EEET 3046 Control Systems (2020)

Lecture 8: Controller design by heuristic tuning

Dr. Yee Wei Law ⟨yeewei.law@unisa.edu.au⟩

Contents

1 Introduction

Lecture 5 introduced PID control, while Lecture 6 explained the roles of P, I and D actions. Furthermore, in the previous lecture, we learnt how to use root loci to tune PID controllers. However, the root locus method requires a model of the plant/system. When this model is hard to come by, we can resort to *empirical/heuristic tuning rules*.

Tuning rules are a set of formulas for calculating controller gains based on

1. some information about the plant, which can be a model or some measurable characteristics of the plant, and
2. some performance criterion, which can be disturbance rejection, desired dominant poles, stability margins, or some other requirement.

Attention: Disturbance

In this lecture, by "disturbance", we mean "load disturbance", aka "input disturbance".

Different rules suit different plants, and there is no one-size-fits-all solution. It is therefore not surprising the topic of PID controller tuning rules has been researched for decades. In fact, the handbook [O'D09] catalogs more than 200 tuning rules. MATLAB itself has at least three PID tuners:

- **Control System Designer's PID tuner**: This is embedded in the Control System tab of the Control System Designer (see Figure 6). This tuner supports rule-based *automatic tuning* (*auto-tuning* for short) — which is the process of automatically identifying the plant model and tuning the controller based on that model [Vis06] — with limited interactivity.
- **GUI-based PID tuner**: This is invocable by the command `pidTuner` and accessible through Simulink (see Figure 12). This tuner is more general and offers a high level of interactivity.

- **Command-based PID tuner**: This is the toned-down version of `pidTuner` invocable by the command `pidtune`. It offers limited and noninteractive configurable options through the function `pidtuneOptions`.

In this lecture, we will learn about two heuristic tuning rules called the *Ziegler-Nichols* tuning rules. Heuristic tuning rules are handy because they do not require a model of the system. They serve as a good starting point for learning, and for tuning, they serve as a baseline for comparison with more advanced rules.

2 Ziegler-Nichols

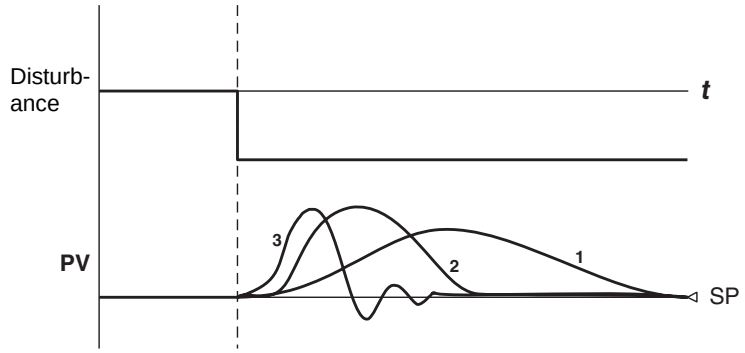


Figure 1: Sample responses to a disturbance [SMY14, Figure 5.1]. Response 3 allows the process variable (PV) to return to the set-point (SP) most quickly, although the peak deviation is larger than that of response 1.

For a controller to be useful, it must be able to track the set-point and reject disturbances. In general, efficient disturbance rejection requires a high-gain controller. However, high-gain controllers are associated with oscillatory step-SP responses [VZ10, Ch. 2]. For example, in Figure 1, we can see three sample step-disturbance responses, and efficient disturbance rejection entails some oscillation in the response. Here lies the basis for the Ziegler-Nichols tuning rules: for efficient disturbance rejection, the controller is tuned to approximately achieve a specified level of damping. The level of damping is specified as *quarter decay ratio* / *quarter amplitude damping*.

Definition: Decay ratio

. . . is the ratio of the second highest peak to the highest peak, as illustrated in Figure 2.

Detail: Decay ratio of second-order systems

Using the definitions of A, B, C in Figure 2, we know from Lecture 4 that for second-order systems, the peak time is given by $t_B = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}}$, and

$$\frac{B}{A} = \exp(-\zeta \omega_n t_B) = \exp\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right),$$

The second peak time is given by $t_C = \frac{3\pi}{\omega_n \sqrt{1-\zeta^2}}$, so

$$\frac{C}{B} = \frac{C}{A} \cdot \frac{A}{B} = \exp\left(-\frac{3\pi\zeta}{\sqrt{1-\zeta^2}}\right) \cdot \exp\left(\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right) = \exp\left(-\frac{2\pi\zeta}{\sqrt{1-\zeta^2}}\right).$$

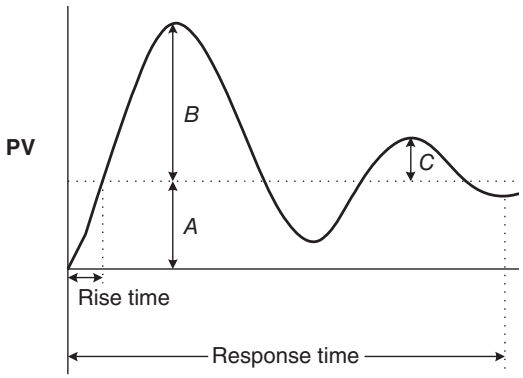


Figure 2: Decay ratio is C/B. Image from [SMY14, Figure 5.2].

