Some Basic Controllers

John Hung & Vic Nelson

ECE Department – Auburn University

ELEC 3040/3050

Notation for signal and systems

Continuous time models:

$$e(t) \qquad \frac{de(t)}{dt} \qquad \int_0^t e(\tau)\, d\tau$$

Laplace transform notation:
(for signals initially at rest)

$$E(s) \qquad sE(s) \qquad \frac{E(s)}{s}$$

Key transform ideas :
Time differentiation $\Leftrightarrow$ "multiply by s"
Time integration $\Leftrightarrow$ "divide by s"

Proportional Control

Control effort $u(t)$ is proportional to error $e(t)$:

$$e(t) = r(t) - y(t)$$
$$u(t) = K_P \times e(t)$$

where r: reference, y: output, K_P: proportional gain.

In transform notation:

$$U(s) = K_P E(s).$$

Proportional Control

The C implementation

```
// Declare variables
 unsigned char ref;          // reference signal
 unsigned char output;       // output signal
 signed int error;           // error signal
 unsigned int Kp=15;         // proportional gain

// Compute error and control effort
 error = ref – output ;      // error
 u = Kp * error;             // control effort
```

Characteristics of Proportional Control

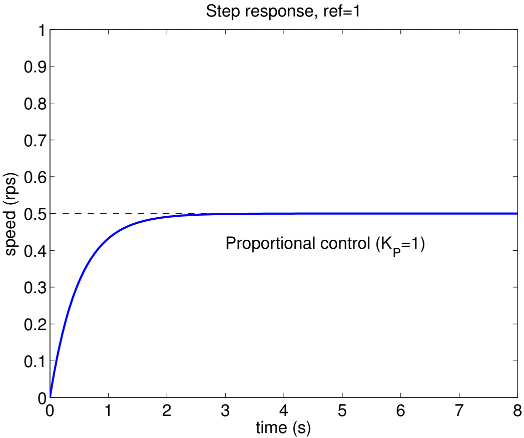
- Easy to program
- Requires finite error to develop non-zero control effort

Example (motor speed control):
If motor speed equals desired speed, then error is zero.
$\downarrow$
Control effort (motor voltage) becomes zero.
$\downarrow$
Motor speed decreases.

An example: Proportional Control

Process model:

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



See the steady-state error!

Integral Control

Control effort $u(t)$ is proportional to accumulated error $e(t)$:

$$u(t) = K_I \int_0^t e(\tau)\, d\tau$$

where K_I: integral gain.

In transform notation:

$$U(s) = \frac{K_I}{s} E(s).$$

Integral Control

The C implementation

```
// Declare variables
unsigned char ref;          // reference signal
unsigned char output;       // output signal
signed int error;           // error signal
unsigned int Ki=3;          // integral gain

// Compute error and control effort
error = ref – output ;      // error
u += Ki * error;            // control effort
or
u = u + Ki * error;
```

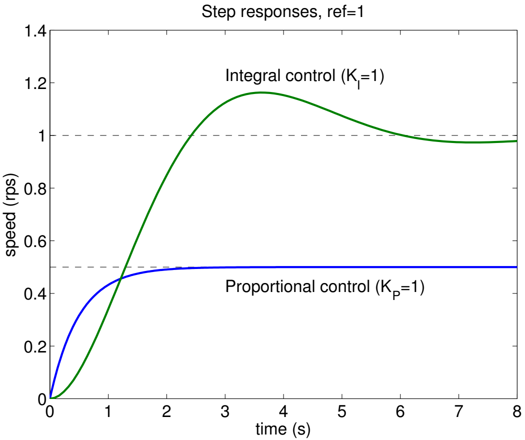
Characteristics of Integral Control

- Can eliminate steady-state error
- Integration raises the order of system dynamics
- Integration introduces delay (takes time to accumulate error)
- Step response has greater tendency to overshoot and oscillate (compared to proportional control)

An example: Integral Control

Comparing against proportional control

- Final error = 0.
- Slower response
- Response overshoots



Proportional + Integral (PI) Control

A linear combination of Proportional and Integral controllers.

In transform notation:

$$U(s) = \left(K_P + \frac{K_I}{s}\right) E(s)$$

or

$$U(s) = \left(\frac{K_P s + K_I}{s}\right) E(s).$$

Deriving a time-domain model of PI control

In transform notation:

$$U(s) = \left(\frac{K_P s + K_I}{s} \right) E(s)$$

or

$$U(s) = \frac{K_P \times sE(s) + K_I \times E(s)}{s}$$

(Recall: Multiply by $s \Leftrightarrow$ time differentiation; divide by $s \Leftrightarrow$ time integration.)

So, in time domain:

$$u(t) = \int_0^t \left(K_P \times \frac{de}{dt} + K_I \times e \right) d\tau.$$

PI Control

The C implementation – part 1

```
// Declare variables
 unsigned char ref;        // reference signal
 unsigned char output;     // output signal
 signed int error;         // error signal
 signed int lasterr;       // previous error
 unsigned int Kp=1;        // proportional gain
 unsigned int Ki=2;        // integral gain
```

The variable “lasterr” is needed to compute the derivative de/dt.

PI Control

The C implementation – part 2

```
// Compute error and control effort, then update error history
error = ref – output ;                        // error
u +=Kp*(error – lasterr) + Ki * error;        // control effort
lasterr = error;                              //update error history
```

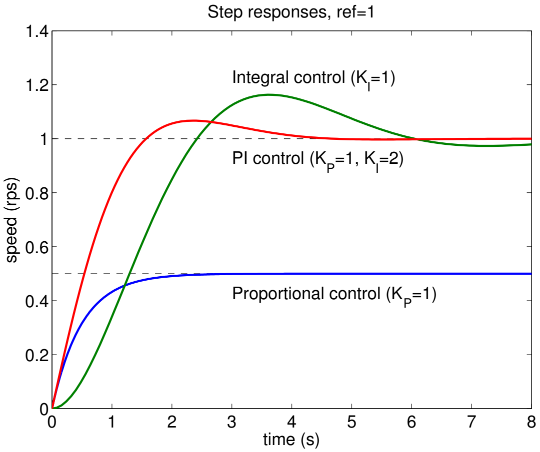
Third line updates the "previous error" for the next iteration!

Characteristics of PI Control

- Less steady-state error (than proportional control)
- Less delay and overshoot (than integral control)
- More complex to program than P or I controllers
- More effort to properly tune

An example: PI Control

Comparing responses to P, I, and PI controllers



Summary of Basic, Linear Controllers

Controller	Notes
P	Simple, but may yield non-zero final error
I	Can reduce final error Transient response tends to be slower and less stable
PI	Can reduce final error Can yield good transient response More complex to program