

High Order Control Design – Advantage Over PI and PID Controllers reference

Yaniv O.¹, Theodor Y. and Safonov S.

Abstract

A robotic application is used to show that advanced controllers are much capable than *PI* controllers. They can achieve higher bandwidth, lower settling time and better disturbance rejection. The increased performance costs little in sensor noise amplification. We show using true-life design examples that advanced control algorithms improve equally well both speed and position controllers.

Introduction

Consider an electrical motor with shaft angle $\theta(t)$, driven by the current $i(t)$. We want the shaft speed, $\dot{\theta}(t)$, to follow a given trajectory, $\dot{\theta}_T(t)$. For this purpose, we embed the motor in a feedback structure as described schematically in **Figure 1**. The controller in **Figure 1** generates a correcting current command, $i(t)$, so as to keep the speed error, $e(t)$, minimal.

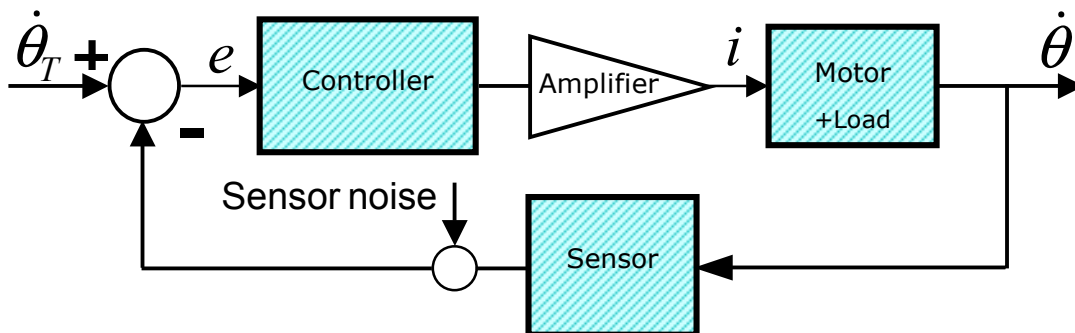
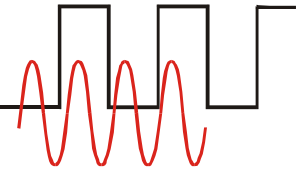


Figure 1: Speed control feedback structure around a motor

The controller is required to minimizing the speed error and in the same time the synthesized current command must remain smooth enough so that (i) no excessive stresses will shorten the system life, and (ii) the current amplifier will be able to effectively follow the current command.

The controller design must consider both small and large signals behavior. The small signal design cares for the behavior when the tracking error is small, and thus the required correction current (torque) is within the amplifier limits. Large signal (nonlinear) design must maintain good stability and performance while the current (torque) requirement goes beyond the amplifier limits. Out of range current (torque) requirements may develop due to extreme reference signal changes or due to extreme disturbances. This article focuses on the small signal (linear) design.

¹ Address for correspondence: O. Yaniv, Elmo Position Control, Shidlovskey 1, Yavne, 81101, POB 13081, Israel and Faculty of Eng. Tel-Aviv University, Tel-Aviv 69978, Israel. oyaniv@elmo.co.il



The controller must have a parameterization, so that users will be able to tune it to their specific applications. The most common controller parameterizations are P , PI and PID . A P (proportional) controller keeps the current $i(t)$ proportional to the speed error,

$\dot{\theta}_T(t) - \dot{\theta}(t)$. A PI controller generates $i(t)$ as the sum of two terms. The P term which is *proportional* to the speed error, and the I term which is proportional to the *integral* of the speed. A PID controller is a PI controller plus the D term, that is, a term proportional to the speed error *derivative*. The worst drawback of PI and PID controllers is their poor high frequency attenuation. Some commercial motion controllers add low-pass filters to their PI controllers, to improve the high frequency attenuation.

