PID Controller

1. Introduction

A **proportional-integral-derivative controller** (**PID controller**) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an *error* value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the *error* by adjusting the process through use of a manipulated variable.

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called **three-term control**: the proportional, the integral and derivative values, denoted *P*, *I*, and *D*. Simply put, these values can be interpreted in terms of time: ***P* depends on the *present* error, *I* on the accumulation of *past* errors, and *D* is a prediction of *future* errors**, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve, a damper, or the power supplied to a heating element.

In the absence of knowledge of the underlying process, a PID controller has historically been considered to be the most useful controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint, and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two actions to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value due to the control action.

2. PID Controller Theory

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output,

Controller manufacturers arrange the Proportional, Integral and Derivative modes into three different controller algorithms or controller structures. These are called

Interactive, Noninteractive, and Parallel algorithms. Some controller manufacturers allow you to choose between different controller algorithms as a configuration option in the controller software. The PID Algorithms are:

1) Interactive Algorithm

$$u(t) = K_c\left[e(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau\right] \times \left[1 + T_d\frac{d}{dt}e(t)\right]$$

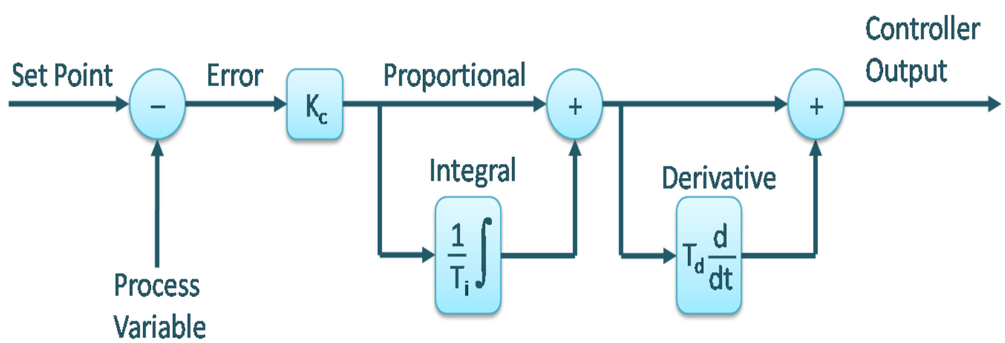


Figure 1: Interactive Algorithm

2) NonInteractive Algorithm

$$u(t) = K_c\left[e(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau + T_d\frac{d}{dt}e(t)\right]$$

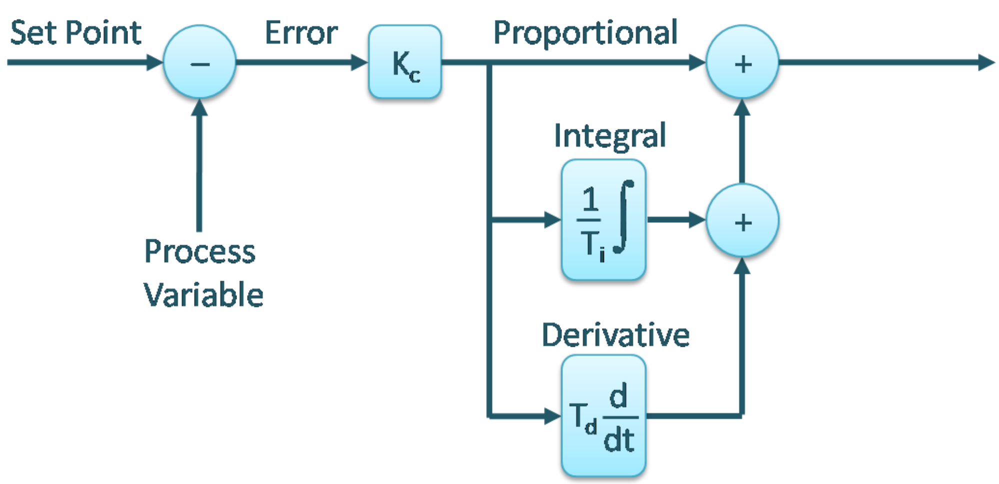


Figure 2: NonIneractive Algorithm

3) Parallel Algorithm

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

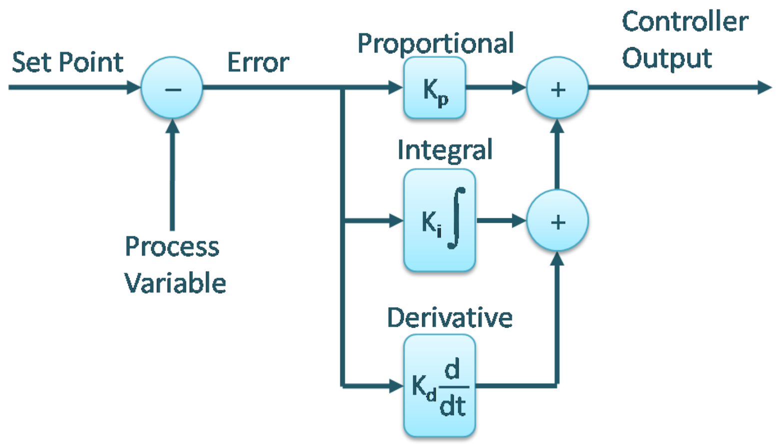


Figure 3: Parallel Algorithm

Where

$K_p = K_c$: *Propotional Gain*

$K_i = \dfrac{K_c}{T_i}$: *Integral Gain*

$K_d = K_c T_d$: *Derivative Gain*

$e(t) = r(t) - y(t)$

2.1 <u>Proportional Term</u>

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant. The proportional term is given by:

$$P_{out} = K_p e(t)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change

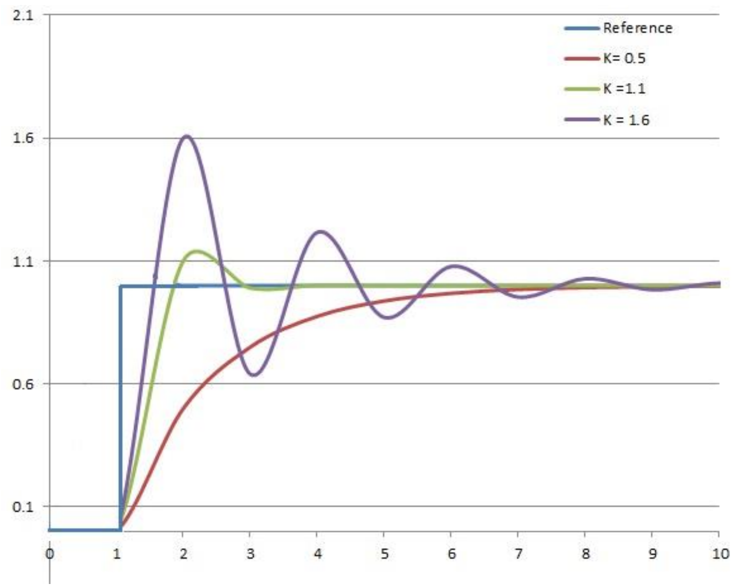


Figure 4: The effect of add K_p ($K_i, and\ K_d$) held constant

2.2 Integral Term

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain $\boldsymbol{K_i}$ and added to the controller output.

$$I_{Out} = K_i \int_0^t e(\tau) d\tau$$

The integral term accelerates the movement of the process towards set-point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set-point value.

