

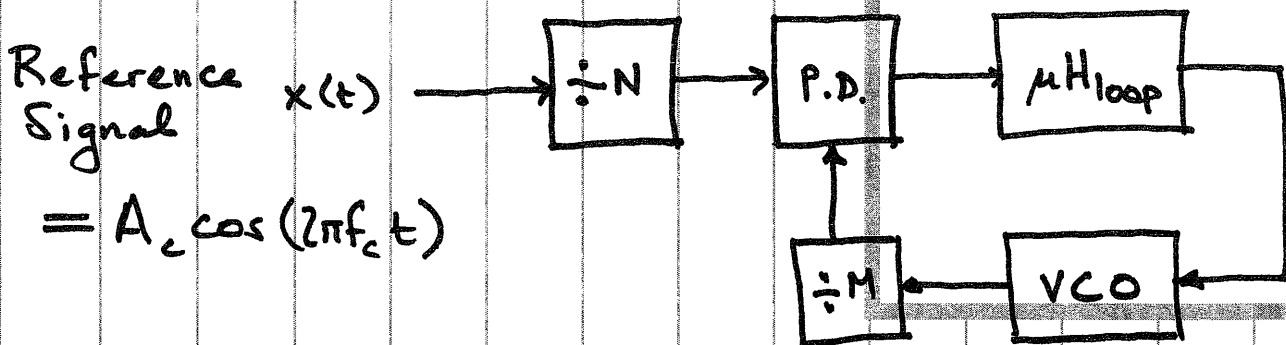
Problem 5. [15 pts. total] *Further Experimentation with Passband Simulation Code.*

Using the code you developed in Problem 4 consider some extensions suggested below

...

- (a) How might you use the passband code to make a frequency synthesizer along the lines of Z&T Problem 4.25? What would the Lissajous figures look like in this case?
- (b) Use the passband code to make an FM demodulator and demonstrate its operation. Be sure to add noise and connect it to a continuous-time SNR.
- (c) Wow me with some other cool demonstration?

Frequency Synthesizer



$$\text{VCO output} = A_v \sin(2\pi f_o t + \hat{\theta}(t))$$

Lock occurs if $\frac{f_c}{N} = \frac{f_o}{M}$ and ideally $\hat{\theta} = 0$.

\Rightarrow VCO output frequency is $f_o = \frac{M}{N} f_c$

There are many interesting considerations in the design of programmable synthesizers — far more than can be explored here — we therefore just consider one item: Tuning Speed.

Recall that when locking to a frequency step, it could be very slow. From the example of problem 4 a freq. step is imposed at about 0.75s with loop params

$$f_n = 10 \text{ Hz}$$

$$\zeta = 0.707$$

$$f_c = 1000 \text{ Hz}$$

Δf	$\Delta f/f_c$	Time to Lock
20	0.02	$1.2 - 0.75 \approx .45s$
30	0.03	$1.4 - 0.75 \approx .65s$
40	0.04	$1.75 - 0.75 \approx 1s$
50	0.05	$2.4 - 0.75 \approx 1.65s$
60	0.06	$> 2.25s$

did not lock by end of the simulation run.

Lets widen the loop bandwidth by setting $f_n = 20Hz$ and run through it again ...

Δf	$\Delta f/f_c$	Time to Lock
20	0.02	
30	0.03	
40	0.04	
50	0.05	$1.05 - 0.75 \approx 0.30s$
60	0.06	$1.2 - 0.75 \approx 0.45s$

Try even wider $f_n = 40Hz$

Δf	$\Delta f/f_c$	Time to lock
50	0.05	$\approx 0.2sec.$
100	0.10	\approx Never locked

Lets try $f_n = 20Hz$. The calculation/experiment above would indicate that we want to keep the freq. step less than or equal to $5\% = 0.05 = 1/20$ and allow about 0.5sec for the loop to settle between switching.

$\Rightarrow N = 20$

The following code will lock in 20 sec for values of $M = 15, 16, 17, 18, 19, 20$

With M set to 20, a number of N values can be also found.

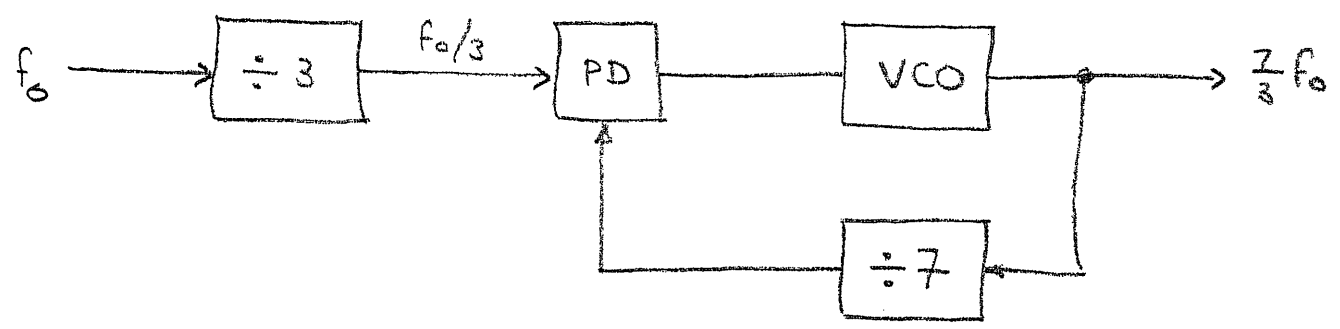
Z+T Sixth Edition Problem 3.44

3.44. Using a single PLL, design a system that has an output frequency equal to $\frac{7}{3}f_0$, where f_0 is the input frequency. Describe fully, by sketching, the output of the VCO for your design. Draw the spectrum at the VCO output and at any other point in the system necessary to explain the operation of your design. Describe any filters used in your design by defining the center frequency and the appropriate bandwidth of each.

