Undesired Changes in PID Controller Model Due to Simulators

A. Paz-Peña
Departamento de Control Automático
CINVESTAV-IPN
México D.F., México 07300
apaz@ctrl.cinvestav.mx

C. Vargas-Jarillo
Departamento de Control Automático
CINVESTAV-IPN
México D.F., México 07300
cvargas@ctrl.cinvestav.mx

Abstract—In this work we propose the simulation of a PID controller to solve two problems, namely: 1) Recovery signal with perturbation, using Simulink. 2) Tracking signal for a motor with perturbation in two ways, using Simulink and using a Matlab program. The aim of this work is to show particular examples, where Simulink changes the expected behavior of the model. We emphasize the importance of numerical integration of differential equations and the bounded signals treatment.

Keywords—PID simulation; PID control; DC motor model; Simulink; Matlab; numerical integration; aliasing.

I. INTRODUCTION

For seven decades PI and PID control structures have been used in control engineering practice [1-3]. The most usual PID control is the parallel controller structure designed by McMillan in 1990, which is used very frequently in many industrial controllers [3] to solve two important problems, one is the recovery of a signal and the other is the tracking signal of a motor. The development of a PID controller for a DC motor generally passes through a simulation stage, where the parameters of the system will be adjusted according to the limits of the hardware. If we work with trademark hardware, default parameters can simplify the implementation, like in the case of data acquisitions cards; however the trademark hardware usually contains hidden hardware corrector that works like a filter, amplifier or other control devices, producing changes in the controller model, i.e. the control actions are limited for using an unknown controller model.

II. MOTOR MODEL

We use the well known linear differential equation of second order with constant coefficients for describing the D.C. motor:

\[ V_i = R_i \left[ J \frac{d\omega(t)}{dt} + B X \omega(t) \right] / K_p \]

\[ + L_i \left[ J \frac{d^2\omega(t)}{dt^2} + B X \frac{d\omega(t)}{dt} \right] / K_p \]

(1)

Where \( X \) denotes the cross product of vectors, \( V_i \) is the induced voltage of the rotor, \( K_p \) is the back emf, \( \omega(t) \) is the angular velocity, \( R_i \) is the resistance of the rotor, \( I \) is the armature current, \( L_i \) is the inductance of the rotor, \( K_p \) is the electromagnetic torque constant, \( j \) is the inertia momentum and \( B \) the viscous friction coefficient. Assuming that \( L_i \) is

When we used the Simulink structure of Figure 1, a Matlab warning appears, advising of an algebraic loop as a consequence of using an ordinary differential equation instead of a delay differential equation, where the delay is produced by the feedback loop. We solved this problem by putting a memory block before the feedback amplifier [6], where the parameters of the memory block depends upon the size of the delay and the processing time of the controller.

A. Recovery signal with perturbation

For problem 1, with a sine input signal of magnitude 1 and frequency of 0.5 rad/sec., with PID controller parameters \( K_p = 0.5, K_i = 0.9, K_d = 0.9 \), output amplification constant of 10, feedback amplification constant of 1 and without feedback delay and white noise with default values and a constant of magnitude 0.5 as perturbation, we obtained figure 2 using the data generated in Matlab workspace. When a signal is recovered with a PID controller with white noise perturbation, we expected a phase change and persistence of white noise in the output signal, however is observed in figure 2 that does not occur. We ran the same problem introducing a feedback delay and we obtained the expected changes.

For problem 1, with a square input signal of magnitude 1, frequency of 0.5 rad/sec and pulse width of 50% of the period, with PID controller parameters \( K_p = 0.5, K_i = 0.9, K_d = 0.9 \), output amplification constant of 10 and without feedback delay and white noise using default values and a constant of magnitude 0.5 as perturbation, we recovered the input signal only with a difference in magnitude, no matters which solver, step, value of maximum step and perturbation signal had been chosen; the difference between the output signal and the input signal is never greater than 10% in width and is never greater than 7.0% in magnitude. We ran the same problem with the same parameters changing the output amplification constant to 100, recovering the input signal as it was expected.
Finally, we recall that the white noise signal in Simulink is produced by a bounded pseudo-random numbers generator [6], where the bound is not symmetrical with respect to the x axis and each value endures more than one sampling time, allowing the controller to diminish the difference between the input signal and the output signal in a soft way while the white noise value is the same.