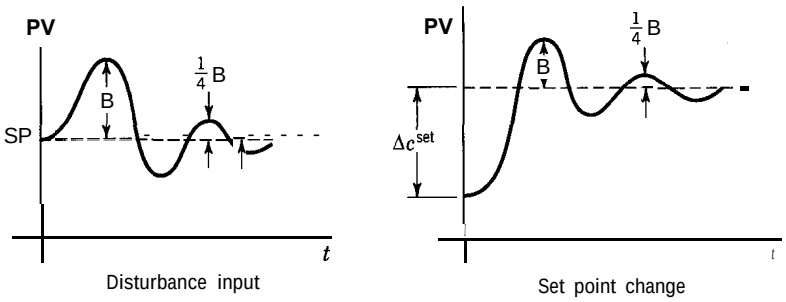


Figure 3: The quarter decay ratio requirement applies to both (a) step-disturbance responses, and (b) step-input responses. Image from [SC97, Figure 7-1.2].

Therefore, the decay ratio of a second-order system is given by

$$\frac{C}{B} = \exp\left(-\frac{2\pi\zeta}{\sqrt{1-\zeta^2}}\right). \tag{1}$$

A quarter decay ratio specifies a decay ratio of 1/4 for both step-disturbance responses, and step-input responses (see Figure 3). The quarter decay ratio criterion has been shown through experience to provide a reasonable trade-off between minimum deviation from the set-point after an upset and the fastest return to the set-point [SMY14, p. 121]. Ziegler and Nichols' empirical observation was that if a PID controller can condition system responses to have a quarter decay ratio, the controller should provide good disturbance rejection — this objective is described as "aggressive" in the literature. However, a quarter decay ratio is widely considered to be too underdamped/oscillatory for most process control applications [SMEDI10, Sect. 12.3.3]. After all, the Ziegler and Nichols designed their rules for US Naval servo control applications [SMY14, p. 126]. Nevertheless, the quarter decay ratio criterion serves as a reasonable starting point for tuning. In fact, Ziegler and Nichol's tuning rules were widely adopted by controller vendors for routine use in the 1940s. A simple demonstration of the disturbance rejection capability of a Ziegler-Nichols-tuned PID controller is available at `https://youtu.be/7qw7vnTGNsA`.

With the quarter decay ratio criterion in mind, Ziegler and Nichols proposed two empirical tuning rules, which are over time known by many names:

1. **step response** / open-loop / step test / process reaction curve;

2. **frequency response** / closed-loop / continuous cycling / constant cycling / sustained oscillation.

Both rules follow the framework of

- performing some system response experiments on the plant;
- extracting the characteristic features of the process dynamics from the experiments; and
- determining the controller parameters from the features.

2.1 Step response method

The method is based on

- measuring the features of a *process reaction curve* (system response to a step input or step disturbance) obtained in open loop (with the controller disconnected); and
- using the measured features to approximately model the plant as
  - either an *integrator plus dead time* (IPDT) transfer function, which takes the form

$$\frac{Ke^{-Ls}}{s}, \tag{2}$$

    where K is the gain, L is the transport lag;
  - or a *first-order plus dead time* (FOPDT) transfer function, which takes the form

$$\frac{Ke^{-Ls}}{Ts+1}, \tag{3}$$

    where K is the gain, T is the time constant, and L is the transport lag.

IPDT transfer functions are for modeling *open-loop unstable* / *non-self-regulating* processes, whereas FOPDT transfer functions are for *open-loop stable* / *self-regulating* processes [SMEDI10, p. 227]. A large variety of plants in the process control industry can in fact be approximately modeled using these transfer functions [XCA07, Sect. 6.2]. Ziegler and Nichols developed their step response method by simulating a larger number of different IPDT and FOPDT processes, and correlating the controller parameters with features of the step response [ÅH04]. The controller parameters correlated with quarter decay ratios gave rise to the tuning rule.

The procedure of the method is as follows:

1. Disconnect the controller and let the system settle into steady state.
2. Introduce a step input of amplitude r, and estimate the reaction rate, $R \stackrel{\text{def}}{=} \max(\dot{y}(t))$, and transport lag L (see Figure 4). The estimation can be done using the MATLAB code in Listing 1.

Listing 1: MATLAB code for fitting the process reaction curve.

```
% Inputs:
%   y: An array representing the output response.
%   t: An array of time stamps. Set-point change assumed to happen at t(1),
%      but t(1) may not be 0.
% Outputs:
%   R: Reaction rate.
%   L: Transport lag.
if s == true                          % self-regulating processes
  [R, I] = max(diff(y)./diff(t));     % reaction rate
  L = t(I)-t(1) - (y(I)-y(1))/R;      % transport lag
  M = y(end)-y(1);                    % amplitude of output
  T = M/R;                            % time constant
```

```
else
  R = (y(end)-y(end-1))/(t(end)-t(end-1));
  M = y(end)-y(1);                    % amplitude of output
  L = t(end)-t(1) - M/R;              % transport lag
end
```

The estimated features are used to approximately model the plant with

- an IPDT transfer function if the process is non-self-regulating:

$$\frac{K}{s}e^{-Ls} = \frac{R/r}{s}e^{-Ls},$$

  where R is the estimated reaction rate, r is the known amplitude of the input, L is the estimated transport lag;

- or an FOPDT transfer function if the process is self-regulating:

$$\frac{K}{Ts+1}e^{-Ls} = \frac{M/r}{Ts+1}e^{-Ls},$$

  where $M = RT$ is the estimated amplitude of the output, r is the known amplitude of the input, T is the estimated time constant, L is the estimated transport lag.

