This paper is accepted for publication at IFAC Conference on Advances in PID Control, PID'12, Brescia, 28-30 March 2012

OptimPID: A MATLAB Interface for Optimum PID Controller Design [★]

Dingyü Xue [*] YangQuan Chen [**]

[*] *School of Information Science and Engineering*
Northeastern University, Shenyang 110004, P R China
(e-mail: xuedingyu@mail.neu.edu.cn).
[**] *Department of Electrical & Computer Engineering*
Center for Self-Organizing & Intelligent Systems (CSOIS)
Utah State University, Logan, UT 84322-4120, USA
(e-mail: yqchen@ieee.org).

Abstract: In the paper, a MATLAB/Simulink based graphical user interface OptimPID is presented for designing optimum PID controllers of different types, under different criteria. The user is required only to specify the plant, linear and nonlinear, in a Simulink model, and the controller can be optimized in a visual way. Illustrative examples of optimum PID design are given and the controllers designed are much better than the existing algorithms and the leading-edge tools.

Keywords: PID controller, optimum PID, optimal control, optimization, graphical user interface, global optimization, ITAE criterion, integral performance indices, actuator saturation, MATLAB and Simulink

1. INTRODUCTION

Since PID-type controllers are widely used in process industry, there are a great amount of published algorithms and applications of PID controllers, see O'Dwyer (2003), Åström and Hägglund (1995), Johnson and Moradi (2005), Silva et al. (2005), and some of the tuning algorithms collected in the books are already adopted in real applications. There are of course limitations in most of the existing tuning formula, such that

(1) Most of the available tuning algorithms are based on the assumption that the plant models are linear and time invariant. If there are nonlinearities in the plant model, or with actuator saturation in the controller, the existing methods can no longer be used.
(2) Most of the available design algorithms are established on the approximation of the plants to certain typical forms such as FOPDT given by $G(s) = k\mathrm{e}^{-\tau s}/(Ts+1)$, not the actual plants. If the plants cannot be approximated well with those model formats, good controllers may not be designed using the existing algorithms.
(3) The qualities of the proposed algorithms are not always good, and some of the published algorithms may even give misleading results.

Thus, it is more important to have a design tool for designing optimum conventional PID controllers for the actual plants directly. Also since actuator saturation is usually unavoidable in real process control systems, and there may also be nonlinearities in the plants, nonlinear behaviors should not be neglected. There are attempts to solve similar problems, for instance, the leading-edge interactive automated tuning facilities and the function `pidtune()` provided in new versions of Control System Toolbox of MATLAB, in MathWorks Inc (2011a). Unfortunately, the tools are not quite good for solving the above problems. Besides, they are not suitable for unstable or nonlinear plants, and the facilities are not quite handy for unexperienced users.

In this paper, a MATLAB/Simulink based graphical user interface, named OptimPID, is developed and presented for optimum conventional PID controller design in servo control systems. Integral performance indices are used, and the most meaningful criterion is recommended. The user needs only to provide the plant in a Simulink model, and OptimPID interface can be used to find the optimum parameters in a visual way. Different types of integral performance indices in servo control systems are summarized and commented in Section 2. In Section 3, a brief tutorial and descriptions to OptimPID is given, and in Section 4, some illustrative examples are given to show the benefit of the proposed interface. Also the examples and designed controllers can be used as benchmarks for further research on PID controller design. Global optimization tools can also be used with extra toolboxes such as Global Optimization Toolbox in MathWorks Inc (2011b), GAOT in Houck et al. (1995) and PSOt in Birge (2003).

2. INTEGRAL PERFORMANCE INDICES

A typical PID control framework for process system is shown in Fig. 1, where the PID controller is often followed by an actuator saturation, $|u(t)| \le u_\mathrm{m}$. The actuator saturation is practical in real-world PID control systems.

[★] Supported financially by the National Natural Science Foundation China, under Grant 61174145.

