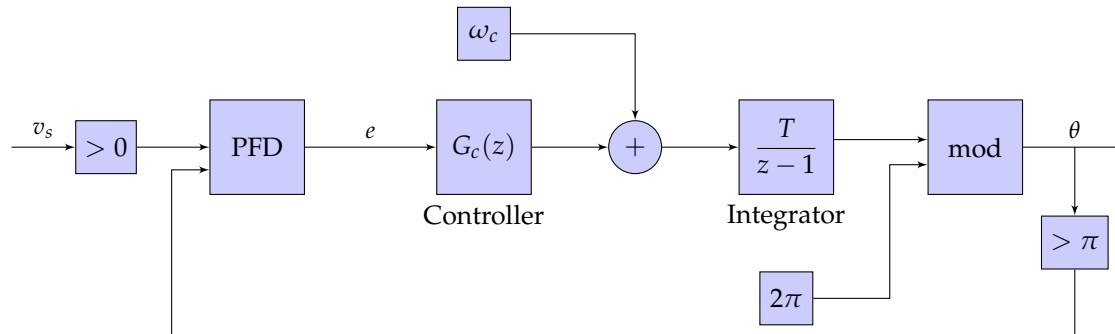


# DIGITAL CONTROL OF POWER ELECTRONICS

## 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.

$$k = 105075 \quad \omega_p = 258.6 \quad (1)$$

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

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

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

```

1  %% low pass single phase pll
2  clear
3  syms s wp k p
4
5  gol = 1/(2*p) * k/(s+wp) * 1/s
6  gcl = gol/(1+gol)
7
8  clear
9  s=tf('s');
10 opts = bodeoptions('cstprefs');
11 opts.FreqUnits = 'Hz';
12 opts.XLim= {[.1 10000]}
13
14 kvco = 1
15
16 fbw = 10
17 zeta = 1
18 w = 2 * pi * fbw
19 a = 1
20 b = -(2^2*zeta^2*2*pi*w^2)
21 c = -(2*pi)^2*w^4
22
23 k1 = (-b+sqrt(b^2-4*a*c))/2
24 k2 = (-b-sqrt(b^2-4*a*c))/2
25 wf = 2*zeta*sqrt(k1/(2*pi))
26
27 GLPF = k1/(s+wf)
28 %digital controller

```

```
29 Ts = 1/10e3
30 Z=tf('z',Ts)
31 sback = ((Z-1)/(Z*Ts))
32 gc_dig_backward = k1 / (sback +wf)
33 gc_dig_zoh = c2d(GLPF,1/10e3,'zoh')
34 gc_dig_tustin = c2d(GLPF,1/10e3,'tustin')
35
36 %% calculate coeff for floating point backward euler
37
38 Bcoeff = single(gc_dig_backward.Numerator{1});
39 Acoeff = single(gc_dig_backward.Denominator{1});
40 fprintf('float A1 = %f;\n',Acoeff(1))
41 fprintf('float A0 = %f;\n',Acoeff(2))
42 fprintf('float B1 = %f;\n',Bcoeff(1))
43 fprintf('float B0 = %f;\n',Bcoeff(2))
44
45 %% calculate coeff for floating point tustin
46
47 Bcoeff = single(gc_dig_tustin.Numerator{1});
48 Acoeff = single(gc_dig_tustin.Denominator{1});
49 fprintf('float A1 = %f;\n',Acoeff(1))
50 fprintf('float A0 = %f;\n',Acoeff(2))
51 fprintf('float B1 = %f;\n',Bcoeff(1))
52 fprintf('float B0 = %f;\n',Bcoeff(2))
```

Below are the coefficients for backwards euler.

