

Problem 14: Z&T Sixth Ed., Problem 3.47 (Seventh Ed., Problem 4.28).

3.47. The imperfect second-order PLL is defined as a PLL with the loop filter

$$F(s) = \frac{s+a}{s+\lambda a}$$

in which λ is the offset of the pole from the origin relative to the zero location. In practical implementations λ is small but often cannot be neglected. Use the linear model of the PLL and derive the transfer function for $\Theta(s)/\Phi(s)$. Derive expressions for ω_n and ζ in terms of K_t , a , and λ .

Problem 15: Z&T Sixth Ed., Problem 3.48 (Seventh Ed., Problem 4.29).

3.48. Assuming the loop filter model for an imperfect second-order PLL described in the preceding problem, derive the steady-state phase errors under the three conditions of θ_0 , f_Δ , and R given in Table 3.5.

Table 3.5 Steady-State Errors

PLL order	$\theta_0 \neq 0$ $f_\Delta = 0$ $R = 0$	$\theta_0 \neq 0$ $f_\Delta \neq 0$ $R = 0$	$\theta_0 \neq 0$ $f_\Delta \neq 0$ $R \neq 0$
1 ($a = 0, b = 0$)	0	$2\pi f_\Delta / K_t$	∞
2 ($a \neq 0, b = 0$)	0	0	$2\pi R / K_t$
3 ($a \neq 0, b \neq 0$)	0	0	0

Problem 16: Z&T Sixth Ed., Computer Exercise 3.8 (Seventh Ed., Computer Exercise 4.4).

3.8. Referring to Computer Example 3.4, draw the block diagram of the system represented by the simulation loop, and label the inputs and outputs of the various loop components with the names used in the simulation code. Using this block diagram, verify that the simulation program is correct. What are the sources of error in the simulation program?

COMPUTER EXAMPLE 3.4

A simulation program is easily developed for the PLL. Two integration routines are required; one for the loop filter and one for the VCO. The trapezoidal approximation is used for these integration routines. The trapezoidal approximation is

$$y[n] = y[n-1] + (T/2)[x[n] + x[n-1]]$$

where $y[n]$ represents the current output of the integrator, $y[n-1]$ represents the previous integrator output, $x[n]$ represents the current integrator input, $x[n-1]$ represents the previous integrator input, and T represents the simulation step size, which is the reciprocal of the sampling frequency. The values of $y[n-1]$ and $x[n-1]$ must be initialized prior to entering the simulation loop. Initializing the integrator inputs and outputs usually result in a transient response. The parameter `settle`, which in the simulation program to follow is set equal to 10% of the simulation run length, allows any initial transients to decay to negligible values prior to applying the loop input.

The following simulation program is divided into three parts. The preprocessor defines the system parameters, the system input, and the parameters necessary for execution of the simulation, such as the sampling frequency. The simulation loop actually performs the simulation. Finally, the postprocessor allows for the data generated by the simulation to be displayed in a manner convenient for interpretation by the simulation user. Note that the postprocessor used here is interactive in that a menu is displayed and the simulation user can execute postprocessor commands without typing them.

The simulation program given here assumes a frequency step on the loop input and can therefore be used to generate Figures 3.51 and 3.52.

```
% File: c3ce4.m
% beginning of preprocessor
clear all % be safe
fdel = input('Enter frequency step size in Hz > ');
fn = input('Enter the loop natural frequency in Hz > ');
zeta = input('Enter zeta (loop damping factor) > ');
npts = 2000; % default number of simulation points
fs = 2000; % default sampling frequency
T = 1/fs;
t = (0:(npts-1))/fs; % time vector
nsettle = fix(npts/10); % set nsettle time as 0.1*npts

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

filt_in_last = 0; filt_out_last = 0;
vco_in_last = 0; vco_out = 0; vco_out_last = 0;
% end of preprocessor

% beginning of simulation loop
for i = 1:npts
    if i < nsettle
        fin(i) = 0;
        phin = 0;
    else
        fin(i) = fdel;
        phin = 2*pi*fdel*T*(i-nsettle);
    end
    s1 = phin - vco_out;
    s2 = sin(s1); % sinusoidal phase detector
    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;
    phierror(i)=s1;
    fvco(i)=vco_in/(2*pi);
    freqerror(i) = fn(i)-fvco(i);
end
%end of simulation loop

%beginning of postprocessor
kk = 0;
while kk == 0
    k = menu('Phase Lock Loop Postprocessor', ...
            'Input Frequency and VCO Frequency', ...
            'Phase Plane Plot', ...
            'Exit Program');
    if k == 1
        plot(t, fn, t, fvco)
        title('Input Frequency and VCO Frequency')
        xlabel('Time - Seconds')
        ylabel('Frequency - Hertz')
        pause
    elseif k == 2
        plot(phierror/2/pi, freqerror)
        title('Phase Plane')
        xlabel('Phase Error / pi')
        ylabel('Frequency Error - Hz')
        pause
    elseif k == 3
        kk = 1;
    end
end
%end of postprocessor

```

Problem 17: Modify the simulation program in Computer Example 3.4 to replace the perfect loop filter with the imperfect loop filter of Z&T Problem 3.47 where $\lambda = 0.1$.