Name: Mohammed Al-Sayegh, ECE 414 Homework - PID Tuner

Table of Contents

Part 1, P control

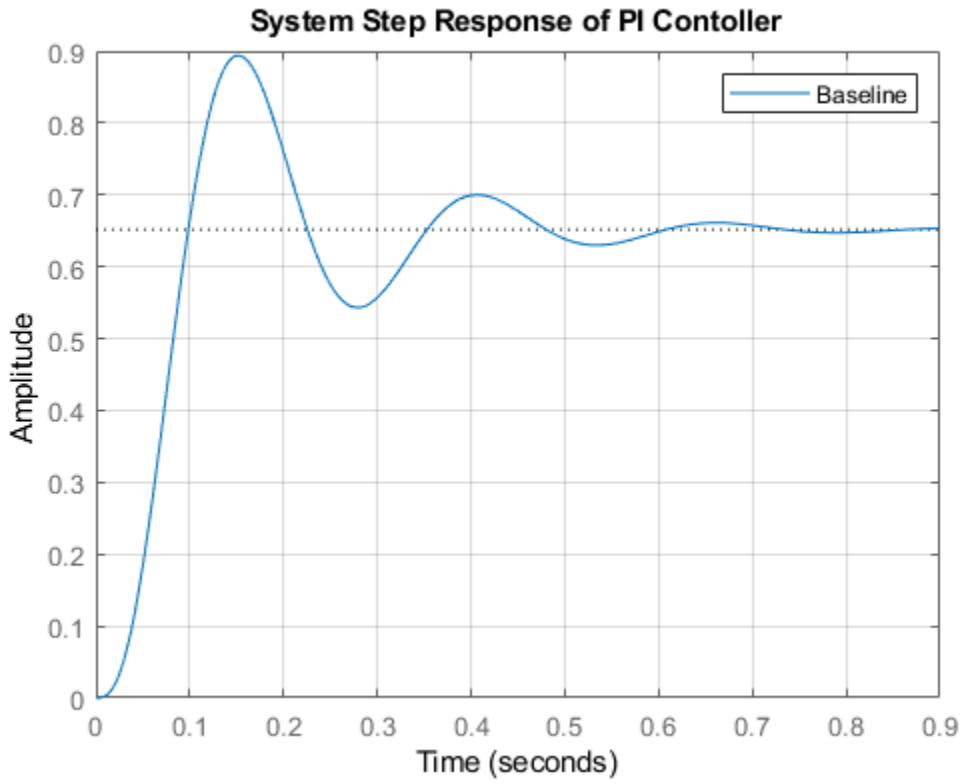
```
clc
close all

% Using pidtuner to generate a PI controller
C_p = pidtune(G, 'P');

% Pass the default terms of Kp and Ki baseline tune of PI by pidTuner
Tune = pidTuner(G, C_p);
waitfor(Tune);

% Display base terms that given by pidTuner
disp('Parameters from pidTuner');
disp('Kp = ');
disp(C_p.Kp);
disp('The step info of P controller using stepinfo:');
disp(stepinfo(C_p));

% Baseline controller effort and system transfer functions
% Baseline System Transfer Function
T_p = (C_p*G)/(1+(G*C_p));
```

```
Parameters from pidTuner
Kp =
```

```
  437.7165

The step info of P controller using stepinfo:
        RiseTime: 0
    SettlingTime: 0
     SettlingMin: 437.7165
     SettlingMax: 437.7165
       Overshoot: 0
      Undershoot: 0
            Peak: 437.7165
        PeakTime: 0
```