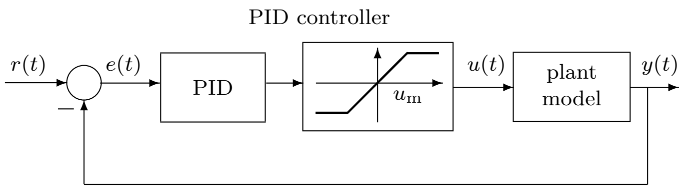


Fig. 1. PID control structure

The default form of the PID controller is

$$G_c(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{Ts+1} \tag{1}$$

with $T = 0.01$. The controller structure and parameters can be changed to other forms, such as discrete-time ones or standard PID controllers easily in the OptimPID interface through the PID controller block in Simulink.

The target of PID control is to keep the dynamic error signal $e(t)$ as small as possible. Integral performance indices to the error signal $e(t)$ are often good choices to do so. In particular, the following integral performance indices are used in the interface

$$I_1 = \int_0^\infty e^2(t)\mathrm{d}t, \quad I_2 = \int_0^\infty |e(t)|\mathrm{d}t, \quad I_3 = \int_0^\infty t|e(t)|\mathrm{d}t \tag{2}$$

$$I_4 = \int_0^\infty te^2(t)\mathrm{d}t, \quad I_5 = \int_0^\infty t^2e^2(t)\mathrm{d}t, \quad I_6 = \int_0^\infty t^2|e(t)|\mathrm{d}t \tag{3}$$

The above criteria are abbreviated respectively ISE, IAE, ITAE, ISTE, IT^{2}SE, and IT^{2}AE criteria respectively, and some of those are usually adopted in literatures.

It can be seen that in the ISE and IAE criteria, the values of the error signals $e(t)$ at any time instances are treated equally, whereas in the ITAE criterion, the value of error signal is penalized when the time t gets larger. That is to say that the ITAE criterion is more suitable for servo control problems, since the error signal is forced to settle down at zero as soon as possible. Later, an example will be given to demonstrate the advantages of the ITAE criterion.

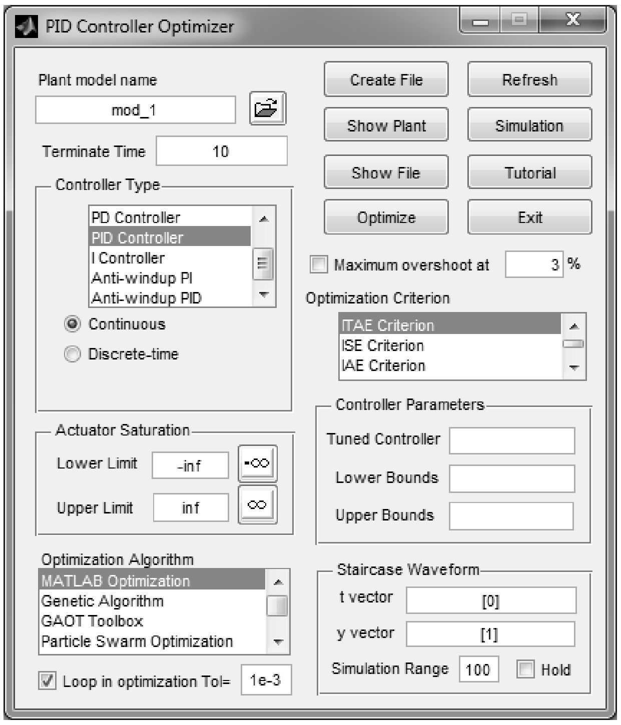
Since ITAE criterion can only be evaluated with simulation approach, infinite integrals cannot be evaluated. Finite-time ITAE integral defined as

$$I_{\mathrm{FT-ITAE}} = \int_0^{t_f} t|e(t)|\mathrm{d}t \tag{4}$$

can be used instead to approximate the ITAE criterion, if the finite time t_f is selected properly, since the error signal $|e(t)|$ may settle down at zero for larger t, such that the integral after t_f may have zero contribution to the total value of the ITAE integral.

3. OPTIMPID: GRAPHICAL USER INTERFACE

A MATLAB based graphical user interface, OptimPID, is developed and it can be used directly in optimum PID controllers design. The package can be downloaded from MathWorks' File Exchange web-site (See Xue (2012)). It can be unzipped to a folder and this folder should be added to the MATLAB search path. Once this is done, the command `optimpid` can be issued and the main interface shown in Fig. 2 can be displayed.