The traditional P , PI or PID controller have one big advantage – they are very simple, and a technician can tune them effectively using simple "cut and try" methods. These simple controllers suffice for simple applications – moderate or low performance requirements, and good enough mechanics and sensors. A very simple control problem is, however, a symptom of too generous mechanics and sensors design. More complicated controllers can push the tracking and disturbance attenuation performance to the physical limits of the system. An advanced controller can get the desired performance out of a lighter structure, or within degraded, cheaper sensors. For the same mechanics-sensors set, an advanced controller can increase the speed range in which accurate enough motions are possible. We use the term advanced-controllers for controllers of almost free structure and order. Advanced controllers do not preserve the PI simplicity. They have many parameters, and require an automated design suite for effective tuning. The decision to use advanced controllers is psychologically not easy. You have to trust the tuning suite of the Ph.D. guy better than you trust your senses. Moreover, you have to believe that the tuning suite does take appropriate design margins, so that you wont have vibrations when the load changes a bit.

In this paper, we compare the performance of advanced and traditional controllers, controlling a robotic arm. The advanced controllers are shown to do much better than the P PI or PID controllers. Section 2 compares PI controllers and advanced controllers, by a laboratory test. Section 3 extends the comparison of Section 2 to frequency domain. Section 4 shows similar comparison results for cascaded position control. Embedding the speed controller of **Figure 1** in an outer position feedback loop makes a cascaded position controller – see **Figure 2**. The position controller is required to follow a trajectory $\theta_{PT}(t)$.

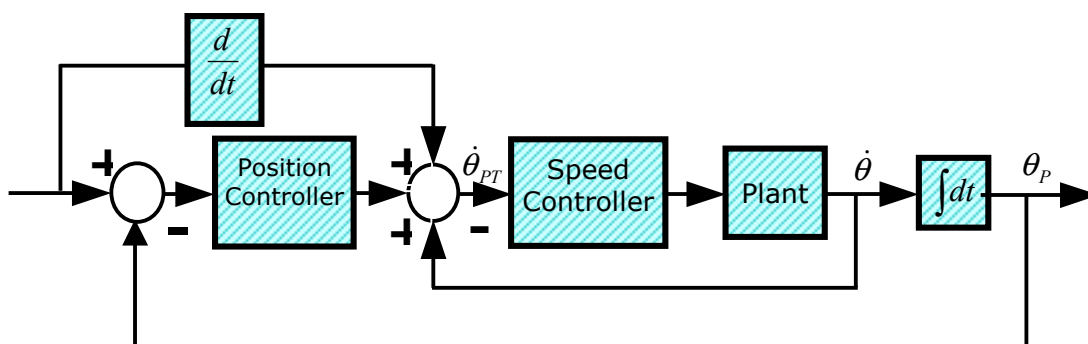
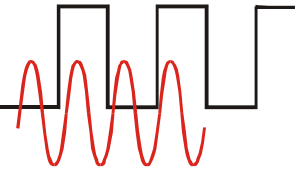


Figure 2: Cascaded position control feedback loop



Speed Control Comparison by Laboratory Tests

Our design example deals with a two-joint robot, see **Figure 3**. The robot is lightweight, and pays for the lightweight with high link compliances. An electrical motor drives each joint. For each motor, a tachometer measures the motor shaft speed and an encoder measures the motor shaft angle. The upper motor (motor 1) drives the internal link, and the lower motor (motor 2) drives the external link.

For this robot, a *PI* speed controller proved useless, since the robot became unstable for very low gains. We helped the *PI* with an additional high frequency low-pass pole. The *PI* plus low-pass performance shown in the next figures is probably better than what an experienced technician could achieve. This is since this robot exhibits high coupling between its articulated axes. If one use traditional *PI* tuning methods to optimize each axis when the other axis is inactive, the integrated system may become unstable or might loose some of its gain and phase margins due to the two axes interaction. If on the other hand, one use traditional *PI* tuning methods to optimize each axis when the other axis is active, stability of the integrated system is guaranteed but again the closed loop might loose some of its gain and phase margins.