3. Calculate the controller gains based on the formulas in Table 1.

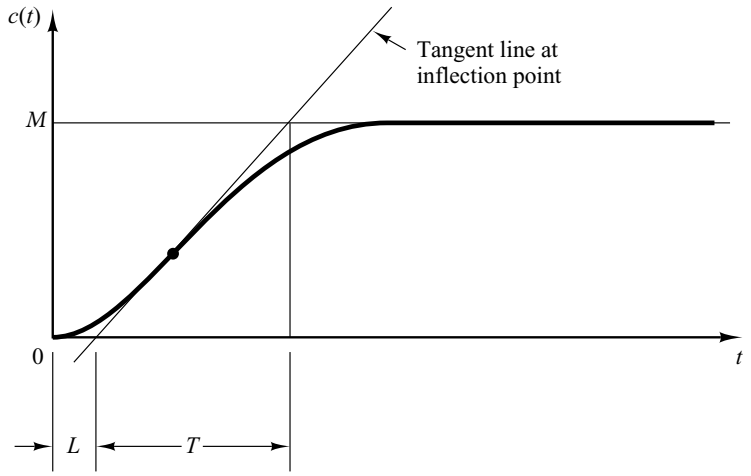


Figure 4: The process reaction curve of a self-regulating process [Oga10, Figure 8-3]. The inflection point is where the angle of the tangent reaches a maximum, or equivalently, where the second-order derivative of the curve reaches 0. Note $M = RT$.

Table 1: Formulas for the Ziegler-Nichols step response method [SMY14, p. 126], [DB11, Table 7.8], [Oga10, Table 8-1].

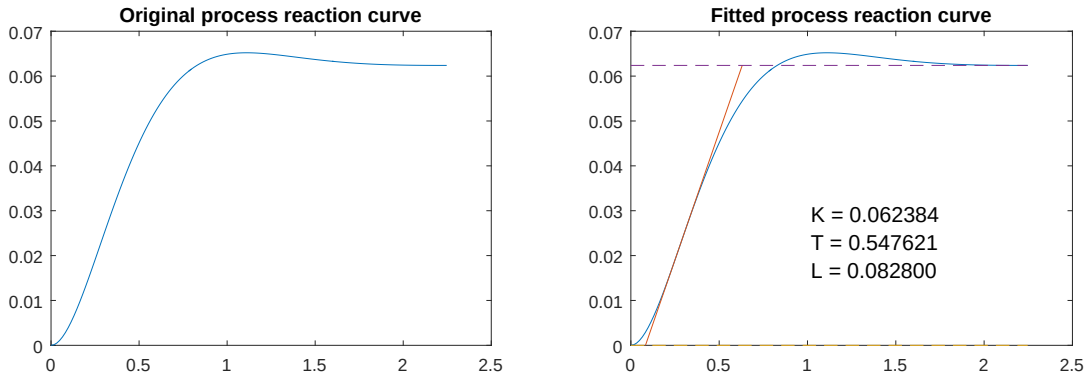
Controller type	K_p	T_i	T_d
P	$r/(RL)$		
PI	$0.9r/(RL)$	$L/0.3$	
PID	$1.2r/(RL)$	$2L$	$0.5L$

A PID controller tuned using the step response method has the transfer function

$$C(s) = \frac{1.2r}{RL}\left(1 + \frac{1}{2Ls} + 0.5Ls\right) = \frac{1.2r}{RL}\left(\frac{2Ls + 1 + L^2s^2}{2Ls}\right) = \frac{0.6r}{RL^2}\frac{(Ls+1)^2}{s},$$

$$\therefore C(s) = \frac{0.6r}{R}\frac{(s+1/L)^2}{s}. \tag{4}$$

Thus, the step response method places double zeros at $-1/L$, allowing the controller to be implemented in the series form. This was intentional since the series form was the norm at the time Ziegler and Nichols worked on their tuning rules.



(a) (b)

Figure 5: Process reaction curves for Example 1.

Example 1

Suppose a process reaction curve is obtained as in Figure 5(a). Fitting the curve using Listing 1 gives us the parameters in Figure 5(b).

(a) Apply the Ziegler-Nichols step response method to tune a PID controller. Denote this controller by $C_1(s)$.

(b) Suppose the plant transfer function is actually

$$G(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \quad \text{where } \zeta = 1/\sqrt{2}, \omega_n = 4.$$

Use Control System Designer's PID tuner and its Ziegler-Nichols step response rule to tune a PID controller. Denote this controller by $C_2(s)$.

(c) Compare $C_1(s)$ with $C_2(s)$ in terms of their step-input response and step-disturbance response.

Solution:

(a) From Figure 5(b), we have K, T and L, but not r and R which are required by the formulas in Table 1. However,

$$Kr = M = RT \implies \frac{r}{R} = \frac{T}{K} = 8.7783.$$

Plugging the values $r/R = 8.7783$ and $L = 0.0828$ into Eq. (4), we get the PID transfer function

$$C_1(s) = \frac{0.6r}{R}\frac{(s+1/L)^2}{s} = \frac{5.2669(s+12.0773)^2}{s}.$$

(b) Launch Control System Designer with the following code:

```
zeta = 1/sqrt(2); omegan = 4;
G = tf(1, [1 2*zeta*omegan omegan^2]);
controlSystemDesigner(G);
```

On the "CONTROL SYSTEM" tab, among "Tuning Methods", select "PID Tuning", as shown in Figure 6. The PID transfer function can be obtained as

$$C_2(s) = \frac{3.3534(s+12.1)^2}{s},$$

which is different from $C_1(s)$.

