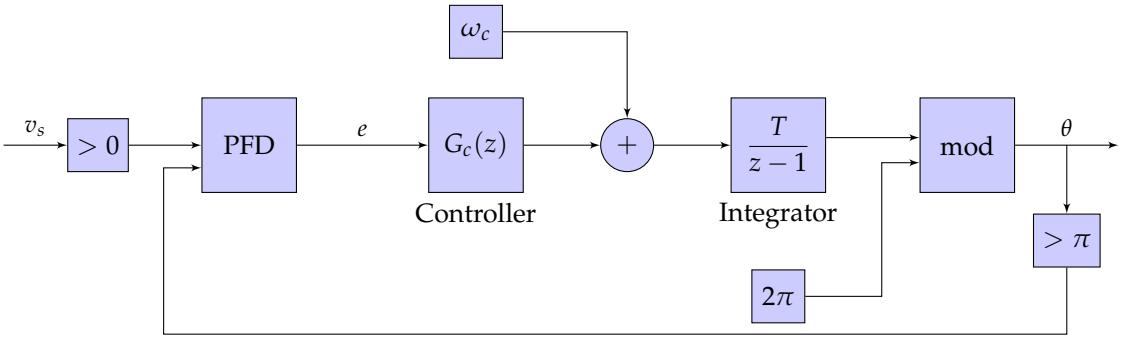
DIGITAL CONTROL OF POWER ELECTRONICS

Single Phase PLL using single precision floating point



We are going to convert the low pass filter design to a digital controller. Below are the gain, pole location, and transfer function for the controller/ low pass filter/ loop filter.

$$k = 105075 \quad \omega_p = 258.6 \tag{1}$$

$$G_{LF}(s) = \frac{105075}{s + 258.6} \tag{2}$$

$$G_{VCO}(s) = \frac{T}{z-1} \tag{3}$$

Below is the matlab code to develop the digital controllers for a low pass controller

```
%% low pass single phase pll
clear
syms s wp k p

gol = 1/(2*p) * k/(s+wp) * 1/s
gcl = gol/(1+gol)

clear
s=tf('s');
opts = bodeoptions('cstprefs');
opts.FreqUnits = 'Hz';
opts.XLim= {[.1 10000]}

kvco = 1

fbw = 10
zeta = 1
w = 2 * pi * fbw
a =1
b = -(2^2*zeta^2*2*pi*w^2)
c = -(2*pi)^2*w^4

k1 = (-b+sqrt(b^2-4*a*c))/2
k2 = (-b-sqrt(b^2-4*a*c))/2
wf = 2*zeta*sqrt(k1/(2*pi))

GLPF = k1/(s+wf)
%digital controller
```

```
Ts = 1/10e3
Z=tf('z',Ts)
sback = ((Z-1)/(Z*Ts))
gc_dig_backward = k1 / (sback +wf)
gc_dig_zoh = c2d(GLPF,1/10e3,'zoh')
gc_dig_tustin = c2d(GLPF,1/10e3,'tustin')

%% calculate coeff for floating point backward euler

Bcoeff = single(gc_dig_backward.Numerator{1});
Acoeff = single(gc_dig_backward.Denominator{1});
fprintf('float A1 = %f;\n',Acoeff(1))
fprintf('float A0 = %f;\n',Acoeff(2))
fprintf('float B1 = %f;\n',Bcoeff(1))
fprintf('float B0 = %f;\n',Bcoeff(2))

%% calculate coeff for floating point tustin

Bcoeff = single(gc_dig_tustin.Numerator{1});
Acoeff = single(gc_dig_tustin.Denominator{1});
fprintf('float A1 = %f;\n',Acoeff(1))
fprintf('float A0 = %f;\n',Acoeff(2))
fprintf('float B1 = %f;\n',Bcoeff(1))
fprintf('float B0 = %f;\n',Bcoeff(2))
```

Below are the coefficients for backwards euler.

