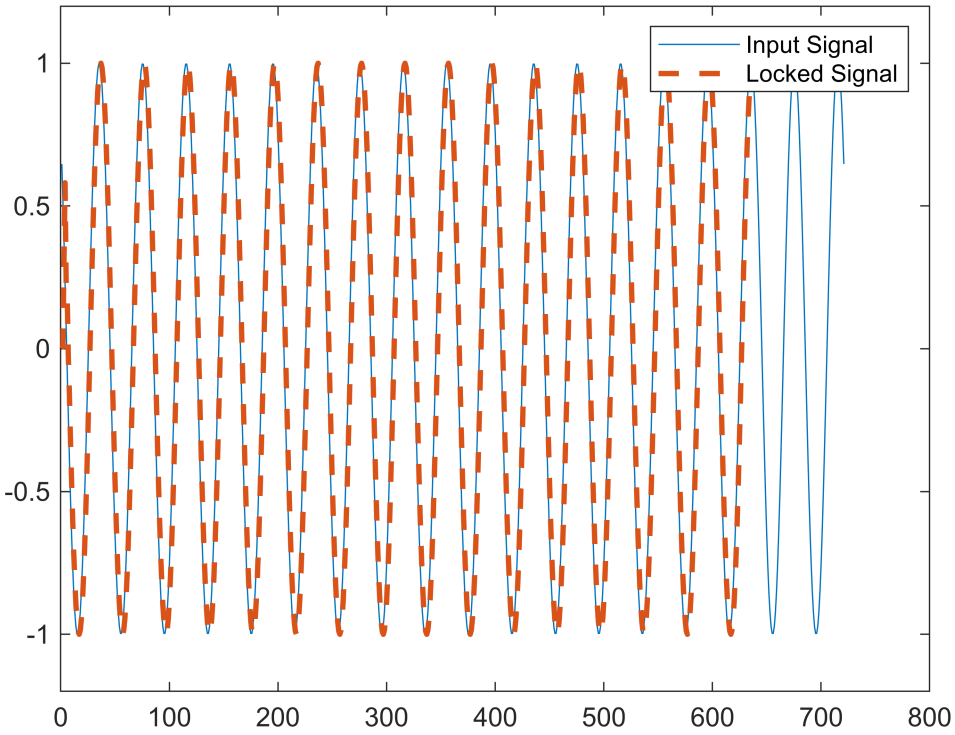
```
fs = 4000*(M+L);                    % fs = 36 kHz
oscil_freq = 750;                   % given initially
lock_freq = 100*(L+M);              % lock_freq = 900
lock_phase =  pi * rand;            % random phase
loop_gain = 0.4 ;
suggested_rate_change = oscil_freq / lock_freq ;
x = 0 : 1/fs: 18/lock_freq;                 % 18 waveforms will be seen
figure;
input_signal = cos(2 * pi * x * lock_freq + lock_phase);
plot(input_signal, LineStyle="-")
hold on;
[y, phase] = DPLL(input_signal, 0, loop_gain, oscil_freq, fs,suggested_rate_change);
plot(y, LineStyle="--", LineWidth=2);
legend('Input Signal', 'Locked Signal');
ylim([-1.2,1.2]);
```

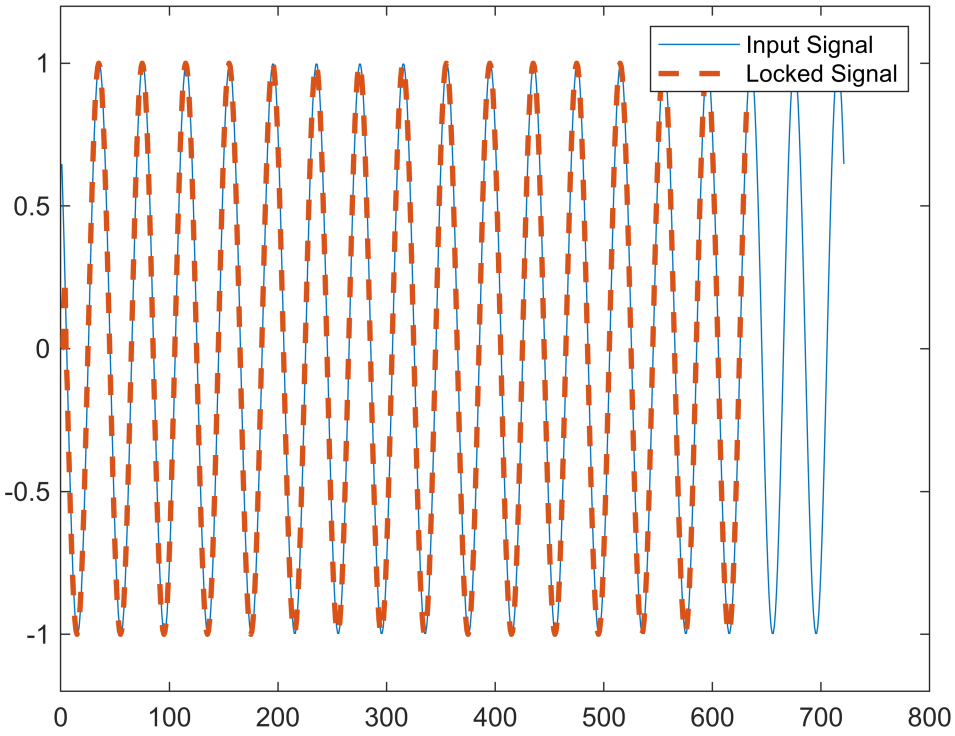


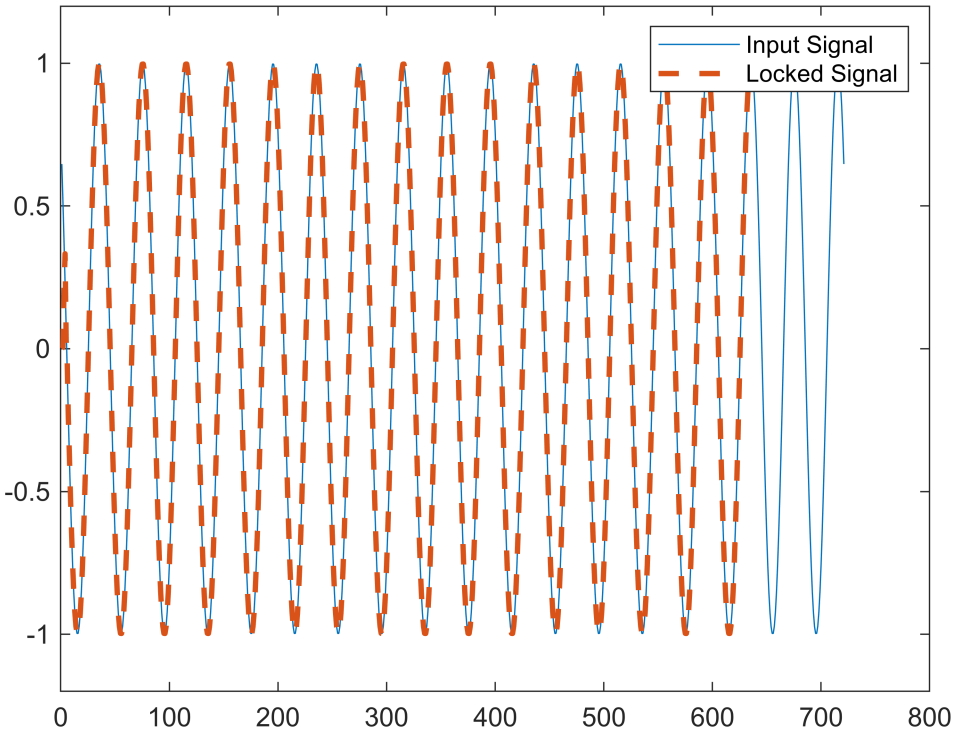

```
pause(1)
for k = 1:15
    figure;
    ylim([-1.1,1.1])
    plot(input_signal, LineStyle="-")
    hold on;
    [y, phase] = DPLL(input_signal, phase, loop_gain, oscil_freq, fs,suggested_rate_change);
    plot(y, LineStyle="--", LineWidth=2);
    legend('Input Signal', 'Locked Signal');
    ylim([-1.2,1.2]);
    %pause(1)

end
```