PID Controller Optimizer

Plant model name: mod_1

Terminate Time: 10

Controller Type: PD Controller, PID Controller, I Controller, Anti-windup PI, Anti-windup PID

Continuous / Discrete-time

Actuator Saturation — Lower Limit: -inf; Upper Limit: inf

Optimization Algorithm: MATLAB Optimization, Genetic Algorithm, GAOT Toolbox, Particle Swarm Optimization

Loop in optimization Tol= 1e-3

Maximum overshoot at 3 %

Optimization Criterion: ITAE Criterion, ISE Criterion, IAE Criterion

Controller Parameters — Tuned Controller; Lower Bounds; Upper Bounds

Staircase Waveform — t vector: [0]; y vector: [1]; Simulation Range: 100; Hold

Fig. 2. The main interface of OptimPID

If the plant model to be controlled is linear time invariant, the delay-free part should be entered into MATLAB workspace by Control System Toolbox functions, `tf()`, `ss()` or `zpk()`, in the object G, and the delay constant should be entered into the variable `tau`. The pre-constructed Simulink model mod_lti.mdl should be used to describe the plant model. The variable G can either be continuous or discrete. If the plant is nonlinear, a Simulink model should be constructed first.

The essential procedures of optimum PID controller design with OptimPID are:

(1) The model name should be entered into the edit box labeled Plant model name in the interface. For linear plants, enter mod_lti, defined in Fig. 3, and the LTI object G and variable `tau` should be specified first in MATLAB workspace.

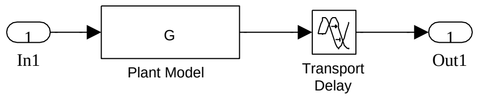


Fig. 3. The Simulink description of the linear plant

(2) The finite time t_f should also be entered into the Terminate time edit box.
(3) One can click the Create File button to establish a *.m file for the objective function, named as optpid_fun*.m. Redundant files can be removed with the Refresh button.
(4) Click Optimize button to start and visualize controller design process. The scopes of the signals $y(t)$, $u(t)$, and $I(t)$ are displayed during the optimization process. The checkbox Loop in optimization can be checked to execute loops in the optimization process, to ensure an accurate optimization results.

Below are optional settings in controller design:

(1) Different forms of PID controllers, P, PI, PD, I, PID, and the ones with anti-windup, continuous and discrete, ones with actuator saturation can be selected, can be selected from the controls in PID Type and Actuator saturation, in an easy manner.
(2) Different criteria ITAE, ISE, IAE, and so on as defined in (1) and (2), can be adopted from the Optimization Criterion list box, with ITAE the recommended one.
(3) Upper and lower bounds on the PID controller parameters can be specified in the relevant edit boxes in the Controller Parameters group.
(4) Different optimization tools and algorithms can be used, including GAOT and PSOt, downloadable from File-exchange and used for global optimal controllers
(5) System simulation under PID controller can be performed, if the input signal is specified as staircase waveform. One can specify the t-vector, y-vector and Simulation Range edit boxes, and click Simulation button. If Hold checkbox is checked, later optimum design will be made according to the staircase waveform.
(6) The optimized controller parameters are returned in MATLAB workspace in the variables `Kp`, `Ki` and `Kd`. Also the optimized parameters are returned in the Tuned Controller edit box.

A hidden internal model in Simulink is given in Fig. 4, where the parameters of the PID controller, including those with possible actuator saturation and anti-windup, can be assigned directly with the OptimPID interface.

