4.2.3. Tuning a PID Compensator

This Section presents an example that illustrates the effects of tuning the PID parameters on the step and error response of a simple system. The system model used is that of Figure 3.8 in Franklin, Powell and Emami-Naeni's "Feedback Control of Dynamic Systems" (2nd Ed.).

Lecturer

Set up MATLAB

```
%cd matlab
pwd
clear all
imatlab_export_fig('print-svg')  % Static svg figures.
format compact
```

```
ans =
    '/Users/eechris/code/src/github.com/cpjobling/eglm03-textbook/04/2'
```

4.2.3.1. The PID Compensator

A PID compensator has the transfer function:

$$D(s) = \frac{U(s)}{E(s)} = K_{\mathrm{prop}} + \frac{K_I}{s} + sK_D(s)$$

$$D(s) = \frac{K_D s^2 + K_{\mathrm{prop}} s + K_I}{s}$$

This can be rearranged into a form that is often found in commercial implementations of the PID compensator:

$$\begin{aligned} D(s) &= \frac{K_{\mathrm{prop}}\left(\left(K_D/K_{\mathrm{prop}} s^2\right) + s + \left(K_I/K_{\mathrm{prop}}\right)\right)}{s} \\ &= \frac{K_{\mathrm{prop}}\left(T_D s^2 + s + 1/T_I\right)}{s} \\ D(s) &= K_{\mathrm{prop}}\left(1 + T_D s + 1/(T_I s)\right) \end{aligned}$$

where K_{prop} is called the "proportional gain" (with units of percentage, i.e. 100% is equivalent to $D(s) = 1$), $T_D = K_D/K_{\mathrm{prop}}$ is known as the "derivative time", $T_I = K_I/K_{\mathrm{prop}}$ is the "integral or *reset time*" and $1/T_I$ is called the "reset rate". The tuning parameters provided with commercial PID compensators are often calibrated in these units.

4.2.3.2. Tuning the PID

In this example a simple type 0 control system with open loop transfer function:

$$G(s) = \frac{1}{(s+1)(5s+1)}$$

is to be compensated with a PID compensator and the step and error responses are considered as each parameter is adjusted.

We first define the plant:

```
s = tf('s');
G = 1/((s + 1)*(5*s + 1))
```

```
G =


           1

  ---------------
  5 s^2 + 6 s + 1


Continuous-time transfer function.
```

We then obtain the step response and error responses of the plant with unity gain feedback and no controller.

The closed loop transfer function of the system is

$$G_c(s) = \frac{G(s)}{1 + G(s)}$$

```
Gc=feedback(G,1)
```

```
Gc =


           1
```

```
---------------

5 s^2 + 6 s + 2



Continuous-time transfer function.
```

The error transfer function is given by

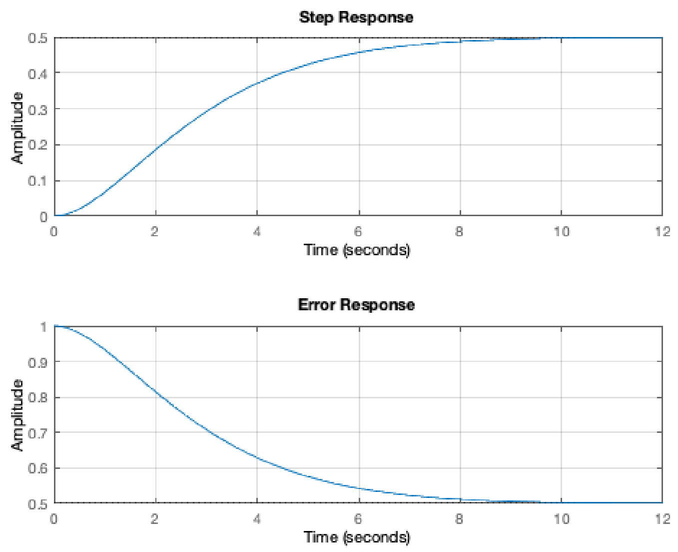
$$G_e(s) = \frac{1}{1 + G(s)}$$

```
Ge=1/(1+G)
```

```
Ge =



5 s^2 + 6 s + 1

---------------

5 s^2 + 6 s + 2



Continuous-time transfer function.
```

To simulate these responses:

```
subplot(211),step(Gc),title('Step Response'),grid
subplot(212),step(Ge),title('Error Response'),grid
```



The incompensated system has a large steady-state step error of 0.5 (or 50%). It is also overdamped and therefore slow.

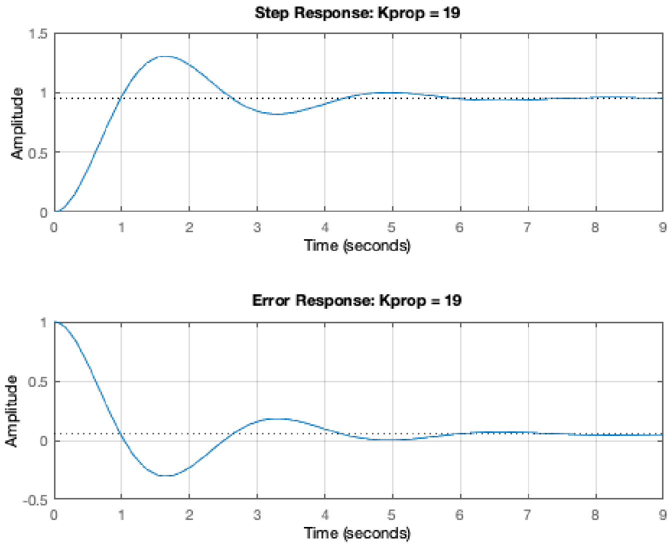
4.2.3.3. First Tuning Stage: Add Proportional Gain

Let us add some proportional control to decrease the rise time and to reduce the error to 5%. Increasing the proportional gain of the system can satisfy the error requirement

```
Kprop = 19;
D = Kprop;
Go = D*G
```

```
Go =



      19

---------------

5 s^2 + 6 s + 1



Continuous-time transfer function.
```

```
Gc = feedback(Go,1);
Ge = 1/(1 + Go);
subplot(211),step(Gc),title('Step Response: Kprop = 19'),grid
subplot(212),step(Ge),title('Error Response: Kprop = 19'),grid
```



Note that the steady-state error is now 5% as required and that the proportional controller has speeded up the response time, but at the cost of reduced damping and excessive overshoot and settling time.