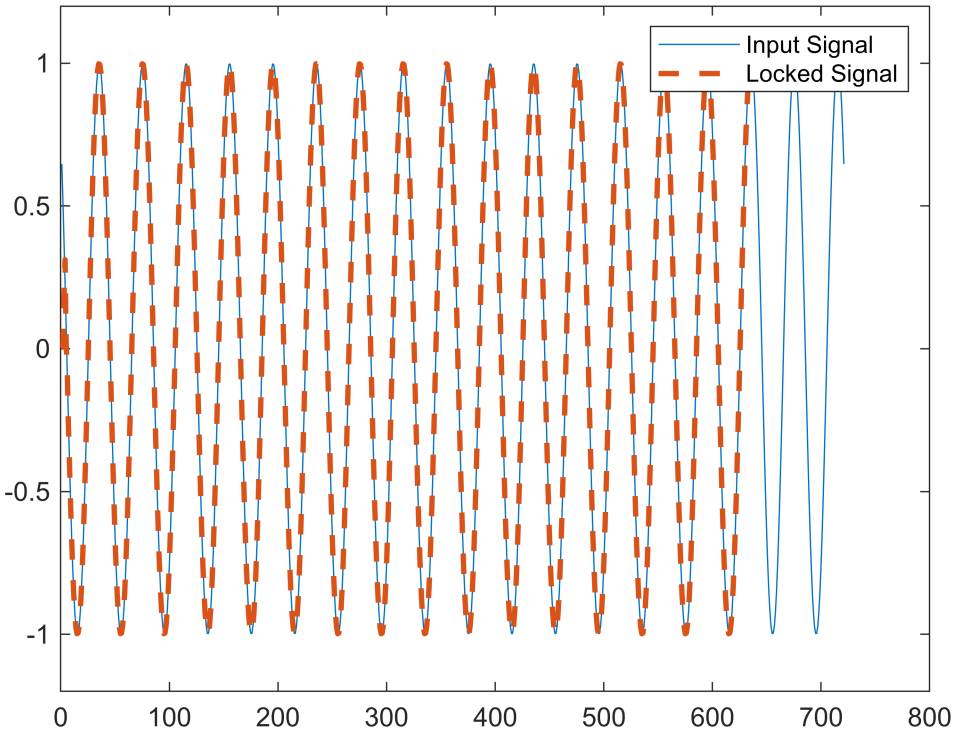

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

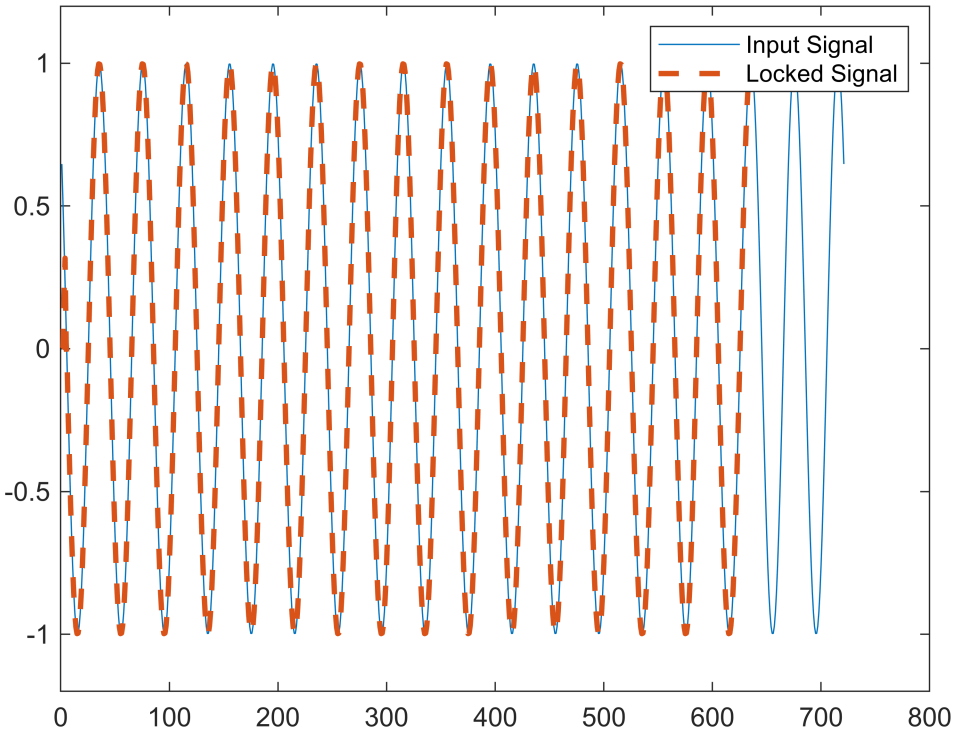

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

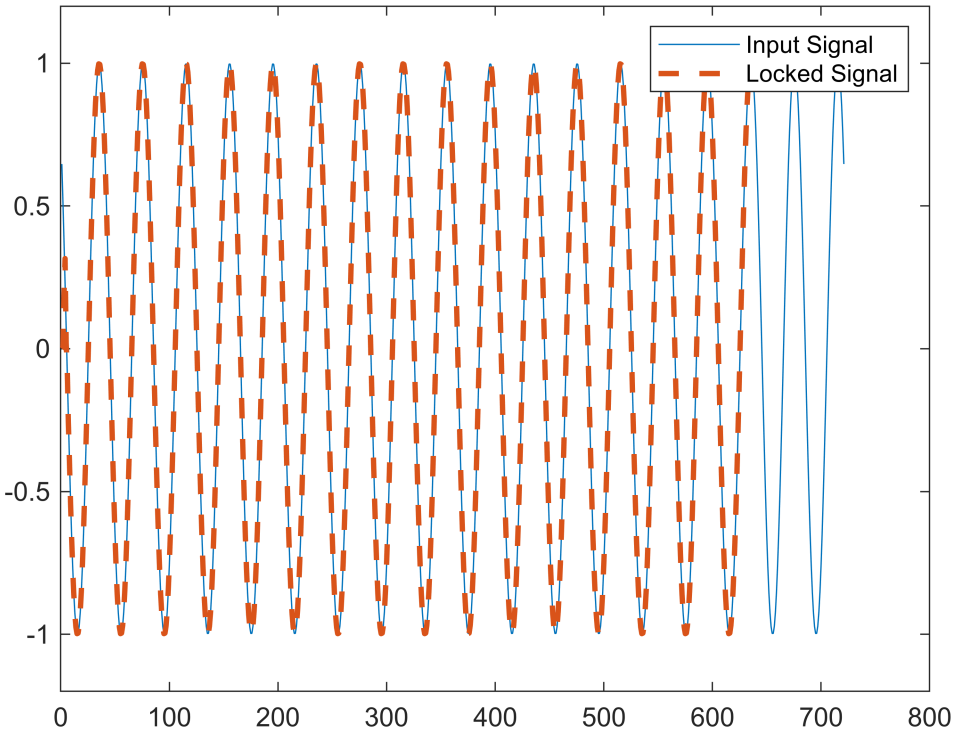

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

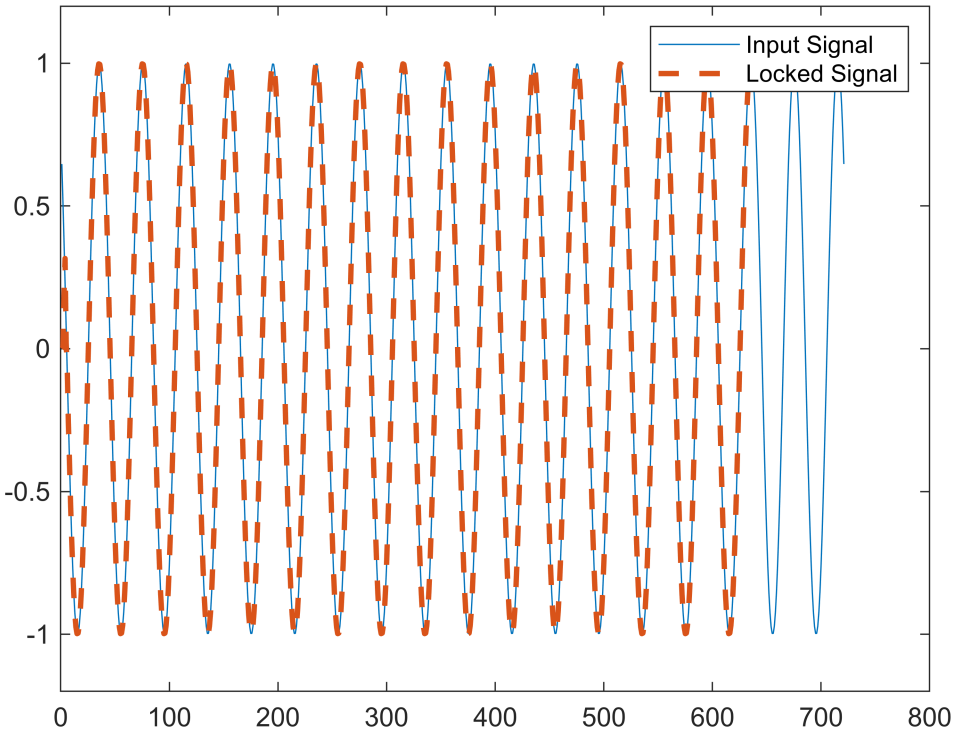

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