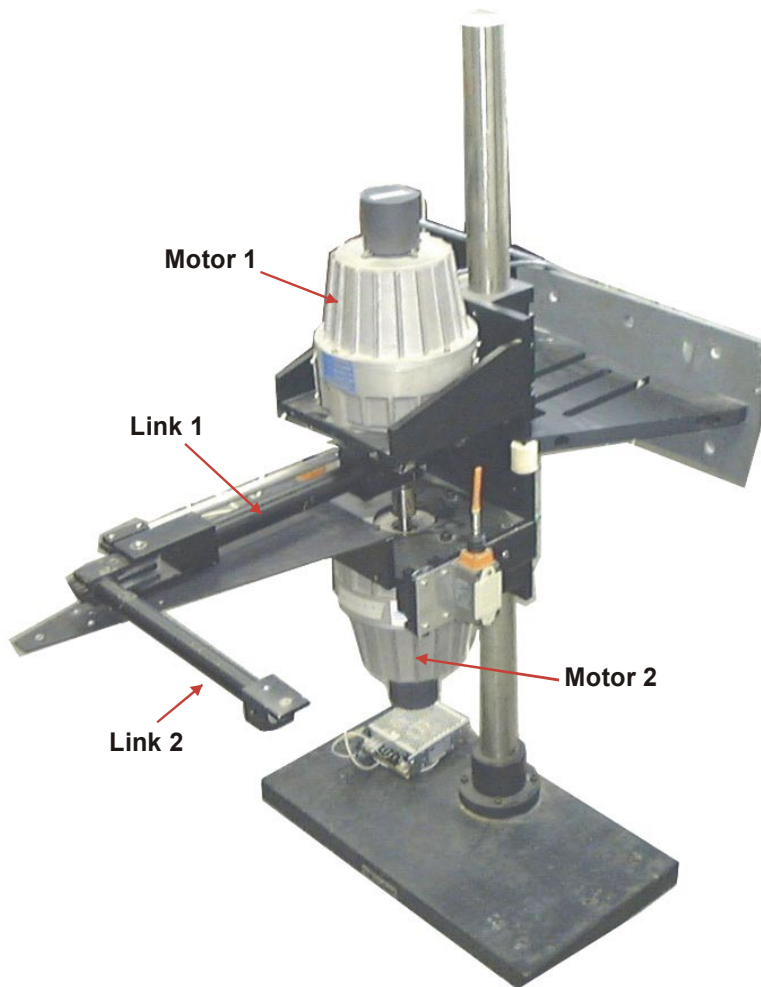
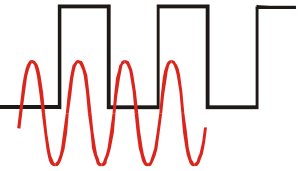


Figure 3: Robot for laboratory tests

a two joint robot with two motors, two tachometers and two encoders.



The robot was tested for several speed reference commands. **Figure 4** and **Figure 5** show the step response of *PI* plus low-pass controller and of an advanced controller against the reference step, for motors 1 and 2, respectively.

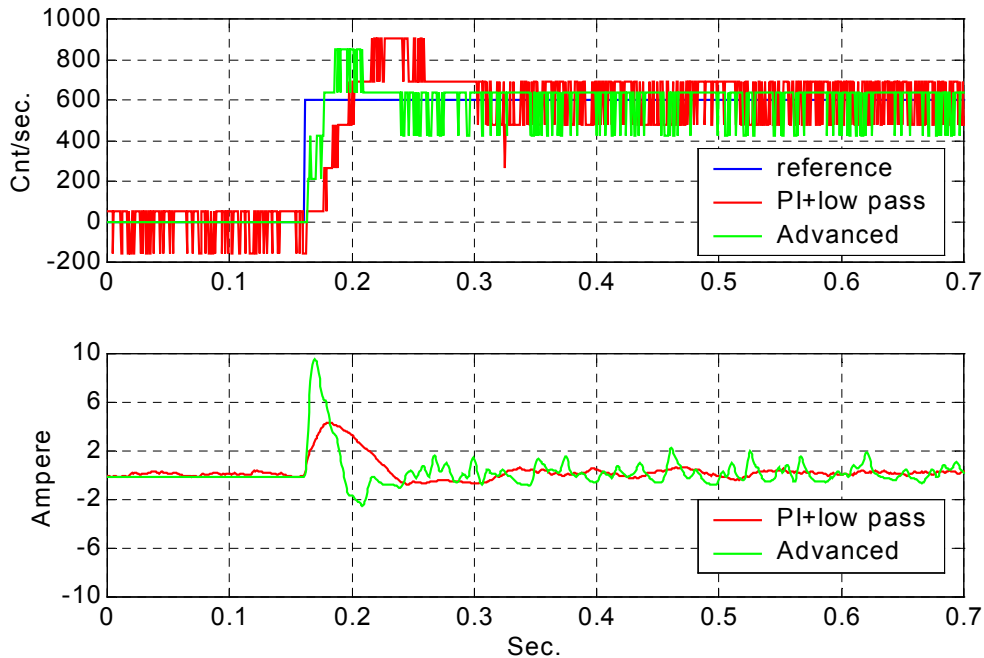


Figure 4: Comparison between *PI* plus low-pass controller and advanced controller.