(c) While obtaining the step-input response is straightforward, the step-disturbance response requires some effort. From Lecture 3, we know

$$E(s) = \frac{R(s)}{1+C(s)G(s)} - \frac{G(s)D(s)}{1+C(s)G(s)},$$

assuming the disturbance going into the plant is $D(s)$, not $-D(s)$. When $R(s) = 0$,

$$Y(s) = R(s) - E(s) = \frac{G(s)D(s)}{1+C(s)G(s)},$$

so the disturbance-to-output transfer function is $\dfrac{G(s)}{1+C(s)G(s)}$ (this is actually the *load sensitivity function* introduced in Lecture 3). Using this transfer function, we can plot the step-disturbance responses. Figure 7 compares the step-input responses attributed to $C_1(s)$ and $C_2(s)$; and the step-disturbance responses attributed to $C_1(s)$ and $C_2(s)$. Interestingly, $C_1(s)$, tuned using the textbook version of the step response method, provides better transient response and disturbance rejection than $C_2(s)$, which is tuned using the MATLAB version of the step response method. Nevertheless, in both cases, the step-input response is more oscillatory than is generally acceptable, so some further manual tuning is necessary.

2.2 Frequency response method

The idea is to determine the proportional gain that sends the closed-loop system to the verge of instability, where the system stays in a state of *continuous cycling* (sustained oscillation with a constant amplitude). The proportional gain is then decreased to the point that gives a quarter decay ratio.

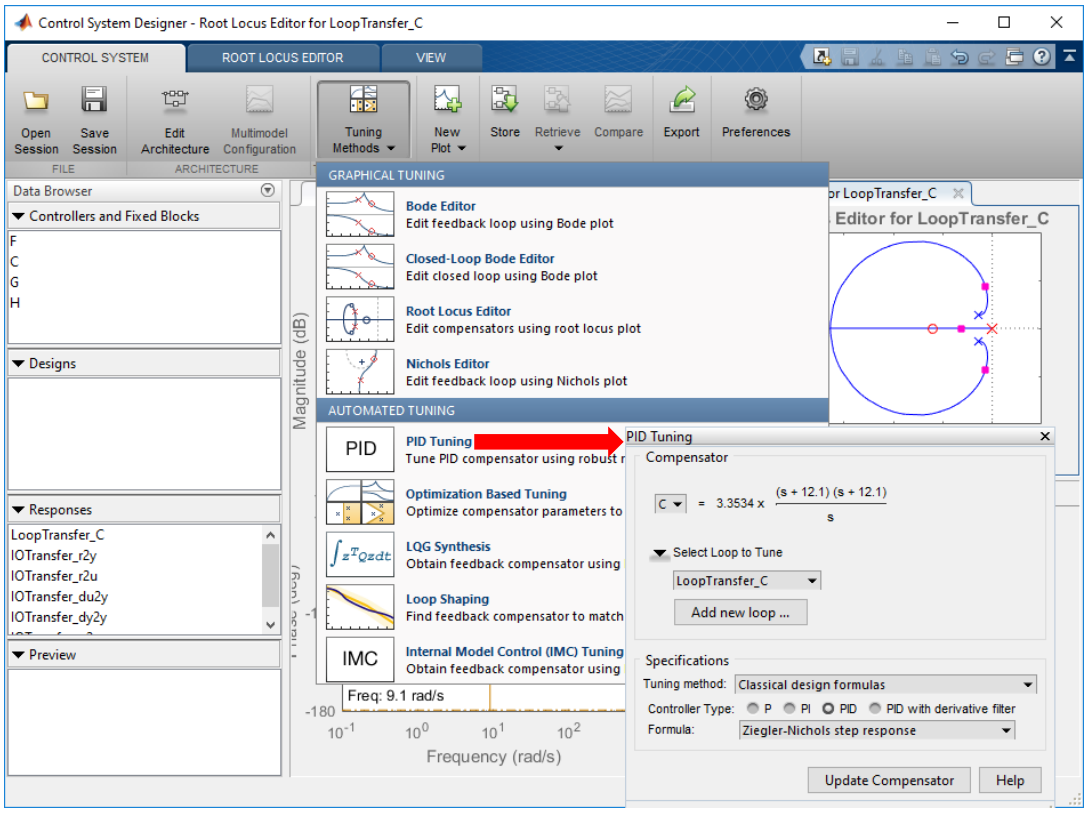


Figure 6: The PID tuner in Control System Designer.