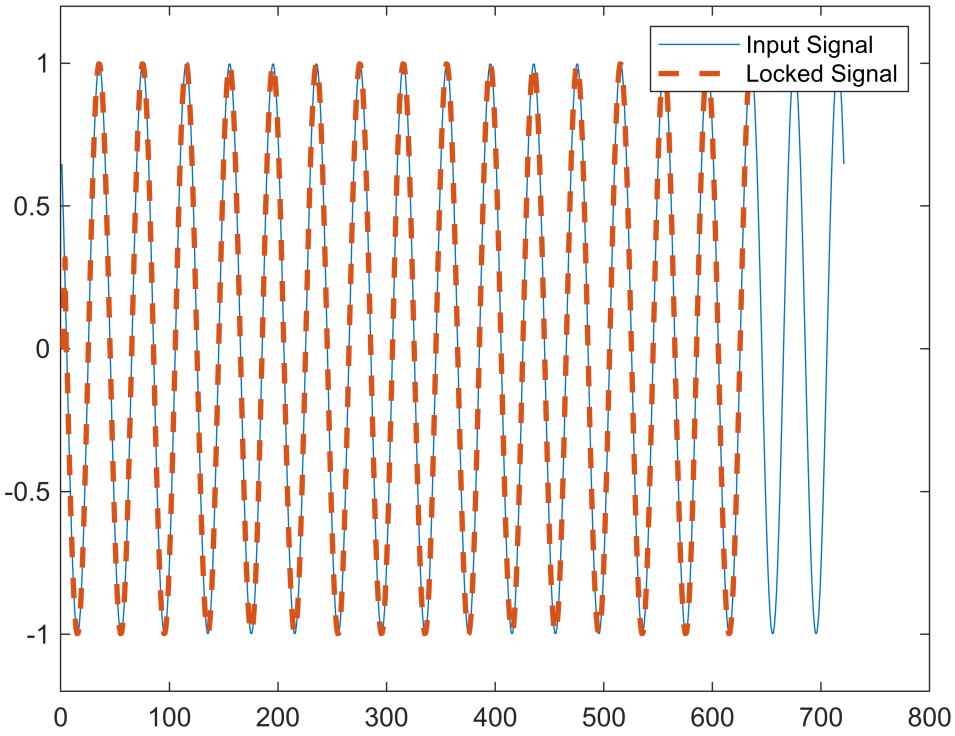

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

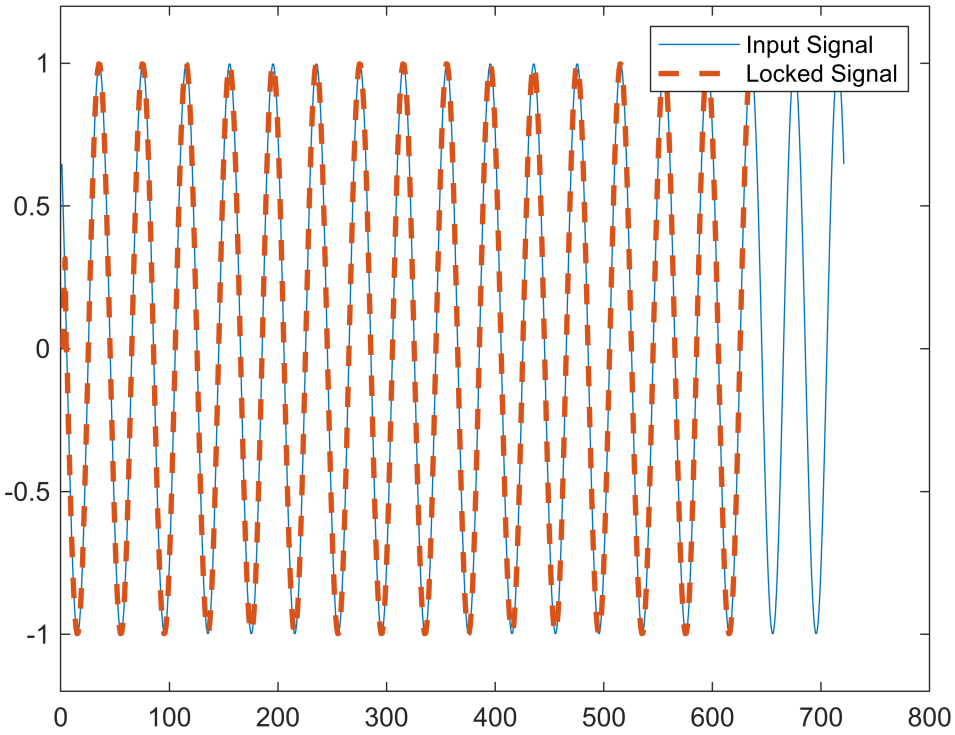

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

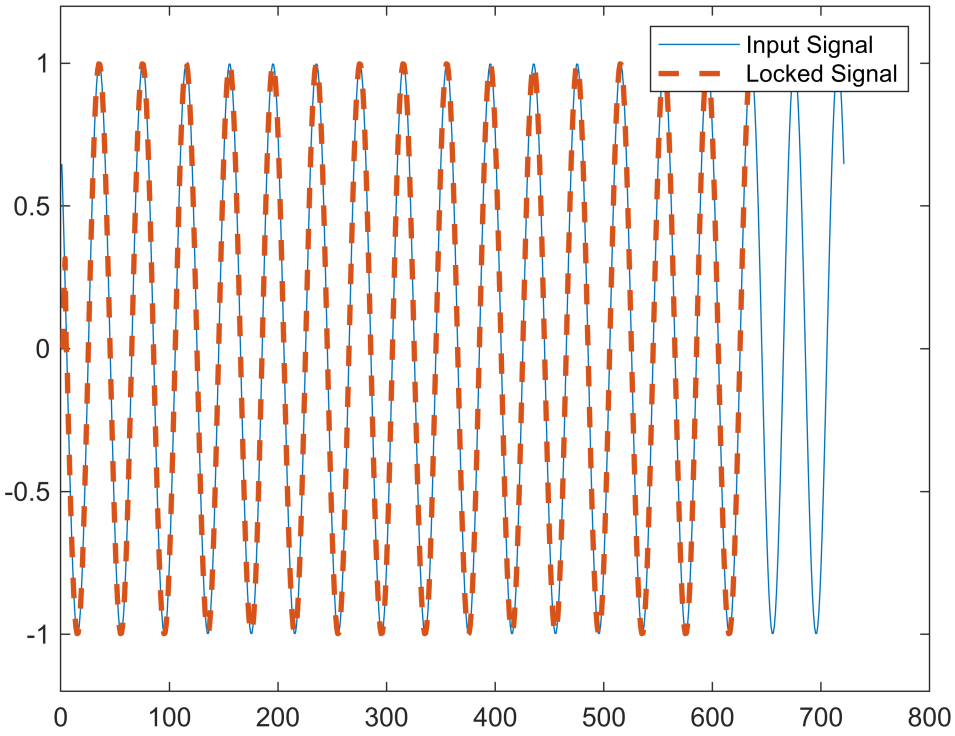

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

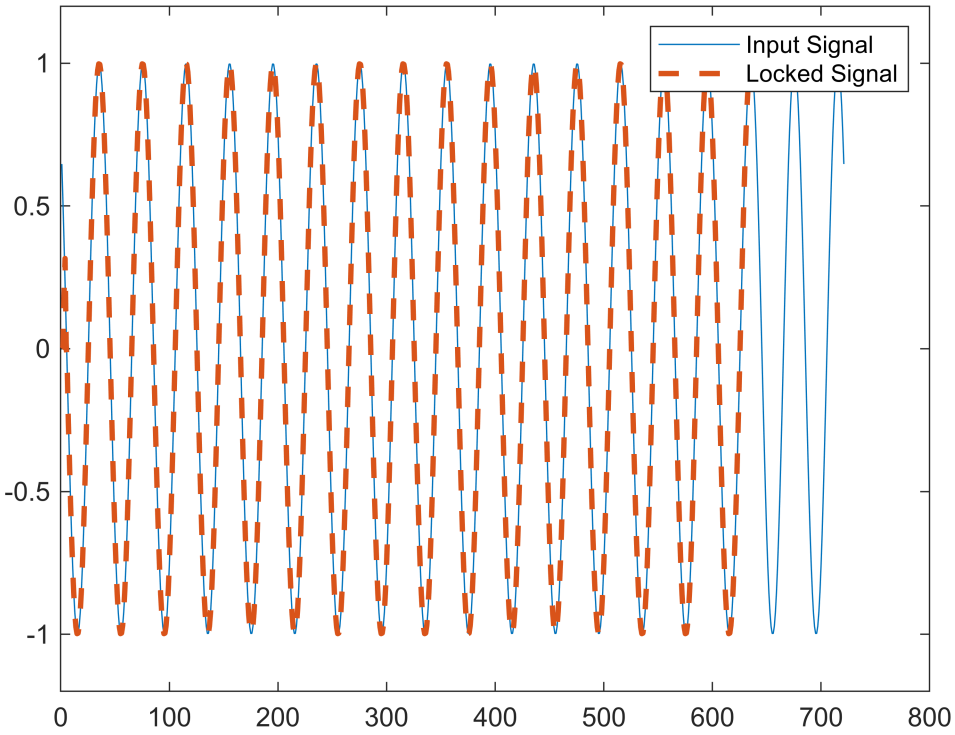

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