2.3 Derivative Term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_{d}. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_{d}. The derivative term is given by

$$D_{out} = K_d \frac{d}{dt} e(t)$$

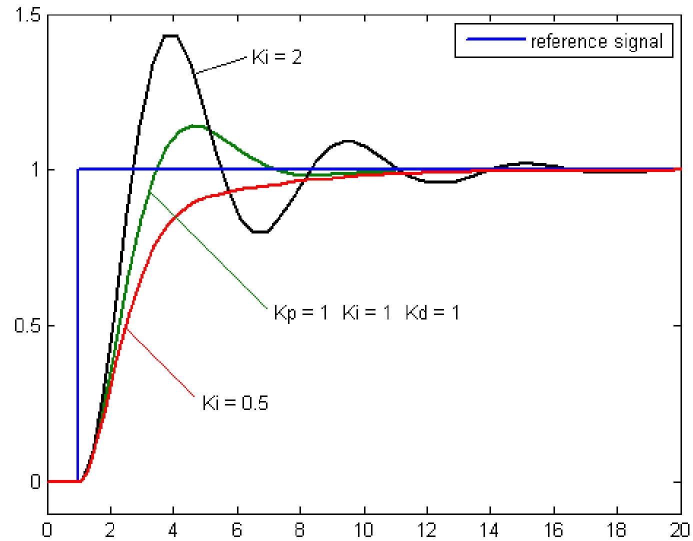


Figure 5: The effect of add K_i ($K_\mathrm{p}, and\ K_\mathrm{d}$) held constant

Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise. Derivative action is seldom used in practice though - by one estimate in only 20% of deployed controllers- because of its variable impact on system stability in real-world applications.

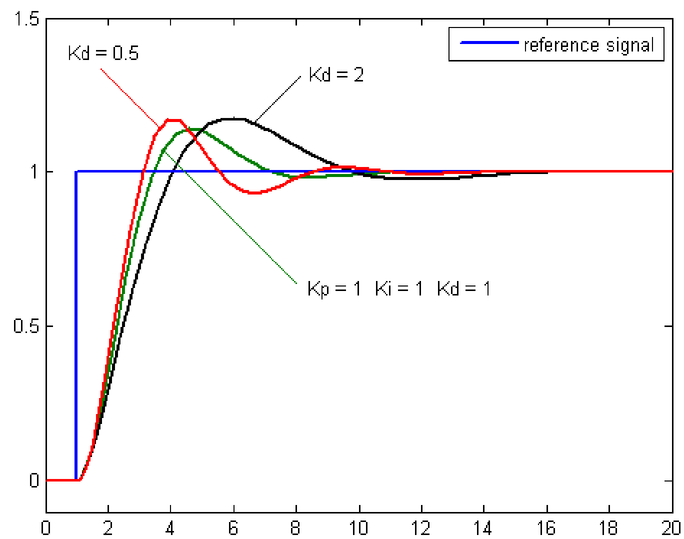


Figure 6 The effect of add K_d ($K_\mathrm{p}, and\ K_\mathrm{i}$) held constant

Table 1: Effect of increasing parameter independently

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
K_p	Decrease	Increase	Small Change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor Change	Decrease	Decrease	No Effect	Improve if K_d small

3. Overview of Methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and on the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Table 2: Choosing a Tuning Method

Method	Advantages	Disadvantages
Manual Tuning	No math required , Online	Requires experienced personnel
Ziegler-Nichols	Proven Method, Online	Process upset, some trial-and-error, very aggressive tuning
Cohen-Coon	Good process models	Some math; offline; only good for first-order processes
Software Tools	Consistent tuning; online or offline - can employ computer-automated control system design (CAutoD) techniques;	Some cost or training involved

4. Open Loop Method

In these methods, the PID is being tuned in open loop, isolated from the process plant. First a step input is applied to the plant and the process reaction curve is obtained. Using the process reaction curve with one of the First Order Plus Dead Time (FOPDT) estimation methods an approximation of the process is calculated. Knowing K_m, τ_m and t_d the PID parameters can be evaluated from the related correlations according to the method used.

First Order Plus Dead Time (FOPDT) is given by

$$G(s) = \frac{K_m}{\tau_m s + 1} e^{-t_d s}$$

4.1 Ziegler-Nichols Open Loop Method

In the 1940's, Ziegler and Nichols devised two empirical methods for obtaining controller parameters. Their methods were used for first order plus dead time situations, and involved intense manual calculations. With improved optimization software, most manual methods such as these are no longer used. However, even with computer aids, the following two methods are still employed today, and are considered among the most common.

This method remains a popular technique for tuning controllers that use proportional, integral, and derivative actions. The Ziegler-Nichols open-loop method is also referred to as S-shaped curve method, because it tests the open-loop reaction of the process to a change in the control variable output. This basic test requires that the response of the system be recorded, preferably by a plotter or computer. Once certain process response values are found, they can be plugged into the Ziegler-Nichols equation with specific multiplier constants for the gains of a controller with either P, PI, or PID actions.

In this method, we obtain experimentally the open loop response of the FOPDT to a unit step input. This method only applied if the response to a step input exhibits an *s*-shaped curve as shown in figure 7. This means that if the plant involves integrators (like 2^{nd} order prototypes system) or complex-conjugate poles (general 2^{nd} order system), then this method can't be applied since *s*-shaped will not be obtained.

This method remains a popular technique for tuning controllers that use proportional, integral, and derivative actions. The Ziegler-Nichols open-loop method is also referred to as a process reaction method, because it tests the open-loop reaction of the process to a change in the control variable output.

The Tuning Procedure:

To use the Ziegler-Nichols open-loop tuning method, you must perform the following steps:

1. Make an open loop step test

2. From the process reaction curve determine the transportation lag or dead time, t_d, the time constant or time for the response to change, τ_m, and the ultimate value that the response reaches at steady-state, K_m, for a step change of X_o.
3. Determine the loop tuning constants. Plug in the reaction rate and lag time values to the Ziegler-Nichols open-loop tuning equations for the appropriate controller (P, PI, or PID) to calculate the controller constants. Use the table 3.

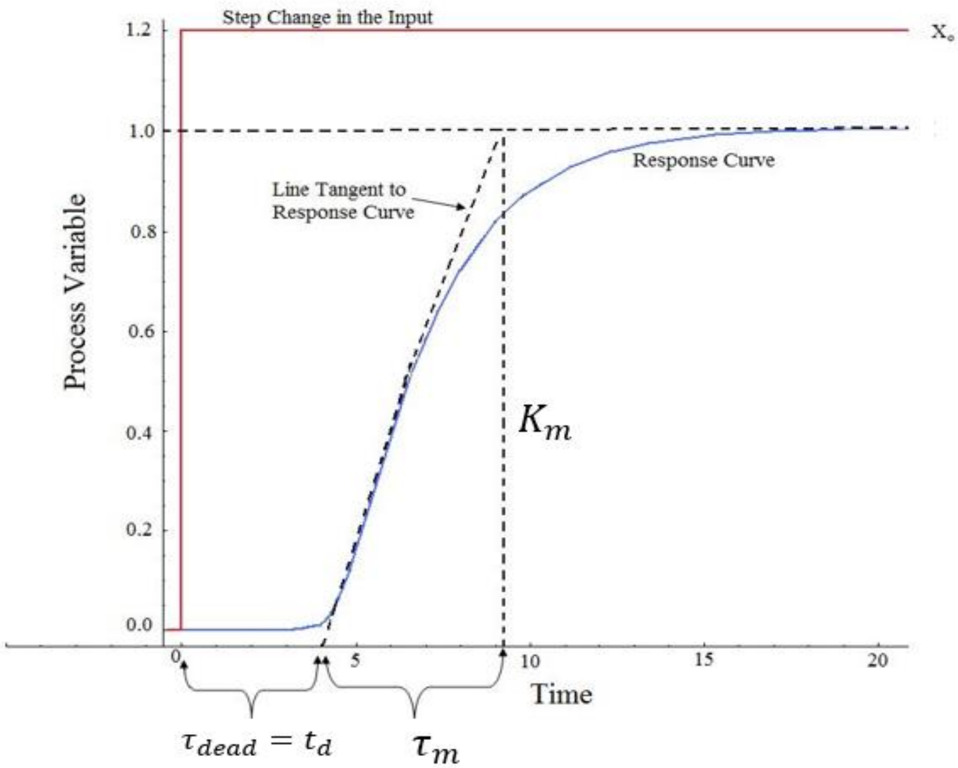


Figure 3: Open Loop of First order system plus dead Time (s-shaped curve)

Table 3: Open-loop Calculation of $(K_p.T_i.T_d)$

	K_p	T_i	T_d
P- Controller	$\frac{X_o}{K_m}\frac{\tau_m}{t_d}$	∞	0
PI- Controller	$0.9\,\frac{X_o}{K_m}\frac{\tau_m}{t_d}$	$3.3\,t_d$	0
PID- Controller	$1.2\,\frac{X_o}{K_m}\frac{\tau_m}{t_d}$	$2\,t_d$	$0.5\,t_d$

The PID controller tuned by this method gives (according to the formula shown in Table (8.4)).