B. Tracking signal in a motor with perturbation

1) Using Simulink: For problem 2, with a sine input signal of magnitude 1 and frequency of 0.5 rad/sec., with PID controller parameters $K_p=0.5$, $K_d=0.3$, $K_i=0.9$, output amplification constant of 1, feedback amplification constant of 1 and without feedback delay and constant of magnitude 0.5 and a decreasing monotone signal in the interval $[1.00, 1.06]$, generated by (2) with minimum value of -0.74 as perturbation.

$$P(t) = \left( \frac{0.04 \sin(t)}{0.04 \sin(t+\pi)} \right) + 625 \quad (2)$$

We obtained the same results as in the problem 1 in A, but the presence of errors depends on the integrator, for example, we found that the output signal is equal to the input signal with the stiff solver Ode23t, variable step, maximum step of 0.001, value of 1000 in the output control amplification and delay feedback.

For problem 2, with input and perturbation sinusoidal signals, we obtained the same results as in the problem 1 in A.

For problem 2, with a square signal of magnitude 1 and frequency of 0.5 rad/sec as input and perturbation signal, where the input signal is produced by a different source than the source that produces the perturbation signal, and with feedback delay, we obtained, after many numerical experiments, that the best tuning of the PID controller parameters were: $K_p=0.3$, $K_d=0.001$, $K_i=0.4$, output amplification constant of 0.1, feedback amplification constant of 10 and using the data generated in Matlab workspace we obtained figure 3.

We can see in figure 3 that the output signal has not a square form, therefore is useless in a tracking signal process; however the output signal still can be useful in recovery signal, for the case of electronic D.C. amplifiers, which use digital inputs.

Finally, in Simulink we can use values like 0.1 in one or more parameters, which are multiplied many times during the simulation, without observing the typical behavior for using truncated binary numbers [7].

2) Motor model with P and PID control via Matlab. If we write a Matlab program to simulate the classical PID control structure, there are some differences with Matlab Simulink because we need to specify the way to obtain the derivative of the error, the integration of the error, the time delay, and how to compare the input signal with the feedback signal. Moreover we can change the sampling times for different intervals of time according to the performance of the structure.

It is very conspicuous that the obtained results with Simulink have in general such a nice behavior. This made us wonder if we can obtain the same results by using our own code. Since Simulink uses the subroutines of Matlab, one could write his own Matlab program, expecting to obtain the same results.

We simulated a PID controller with a Matlab program with a sinus input signal, with sampling time of $((2\pi)/100)$ seconds. We took the derivative error value at $x_i$ as the slope of the straight line that passes through the points of the error signal at $(x_i, e(x_i))$ and $(x_{i-1}, e(x_{i-1}))$. We took the integral error value at $x_i$ as the result of trapezoidal integration in the interval $[x_i, x_{i-1}]$, with 30 subintervals for the method. Using the same parameters used in Simulink for the problem 2 with a sinusoidal signal and constant of magnitude 0.5 as perturbation, we obtained very different results, for example, the output signal grew excessively in magnitude, this behavior is a consequence of some divisions by small denominators; therefore we must have a criterion to avoid zero crossings, by mean of the integration step or averaging the boundary values in the intervals that present these behaviors. Also we observed in every run, some points where the values of the magnitude are too big, it is not
exclusive for the output signal; thus it is necessary to handle
saturations limits for the signals.

<table>
<thead>
<tr>
<th>Sampling</th>
<th>Time without big Oscillations (sec)</th>
<th>Sampling</th>
<th>Time without big Oscillations (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td></td>
<td>time</td>
<td></td>
</tr>
<tr>
<td>0.0010</td>
<td>0</td>
<td>0.2500</td>
<td>100</td>
</tr>
<tr>
<td>0.0028</td>
<td>65</td>
<td>0.5000</td>
<td>110</td>
</tr>
<tr>
<td>0.0625</td>
<td>65</td>
<td>0.7500</td>
<td>19</td>
</tr>
<tr>
<td>0.1000</td>
<td>85</td>
<td>1.0000</td>
<td>60</td>
</tr>
<tr>
<td>0.1250</td>
<td>90 increase</td>
<td>decrease</td>
<td></td>
</tr>
</tbody>
</table>

K_p=1.1, K_d=0.05, K_i=0.02, Feedback Amp. =0.06 and Output Amp. =1

Figure 3. Motor model with PID control via Matlabback

In the case of integration and derivation, we should also
consider the sampling time as our first parameter for
discriminate if it is necessary to use different regions of
sampling and choose a numerical integration method and a
derivation method according to the model [8].