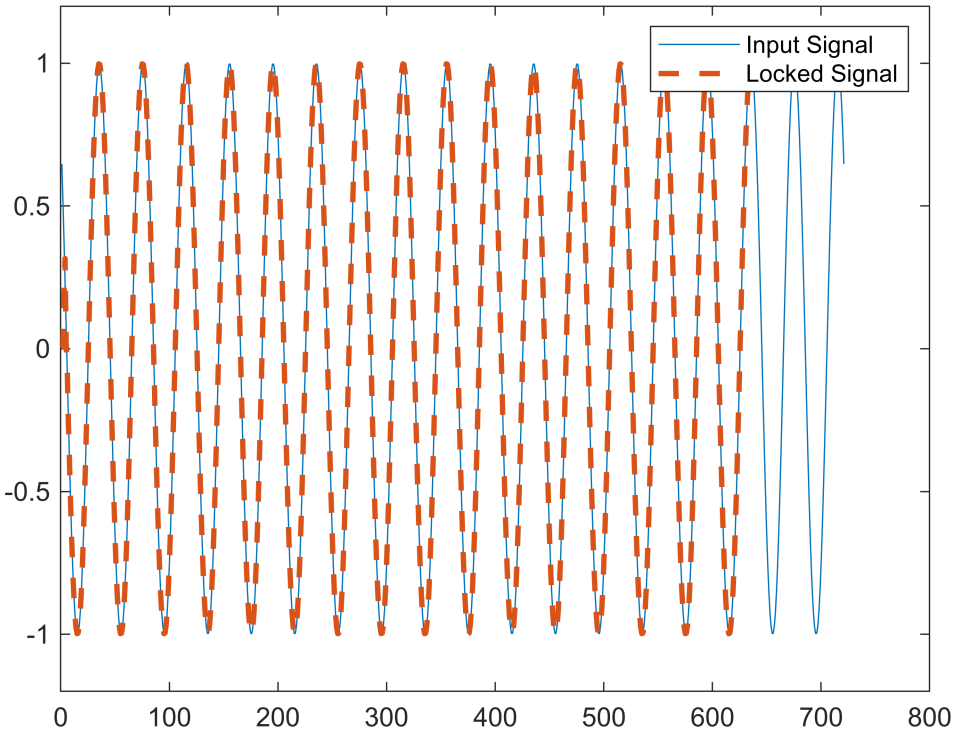

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

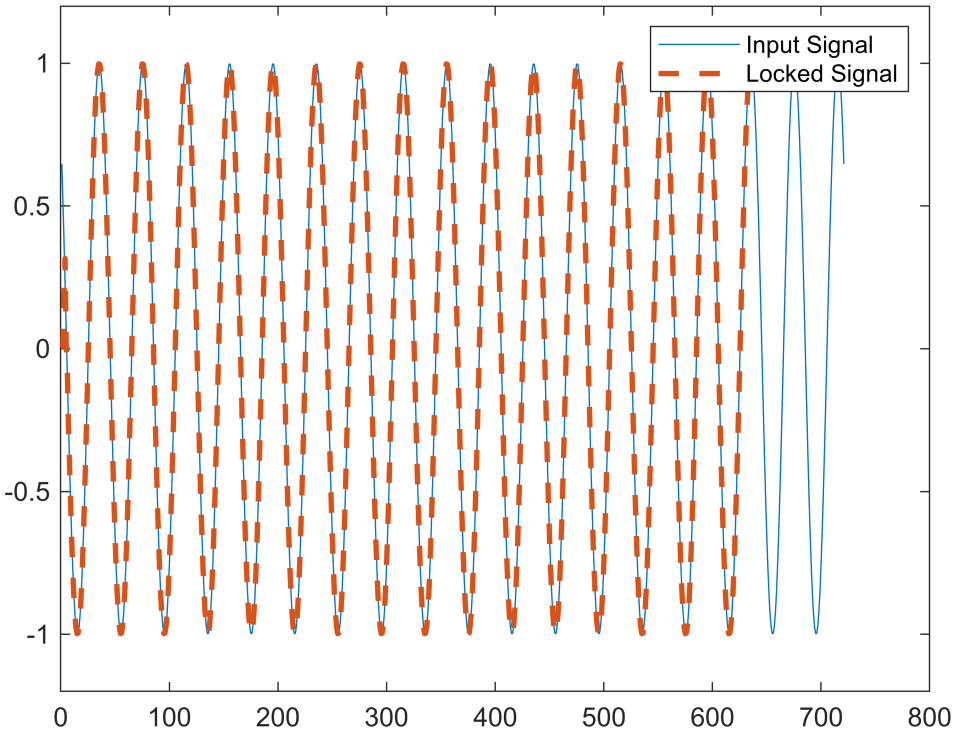

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

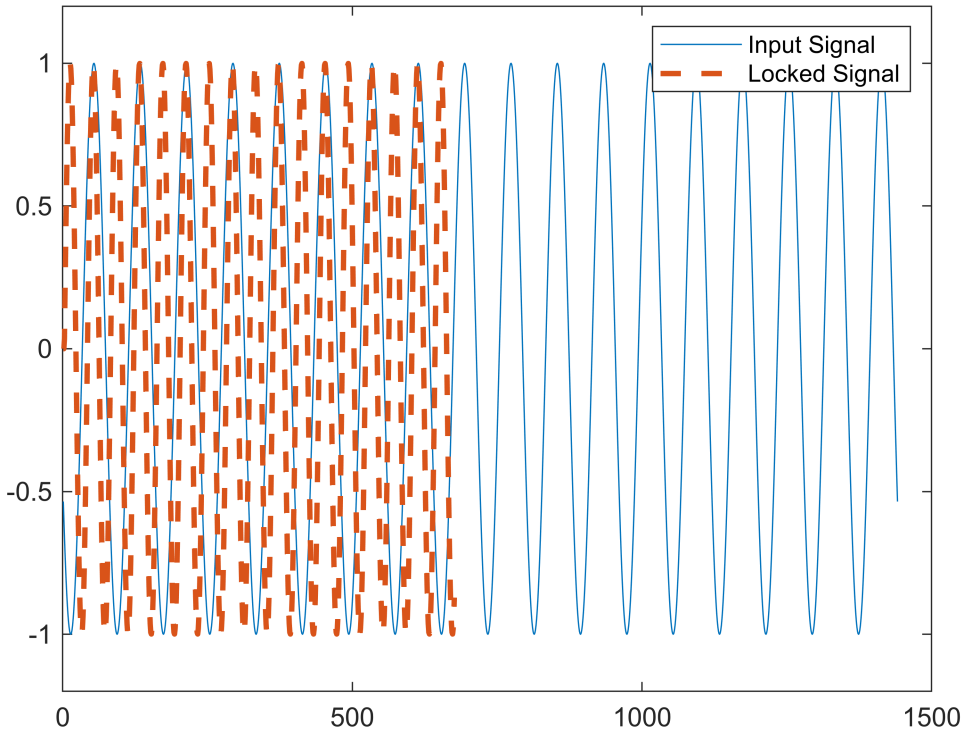

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800

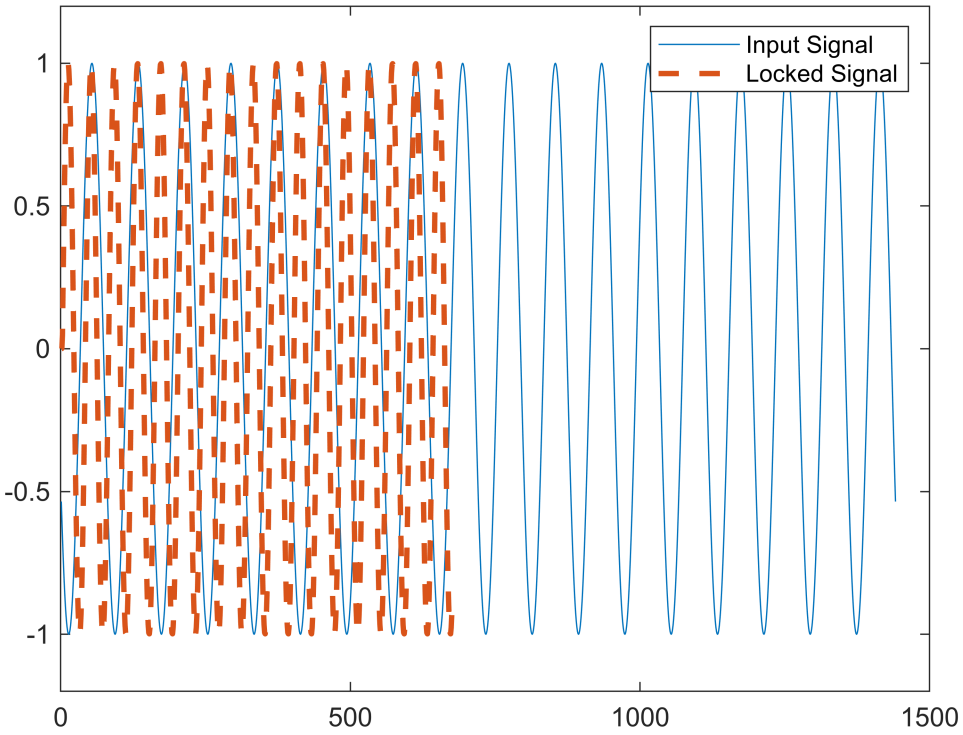


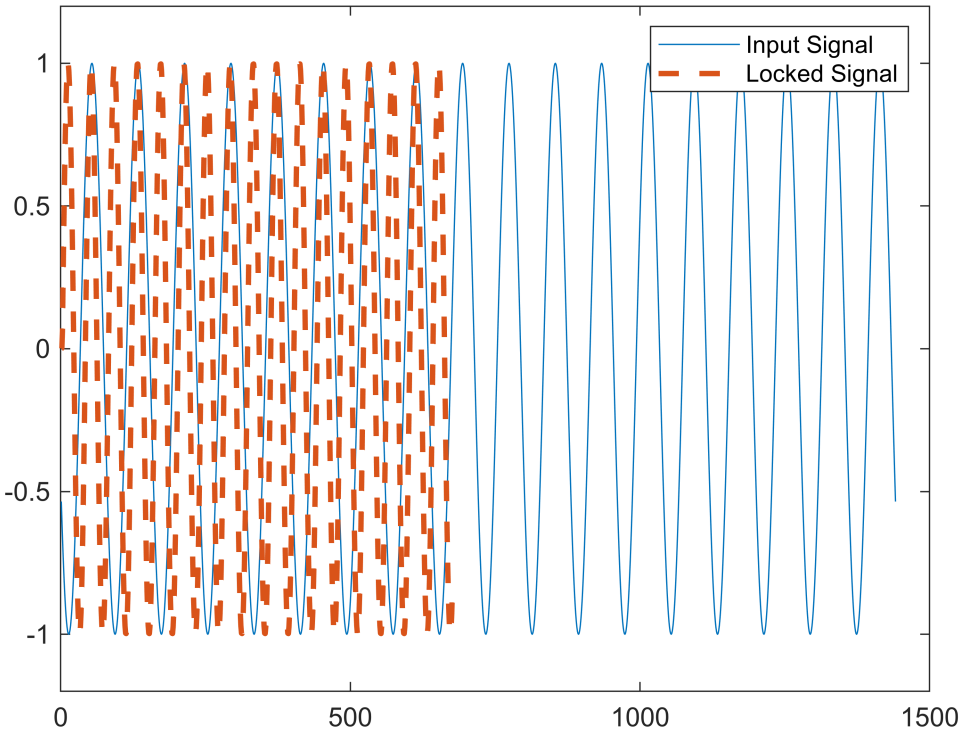
d.2)