.

$$G_c(s) = K_p\left[1 + \frac{1}{T_i s} + T_d s\right]$$
$$= 1.2\frac{X_o}{K_m}\frac{\tau_m}{t_d}\left(1 + \frac{1}{2t_d s} + 0.5t_d s\right)$$
$$= 0.6\tau_m\frac{(s + 1/t_d)^2}{s}$$

- This mean that the controller adds double zero at $s = -\frac{1}{t_d}$, and pole at origin
- Advantages Ziegler-Nichols Open Loop Tuning Methods
  1. Quick and easier to use than other methods
  2. It is a robust and popular method

- Disadvantages Ziegler-Nichols Open Loop Tuning Methods
  1. It depends upon purely t_d to estimate I and D controllers.
  2. Approximations for the K_p. T_i, and T_d values might not be entirely accurate for different systems.
  3. It does not hold for I, D and PD controllers

4.2 Cohen-Coon Method

The Cohen-Coon tuning rules are suited to a wider variety of processes than the Ziegler-Nichols tuning rules. The Ziegler-Nichols rules work well only on processes where the dead time is less than half the length of the time constant. The Cohen-Coon tuning rules work well on processes where the dead time is less than two times the length of the time constant (and you can stretch this even further if required). Also it provides one of the few sets of tuning rules that has rules for PD controllers.

Like the Ziegler-Nichols tuning rules, the Cohen-Coon rules aim for a quarter-amplitude damping response. Although quarter-amplitude damping-type of tuning provides very fast disturbance rejection, it tends to be very oscillatory and frequently interacts with similarly-tuned loops. Quarter-amplitude damping-type tuning also leaves the loop vulnerable to going unstable if the process gain or dead time doubles in value.

In this method the process response curve is obtained first, by an open loop test as shown in figure8 and then the process dynamics is approximated by a first order plus dead time model, with following parameters:

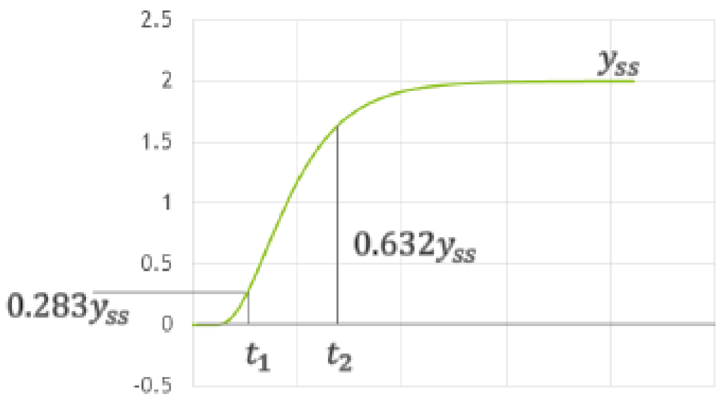


Figure 7: The open loop response of plant

$$\tau_m = \frac{3}{2}(t_2 - t_1)$$

$$t_d = t_2 - \tau_m$$

Again the particular rules for this method are used to calculate the PID parameters. They are listed in table 4

Table 3: the parameter of Cohen-Coon Method

Controller	K_c	T_i	T_d
P	$\frac{\tau_m}{Kt_d}\left(1 + \frac{t_d}{3\tau_m}\right)$	-	-
PI	$\frac{\tau_m}{Kt_d}\left(0.9 + \frac{t_d}{12\tau_m}\right)$	$t_d\left(\frac{30 + \frac{3t_d}{\tau_m}}{9 + \frac{20t_d}{\tau_m}}\right)$	-
PD	$\frac{\tau_m}{Kt_d}\left(1.25 + \frac{t_d}{6\tau_m}\right)$	-	$t_d\left(\frac{6 - \frac{2t_d}{\tau_m}}{22 + \frac{3t_d}{\tau_m}}\right)$
PID	$\frac{\tau_m}{Kt_d}\left(1.33 + \frac{t_d}{4\tau_m}\right)$	$t_d\left(\frac{32 + \frac{6t_d}{\tau_m}}{13 + \frac{8t_d}{\tau_m}}\right)$	$t_d\left(\frac{4}{11 + \frac{2t_d}{\tau_m}}\right)$

5. Ziegler-Nichols Closed - Loop Tuning Method

The Ziegler-Nichols closed-loop tuning method allows you to use the critical gain value, K_{cr}, and the critical period of oscillation, P_{cr}, to calculate K_p . It is a simple method of tuning PID controllers and can be refined to give better approximations of the controller. You can obtain the controller constants K_p , T_i , and T_d in a system with feedback. The Ziegler-Nichols closed-loop tuning method is limited to tuning processes that cannot run in an open-loop environment.

Determining the ultimate gain value, K_{cr}. is accomplished by finding the value of the proportional-only gain that causes the control loop to oscillate indefinitely at steady state. This means that the gains from the I and D controller are set to zero so that the influence of P can be determined. It tests the robustness of the K_p value so that it is optimized for the controller. Another important value associated with this proportional-only control tuning method is the critical period (P_{cr}). The ultimate period is the time required to complete one full oscillation while the system is at steady state. These two parameters, K_{cr} and P_{cr}, are used to find the loop-tuning constants of the controller (P, PI, or PID). To find the values of these parameters, and to calculate the tuning constants, use the following procedure:

- The Tuning Procedure:

  1. Remove integral and derivative action. Set integral time (T_i) to ∞ or its largest value and set the derivative controller (T_d) to zero.
  2. Create a small disturbance in the loop by changing the set point. Adjust the proportional, increasing and/or decreasing, the gain until the oscillations have constant amplitude.
  3. Record the gain value (K_{cr}) and period of oscillation (P_{cr}).
  4. Plug these values into the Ziegler-Nichols closed loop equations and determine the necessary settings for the controller.