The step command to motor 1 is 600[cnt/sec], and to motor 2 zero.

Clearly for motor 1 (**Figure 4**), the tracking error, rise time and settling time of the advanced controller are much lower than the corresponding results of the *PI* plus low-pass. The rise time of the advanced controller is 0.017seconds, about 43% of the 0.04 seconds rise time of the *PI* plus low-pass. The same relation holds for the settling time. The prices for the higher performance of the advanced controller are twice the current peak and larger high frequency noise.

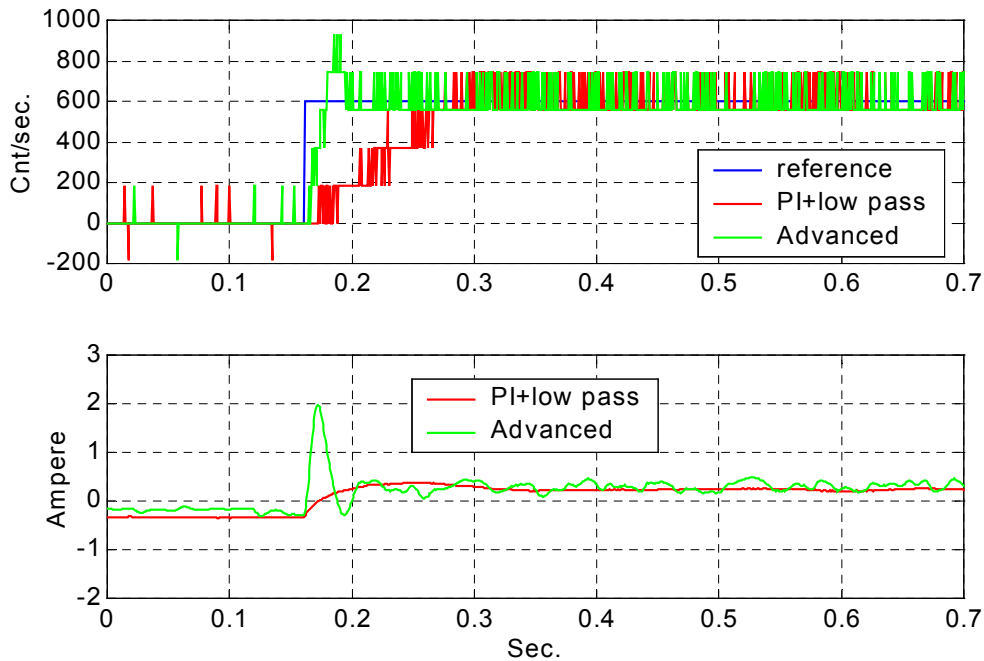
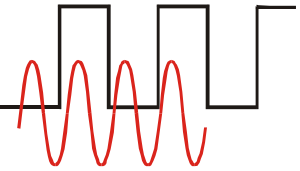


Figure 5: Comparison between PI plus low-pass controller and advanced controller.

The step command to motor 2 is 600[cnt/sec]; and to motor 1 is zero.

For motor 2, the tracking error, (**Figure 5**), rise time and settling time of the advanced controller are much lower than that of the *PI* plus low-pass. The rise time of the advanced controller is 0.02 seconds, about 20% the 0.1 seconds rise time of the *PI* plus low-pass. The same relation holds for the settling time. The price for the higher performance of the advanced controller is again the current peak. In most real applications however, the reference command is smooth and the peak current is dictated by the reference command rather than by the controller. This is emphasized by the next comparison, which compares the controllers for acceleration-limited step response.

Figure 6 and **Figure 7** compare between our advanced controller and the *PI* plus low-pass controller for a smooth acceleration limited speed command. Again, the tracking error for the advanced controller is much lower than for the *PI* plus low-pass. The overshoot of the advanced controller is by far lower, and the current consumption is about the same for both controllers.