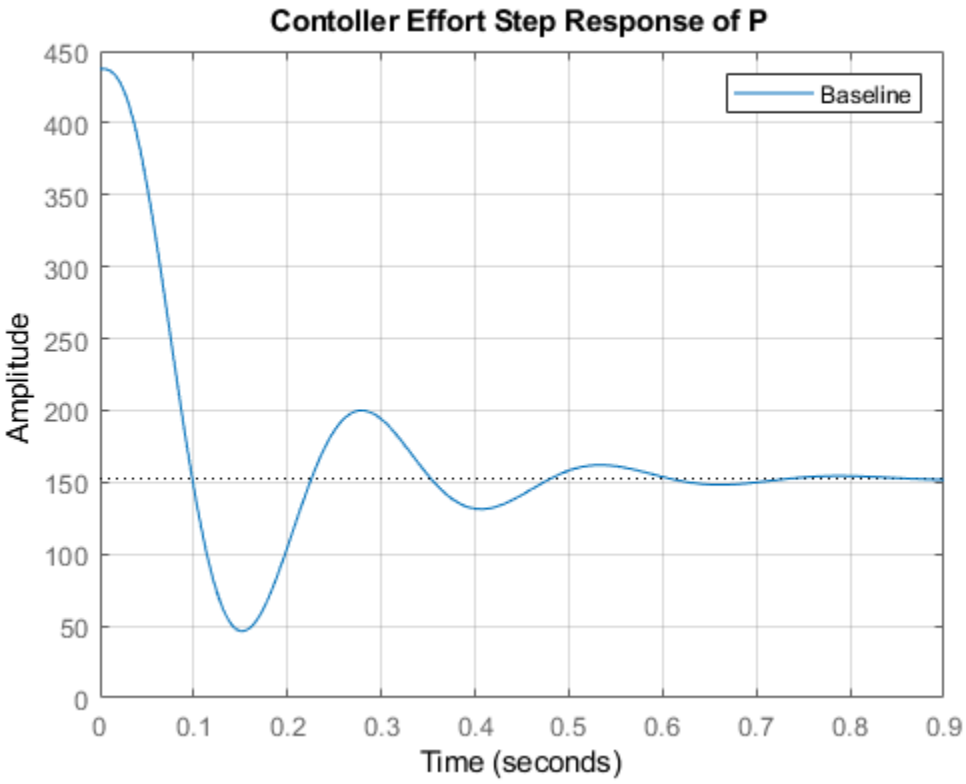
Plot the Baseline vs. tuned system Step Response

```
figure(1);
hold on;

step(T_p)
grid on;

legend('Baseline');
title('System Step Response of PI Contoller');
hold off;
```



Plot the Baseline vs. tuned system Contoller Effort Step Response

```
figure(2)
hold on;

Tu = T_p/G;
step(Tu)

grid on;

legend('Baseline');
title('Contoller Effort Step Response of P');
hold off;
```