We tuned many times to obtain the closer output signal to
the input signal, founding these PID controller constants:
K_p=1.1, K_d=0.05, K_i=0.02, feedback amplification constant
of 0.06 and any output amplification constant greater than 1.
With those parameters we assure that the output signal
remains at least 65 seconds without oscillations, except the
one shown on the figure 4. But the output always presents
offset and its magnitude is around 50% less than the desired.
Table 1 shows a comparison for different sampling times.

We simulated a P controller, with a Matlab program, to
show how even without including the integral and derivative
controls, is not possible to avoid big oscillations in problem
1, even without white noise, when is used a sine signal as
input with sampling time of ((2*π)/100) seconds and
without perturbations. After many numerical experiments
the best tuning of the P controller parameters were 0.1<K_p<1.1
and feedback amplification constant equal to 1,
where the output amplification constant is included in the
constant K_p. With this tune the output signal never presents
offset in phase, but its magnitude is always around 50% less
than the desired value and we assured an output signal with
at least 120 seconds without oscillations.

Besides we obtained one example with K_p= 0.1 which
presents oscillations after 1 second maintaining the rest of
the parameters without changes.

Finally if the signals are developed in series is necessary
to take into account the number of terms in the series that
represent the signal without loss of important information.
In some cases Lanczos filter can help to reduce time [9]. For
stochastic signals, as the white noise, the Stratonovich
integral is used, because of its nice characteristics [10].

III. ALIASING

One usually thinks that the aliasing problem in PI and
PID control is only presented in the generation of signals
[11]. However this problem appears also in the numerical
integration [12] and in the recovery of signals. Particularly
in communications systems, the Fourier series and Fourier
transform [13] are tools to generate or recover signals from
cardinal series. The aliasing phenomenon in Shannon
sampling reconstruction procedure has been discussed in
numerous papers [14-16]. The presence of aliasing in the
generation of signals often is associated to truncated cardinal
series [14]. FFT does not corrects the aliasing problem, only
diminishes the CPU time [17]. In this work we used the
Lanczos sigma factors to reduce the number of steps in the
generation of square signals [18]. We chose the sigmoidal
sigma factor form as in (3), because this reduces or at least
maintains the CPU time of the problems that we ran before.
In this equation α is a variable that depends directly of
time sample.

\[ \sigma = \text{sinc} \left( \frac{\pi x}{\alpha m} \right) \]  

(3)

When the signals are not square (3) can help only in a few
cases, thus inclusion of these sigma factors is based just on
heuristic knowledge.

Finally, when the sampling time is smaller than the time to
build the integration or derivation of the error signal, we could
obtain a false signal, whether square or not [8].

IV. CONCLUSIONS

We have simulated PID controllers on Matlab Simulink
and also using a Matlab program. We showed that the
simulations with Simulink can present discrepancies with
the theoretical expected results [19]. We showed how the
complexity increases in direct relation with the number of
specifications that the Matlab program requires. We
presented examples and explanations about how is not
possible to cancel a white noise perturbation just tuning the
parameters in the structure, and how for this and other
problems that present values that increase in excess, the way
to solve the problems can be using saturations limits for
the signals, for example with filters or using bounded signals.
We showed that the motor model presented is changed, either
assuming small delays when they are big, to choose wrong
differential equations solvers, hidden filters or ignore
predetermined actions of the hardware devices or software
packages used in the controller system. Besides, we showed
that the aliasing phenomenon is not exclusive of the

417
construction of square signals by means of cardinal series and also can be presented in the numerical integration.

Finally we showed that the controller performance changes, when we avoid the compute intervals with values that tend to indeterminate the numerical integration, of the differential equations used by the solver.

REFERENCES