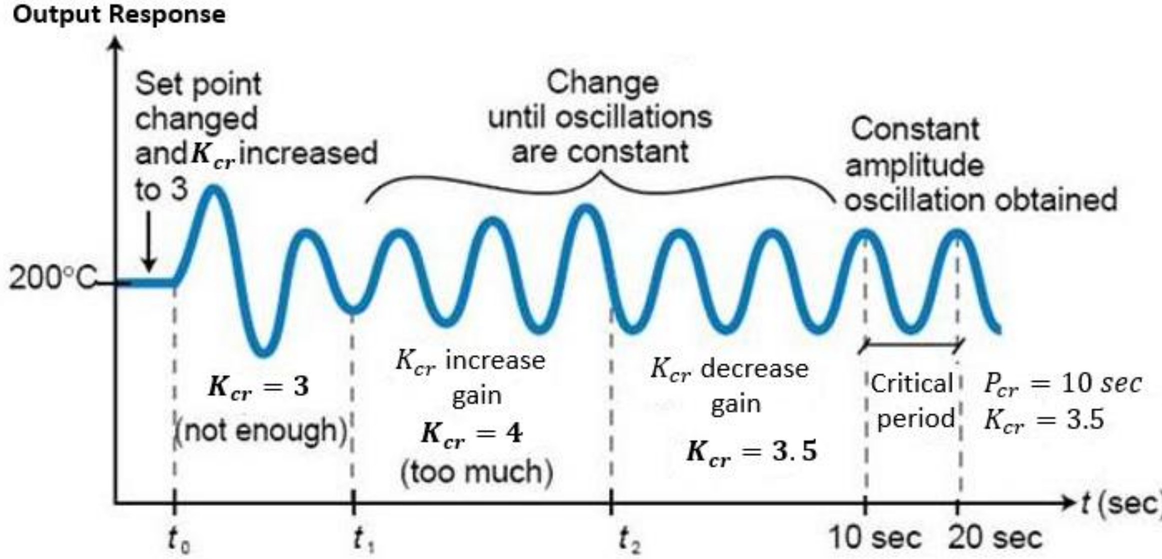


Figure 9: System tuned using the Ziegler-Nichols closed-loop tuning method

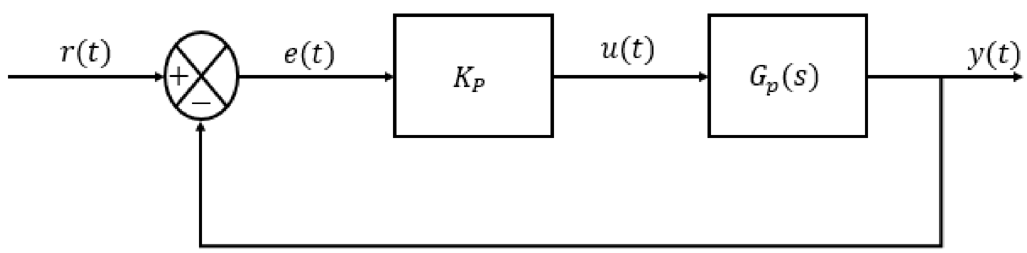


Figure 10: The Control system with Gain K_p

Table 4: Closed-Loop Calculation of $(K_p . T_i . T_d)$

	K_p	T_i	T_d
P- Controller	$\frac{K_{cr}}{2}$	∞	0
PI- Controller	$\frac{K_{cr}}{2.2}$	$\frac{P_{cr}}{1.2}$	0
PID- Controller	$\frac{K_{cr}}{1.7}$	$\frac{P_{cr}}{2}$	$\frac{P_{cr}}{8}$

The PID controller tuned by this method gives (according to the formula shown in Table 8.5).

$$G_c(s) = K_p\left[1+\frac{1}{T_i s}+T_d s\right]$$
$$= 0.6K_{cr}\left(1+\frac{1}{0.5P_{cr}s}+0.125P_{cr}s\right)$$
$$= 0.075K_{cr}P_{cr}\frac{(s+4/P_{cr})^2}{s}$$

Thus the PID Controllers adds a pole at origin and double zeros at $s=-\frac{4}{P_{cr}}$

If the system has a known mathematical model (Transfer function is given), then RL method can be used to find K_{cr} value (critical gain) and the frequency of the sustained oscillations w_{cr}. After that P_{cr} is found from

$$P_{cr} = \frac{2\pi}{w_{cr}}$$

These values can be found from the crossing points of the root locus branches with the *jw* axis. This method doesn't apply if the root locus doesn't cross the *jw* axis.

- Advantages Ziegler-Nichols Closed-Loop Tuning Methods
  1. Easy experiment; only need to change the P controller
  2. Includes dynamics of whole process, which gives a more accurate picture of how the system is behaving
- Disadvantages Ziegler-Nichols Closed-Loop Tuning Methods
  1. Experiment can be time consuming
  2. Can venture into unstable regions while testing the P controller, which could cause the system to become out of control

6. Software Method (PID Tuning Toolbox In MATLAB)

6.1 Automatically Tune PID Controller Gains

PID tuning is the process of finding the values of proportional, integral, and derivative gains of a PID controller to achieve desired performance and meet design requirements.

PID controller tuning appears easy, but finding the set of gains that ensures the best performance of your control system is a complex task. Traditionally, PID controllers are tuned either manually or using rule-based methods. Manual tuning methods are iterative and time-consuming, and if used on hardware, they can cause damage. Rule-based methods also have serious limitations: they do not support certain types of plant models, such as unstable plants, high-order plants, or plants with little or no time delay.

You can automatically tune PID controllers to achieve the optimal system design and to meet design requirements, even for plant models that traditional rule-based methods cannot handle well.

An automated PID tuning workflow involves:

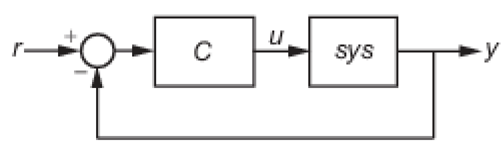
- Identifying plant model from input-output test data
- Modeling PID controllers in MATLAB using PID objects or in Simulink using PID Controller blocks
- Automatically tuning PID controller gains and fine-tune your design interactively
- Tuning multiple controllers in batch mode
- Tuning single-input single-output PID controllers as well as multiloop PID controller architectures

6.2 PID Tuning Toolbox

Can be use the PID tuning toolbox to determine the parameter of controller depend on the system form MATLAB or SIMULINK as following step:

1) MATLAB

Use the PID Tuner to interactively design a SISO PID controller in the feed-forward path of single-loop, unity-feedback control configuration



The PID Tuner automatically designs a controller for your plant. You specify the controller type (P, I, PI, PD, PDF, PID, PIDF) and form (parallel or standard). You can analyze the design using a variety of response plots, and interactively adjust the design to meet your performance requirements.

To launch the PID Tuner, use the pidTuner command:

pidTuner(sys,type)

where ***sys*** is a linear model of the plant you want to control,

type is a string indicating the controller type to design

- **PID Controller Type**

The PID Tuner can tune up to seven types of controllers. To select the controller type, use one of these methods:

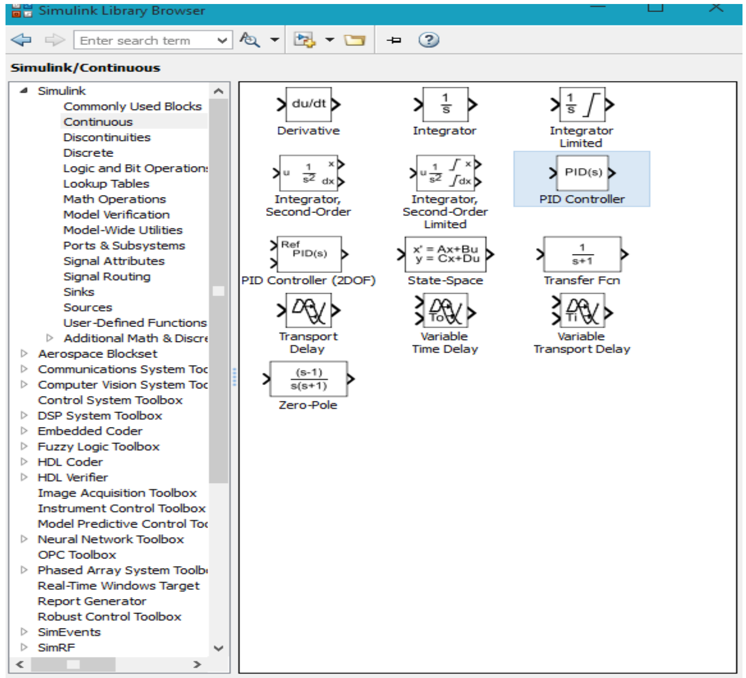
Provide the type argument to the launch command pidTuner.