Part 2, PD control

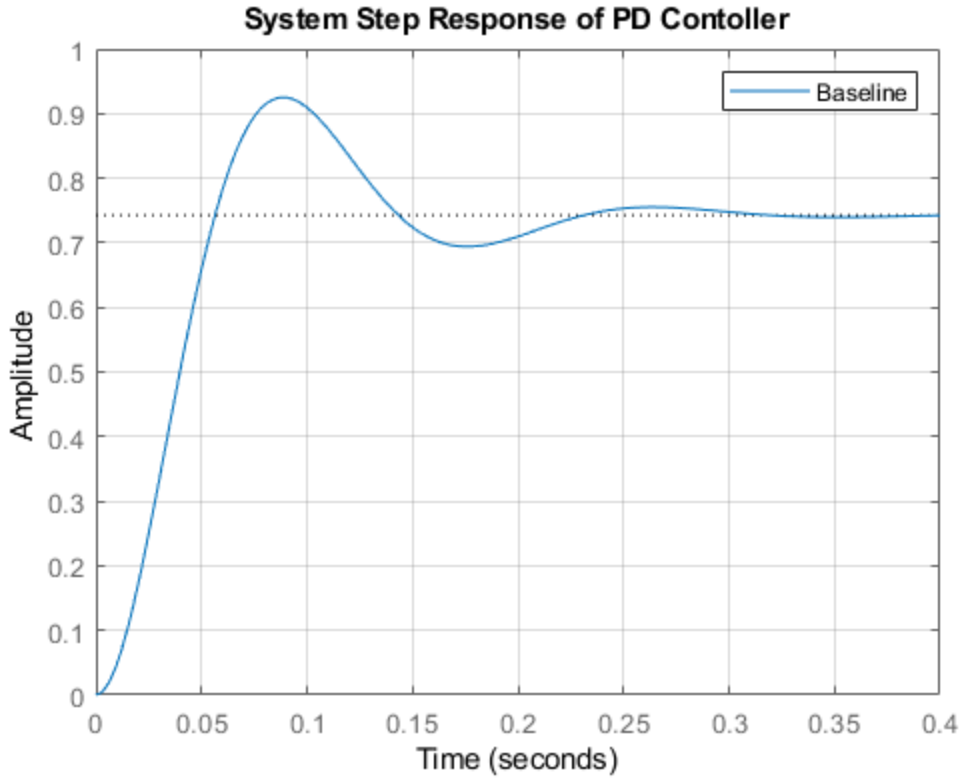
```
clc
close all

% Using pidtuner to generate a PD controller
C_pd = pidtune(G, 'PD');

% Pass the default terms of Kp and Ki baseline tune of PI by pidTuner
```

```
Tune = pidTuner(G, C_pd);
waitfor(Tune);

% Display base terms that given by pidTuner
disp('Parameters from pidTuner');
disp('Kp = ');
disp(C_pd.Kp);
disp('Kd = ');
disp(C_pd.Kd);
disp('Cannot simulate the time response of improper (non-causal) PD
 contorller');

% Baseline controller effort and system transfer functions
% Baseline System Transfer Function
T_pd = (C_pd*G)/(1+(G*C_pd));
```

```
Parameters from pidTuner
Kp =
  675.9562

Kd =
   26.3920

Cannot simulate the time response of improper (non-causal) PD
 contorller
```

Plot the Baseline vs. tuned system Step Response

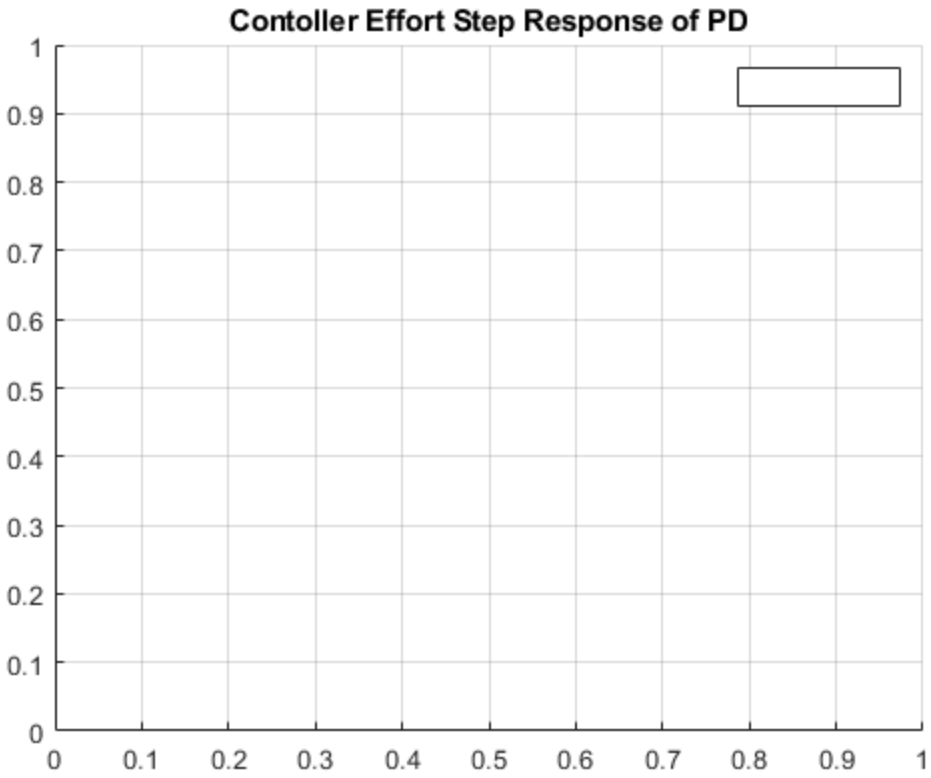
```
figure(1);
hold on;

step(T_pd)
grid on;

legend('Baseline');
title('System Step Response of PD Contoller');
hold off;
```