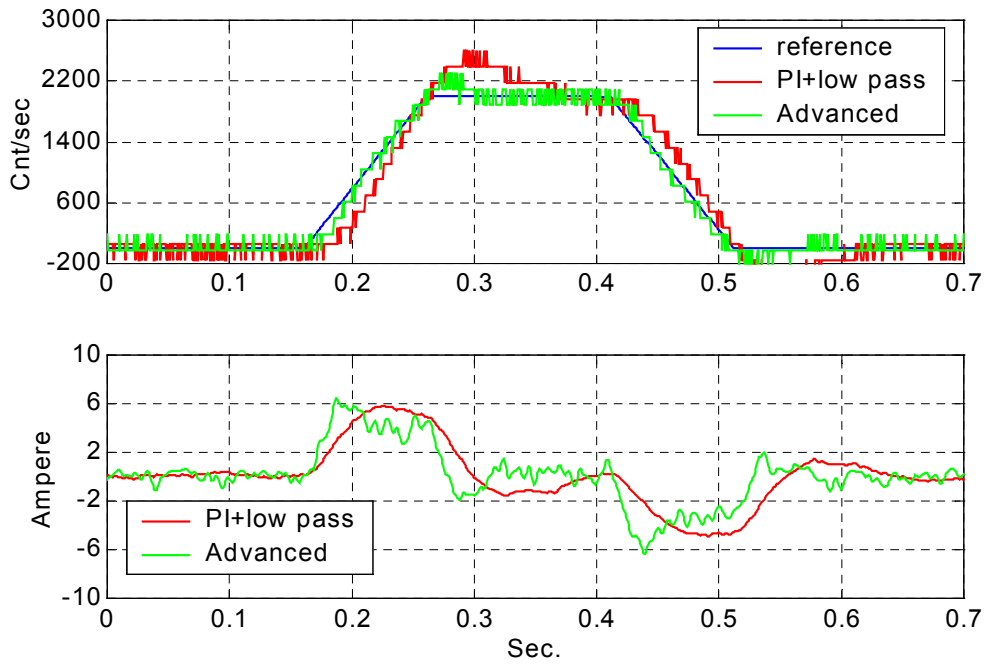
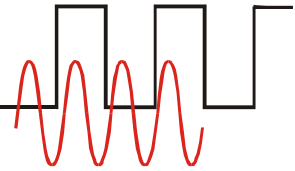


Figure 6: Comparison between *PI* plus low-pass controller and advanced controller.

Trajectory command to motor 1 is 2000[cnt/sec], acceleration limitation 20000[cnt/sec²]. Motor 2 is commanded to stop.

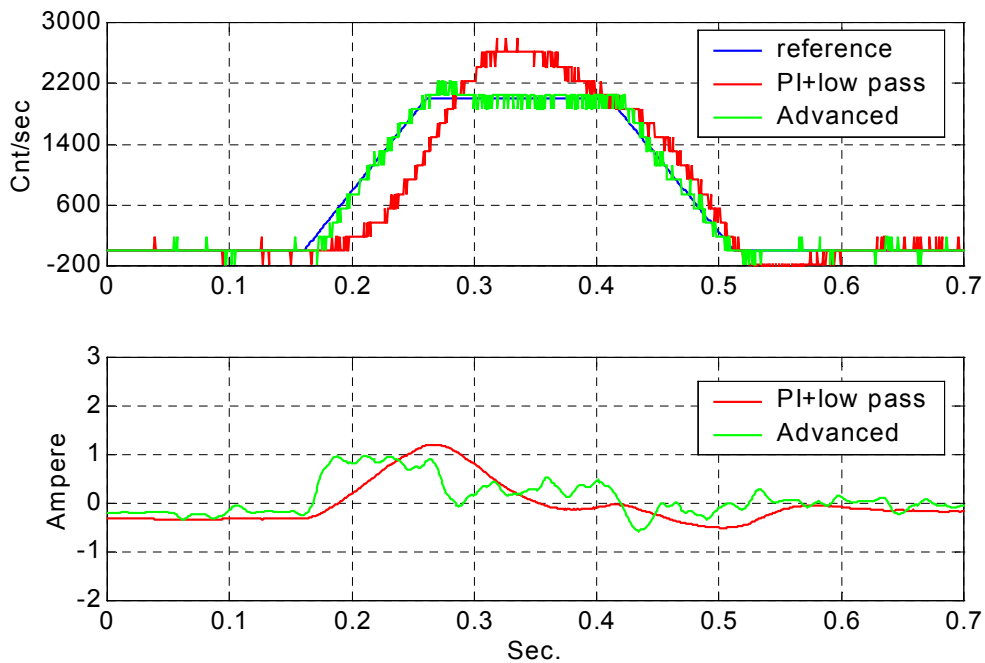
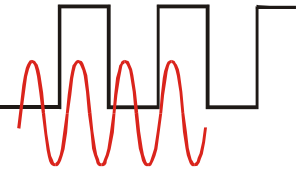