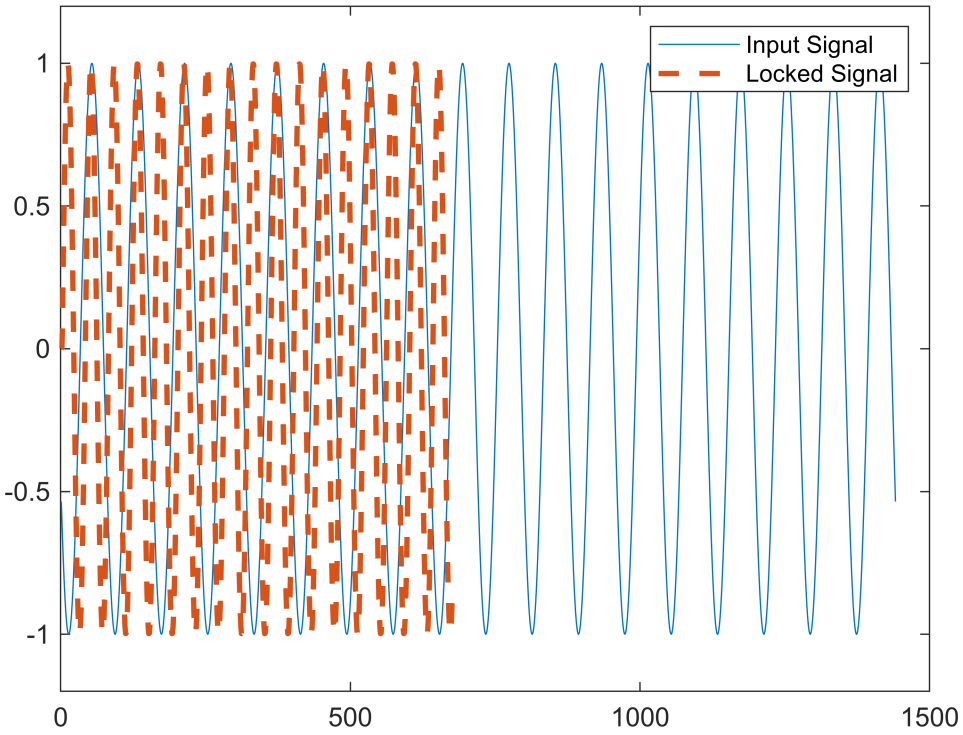
```
fs = 4000*(M+L);                    % fs = 36 kHz
oscil_freq = 400;                   % given initially
lock_freq = 100*(L+M);              % lock_freq = 450
lock_phase =  pi * rand;            % random phase
loop_gain = 0.4 ;
suggested_rate_change = oscil_freq / lock_freq ;
lock_freq = 50*(L+M);
x = 0 : 1/fs: 18/lock_freq;                 % 18 waveforms will be seen
figure;
input_signal = cos(2 * pi * x * lock_freq + lock_phase);
plot(input_signal, LineStyle="-")
hold on;
[y, phase] = DPLL(input_signal, 0, loop_gain, oscil_freq, fs,suggested_rate_change);
plot(y, LineStyle="--", LineWidth=2);
legend('Input Signal', 'Locked Signal');
ylim([-1.2,1.2]);
```

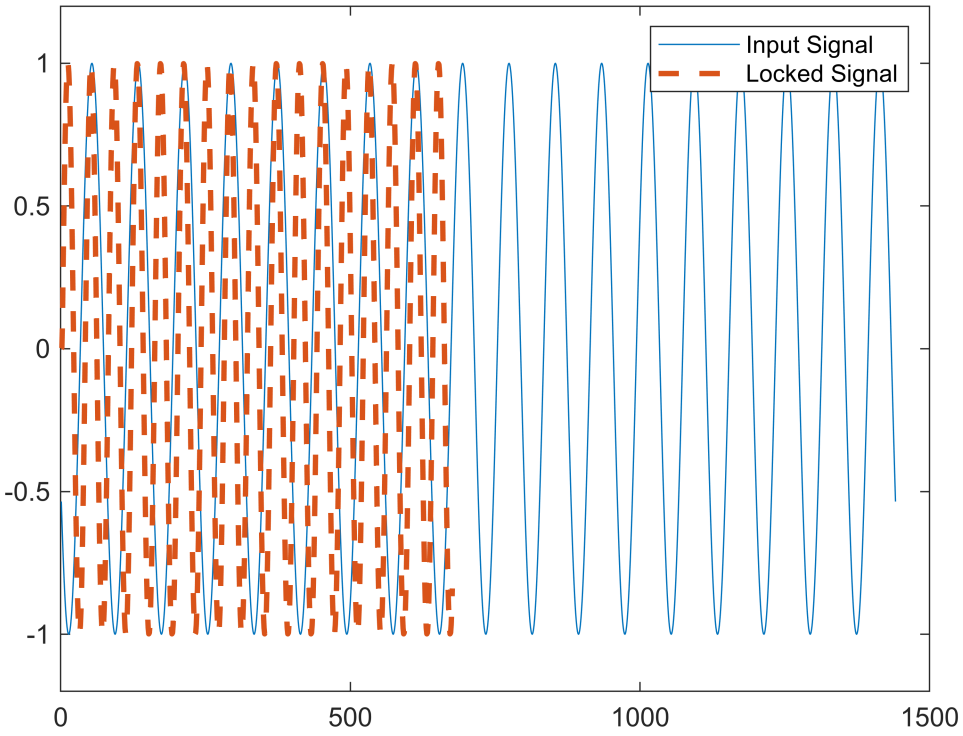


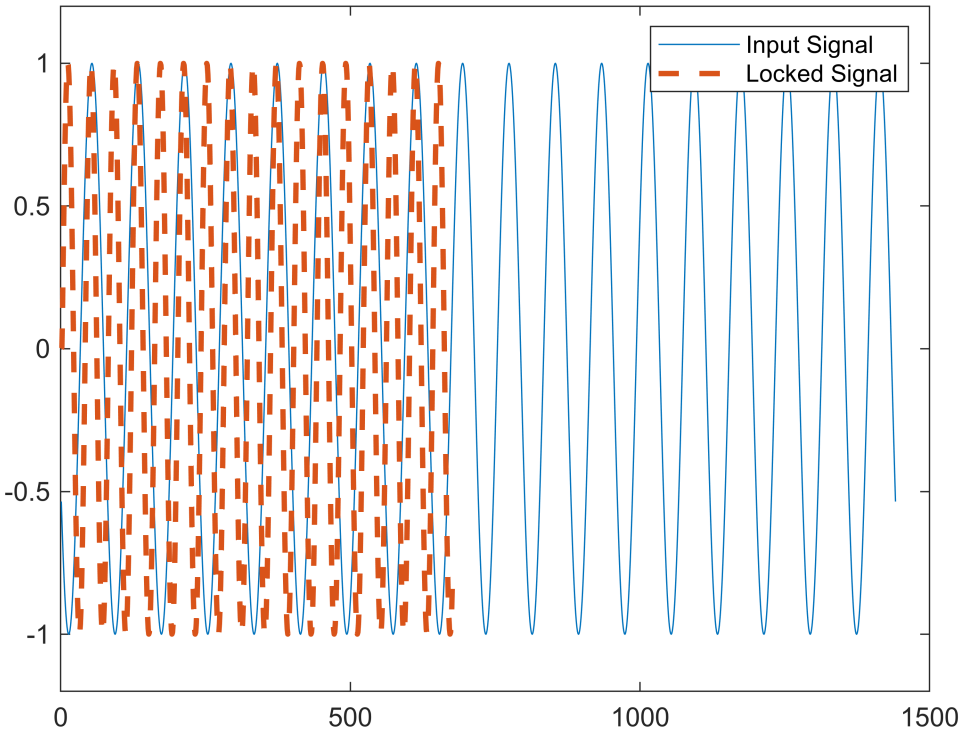

```
pause(1)
for k = 1:15
    figure;
    ylim([-1.1,1.1])
    plot(input_signal, LineStyle="-")
    hold on;
    [y, phase] = DPLL(input_signal, phase, loop_gain, oscil_freq, fs,suggested_rate_change);
    plot(y, LineStyle="--", LineWidth=2);
    legend('Input Signal', 'Locked Signal');
    ylim([-1.2,1.2]);
    %pause(1)

end
```