Plot the Baseline vs. tuned system Contoller Effort Step Response

```
figure(2)
hold on;

Tu = T_pd/G;
% step(Tu)

grid on;

legend('Baseline', 'Tuned');
title('Contoller Effort Step Response of PD');
hold off;
```

```
Warning: Ignoring extra legend entries.
```



Part 3, PI control

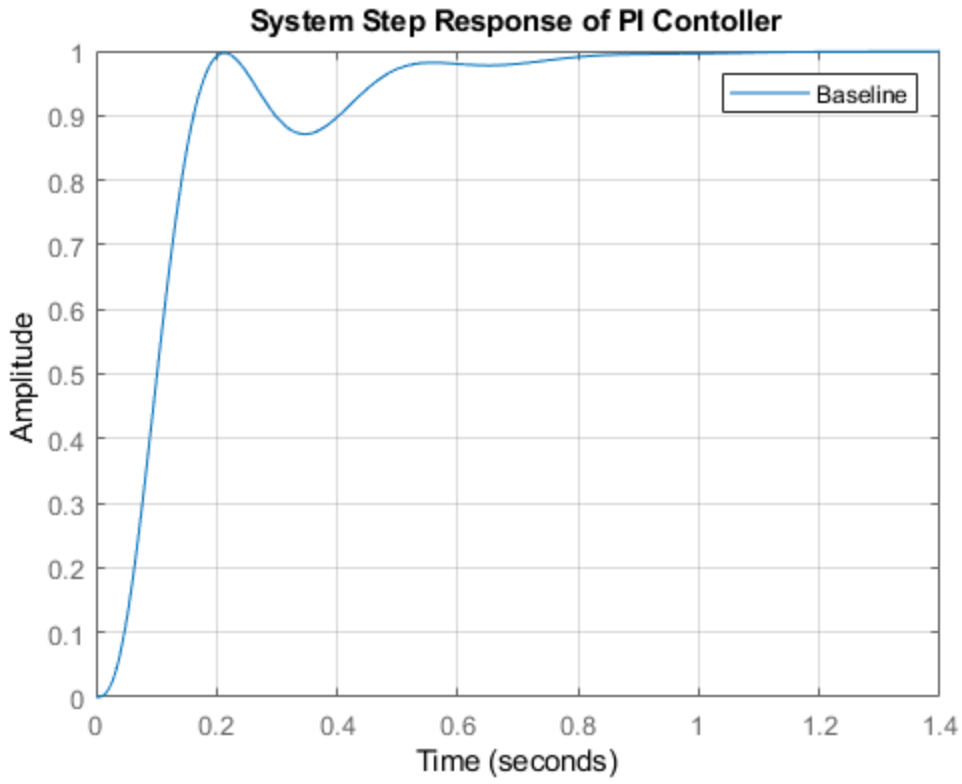
```
clc
close all

% Using pidtuner to generate a PI controller
C_pi = pidtune(G, 'PI');

% Pass the default terms of Kp and Ki baseline tune of PI by pidTuner
Tune = pidTuner(G, C_pi);
waitfor(Tune);

% Display base terms that given by pidTuner
disp('Parameters from pidTuner');
disp('Kp = ');
disp(C_pi.Kp);
disp('Ki = ');
disp(C_pi.Ki);
disp('The step info of PI controller using stepinfo:');
disp(stepinfo(C_pi));

% Baseline controller effort and system transfer functions
% Baseline System Transfer Function
T_pi = (C_pi*G)/(1+(G*C_pi));
```

```
Parameters from pidTuner
```

```
Kp =
  245.4567

Ki =
   1.7656e+03

The step info of PI controller using stepinfo:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

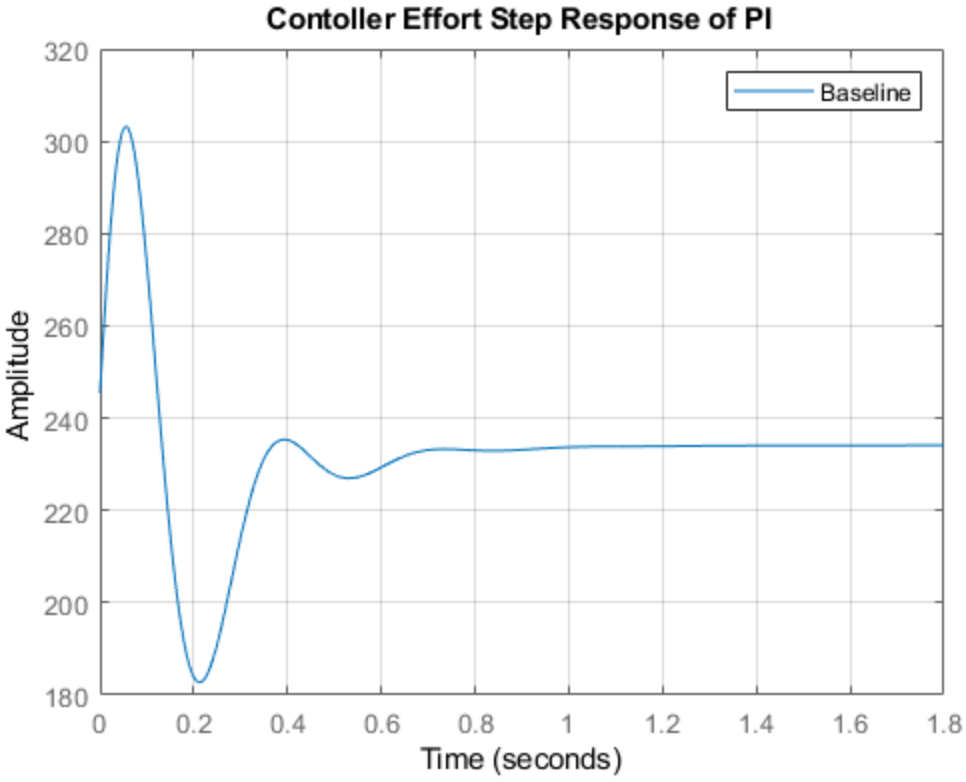
Plot the Baseline vs. tuned system Step Response