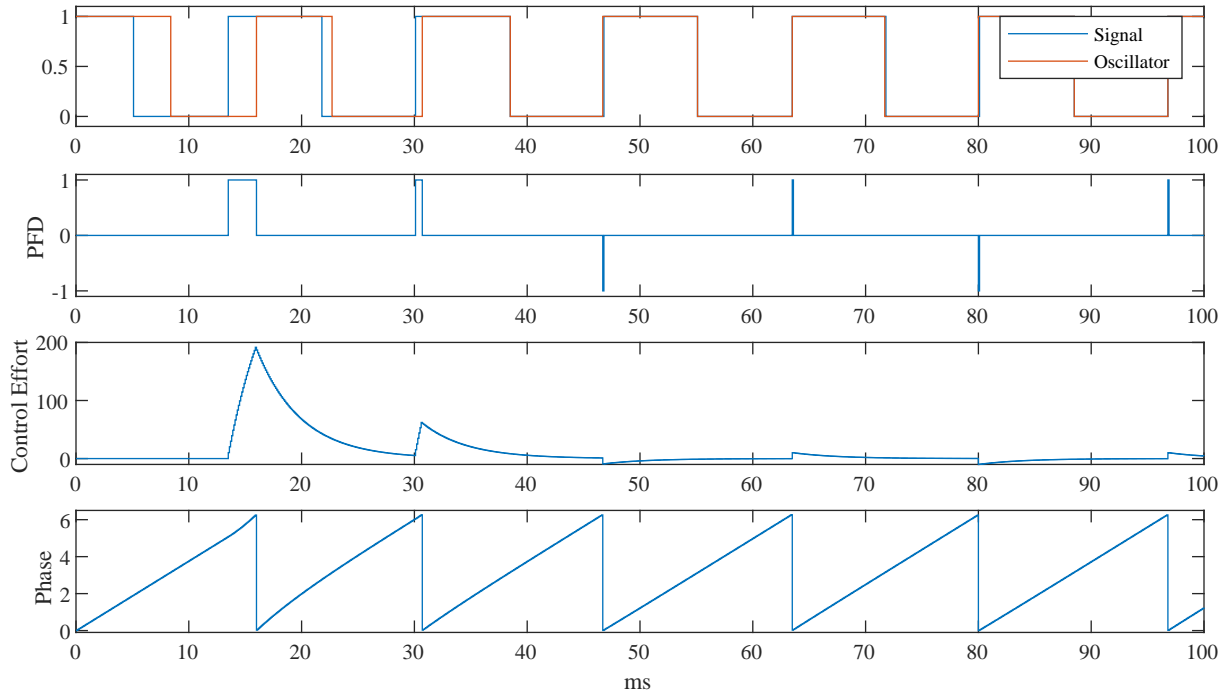
```
1 float A1 = 1.025864;
2 float A0 = -1.000000;
3 float B1 = 10.507576;
4 float B0 = 0.000000;
```

Below are the coefficients for tustin.

```
1 float A1 = 1.000000;
2 float A0 = -0.974466;
3 float B1 = 5.186714;
4 float B0 = 5.186714;
```

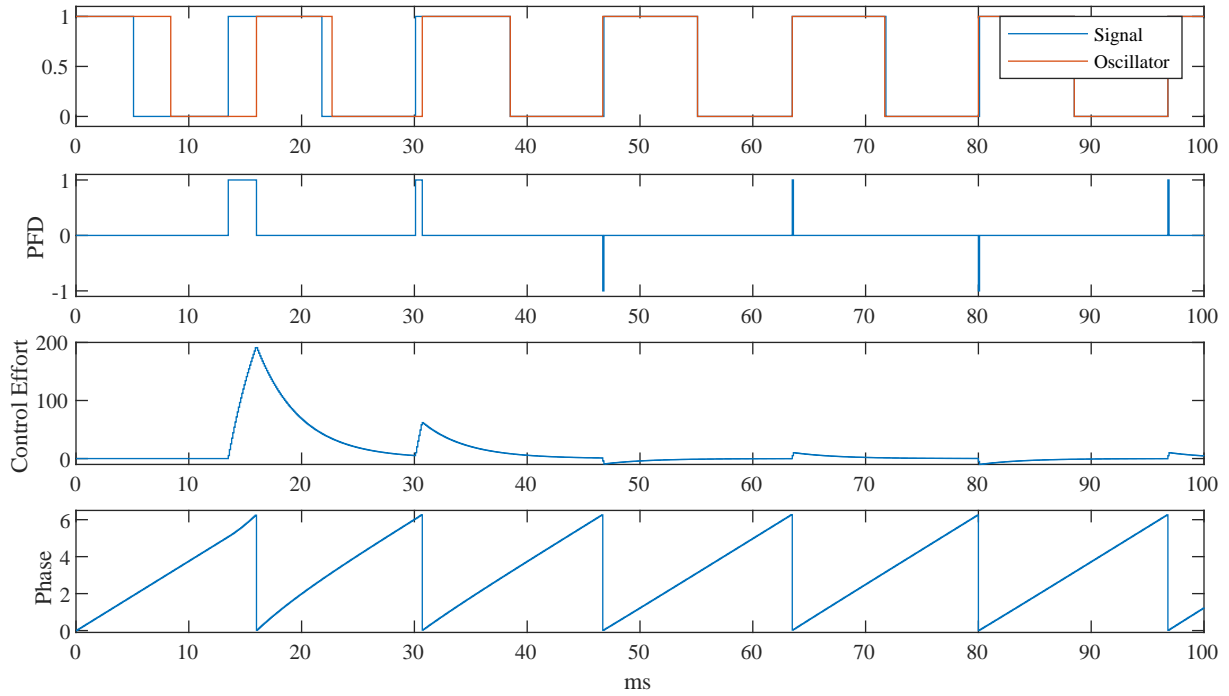
Use backward euler we arrive at a controller with.