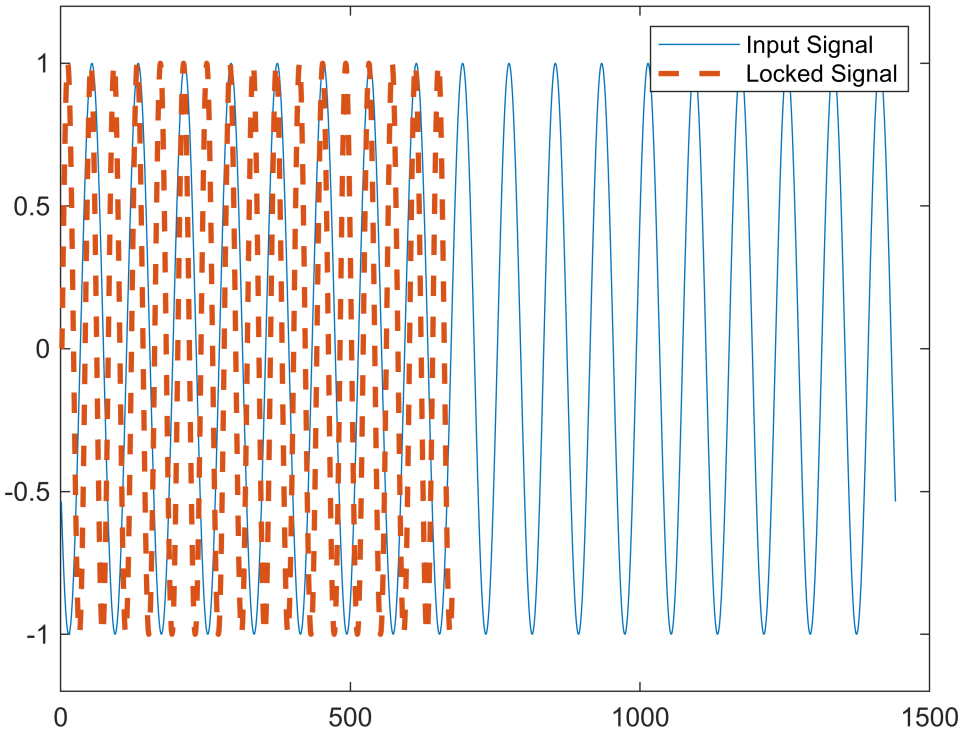

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

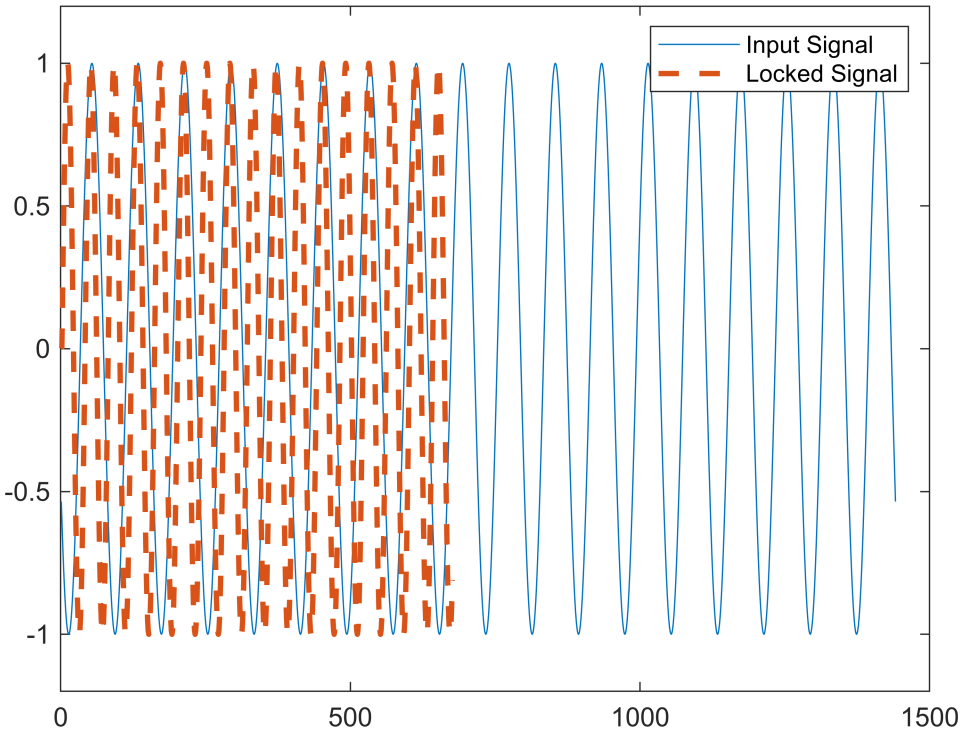

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

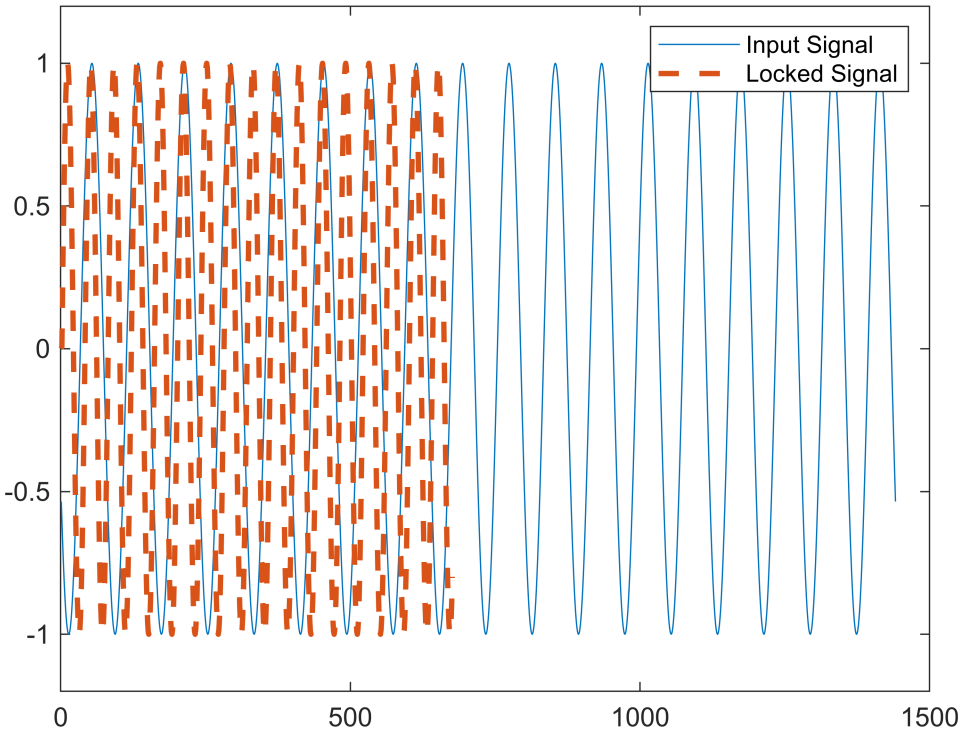

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

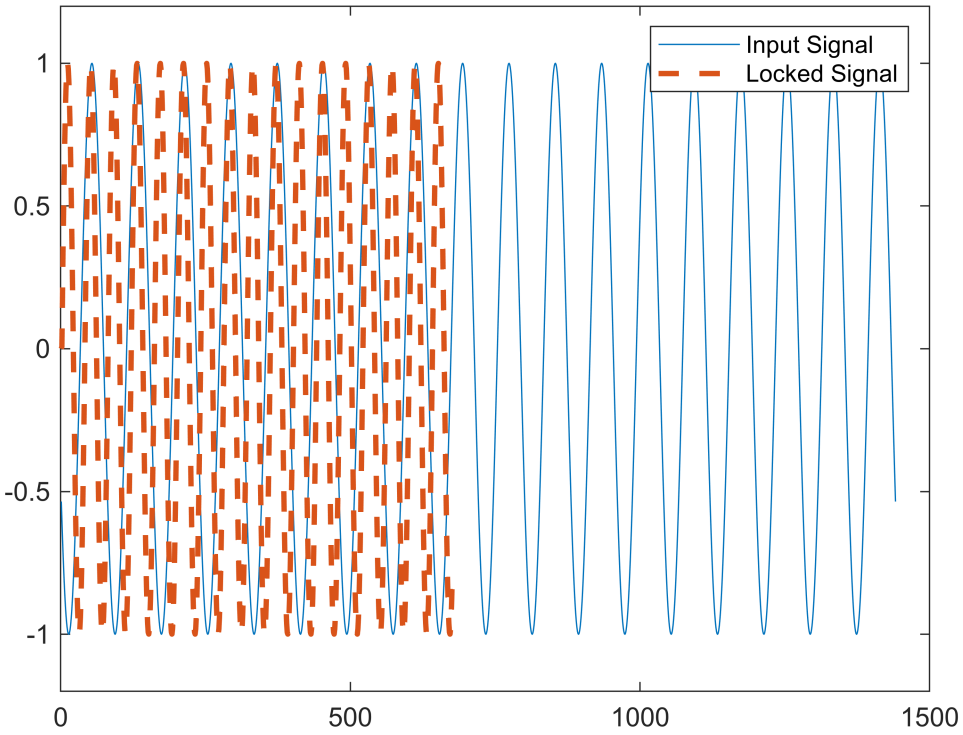

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

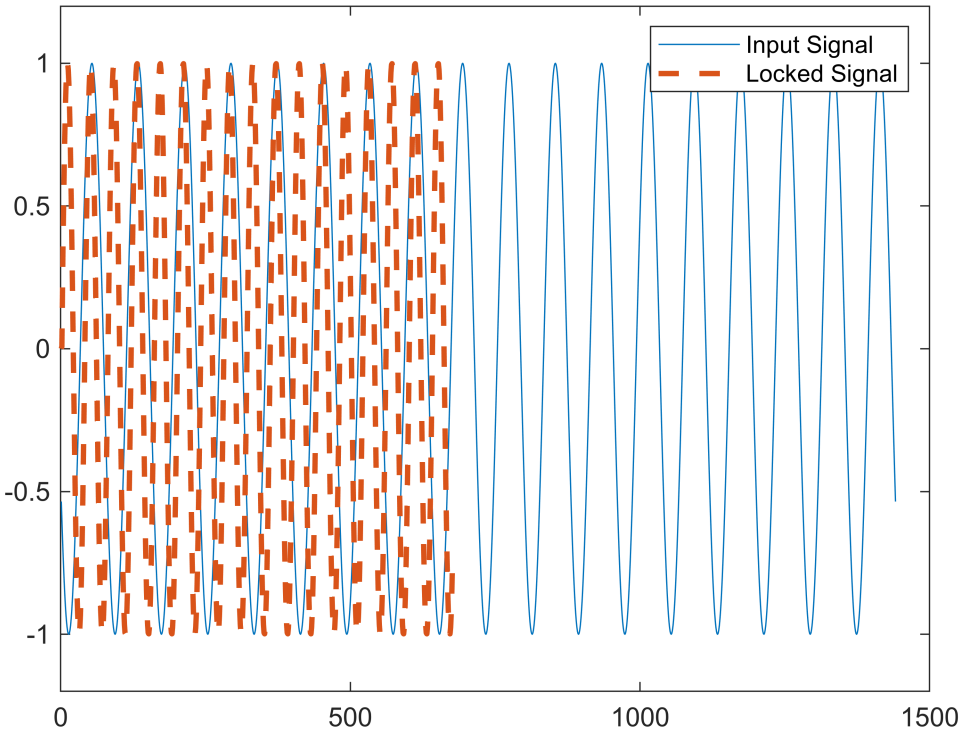

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

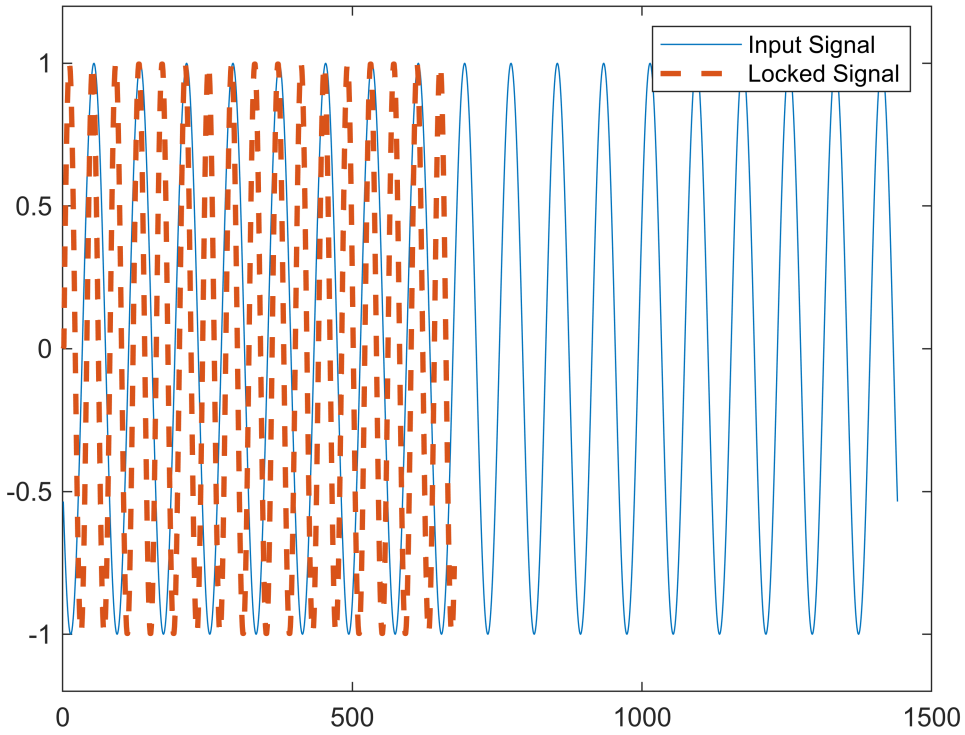

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

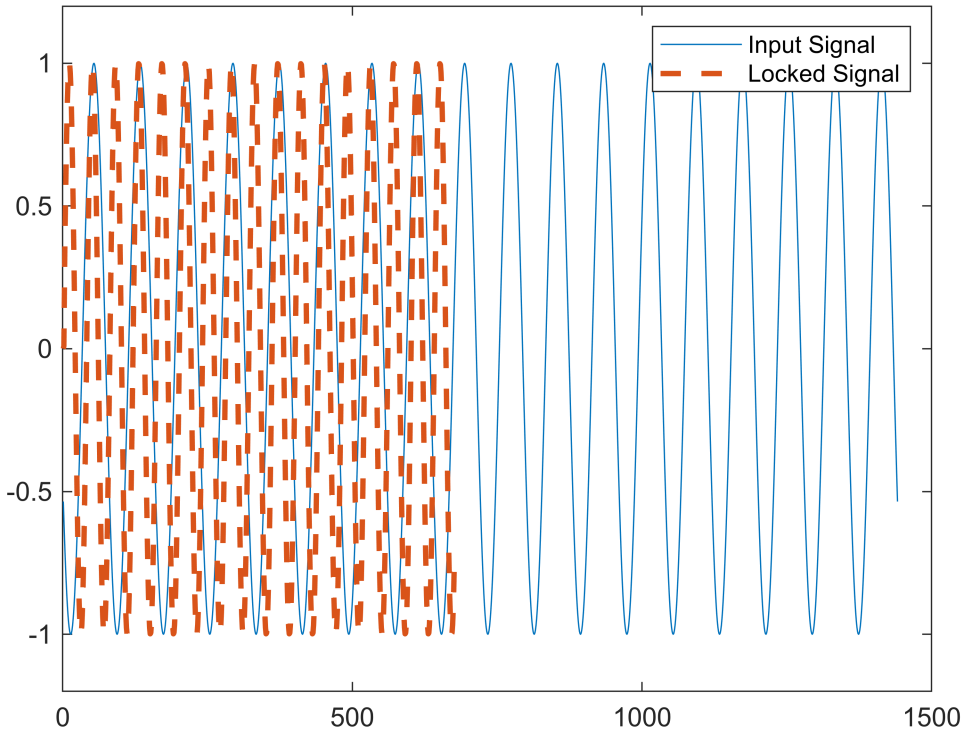

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

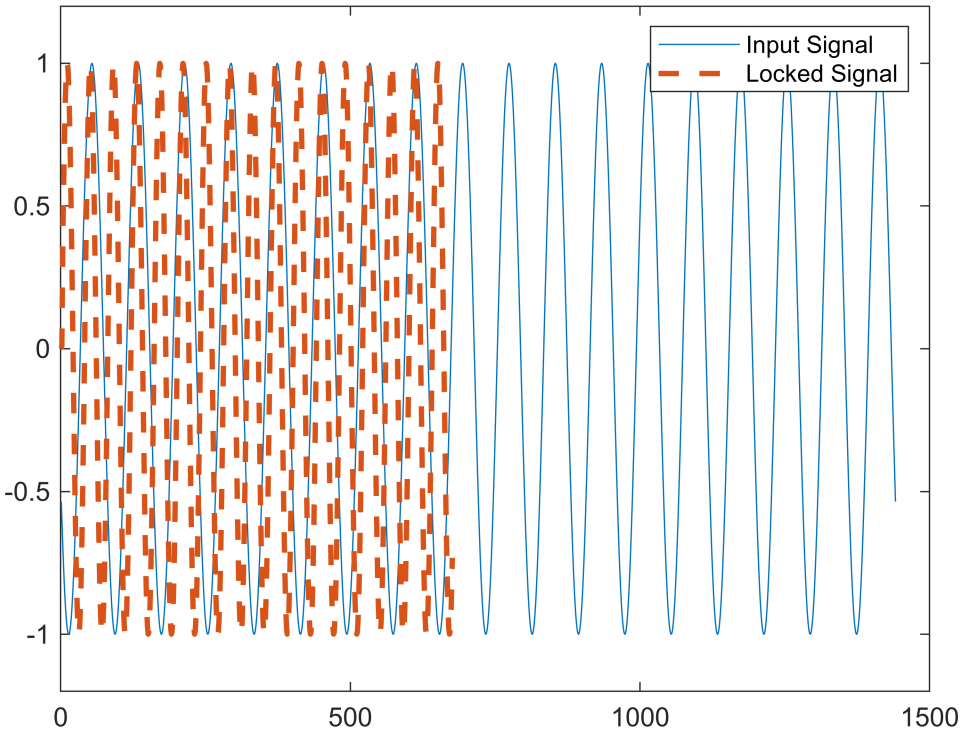

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

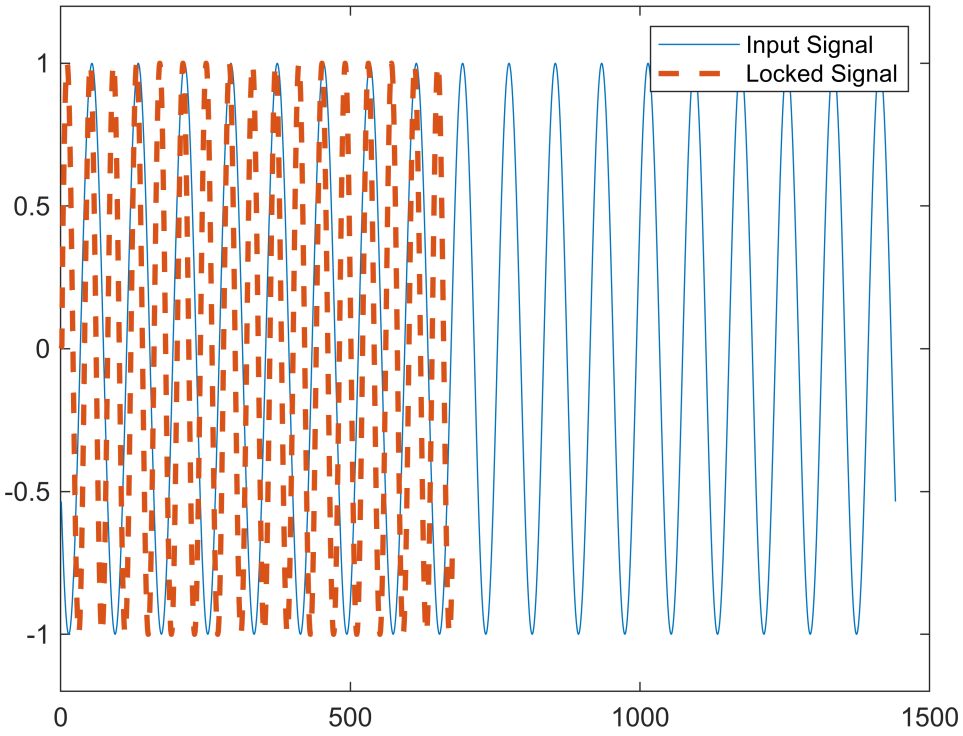

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500

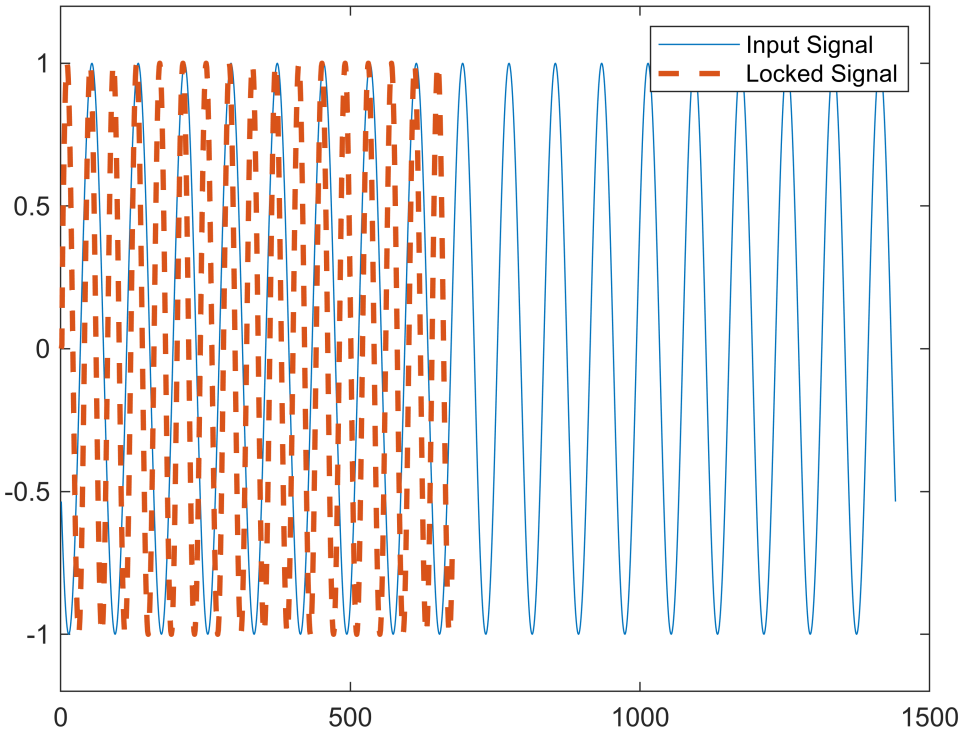

Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500


Input Signal
Locked Signal
1
0.5
0
-0.5
-1
0
500
1000
1500



REAL TIME APPLCIATION :

```
% clear;
% close all;
% oscil_freq = 0.2;
% lock_freq = 0.4;
% lock_phase =  0*pi;
% loop_gain = 0.94;
% fs = 250;
% counter = 0;
% int1 = 10;
% int2 = 10;
% figure;
% t = 0;
% % phase = 0.1*pi;                                  % given initially
%             x = t : 1/fs: t+15e-3;
%             input_signal = sin(2 * pi * x *lock_freq + lock_phase);
%             [y, phase] = DPLL(input_signal, phase, loop_gain, oscil_freq, fs);
%             if lock_freq > oscil_freq
%                 y = cat(2, y, zeros(1, (length(input_signal)- length(y))));
%             end
%             if lock_freq < oscil_freq
%                 input_signal = cat(2, input_signal, zeros(1, (length(y) - length(input_signa
%                 x = t : 1/fs: t+length(input_signal)/fs-1/fs;
%             end
%             plot(x, input_signal, LineStyle="-")
```

```
%             hold on;
%             ylim([-1 1]);
%             plot(x, y, LineStyle="--", LineWidth=2);