```
figure(1);
hold on;

step(T_pi)
grid on;

legend('Baseline');
title('System Step Response of PI Contoller');
hold off;
```



Plot the Baseline vs. tuned system Contoller Effort Step Response

```
figure(2)
hold on;

Tu = T_pi/G;
step(Tu)
grid on;

legend('Baseline');
title('Contoller Effort Step Response of PI');
hold off;
```



Part 4, PID control

```
pause(1);
close all;
```

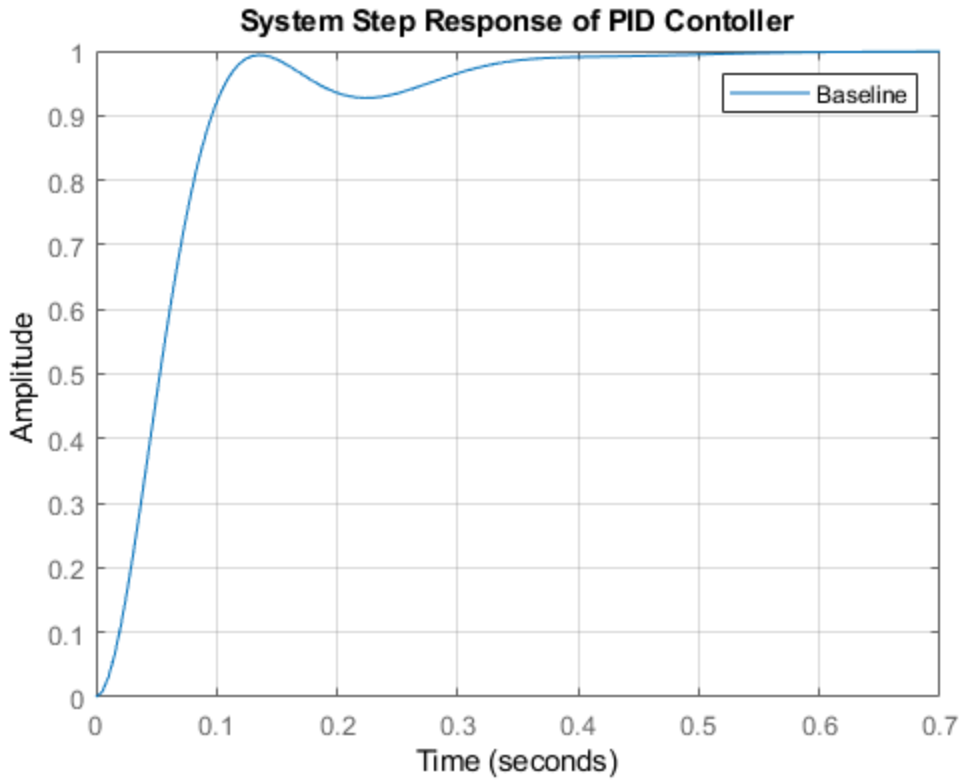
Using pidtuner to generate a PID controller

```
C_pid = pidtune(G, 'PID');

% Pass the default terms of Kp and Ki baseline tune of PI by pidTuner
Tune = pidTuner(G, C_pid);
waitfor(Tune);

% Display base terms that given by pidTuner
disp('Parameters from pidTune');
disp('Kp = ');