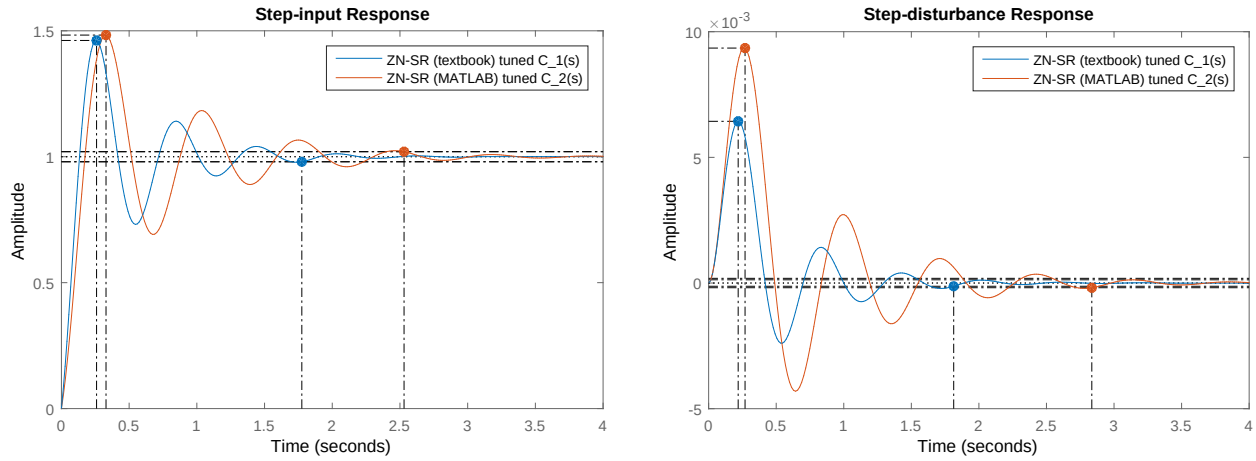


Figure 7: Step-input responses and step-disturbance responses for comparison in Example 1.

The procedure of the method is as follows:

1. Set proportional gain to low, and both integral gain and derivative gain to zero.

2. Increase the proportional gain until the system reaches the boundary of instability; this is when a constant-amplitude limit cycle occurs.
   - The corresponding proportional gain is called the *ultimate gain* or *critical gain*, denoted K_u.
   - The corresponding period of oscillation is called the *ultimate period* or *critical period*,

denoted T_u.

If K_{gm} is the plant's *gain margin* (see next lecture), and ω_{gm} is the *gain margin frequency* (also see next lecture), then $K_u = K_{gm}$, and $T_u = 2\pi/\omega_{gm}$.

3. Calculate the controller gains based on the formulas in Table 2.

Table 2: Formulas for the Ziegler-Nichols frequency response method [SMY14, p. 128], [DB11, Table 7.7], [Oga10, Table 8-2].

Controller type	K_p	T_i	T_d
P	$0.5K_u$		
PI	$0.45K_u$	$T_u/1.2$	
PID	$0.6K_u$	$T_u/2$	$T_u/8$

A PID controller tuned using the frequency response method has the transfer function

$$C(s) = 0.6K_u\left(1 + \frac{2}{T_u s} + \frac{T_u s}{8}\right) = \frac{0.6K_u}{8T_u}\left(\frac{8T_u s + 16 + T_u^2 s^2}{s}\right) = \frac{0.075K_u}{T_u}\frac{(T_u s + 4)^2}{s},$$

$$\therefore C(s) = 0.075K_u T_u \frac{\left(s + 4/T_u\right)^2}{s}. \tag{5}$$

Thus, the frequency response method places double zeros at $-4/T_u$, allowing the controller to be implemented in the series form.

Example 2

Consider the non-self-regulating plant

$$G(s) = \frac{1}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)}, \quad \text{where } \zeta = 1/\sqrt{2}, \omega_n = 4.$$

The ultimate gain and ultimate period are found using the MATLAB code below to be $K_u = 90.5097$ and $T_u = 1.5708$.

```
G = tf(1, [1 2*zeta*omegan omegan^2 0]);
[Ku, ~, Wgm, ~] = margin(G);
Tu = 2*pi/Wgm;
```

(a) Apply the Ziegler-Nichols frequency response method to tune a PID controller. Denote this controller by $C_1(s)$.

(b) Use Control System Designer's PID tuner and its Ziegler-Nichols frequency response rule to tune a PID controller. Denote this controller by $C_2(s)$.

(c) Apply the Ziegler-Nichols step response method to tune a PID controller. Denote this controller by $C_3(s)$.

(d) Compare $C_1(s)$, $C_2(s)$ and $C_3(s)$ in terms of their step-input response and step-disturbance response.

Solution:

(a) Plugging $K_u = 90.5097$ and $T_u = 1.5708$ into Eq. (5), we get the PID transfer function

$$C_1(s) = 0.075 K_u T_u \frac{(s + 4/T_u)^2}{s} = \frac{10.6630(s + 2.546)^2}{s}.$$

At the ultimate gain, we can clearly see from Figure 8 that the uncompensated system is indeed in a state of sustained oscillation.

(b) Following the method in Example 1(b) (except this time we choose "Ziegler-Nichols frequency response" rather than "Ziegler-Nichols step response"), we can obtain the PID transfer function as

$$C_2(s) = \frac{10.6630(s^2 + 5.09s + 6.48)}{s},$$

which is very close to $C_1(s)$.

(c) Using Listing 1, we can fit the unit step response of $G(s)$ with $R = 0.0650$ and $L = 0.3536$ (see Figure 9). Plugging these values into Eq. (4), we get the PID transfer function

$$C_3(s) = \frac{9.6(s^2 + 5.6569s + 8)}{s}.$$

(d) Following the method in Example 2(c), we can obtain the plots in Figure 10. $C_1(s)$ and $C_2(s)$ are so close that the corresponding responses overlap in the plots. The plots show that $C_1(s)$, tuned by the frequency response method, provides better transient response and disturbance rejection than $C_3(s)$, which is tuned by the step response method.