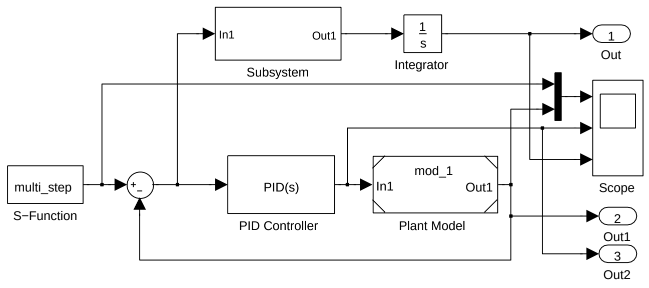


Fig. 4. Hidden Simulink main model

In the model, the multi_step block is a masked Simulink block described by the S-function, describing the staircase waveform of the input signal

```
function [sys,x0,str,ts]=multi_step(t,x,u,flag,tTime,yStep)
switch flag,
  case 0, sizes = simsizes;
    sizes.NumContStates = 0; sizes.NumDiscStates = 0;
    sizes.NumOutputs = 1; sizes.NumInputs = 0;
    sizes.DirFeedthrough = 0; sizes.NumSampleTimes = 1;
    sys = simsizes(sizes);
    x0 = []; str = []; ts = [0 0];
  case 3, i=find(tTime<=t); sys=yStep(i(end));
  case {1, 2, 4, 9}, sys = [];
  otherwise, error(['handled flag = ,num2str(flag)]);
end;
```

where the staircase waveform is expressed as the two vectors tTime and yStep, which should be specified in the lower-right corner of the main interface. The integral performance indices subsystem is shown in shown Fig. 5, and it is controlled by the constant `keyCriterion`. This constant can also be assigned by the interface. One may also add more performance indices in the Simulink model. Besides, in order to design satisfactory controllers, the following key points should also be considered:

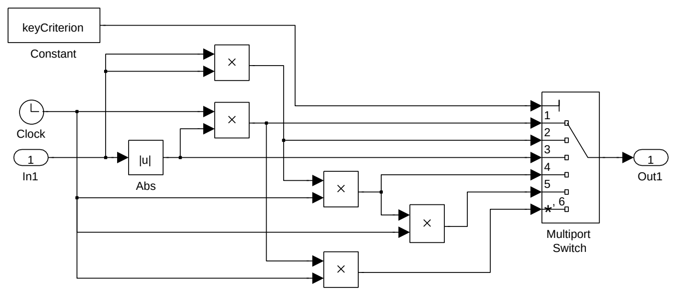


Fig. 5. The subsystem for the integral performance indices

(1) The selection of the terminate time t_f is sometimes crucial in optimum controller design. Since the integrands in the performance indices are non-negative, the integrals are always nondecreasing functions. The curves will remain flat when the integrand, or the error signal $e(t)$, settles down at zero. The strategy of selecting t_f will be depicted through examples.
(2) In order to keep the speed of the optimum controller design process, the number of iterations is set to a moderate value in each run. Thus, the Loop in optimization checkbox should be checked for a good controller design request, otherwise, the Optimize button may be clicked twice or three times to get a converged result.
(3) Sometimes, especially in the case of unstable plants, the conventional searching algorithms may not result in stabilizing controllers, global optimization approaches such as genetic algorithms, simulated annealing algorithms, pattern search algorithms as well as particle swarm optimization techniques should be adopted from the interface. However it should be noted that the speed of using these algorithms may sometimes be extremely slow.

4. ILLUSTRATIVE EXAMPLES

In this section, several illustrative examples are presented. Commonly used plant models in literatures are adopted and optimal PID controllers are designed. These controllers can later be used as benchmark PID controllers.

[Example 1]: Non-minimum phase plant

$$G(s) = \frac{5(-s+3)}{s^2(s+6)(s+10)}$$

the model should first be entered into MATLAB workspace

```
>> s=tf('s'); G=5*(-s+3)/s^2/(s+6)/(s+10); tau=0;
```

Since there is an integrator in the plant, PD controllers are adequate for the system. With MATLAB function `pidtune()`, a PD controller can be designed, however the quality of the controller is very poor. For the plant model, terminate time of $t_\mathrm{f} = 10$ is sufficient. Thus in the interface shown in Fig. 2, the following controls should be entered:

- Plant model name: mod_lti
- Terminate Time: 10
- Controller Type: PD Controller

Clicking Create File button to establish a *.m file for the objective function, then click Optimize button to design optimum PD controller. Change the lower/upper boundaries of the actuator saturation to 10, 8, 6, 4,2 respectively, the optimum PD controllers can also be designed. Also set the boundaries to `inf` again, and select ISE Criterion from the Optimization Criterion listbox, an optimum PD controller for such a criterion can be designed, as shown in Table 1. The step responses of those controllers are shown in Fig. 6, and it can be seen that the ISE criterion is oscillatory, and not suitable for the plant, while the ITAE criterion forces the time response settle down to zero as quickly as possible and it is more suitable.