disp(C_pid.Kp);
disp('Ki = ');
disp(C_pid.Ki);
disp('Kd = ');
disp(C_pid.Kd);
disp('The step info of P controller using stepinfo:');
disp(stepinfo(C_p));

% Baseline controller effort and system transfer functions
```

```
% Baseline System Transfer Function
T_pid = (C_pid*G)/(1+(G*C_pid));

Parameters from pidTune
Kp =
  462.5223

Ki =
   3.5005e+03

Kd =
   15.2785

The step info of P controller using stepinfo:
        RiseTime: 0
    SettlingTime: 0
     SettlingMin: 437.7165
     SettlingMax: 437.7165
       Overshoot: 0
      Undershoot: 0
            Peak: 437.7165
        PeakTime: 0
```

Plot the Baseline vs. tuned system Step Response

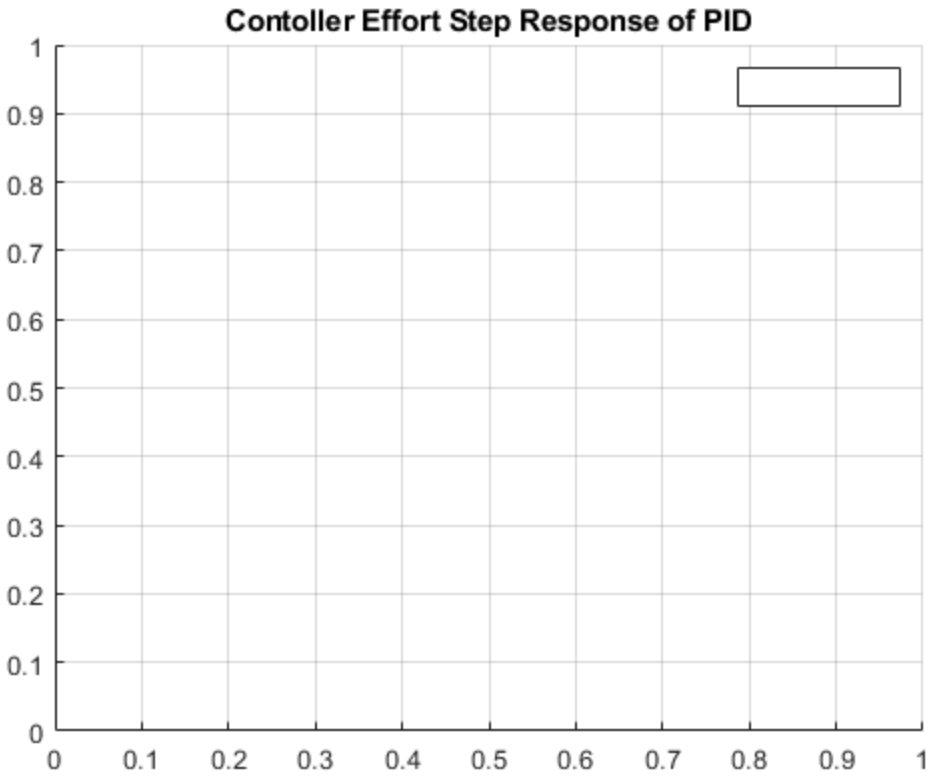
```
figure(1);
hold on;