4.2.3.4. Second Tuning Stage: Add Derivative Action

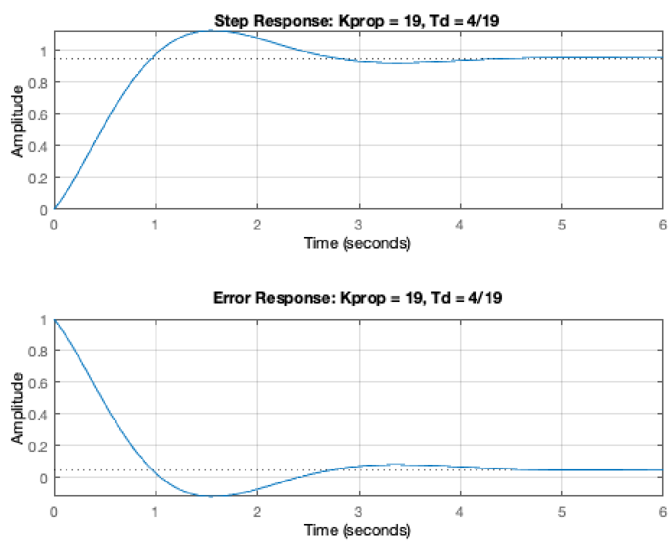
To reduce the overshoot and settling time we add some derivative action to dampen the response without affecting the steady state error.

```
Td = 4/19;
Deriv = Td*s;
D = Kprop *(1 + Deriv);
Go = D*G;
```

```
Gc = feedback(Go,1)
Ge=1/(1 + Go)
```

```
Gc =


      4 s + 19

  ----------------

  5 s^2 + 10 s + 20


Continuous-time transfer function.

Ge =


   5 s^2 + 6 s + 1

  ----------------

  5 s^2 + 10 s + 20


Continuous-time transfer function.
```

```
subplot(211),step(Gc),title('Step Response: Kprop = 19, Td = 4/19'),grid
subplot(212),step(Ge),title('Error Response: Kprop = 19, Td = 4/19'),grid
```



4.2.3.5. Third Tuning Stage: Add Integral Action

Finally adding some integral action will eliminate the steady-state step error altogether.

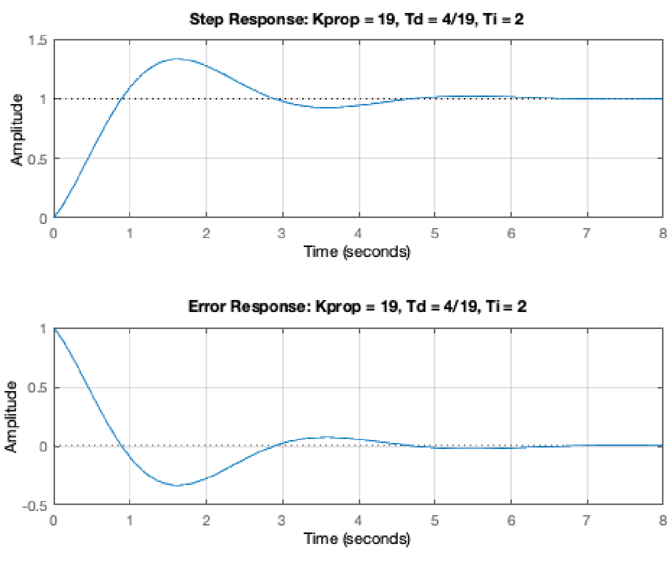
```
Ti = 2;
Integ = (1/Ti)/s;
D = Kprop*(1 + Deriv + Integ);
Go = D*G
```

```
Go =


  4 s^2 + 19 s + 9.5

  ------------------

  5 s^3 + 6 s^2 + s


Continuous-time transfer function.
```

```
Gc=feedback(Go,1)
Ge=1/(1 + Go)
```

```
Gc =


      4 s^2 + 19 s + 9.5

  ---------------------------

  5 s^3 + 10 s^2 + 20 s + 9.5


Continuous-time transfer function.

Ge =


       5 s^3 + 6 s^2 + s

  ---------------------------

  5 s^3 + 10 s^2 + 20 s + 9.5


Continuous-time transfer function.
```

```
subplot(211),step(Gc),title('Step Response: Kprop = 19, Td = 4/19, Ti = 2'),grid
subplot(212),step(Ge),title('Error Response: Kprop = 19, Td = 4/19, Ti = 2'),grid
```



4.2.3.6. Comments

One would probably try further adjustments to reduce the overshoot that the addition of integral action produces

4.2.3.7. Exercises

You might like to try further adjustments of the parameters K_{prop}, T_d and T_I.

See also the companion documents on the Analytical design of a PID compensator and Zeigler-Nichols tuning and Autotuning a PID Compensator with MATLAB.

4.2.3.8. Resources

An executable version of this document is available to download as a MATLAB Live Script file pid_tuning.mlx. You can simulate this design using the file pid_cl.slx.

By Dr Chris P. Jobling