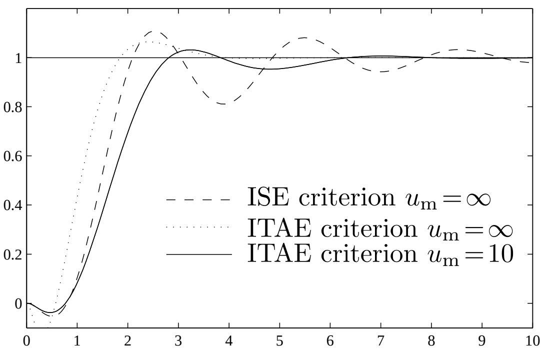


Fig. 6. Comparison of different criteria

Table 1. PD controllers with performance indices

criterion	saturation	K_p	K_d	perf. index
ITAE	`inf`	2.65×10^{-5}	3.6934	0.8866
	10	2.4577	4.8103	2.0979
	8	2.4997	4.8526	2.1555
	6	2.5197	4.8732	2.2200
	4	2.5525	4.9076	2.2959
	2	2.3112	4.5811	2.6929
ISE	10	3.6928	6.2878	3.2254
`pidtune()`	`inf`	0.0021	0.115	21.6697

The integral curve of the error signal can also be obtained in Fig. 7, for the ITAE criterion, with $u_\mathrm{m}=10$. It can be seen that the integral curve settled down at $\hat{t}_\mathrm{f}=7.2$, thus the terminate time of 10 is correct for the example.

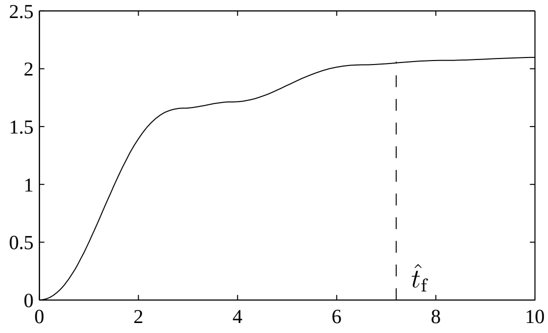


Fig. 7. Simulink description of the plant

Normally it is crucial in selecting t_f in controller design. Here different values of t_f's are used in designing PD controllers and the results are consistent. The step responses under these controllers are shown in Fig. 8, and it can be seen that the response curves are almost the same. It can be seen from the example that, for this plant model, if the terminate time t_f is selected larger than $\hat{t}_\mathrm{f}$, the designed optimum controllers are all acceptable. Thus it is important to check after design whether the error signal is settled down at zero, or whether the ITAE integral curve is flat at the selected t_f.

Table 2. Controllers designed with different t_f's

selected t_f	K_p	K_d	performance index
7	2.5395	4.8911	2.0368
8	2.5664	4.9585	2.0644
9	2.4579	4.8087	2.0819
10	2.4577	4.8103	2.0979
12	2.4507	4.7989	2.1050
14	2.4534	4.8049	2.1075
16	2.4297	4.7768	2.1084
20	2.4337	4.7836	2.1086
50	2.4384	4.7856	2.1086
100	2.4224	4.7666	2.1087
500	2.4424	4.7938	2.1085

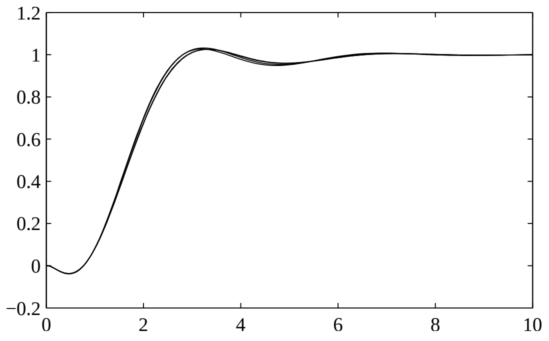


Fig. 8. Control results with the PD controllers

[Example 2] Consider the plant model given by

$$G(s)=\frac{1+\dfrac{3\mathrm{e}^{-s}}{s+1}}{s+1}$$

It is not possible to express the plant model with linear delay-free standard form. With MATLAB `pidtune()` function, only PI controller can be obtained, such that $K_\mathrm{p}=0.063$, $K_\mathrm{i}=0.084$, unfortunately the tuning behavior is very poor. To design optimum PID controllers with OptimPID, a Simulink model should be established first for the plant as shown in Fig. 9, and saved in file mod_1.mdl.