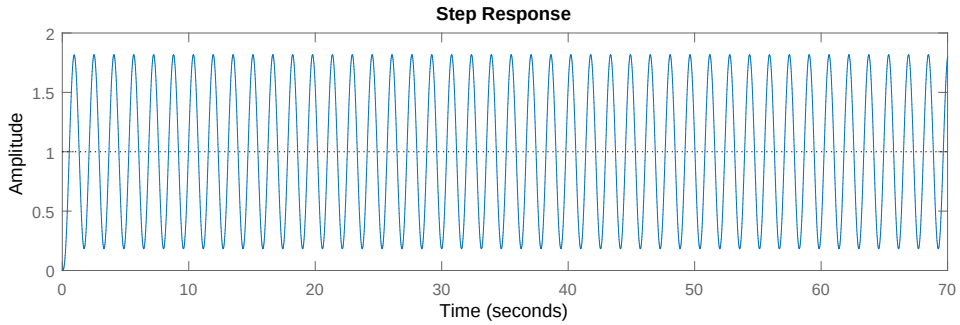


Figure 8: The sustained oscillation of the uncompensated system $G(s)$ in Example 2 when the proportional gain is set to the ultimate gain.

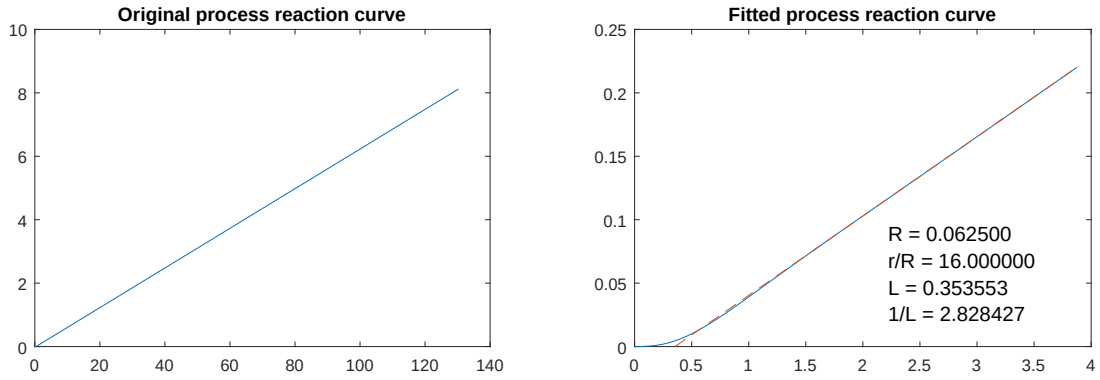


Figure 9: Process reaction curves for Example 2.

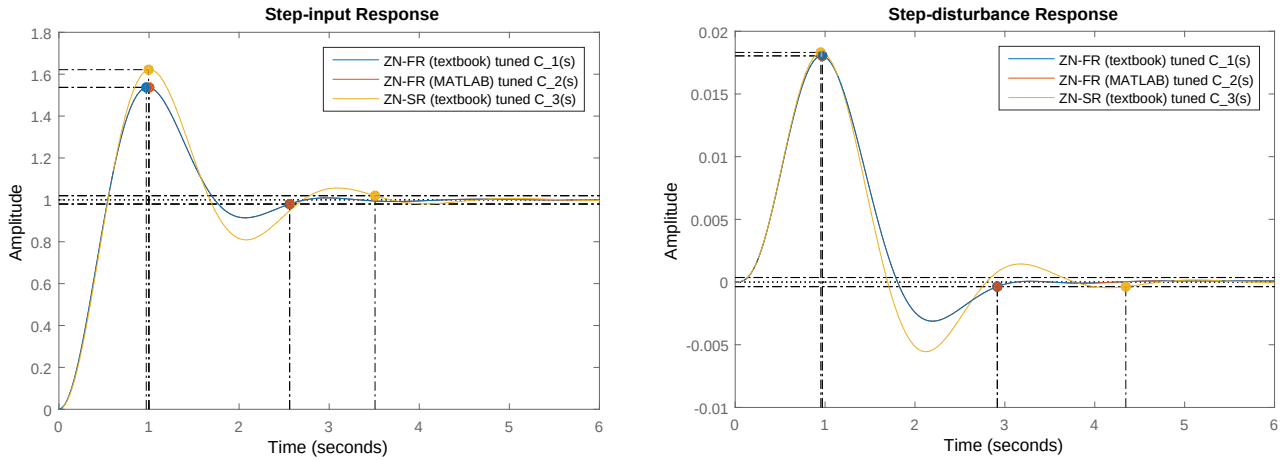


Figure 10: Step-input responses and step-disturbance responses for comparison in Example 2.

3 Comparison

Essentially, the step response method is based on building a parametric model of the plant, whereas the frequency response method is nonparametric. The step response and frequency response methods have their pros and cons:

Step response method [SMEDI10, p. 227]

- Care must be taken to make sure the set-point change is not so large that it sends the process into saturation or any nonlinear region, and yet not so small that the response is overwhelmed by noise. Other than this, the step response method is robust — few things can go wrong.
- Less time-consuming and less disruptive to implement than the frequency response method.
- Applicable to open-loop unstable processes.

Frequency response method [SMEDI10, p. 224]

- In many applications, placing a plant in its continuous cycling state (stability limit) is hazardous — think runaway chemical reactions.
- Can be time-consuming for processes with slow dynamics.
- Not applicable to open-loop unstable processes.

Both methods tend to produce oscillatory responses and large overshoots for set-point changes.

More conservative methods such as the Tyreus-Luyben may be preferable [SMEDI10, p. 225], [Yu06, p. 18]. The Tyreus-Luyben method is similar to the Ziegler-Nichols frequency response method, except that

- It is not applicable to P controllers.
- It trades off disturbance rejection and settling time for higher damping and robustness [ÅH06, Sect. 2.7]. Recall the concept of robustness in Lecture 3.