Figure 7: Comparison between *PI* plus low-pass controller and advanced controller.

Trajectory command on motor 2 is 2000[cnt/sec], acceleration limitation 20000[cnt/sec²]. Motor 2 is commanded to stop.



Speed Control Comparison by Frequency Domain Analysis

In section 2 the robot's tracking performance for different controllers has been studied. We subjected the controller to abrupt reference waveforms, which expose the transient behavior of the closed loop. The time domain tests of Section 2 show the final result, but they offer no explanation to the difference in the results achieved. The frequency domain analysis of this section grants insight to questions such as the feasibility of better designs, and the design margins taken. The frequency plots provide an estimate for settling time and overshoot. This estimate confirms the result of Section 2.

Open Loop

The robot has two motors and four sensors, two tachometers and two encoders, generating eight transfer functions from the current commands introduced into each of the motors to each of the sensors. Let p_{ij} denote the transfer function from current introduced into motor j to the integral of the angle (integral of speed) measured by the tachometer on the shaft of motor i ; and r_{ij} denote the transfer function from the current command introduced into motor j to the encoder coupled to the shaft of motor i . Figure 8 and Figure 9 depict these eight discrete Bode plots.

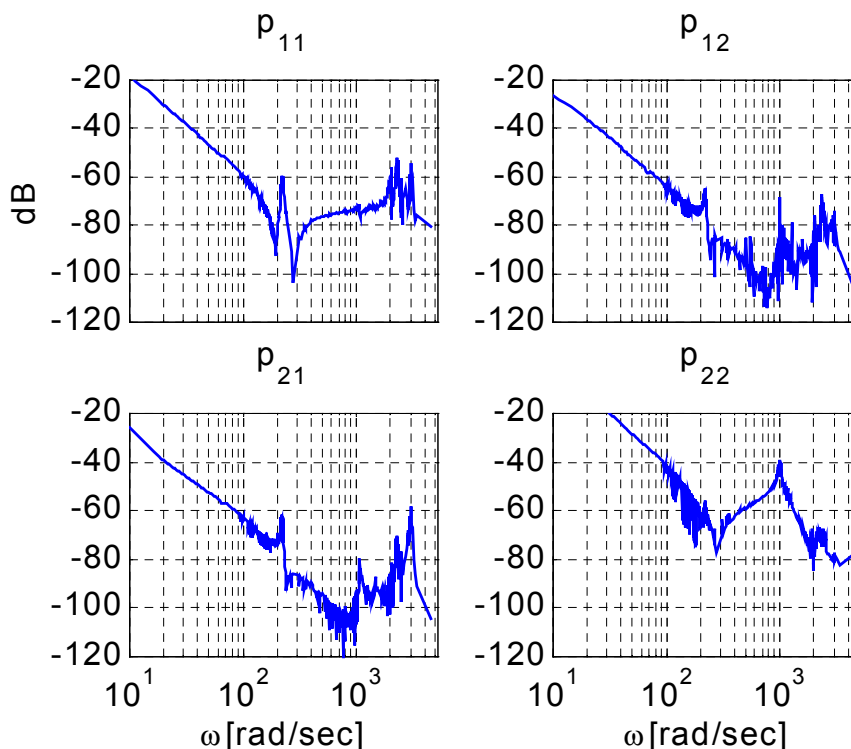


Figure 8: Bode plot of p_{ij} , from input j to integral of the tachometer on link i