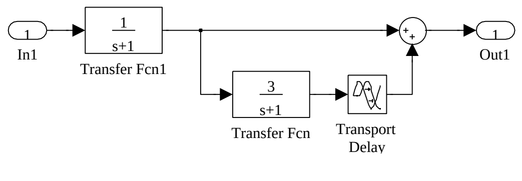


Fig. 9. Simulink description of the plant

Again the following control items in the OptimPID interface should be responded

- Plant model name: mod_1
- Terminate Time: 10
- Controller Type: PID Controller

For different levels of actuator saturation u_m, the PID controllers can be designed and given in Table 3. Also the PI controller designed with `pidtune()` function is also compared, which gives very poor control quality.

The closed-loop step response of the system can be obtained as shown in Fig. 10, and it can be seen that the behaviors under the PID controllers are very satisfactory.

Table 3. PID controllers for different u_m's

saturation	K_p	K_i	K_d	performance index
inf	0.51782	0.23691	0.34761	1.1973
10	0.56615	0.20917	0.37859	1.3283
8	0.58135	0.20515	0.39181	1.3696
6	0.5949	0.19863	0.40535	1.4260
4	0.62109	0.19243	0.43148	1.5053
2	0.64305	0.18355	0.4522	1.6247
with `pidtune()` function				
inf	0.063	0.084	0	8.7198

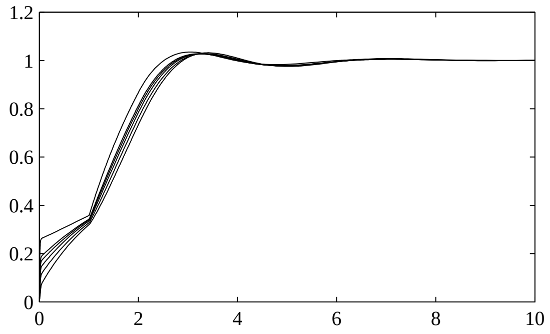


Fig. 10. Closed-loop step response for different u_m's

[Example 3] Consider the time-varying plant model

$$\ddot{y}(t) + \mathrm{e}^{-0.2t}\,\dot{y}(t) + \mathrm{e}^{-5t}\sin(2t+6)y(t) = u(t)$$

For the time varying plant model, it is extremely difficult, if not impossible, to design satisfactory PID controllers with traditional approaches. Again, OptimPID can be used to design PID controllers. A Simulink model for the plant model can be created, as shown in Fig. 11.

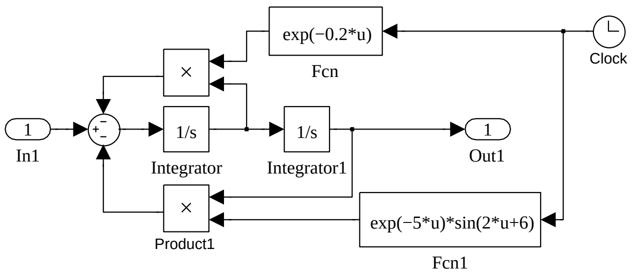


Fig. 11. Simulink description of the plant

If there is no actuator saturation in the control system, the step response can be of any speed, and the control signal could be extremely large. In real applications, this is not acceptable. Again for different levels of allowed control signals, the PID controllers can be designed, as shown in Table 4.

The step responses under different values of u_m's can be obtained as shown in Fig. 12, and it can be seen that the step responses under $u_\mathrm{m} = 10$ and 8 are quite similar, and when u_m decreases, the speed of responses may also be reduced, however the control is acceptable.