%             title("Real-time PLL");
%             legend("input_signal", "locked_signal");
%             hold on;
%             pause(0.04);
%             t = t + 0.04;
% while true
%         for i=0:int1/0.04
%             oscil_freq = 0.5;
%             lock_freq = 0.25;
%             lock_phase =  0*pi;
%             loop_gain = 0.5;
%             clf;
%             x = t : 1/fs: t+15e-3;
%             input_signal = sin(2 * pi * x *lock_freq + lock_phase);
%             [y, phase] = DPLL(input_signal, phase, loop_gain, oscil_freq, fs);
%             if lock_freq > oscil_freq
%                 y = cat(2, y, zeros(1, (length(input_signal)- length(y))));
%             end
%             if lock_freq < oscil_freq
%                 input_signal = cat(2, input_signal, zeros(1, (length(y) - length(input_signal
%                 x = t : 1/fs: t+length(input_signal)/fs-1/fs;
%             end
%             plot(x, input_signal, LineStyle="-")
%             hold on;
%             ylim([-1 1]);
%             plot(x, y, LineStyle="--", LineWidth=2);
%             title("Real-time PLL");
%             legend("input_signal", "locked_signal");
%             hold on;
%             pause(0.04);
%             t = t + 0.04;
%         end
%         for i=0:int2/0.04
%             oscil_freq = 0.28;
%             lock_freq = 0.6;
%             lock_phase =  0.5*pi;
%             loop_gain = 0.5;
%             clf;
%             x = t : 1/fs: t+15e-3;
%             input_signal = sin(2 * pi * x *lock_freq + lock_phase);
%             [y, phase] = DPLL(input_signal, phase, loop_gain, oscil_freq, fs);
%             if lock_freq > oscil_freq
%                 y = cat(2, y, zeros(1, (length(input_signal)- length(y))));
%             end
%             if lock_freq < oscil_freq
%                 input_signal = cat(2, input_signal, zeros(1, (length(y) - length(input_signal
%                 x = t : 1/fs: t+length(input_signal)/fs-1/fs;
```

```
%             end
%             plot(x, input_signal, LineStyle="-")
%             hold on;