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



The tustin transformation will arrive a the below controller

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



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

```
1 %% PI controller
2 damping = 1
3
4 omega_n = 2*pi*4
5
6 kp = 4*pi*damping*omega_n
7 ki = 2*pi*omega_n^2
8 GLPF = ki/s + kp
9
10 %digital controller
11 Ts = 1/10e3
12 Z=tf('z',Ts)
13 sback = ((Z-1)/(Z*Ts))
14 gc_dig_backward = ki/sback + kp
15 gc_dig_zoh = c2d(GLPF,1/10e3,'zoh')
16 gc_dig_tustin = c2d(GLPF,1/10e3,'tustin')
17
18 %% calculate coeff for floating point backward euler
19
20 Bcoeff = single(gc_dig_backward.Numerator{1});
21 Acoeff = single(gc_dig_backward.Denominator{1});
22 fprintf('float A1 = %f;\n',Acoeff(1))
23 fprintf('float A0 = %f;\n',Acoeff(2))
24 fprintf('float B1 = %f;\n',Bcoeff(1))
25 fprintf('float B0 = %f;\n',Bcoeff(2))
26
27 %% calculate coeff for floating point tustin
28
29 Bcoeff = single(gc_dig_tustin.Numerator{1});
30 Acoeff = single(gc_dig_tustin.Denominator{1});
31 fprintf('float A1 = %f;\n',Acoeff(1))
32 fprintf('float A0 = %f;\n',Acoeff(2))
33 fprintf('float B1 = %f;\n',Bcoeff(1))
34 fprintf('float B0 = %f;\n',Bcoeff(2))
```

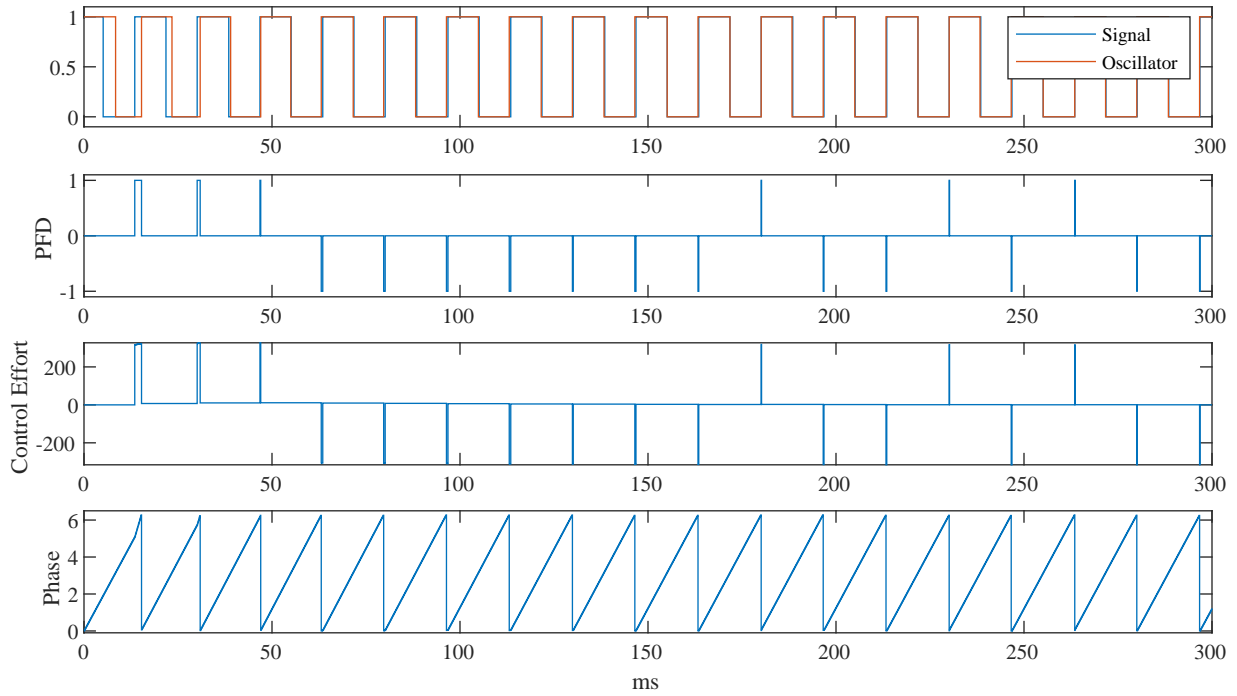
Below are the coefficients for backwards euler.

```
1 float A1 = 1.000000;
2 float A0 = -1.000000;
3 float B1 = 316.224213;
4 float B0 = -315.827332;
```

Below are the coefficients for tustin.

```
1 float A1 = 1.000000;
2 float A0 = -1.000000;
3 float B1 = 316.025787;
4 float B0 = -315.628906;
```

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



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