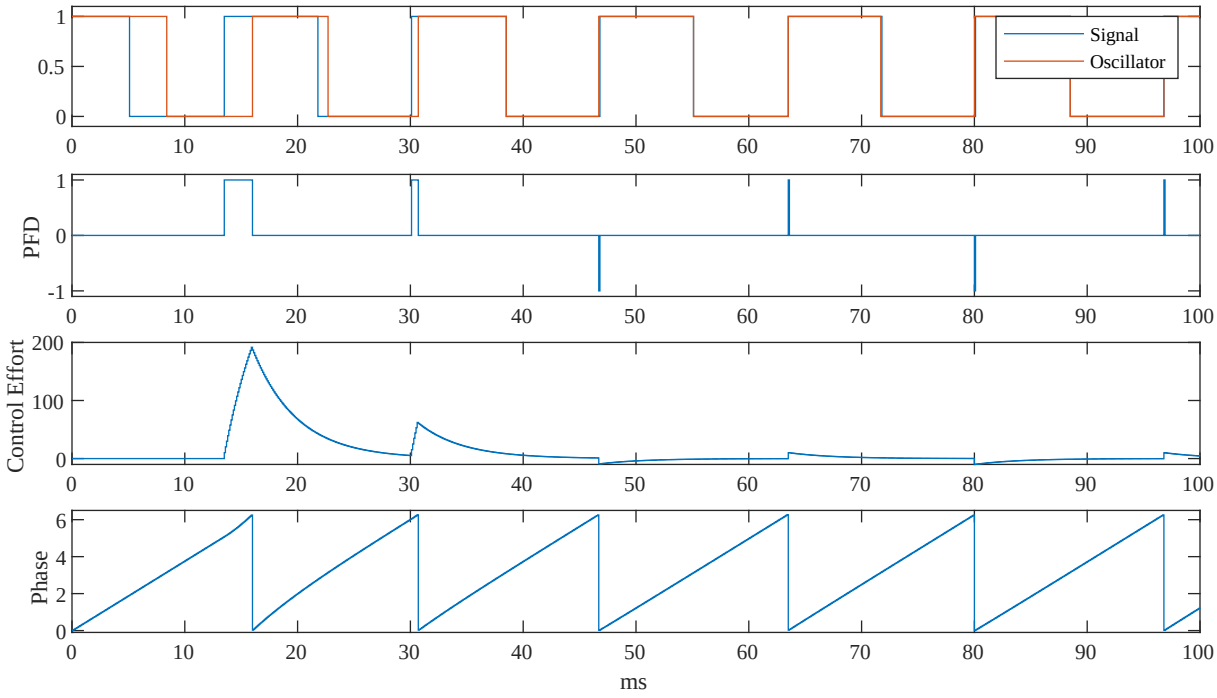
```
float A1 = 1.025864;
float A0 = -1.000000;
float B1 = 10.507576;
float B0 = 0.000000;
```

Below are the coefficients for tustin.

```
float A1 = 1.000000;
float A0 = -0.974466;
float B1 = 5.186714;
float B0 = 5.186714;
```

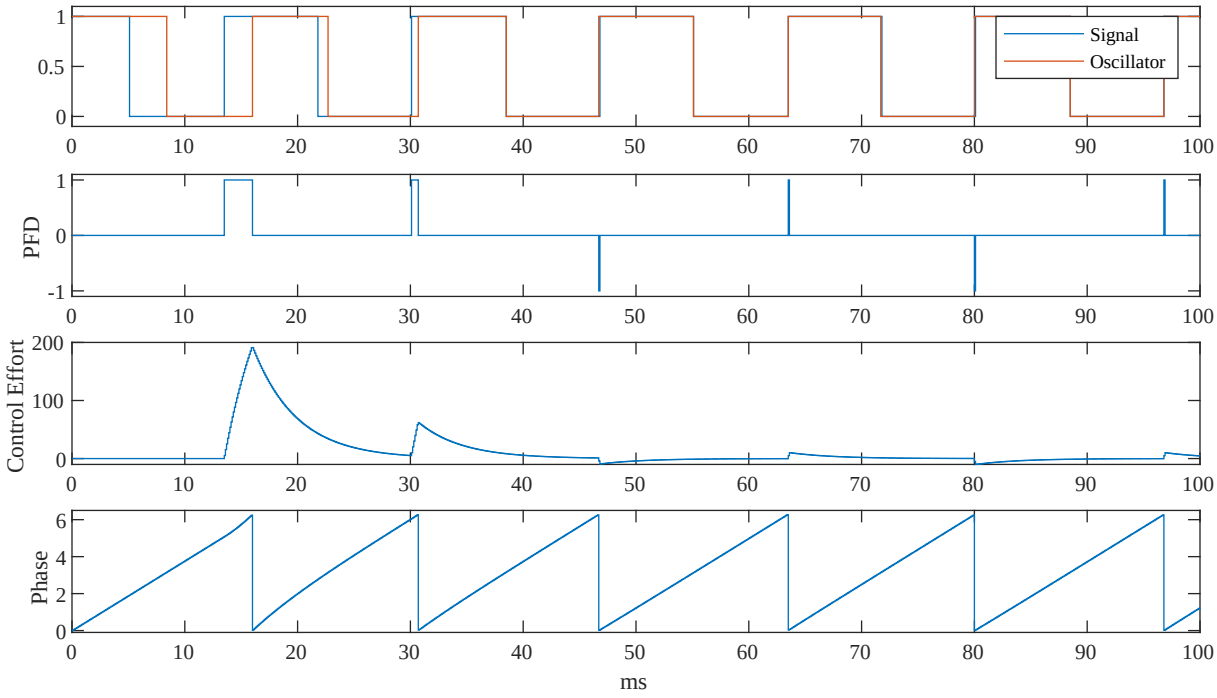
Use backward euler we arrive at a controller with.