step(T_pid)
grid on;

legend('Baseline');
title('System Step Response of PID Contoller');
hold off;
```



Plot the Baseline vs. tuned system Contoller Effort Step Response

```
figure(2)
hold on;

Tu = T_pid/G;
%step(Tu)
grid on;

legend('Baseline');
title('Contoller Effort Step Response of PID');
hold off;
```

```
Warning: Ignoring extra legend entries.
```



Part 3, PIDF control

```
clc;
pause(1);
close all;
```

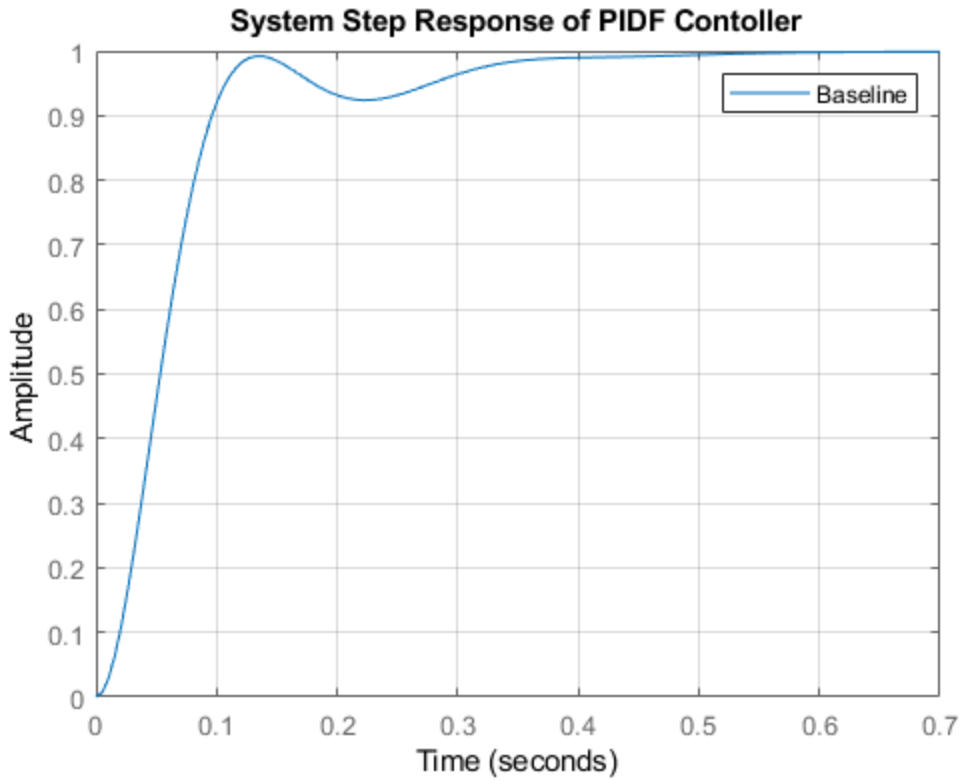
Using pidtuner to generate a PID controller

```
C_pidf = pidtune(G, 'PIDF');