Desired input: $x(t) = A \cos 2\pi f_0 t$

Desired output: $e_o(t) = B \cos 2\pi (\frac{7f_0}{3}) t$

The simplest architecture which will achieve the phase-locked synthesis of $\frac{7}{3}f_0$ from f_0 is:



Here we show the circuit without a loop filter (or just assume the LF is part of of the phase detector).

```
%% Passband Simulation of a PLL. Problem 5A. Freq. Synthesizer.
%
% File = THE_P5_FreqSynth.m
%
% Based on Z&T Problem 3.44 of 6th Edition.

%clear all;                %Be safe
%close all;

%% Beginning of preprocessor
%

% General parameters

f0 = 1000;                  %Nominal VCO frequency
fr = 1000;                 %Nominal carrier/ref frequency
N = 20;                    %Divider in reference path
M = 18;                    %Divider of VCO output
%alpha = 0.95;
%fc = 1000;                %Carrier frequency
Ns = 80;                  %Oversampling factor
fs = Ns*fr;               %Simulation sampling rate
Tf = 12;                  %Simulation length in sec
npts = floor(Tf*fs);      %Number of simulation points
t = (0:(npts-1))/fs;      %Time vector

s = cos(2*pi*fr*t);
sdN = cos(2*pi*(fr/N)*t);
soutIdeal = sin(2*pi*(M*fr/N)*t);

figure(1)
TitleSTR1 = 'Ideal Freq. Synth. Relationship, (N, M) = (';

plot(s,soutIdeal)
axis square;
title([TitleSTR1, num2str(N),',', num2str(M),')'])
xlabel('Reference Sinusoid')
ylabel('Ideal VCO Output Sinusoid')

%%

% Parameters of PLL

fn = 20;                   %Loop natural frequency in Hz
zeta = 0.707;              %Loop damping factor

Kt = 4*pi*zeta*fn;        %Loop gain
a = pi*fn/zeta;           %Loop filter parameter

% Phase detector parameters
nc = 5;                   %Number of carrier cycles over which to avg
navg = floor(nc*fs/f0);

filt_in_last = 0;
filt_out_last = 0;
vco_in_last = 0;
vco_out = 0;
```

```
vco_out_last = 0;

%fvco = zeros(1,npts);

% End of preprocessor

%% Beginning of Simulation Loop
%

T = 1/fs;
svco = zeros(1,npts);
svcodM = zeros(1,npts);

for i = 1:npts

    svco(i) = sin(2*pi*f0*t(i) + vco_out);
    svcodM(i) = sin(2*pi*(f0/M)*t(i) + vco_out/M);

    % This code implements a simple phase detector
    if i <= navg
        temp = sdN(1:i) .* svcodM(1:i);
        s2 = mean(temp);
    else
        temp = sdN(i - navg:i) .* svcodM(i - navg:i);
        s2 = mean(temp);
    end

    s3 = Kt*s2;

    filt_in = a*s3;
    filt_out = filt_out_last + (T/2)*(filt_in + filt_in_last);
    filt_in_last = filt_in;
    filt_out_last = filt_out;
    vco_in = s3 + filt_out;
    vco_out = vco_out_last + (T/2)*(vco_in + vco_in_last);
    vco_in_last = vco_in;
    vco_out_last = vco_out;

end

% End of Simulation Loop

%% Beginning of Postprocessor
%

nn = 10000;
kk = floor(npts/nn);

%Make a movie
pause on

for i = 0:(kk-1)

    figure(2)
    subplot(3,1,1)
    plot(sdN(1+i*nn:(i+1)*nn),svcodM(1+i*nn:(i+1)*nn))
```

```
axis square;
title('Phase Detector Inputs')
xlabel(['Input Divided by N = ',num2str(N)])
ylabel(['VCO Divided by M = ',num2str(M)])

subplot(3,1,2)
plot(s(1+i*nn:(i+1)*nn),svco(1+i*nn:(i+1)*nn))
axis square;
title(['(N, M) = (',num2str(N), ' ',num2str(M), ')'])
xlabel('Reference')
ylabel('Actual VCO Output')

pause(0.02)
end

subplot(3,1,3)
TitleSTR1 = 'Ideal for (N, M) = (';

plot(s,soutIdeal)
axis square;
title([TitleSTR1, num2str(N), ', ' num2str(M), ')'])
xlabel('Reference')
ylabel('Ideal VCO Output')

pause off

% End of Postprocessor
```