Table 3: Formulas for the Tyreus-Luyben method [ÅH06, Table 2.5], [SMEDI10, Table 12.4], [Yu06, Table 2.2].

Controller type	K_p	T_i	T_d
PI	$K_u/3.2$	$2.2T_u$	
PID	$K_u/2.2$	$2.2T_u$	$T_u/6.3$

The procedure for the Tyreus-Luyben method is the same as that for the Ziegler-Nichols frequency response method, with the exception being the formulas in Table 3 are to be used instead. A PID controller tuned using the Tyreus-Luyben method has the transfer function

$$C(s) = \frac{K_u}{2.2}\left(1 + \frac{1}{2.2T_u s} + \frac{T_u s}{6.3}\right) = \frac{K_u}{2.2}\left(\frac{6.3 \cdot 2.2T_u s + 6.3 + 2.2T_u^2 s^2}{6.3 \cdot 2.2T_u s}\right),$$

$$\therefore C(s) = \frac{K_u T_u}{13.86}\frac{(s + 0.4931/T_u)(s + 5.8070/T_u)}{s}. \tag{6}$$

Example 3

For the plant in Example 2, repeated here for convenience:

$$G(s) = \frac{1}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)}, \quad \text{where } \zeta = 1/\sqrt{2}, \omega_n = 4,$$

tune a PID controller using the Tyreus-Luyben method, and compare the controller with the controller $C_1(s)$ from Example 2 in terms of their step-input response and step-disturbance response.

Solution: Plugging the values $K_u = 90.5097$ and $T_u = 1.5708$ from Example 2 into Eq. (6), we obtain the PID transfer function

$$C_2(s) = \frac{K_u T_u}{13.86}\frac{(s + 0.4931/T_u)(s + 5.8070/T_u)}{s} = \frac{10.258(s + 3.697)(s + 0.3139)}{s}.$$

The plots in Figure 11 show that the Tyreus-Luyben version of the controller, $C_2(s)$, is worse in terms of disturbance rejection and settling time, but better in terms of lower overshoot than the Ziegler-Nichols version, $C_1(s)$.

Tuning a PID controller does not need to stop at Ziegler-Nichols, Tyreus-Luyben or any other tuning rule. Software like MATLAB provides easy-to-use graphical tools for PID tuning. Revisiting Example 2, if `G` implements the plant, and `C` implements the controller tuned by the Ziegler-Nichols frequency response method, then the MATLAB command

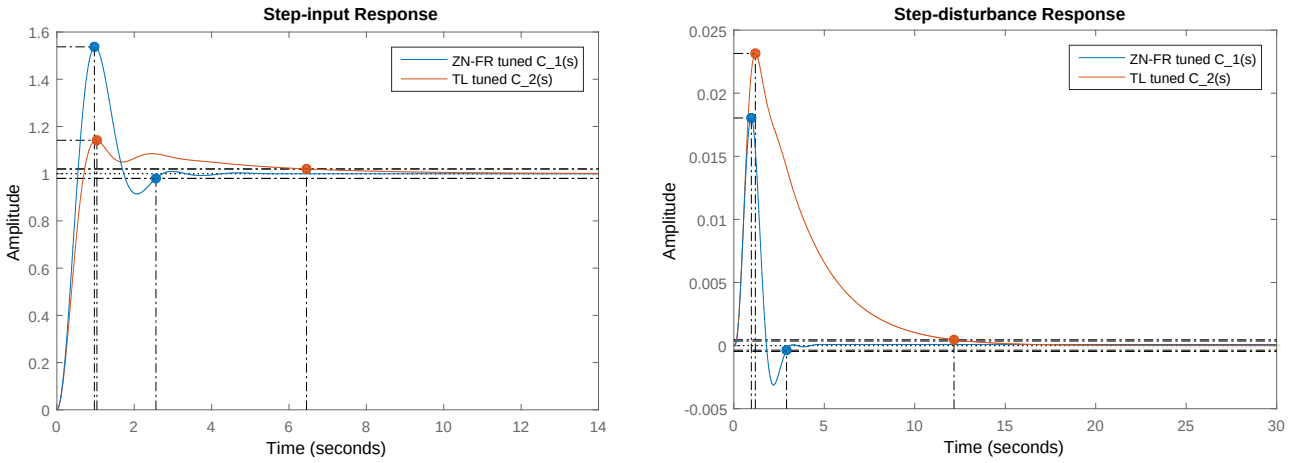


Figure 11: Step-input responses and step-disturbance responses for comparison in Example 3.