In PID Tuner, use the Type menu to change controller types.

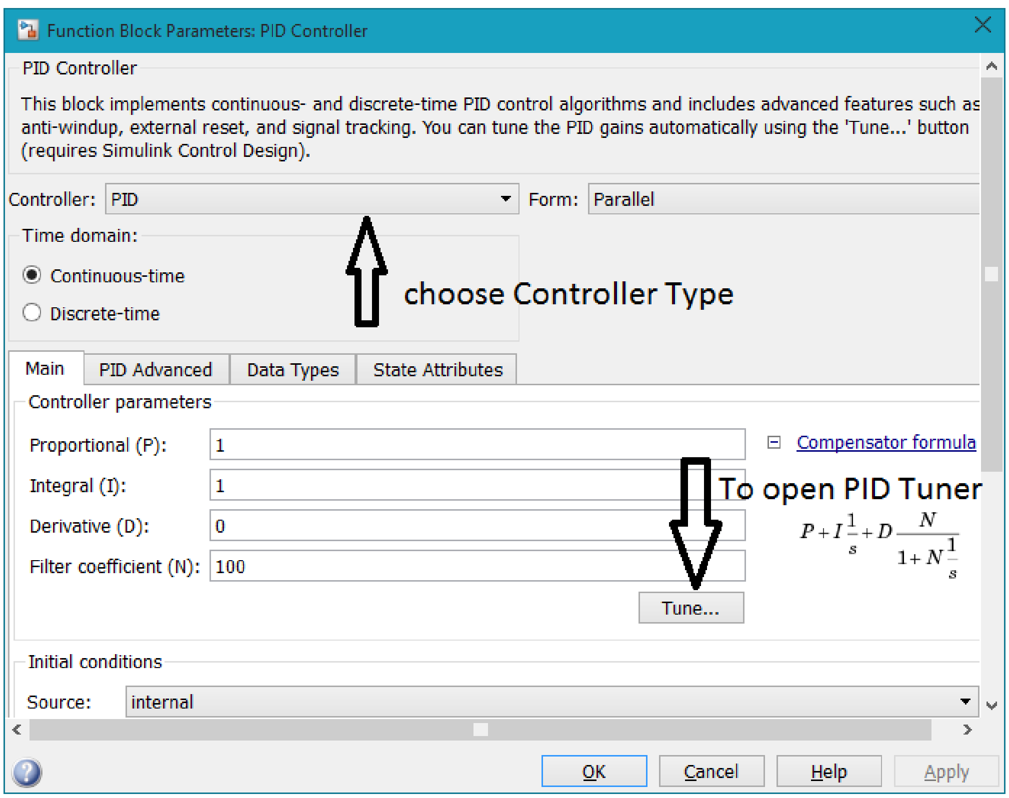
`type` input to `pidTuner`	Entry in Type menu	Controller Type	Continuous-time Controller Formula (parallel form)	Discrete-time Controller Formula (parallel form, `ForwardEuler` integrator formulas)
'p'	P	Proportional only	K_p	K_p
'i'	I	Integral only	$\frac{K_i}{s}$	$K_i \frac{T_s}{z-1}$
'pi'	PI	Proportional and integral	$K_p + \frac{K_i}{s}$	$K_p + K_i \frac{T_s}{z-1}$
'pd'	PD	Proportional and derivative	$K_p + K_d s$	$K_p + K_d \frac{z-1}{T_s}$
'pdf'	PDF	Proportional and derivative with first-order filter on derivative term	$K_p + \frac{K_d s}{T_f s+1}$	$K_p + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$
'pid'	PID	Proportional, integral, and derivative	$K_p + \frac{K_i}{s} + K_d s$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{z-1}{T_s}$
'pidf'	PIDF	Proportional, integral, and derivative with first-order filter on derivative term	$K_p + \frac{K_i}{s} + \frac{K_d s}{T_f s+1}$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$

2) SIMULINK

Select the PID controller block form Simulink Library



Drag the PID controller and place in the Simulink model , and double click on block.



Example 1:

This example shows how to use the PID tuner to design a controller for the plant

$$G(s) = \frac{1}{(s+1)^3}$$

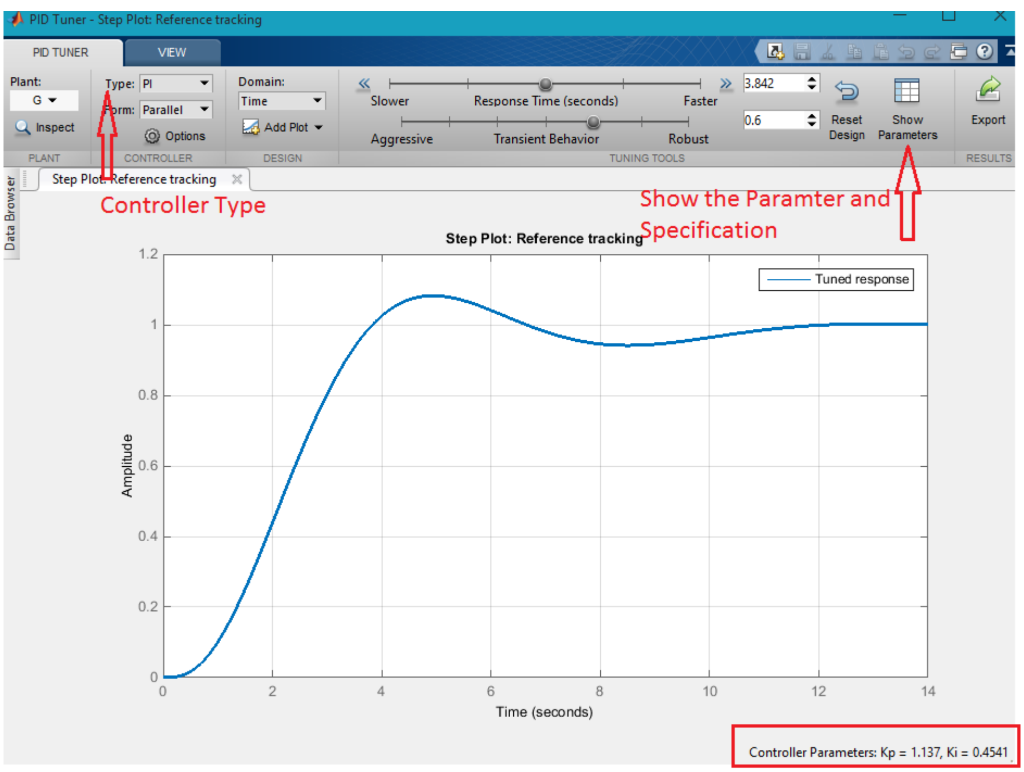
1. Create the plant model and open the PID Tuner to design a PI controller for a first pass design.

```
>> G=zpk([],[-1 -1 -1],1)

G =

       1
   -------
   (s+1)^3

Continuous-time zero/pole/gain model.
```

2.