$$G_c(z) = \frac{10.51z}{1.026z - 1} \tag{4}$$



The tustin transformation will arrive a the below controller

$$G_c(z) = \frac{5.187z + 5.187}{z - 0.9745} \tag{5}$$



Below is the matlab code to develop the digital controllers for a pi floating point controller

```
%% PI controller
damping = 1

omega_n = 2*pi*4

kp = 4*pi*damping*omega_n
ki = 2*pi*omega_n^2
GLPF = ki/s + kp

%digital controller
Ts = 1/10e3
Z=tf('z',Ts)
sback = ((Z-1)/(Z*Ts))
gc_dig_backward = ki/sback + kp
gc_dig_zoh = c2d(GLPF,1/10e3,'zoh')
gc_dig_tustin = c2d(GLPF,1/10e3,'tustin')

%% calculate coeff for floating point backward euler

Bcoeff = single(gc_dig_backward.Numerator{1});
Acoeff = single(gc_dig_backward.Denominator{1});
fprintf('float A1 = %f;\n',Acoeff(1))
fprintf('float A0 = %f;\n',Acoeff(2))
fprintf('float B1 = %f;\n',Bcoeff(1))
fprintf('float B0 = %f;\n',Bcoeff(2))

%% calculate coeff for floating point tustin

Bcoeff = single(gc_dig_tustin.Numerator{1});
Acoeff = single(gc_dig_tustin.Denominator{1});
fprintf('float A1 = %f;\n',Acoeff(1))
fprintf('float A0 = %f;\n',Acoeff(2))
fprintf('float B1 = %f;\n',Bcoeff(1))
fprintf('float B0 = %f;\n',Bcoeff(2))
```

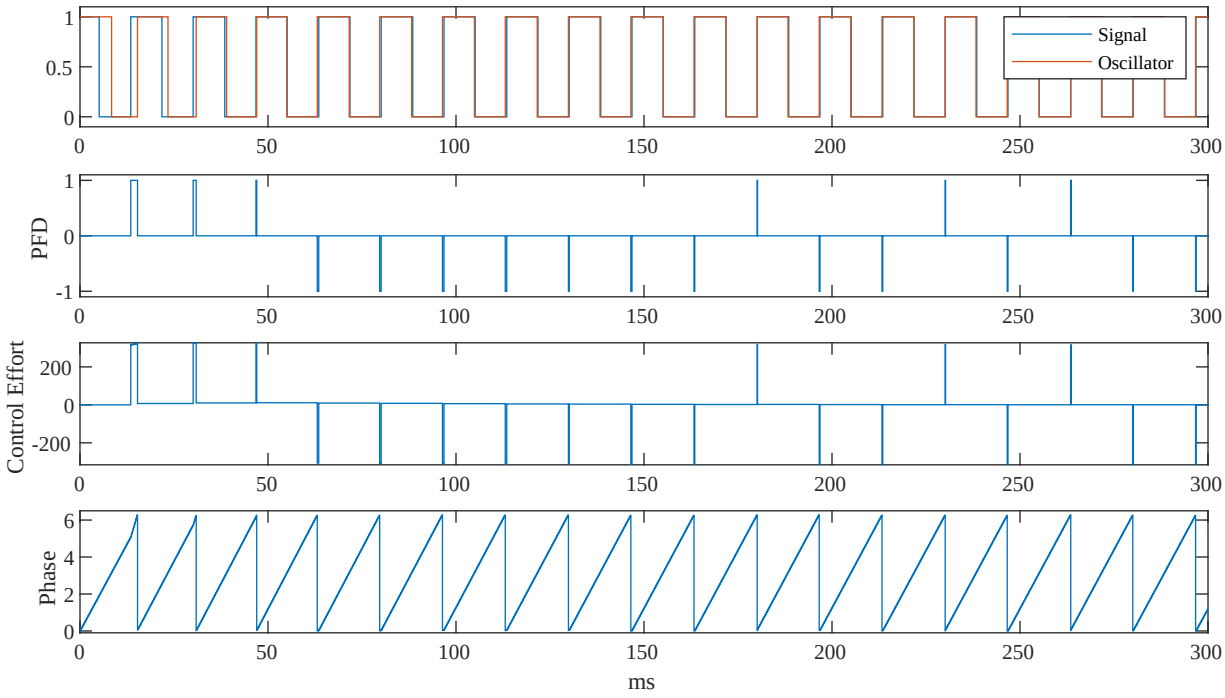
Below are the coefficients for backwards euler.

```
float A1 = 1.000000;
float A0 = -1.000000;
float B1 = 316.224213;
float B0 = -315.827332;
```

Below are the coefficients for tustin.

```
float A1 = 1.000000;
float A0 = -1.000000;
float B1 = 316.025787;
float B0 = -315.628906;
```

$$G_c(z) = \frac{316.2z - 315.8}{z - 1} \tag{6}$$



$$G_c(z) = \frac{316z - 315.6}{z - 1} \tag{7}$$