Table 4. Controller parameters for different t_f's

saturation u_m	K_p	K_i	K_d	ITAE's
10	388.3696	0.001783	51.74334	0.061958
8	373.616	0.001893	54.30331	0.078147
6	357.1526	0.0019094	57.97354	0.10565
4	328.1805	0.001997	62.03096	0.1624
2	282.4096	0.0021302	68.526	0.34422

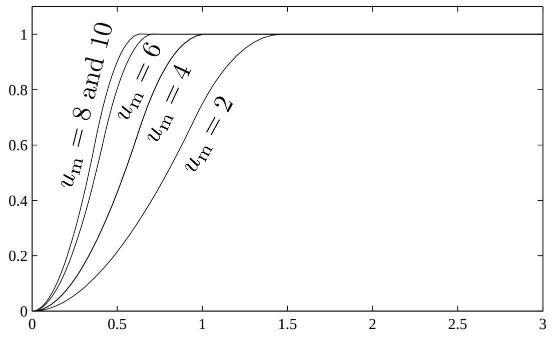


Fig. 12. Step responses under different u_m's

If the controller designed for $u_\mathrm{m} = 2$ is used, the staircase waveform response of the system can be obtained by first setting the waveform as `t_seq= [0,12,26,39,54]`, `y_seq= [1,5,4,2,6]`, and stop time as $t_\mathrm{f} = 70$. Then click Simulation button, the staircase simulation results and the actual control signal can be obtained as shown in Fig. 13. The total ITAE criterion in this case is 900.1046. It can be seen that although the time-varying system under unit step input is perfect, the response under the given staircase waveform is very poor. A new controller for the given input signal should be designed.

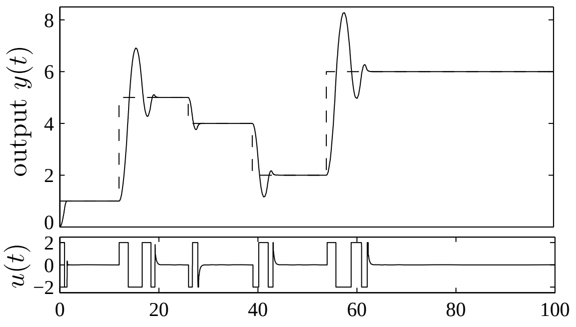


Fig. 13. System response under staircase waveform

One can check the Hold checkbox, and the new stop time of 70 can be filled in automatically to the Terminate Time edit box. One can click the Create File button to generate a new objective function, then click Optimize button to design the controller, which yields $G_1(s) = 106.8637 + 5.89 \times 10^{-8}/s + 72.5773s/(0.01s+1)$, and the new total ITAE value is reduced to 507.7497. The new time response is obtained as shown in Fig. 14. It can be seen that the time response is significantly improved, although the response in the first few seconds is not as good.

[Example 4] Consider an unstable linear plant

$$G(s) = \frac{s+2}{s^4+8s^3+4s^2-s+0.4}$$

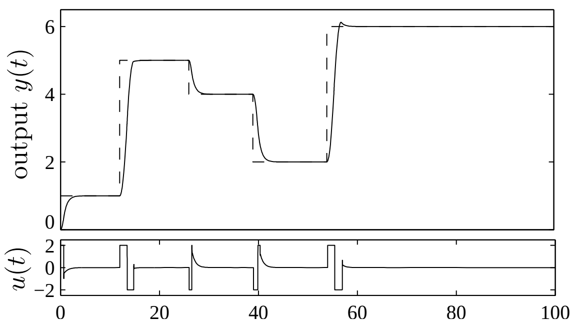


Fig. 14. With new optimal controller

and the actuator saturation limits are $|u_{\rm m}(t)| = 5$. The following MATLAB commands should be given to express the plant model.

```
>> G=tf([1 2],[1 8 4 -1 0.4]); tau=0;
```

In this case, the following control items

- Plant model name: mod_lti
- Terminate Time: 15
- Controller Type: PID Controller
- Actuator saturation: lower and upper limits, −5 and 5

should be specified. Since the plant model is unstable, it might be difficult to start with a stabilizing initial controller. Traditional optimization algorithm may fail to find a global optimum, or even stabilizing PID controller. Global optimization algorithm such as genetic algorithm or pattern search algorithm can be adopted instead. For instance, the Pattern Search item from the Optimization Algorithms listbox can be selected, and the Create File and Optimize buttons can be clicked to find the optimum PID controller parameters. With the interface, the optimum PID controller can be designed as $G_{\rm c}(s) = 50.9569 + 0.1656/s + 58.777s/(0.01s + 1)$. The ITAE criterion under such a controller is 1.0375. The step response and control signals are shown in Fig. 15. It can be seen that despite the plant model is unstable, the closed-loop system behaves well under the controller.