%             ylim([-1 1]);
%             plot(x, y, LineStyle="--", LineWidth=2);
%             title("Real-time PLL");
%             legend("input_signal", "locked_signal");
%             hold on;
%             pause(0.04);
%             t = t + 0.04;
%         end
%         break
% end
%
```

```
function [decimated] = decimate(inp, factor)
    if factor ~= 1
        [b,a] = butter(6, 1/factor, 'low');
        inp = filter(b, a, inp);
    end
decimated = zeros(1, ceil(length(inp)/factor));
    for i= 1: length(decimated)
        decimated(i) = inp(i*factor - factor +1);
    end
end

function [interpolated] = interpol(inp, factor)
if factor == 1
    interpolated = inp;
else

    interpolated = zeros(1, factor*length(inp));
        for i = 1: length(interpolated)
            if rem(i, factor) == 1
                interpolated(i) = inp(ceil(i/factor));
            end
        end

        [b,a] = butter(6, 1/factor, 'low');
        interpolated = factor * filter(b, a, interpolated);

end
end
```

```
function [rate_changed] = rate_change(inp, factor)
[up_factor, down_factor] = rat(factor);
upsampled = interpol(inp, up_factor);
rate_changed = decimate(upsampled, down_factor);
end

function [locked_signal, return_phase] = DPLL(inp, phase, loop_gain, oscil_freq, fs, nfactor)
inp = inp / max(inp);

freq = fft(inp, 100000);
[val, idx] = max(freq);
peak_freq = ((idx-1) * fs / 100000);
if peak_freq > fs/2
    peak_freq = fs - ((idx-1) * fs / 100000);
end

t = 0 : 1/fs : 19/peak_freq;

lag = phase * 2* pi + pi / 2;
oscil = sin(2*pi*oscil_freq*t + lag);

feedback_signal = rate_change(oscil, nfactor);

visualization_signal = sin(2*pi*oscil_freq*t + lag - pi/2);
locked_signal = rate_change(visualization_signal, nfactor);

feedback_signal = loop_gain * feedback_signal;
if length(feedback_signal)< length(inp)
    mixer = inp(1:length(feedback_signal)) .* feedback_signal;
else
    mixer = inp .* feedback_signal(1:length(inp));
end
% locked_signal = feedback_signal / loop_gain;
return_phase = phase + mean(mixer);
end
```