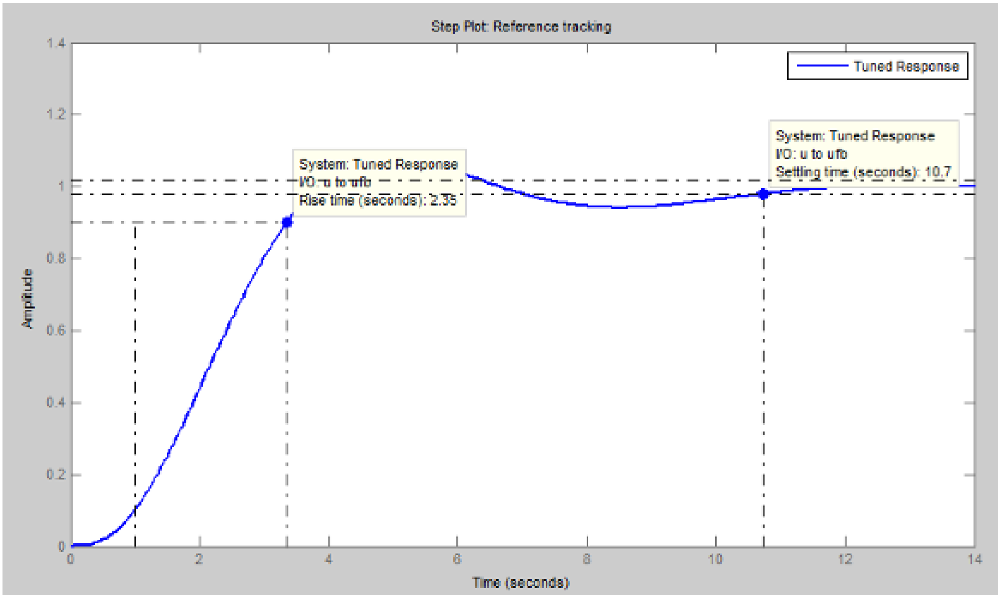
```
Command Window
fx >> pidTuner(G)
```

3. The PIDTuner toolbox

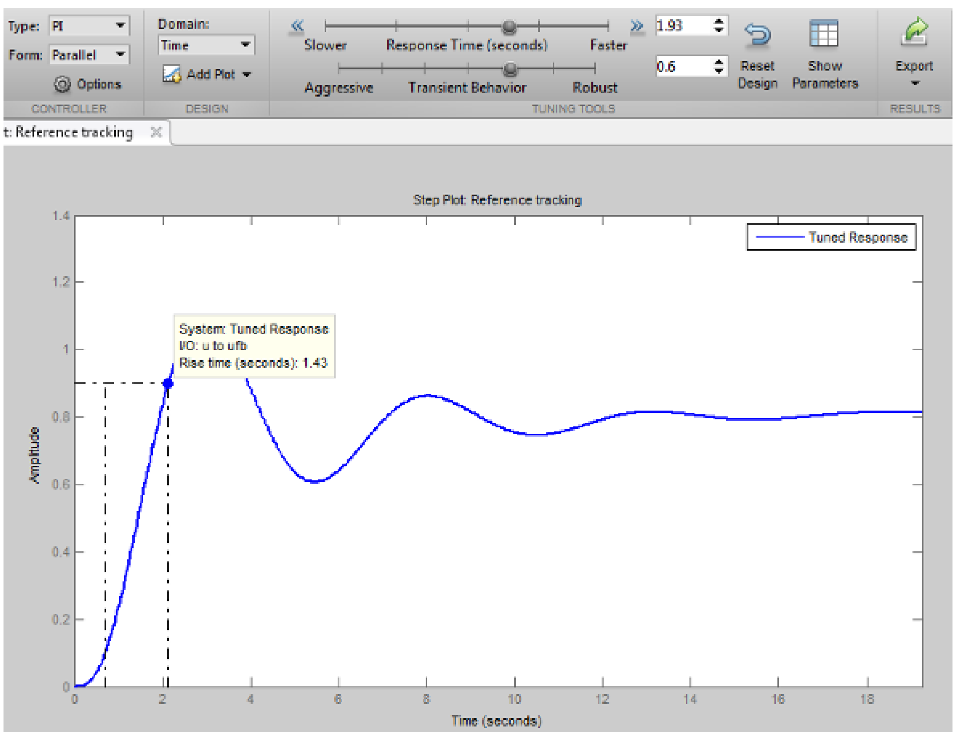


4. Examine the reference tracking rise time and settling time.

Right-click on the plot and select ***Characteristics*** > ***Rise Time*** to mark the rise time as a blue dot on the plot. Select ***Characteristics*** > ***Settling Time to mark the settling time***. To see tool-tips with numerical values, click each of the blue dots.



5. Slide the ***Response time slider*** to the right to try to improve the loop performance. The response plot automatically updates with the new design



Moving the Response time slider far enough to meet the rise time requirement of less than 1.5 s results in more oscillation. Additionally, the parameters display shows that the new response has an unacceptably long settling time.

Controller parameters

	Tuned
Kp	2.9965
Ki	0.054468
Kd	
Tf	

Performance and robustness

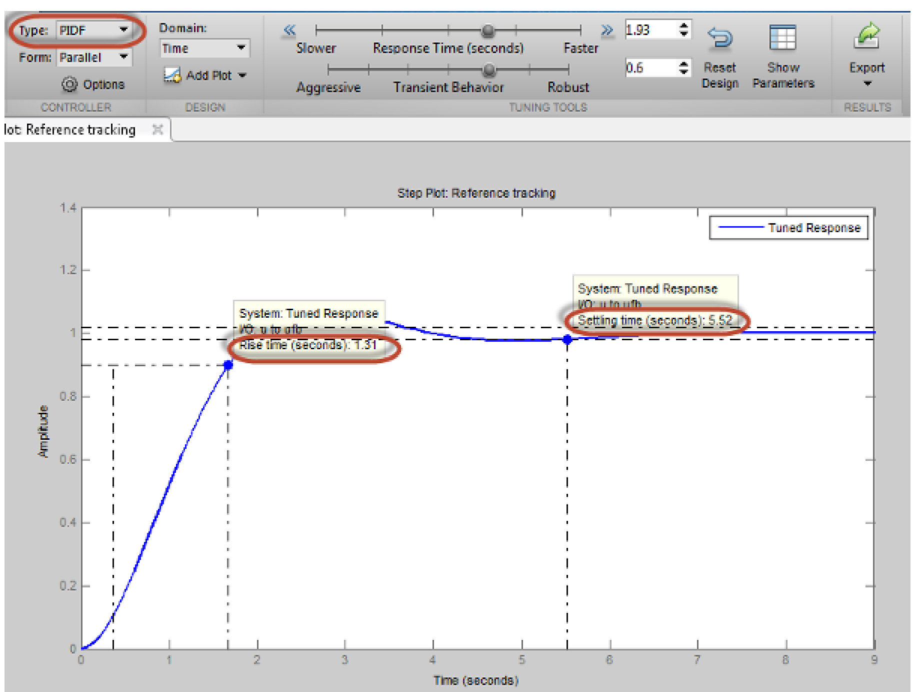
	Tuned
Rise time	1.43 seconds
Settling time	182 seconds
Overshoot	9.42 %
Peak	1.09
Gain margin	8.37 dB @ 1.72 rad/s
Phase margin	40.7 deg @ 1.04 rad/s
Closed-loop stability	Stable

To achieve the faster response speed, the algorithm must sacrifice stability.

6. Change the controller type to improve the response.

Adding derivative action to the controller gives the PID Tuner more freedom to achieve adequate phase margin with the desired response speed.