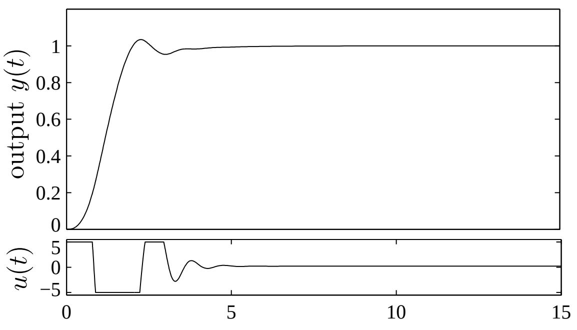


Fig. 15. Step response of the closed-loop system

5. CONCLUSIONS

In the paper, a universal optimum PID controller design interface is presented. With such an interface, optimum PID controllers under different integral performance indices can be designed, when the plant model is represented by a Simulink model. The interface is easy to use, and may be used by unexperienced users directly. The ITAE performance index is recommended for the design, since it is more reasonable than the widely used ISE criterion. Global optimized controllers can be obtained with the use of relevant tools. The conclusions and advantages of the proposed OptimPID interface are:

(1) This interface is much much better, and handy, than the leading-edge MATLAB tools in MathWorks Inc (2011a), and the results are better than the collected algorithms in O'Dwyer (2003) and other published ones, since meaningful optimization is involved.
(2) OptimPID is an innovative interface for the design of optimum PID controllers, since it only requests the user to specify the plants to be controlled in Simulink, the optimum controller can be found with the use of optimization facilities in MATLAB.
(3) ITAE criterion is mainly studied and recommended, rather than the well accepted ISE and other criteria, since it is much better in describing time domain response behaviors. This criterion is meaningful in control applications, although the optimum controllers under other criteria can equally be obtained, if we wish. The tactics of selecting of $t_{\rm f}$ is also presented.
(4) The actuator saturation in the controller is allowed, and SISO nonlinear plant with any complexity can be handles easily, without any restrictions. The specification of overshoot constraints can also be used in the interface.
(5) Different optimization problem solvers are integrated in the interface, and genetic algorithm, particle swarm optimization, simulated annealing as well as pattern search algorithms may lead to global optimization results, which are likely to ensure stabilizing closed-loop behavior even for complicated plants.
(6) A special trick is set to the generated objective function file, and it may successfully avoid unstable solutions which cause abnormal terminations in simulation processes.
(7) Practical plants with noises can be modeled in Simulink directly, however in this case, ITAE may not be the ideal criterion, ISE, IAE can be used instead.

REFERENCES

Åström, K.J. and Hägglund, T. (1995). *PID controllers: theory, design and tuning.* Research Triangle Park, Instrument Society of America.

Birge, B. (2003). Psot, a particle swarm optimization toolbox for matlab. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 182∼186. Indianapolis.

Houck, C.R., Joines, J.A., and Kay, M.G. (1995). *A genetic algorithm for function optimization: a MATLAB implementation.* Electronic version.

Johnson, M.A. and Moradi, M.H. (2005). *PID control — new identification and design methods.* Springer, London.

MathWorks Inc (2011a). *Control System Toolbox user's manual.*

MathWorks Inc (2011b). *Global Optimization Toolbox user's manual.*

O'Dwyer, A. (2003). *Handbook of PI and PID controller tuning rules.* Imperial College Press, London.

Silva, G.J., Datta, A., and Bhattacharyya, S.P. (2005). *PID controllers for time-delay systems.* Birkhäuser, Boston.

Xue, D. (2012). *OptimPID: an optimum PID controller design interface.* MathWorks File-exchange.