% Pass the default terms of Kp, Ki and Kd baseline tune of PID by
 pidTuner
Tune = pidTuner(G, C_pidf);
waitfor(Tune);

% Display base terms that given by pidTuner
disp('Parameters from pidTune');
disp('Kp = ');
disp(C_pidf.Kp);
disp('Ki = ');
disp(C_pidf.Ki);
disp('Kd = ');
disp(C_pidf.Kd);
disp('The step info of PIDF controller using stepinfo:');
disp(stepinfo(C_pidf));
```

```
% Baseline controller effort and system transfer functions
% Baseline System Transfer Function
T_pidf = (C_pidf*G)/(1+(G*C_pidf));

Parameters from pidTune
Kp =
  459.9116

Ki =
   3.4473e+03

Kd =
   15.1426

The step info of PIDF controller using stepinfo:
        RiseTime: 9.6691e-04
    SettlingTime: 0.0017
     SettlingMin: 476.7685
     SettlingMax: 3.8734e+03
       Overshoot: 6.2816e+03
      Undershoot: 0
            Peak: 3.4559e+04
        PeakTime: 0
```

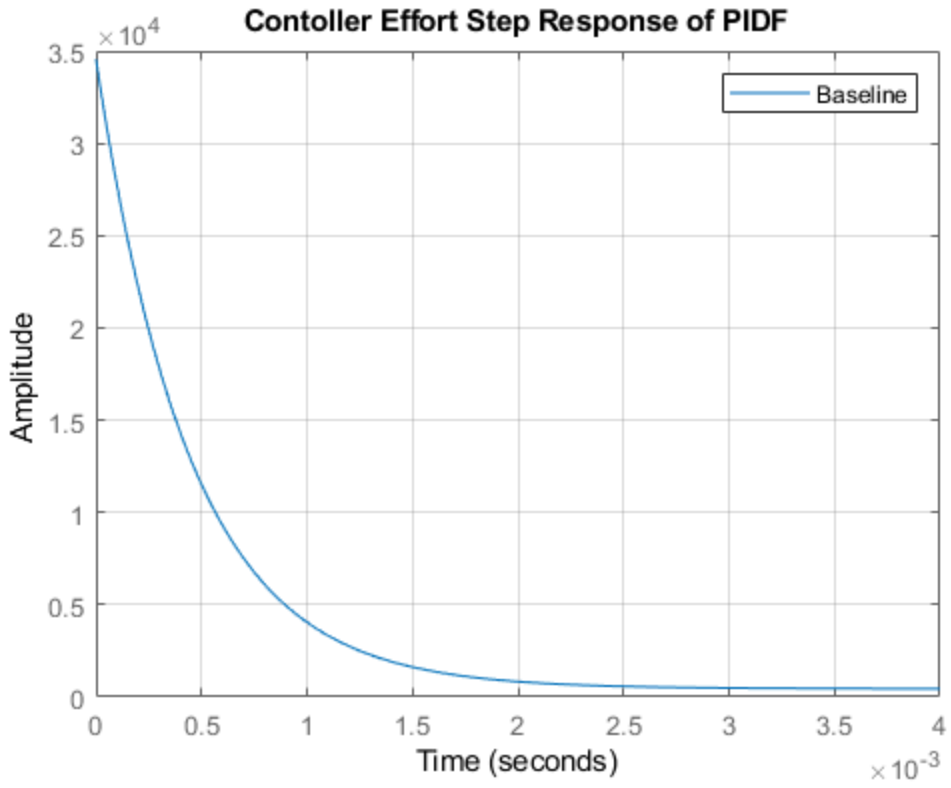
Plot the Baseline vs. tuned system Step Response

```
figure(1);
hold on;

step(T_pidf)
grid on;

legend('Baseline');
title('System Step Response of PIDF Contoller');
hold off;
```



Plot the Baseline vs. tuned system Contoller Effort Step Response

```
figure(2)
hold on;

Tu = T_pidf/G;
step(Tu)
grid on;

legend('Baseline');
title('Contoller Effort Step Response of PIDF');
hold off;
```



Published with MATLAB® R2018b