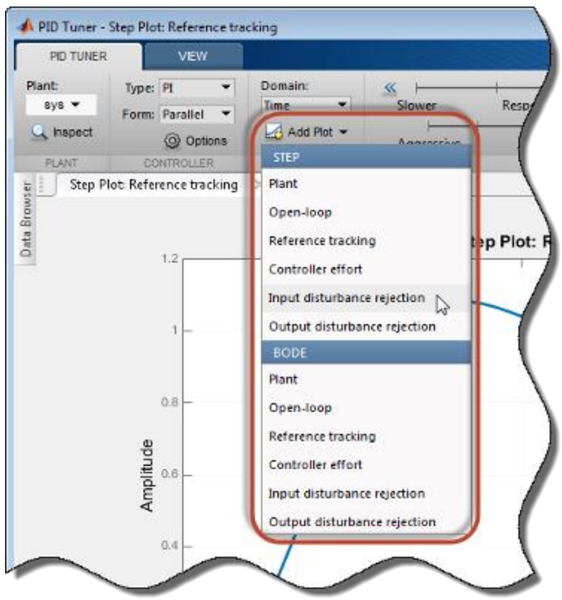
In the Type menu, select PIDF. The PID Tuner designs a new PIDF controller



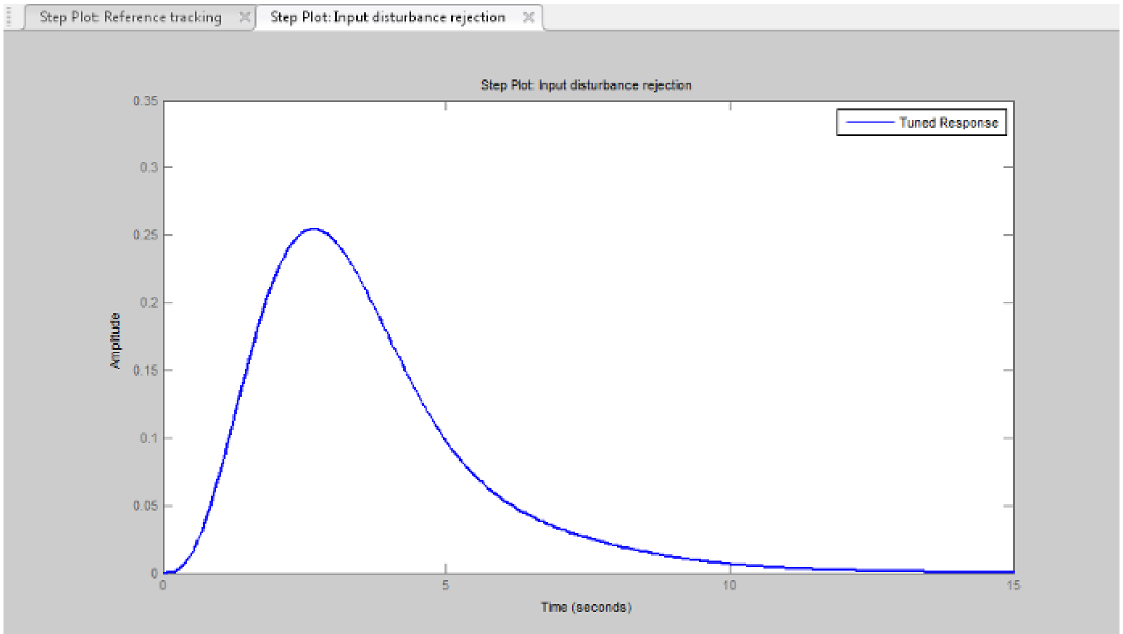
The rise time and settling time now meet the design requirements. You can use the Response time slider to make further adjustments to the response. To revert to the default automated tuning result, click Reset Design.

7. Analyze other system responses, if appropriate.

To analyze other system responses, click Add Plot. Select the system response you want to analyze.



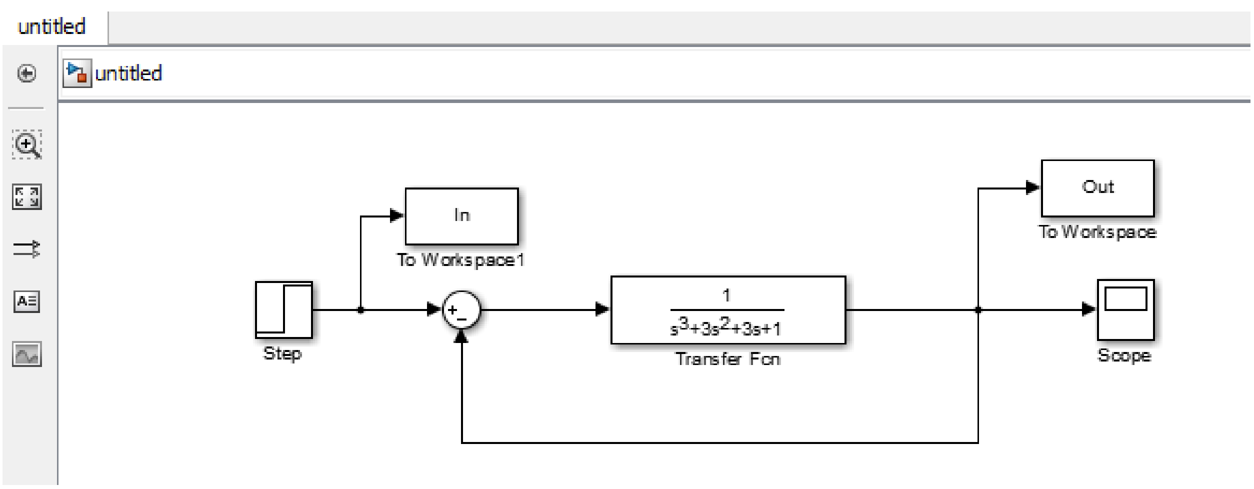
For example, to observe the closed-loop step response to disturbance at the plant input, in the Step section of the **Add Plot** menu, select Input disturbance rejection. The disturbance rejection response appears in a new figure.

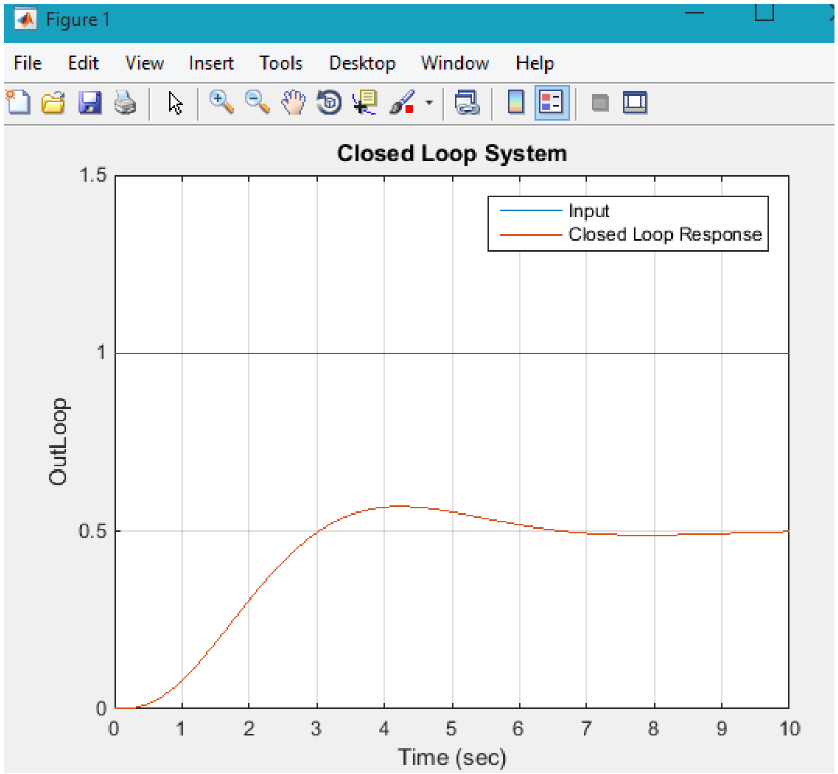


Example 2:

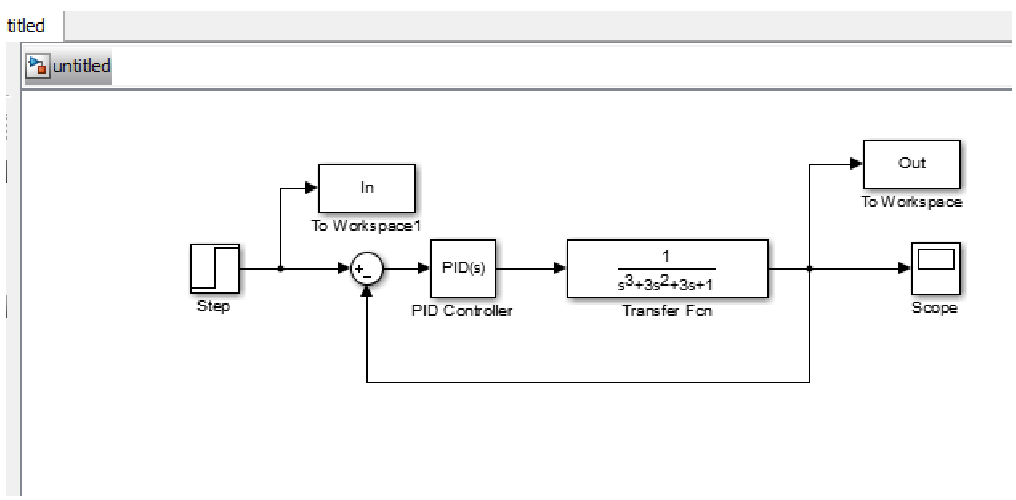
Assume the system in example 1 Bu

1) Build the Simulink block represent the closed loop system

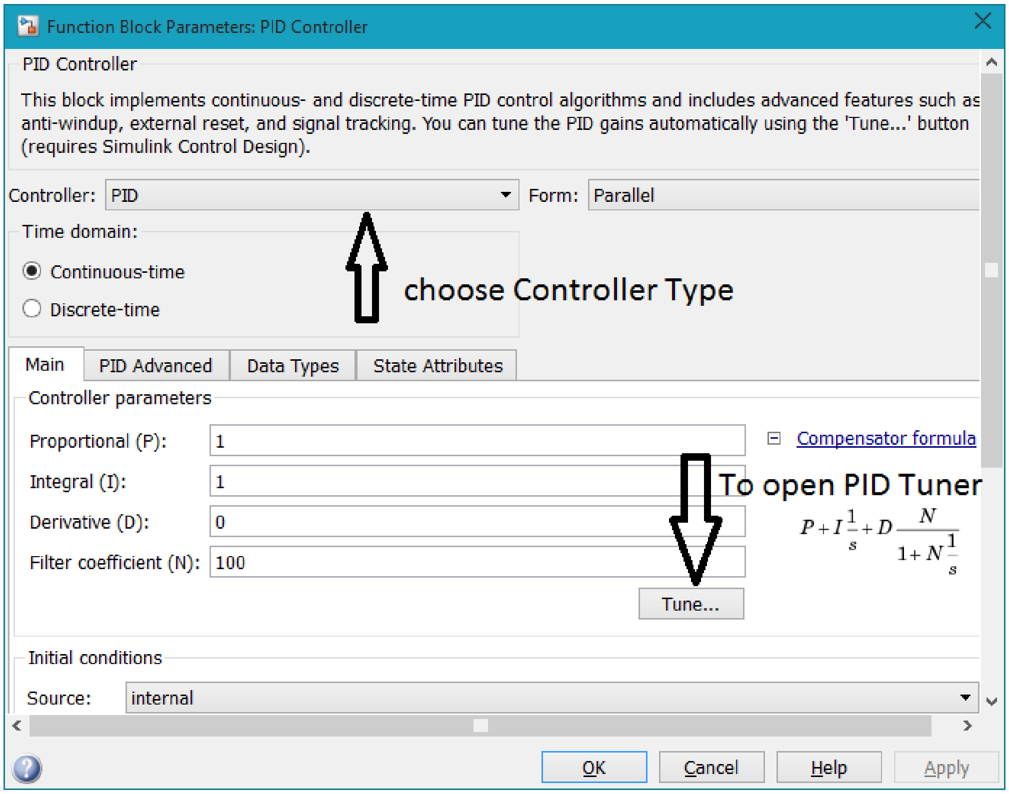




2) Add PID Controller block

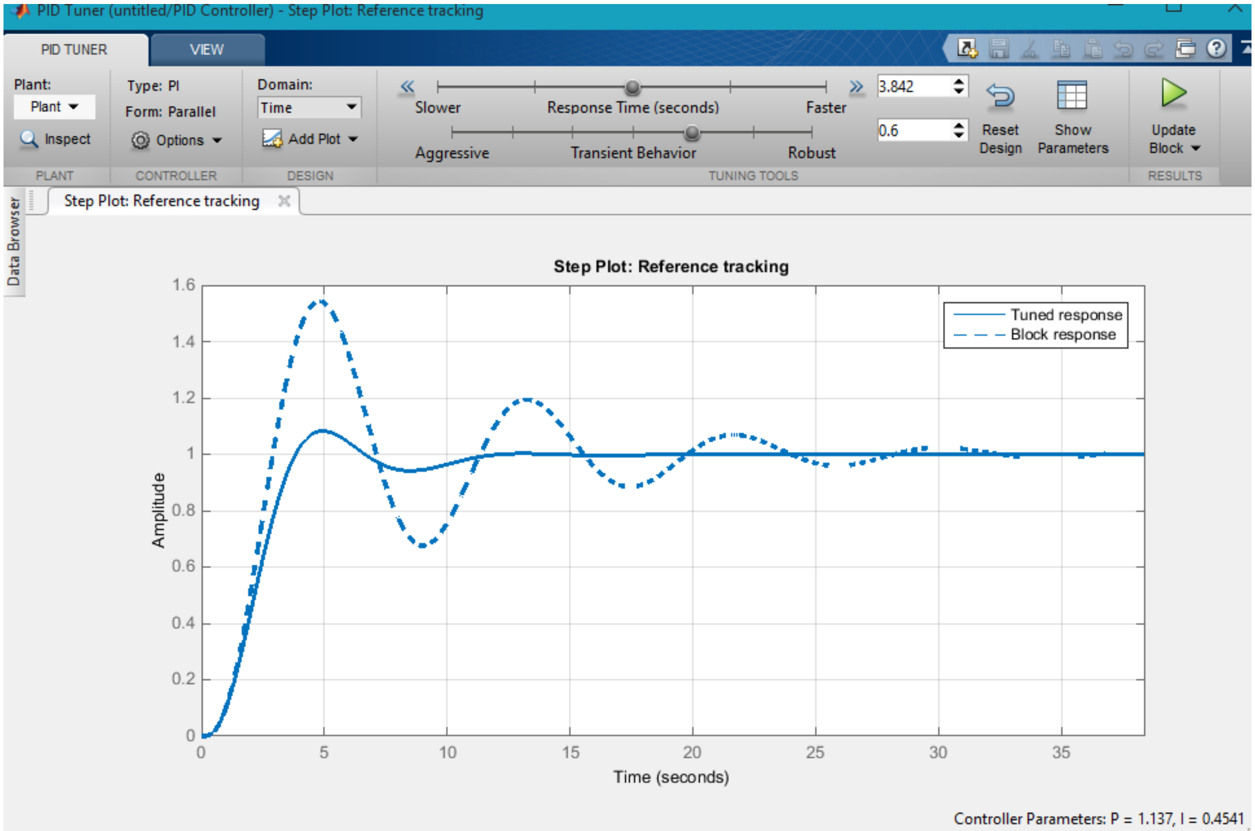


3) Double click on PID controller block



4) Choose the controller type , click apply and click tune

The PID tuner window will appear and can follow the steps as example 1



Note:

If need change the controller type , must be close the tuner window, and repeat step 4

References . . . .

1) http://www.mathworks.com/help/index.html
2) Farid Golnaraghi, Benjamin C.Kuo; Automatic Control Systems; Ninth Edition
3) Norman S.Nise; Control Systems Engineering; Sixth Edition
4) Richard C.Dorf, Robert H.Bishop; Modern Control Systems; Twelfth Edition.