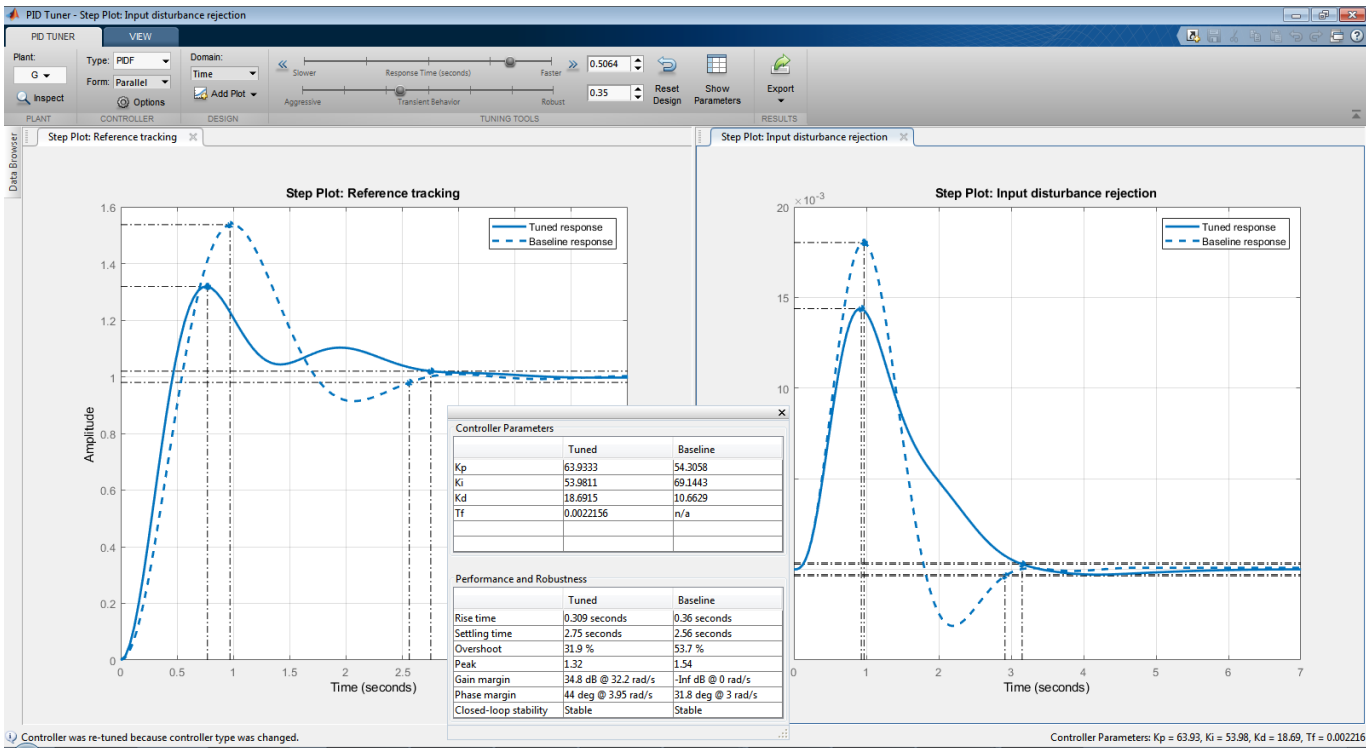


Figure 12: The GUI of `pidTuner` displaying plots for reference tracking and disturbance rejection. Above, G is the plant in Example 2, and the baseline controller is the Ziegler-Nichols frequency-response PID controller in Example 2. We can see the hand-tuned PIDF controller outperforms the baseline PID controller. For more information about `pidTuner`, on a MATLAB Command Window, type "`web(fullfile(docroot, 'control/getstart/tune-pid-controller-to-balance-tracking-and-disturbance-rejection.html'))`".

```
pidTuner(G, C)
```

invokes `pidTuner` with `C` as the baseline controller. The PID Tuner allows us to tune the controller to get better damping and disturbance rejection without sacrificing settling time (see Figure 12). The important options include

- the controller type,

- the response time (a faster response comes at the price of other performance variables),
- the transient behavior (choose more "aggressive" for better disturbance rejection, but more "robust" for robustness).

Nevertheless, neither Control System Designer's PID tuner nor `pidTuner` is the be-all end-all PID tuning tool. For the plant

$$G(s) = \frac{s+6}{(s+1)(s+3)(s^2+4)},$$

both tools fail to generate a stabilizing PID controller. This can be surprising considering how we have successfully used root locus to design a stabilizing PID controller for $G(s)$ in the previous lecture. Repeating the message from the previous lecture: where tuning rules and automated tuners fall short, we always have the basics to turn to, which in this case is root locus.

References

[ÅH04] KJ Åström and T Hägglund. Revisiting the Ziegler–Nichols step response method for PID control. *Journal of Process Control*, 14(6):635–650, 2004.

[ÅH06] Karl Johan Åström and Tore Hägglund. *Advanced PID control*. ISA-Instrumentation, Systems, and Automation Society, 2006.

[DB11] R.C. Dorf and R.H. Bishop. *Modern Control Systems*. Pearson, 12th edition, 2011.

[O'D09] Aidan O'Dwyer. *Handbook of PI and PID controller tuning rules*, volume 57. World Scientific, 2009.

[Oga10] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 5th edition, 2010.

[SC97] Carlos A Smith and Armando B Corripio. *Principles and practice of automatic process control*. Wiley, 2nd edition, 1997.

[SMEDI10] Dale E Seborg, Duncan A Mellichamp, Thomas F Edgar, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 3rd edition, 2010.

[SMY14] William Y. Svrcek, Donald P. Mahoney, and Brent R. Young. *A Real-Time Approach to Process Control*. Wiley, 2014.

[Vis06] Antonio Visioli. *Practical PID control*. Springer-Verlag London Limited, 2006.

[VZ10] Antonio Visioli and Qingchang Zhong. *Control of integral processes with dead time*. Springer Science & Business Media, 2010.

[XCA07] D. Xue, Y. Chen, and D.P. Atherton. *Linear Feedback Control: Analysis and Design with MATLAB*. SIAM, 2007.

[Yu06] Cheng-Ching Yu. *Autotuning of PID controllers: A relay feedback approach*. Springer Science & Business Media, 2006.