Edward Venator

EECS 304 Spring 2012

Helicopter Competition Project Report

## Solution

numG = [125 50 35]

denG = [(1/90) 1 0]

## Methodology

My approach to this problem was to use a PID controller combined with a low-pass filter. The PID controller provides the control signal, and the low-pass filter accounts for sensor noise and quantization effects by removing high frequencies from the error signal. I chose not to implement a prefilter on the reference path, as I did not find it to have any appreciable effect on the results.

I could have approached this problem differently and used a different controller architecture, but I chose not to for a few reasons. First, PID controllers are intuitively easy to understand. This makes them easy to tune quickly if an accurate simulation is available. Second, the PID controller is relatively robust with respect to inaccuracies in the simulation. More advanced techniques like pole-zero cancelation and use of notch filters to cancel the effects of system dynamics are highly dependent on accurate knowledge of the system characteristics. Since I have had no access to the system itself, I cannot verify that the simulation is accurate. Even if the simulation is accurate, it has several parameters that can vary. If, for example, the natural frequency of the system varies from 1.0 to 1.5 rad/sec, it would break any controller that attempts to cancel out the system dynamics. A PID controller can be tuned reasonably well, even if the system parameters change slightly.

After deciding on a PID controller, I began experimentally tuning the controller, validating my gains by testing them in the Simulink model.

## Conclusions

Although my results are not perfect, I think they are nearly as good as can be expected from a PID controller. If the parameters of the system were known perfectly, it would be possible to optimally tune the controller, and rltool has a function to do so. However, the optimally-tuned PID controller is operating right at the edge of stability. As a result, it is highly dependent on the accuracy of the system parameters. Varying the system parameters causes the optimally-tuned PID controller to go unstable. Given the range over which the parameters can vary, I am pleased with my controller.

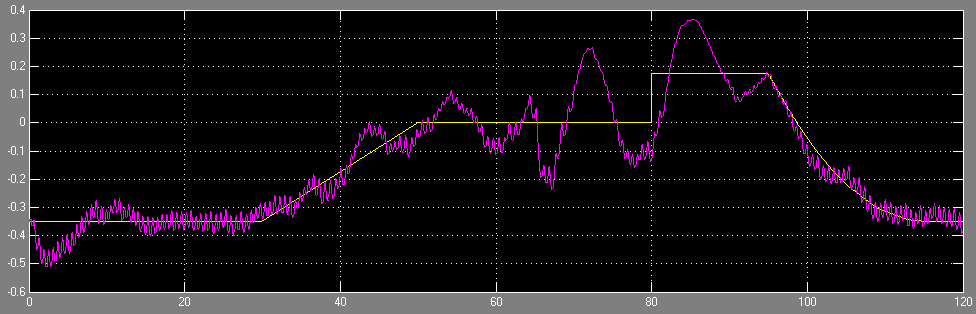


Figure 1: Worst case controller performance

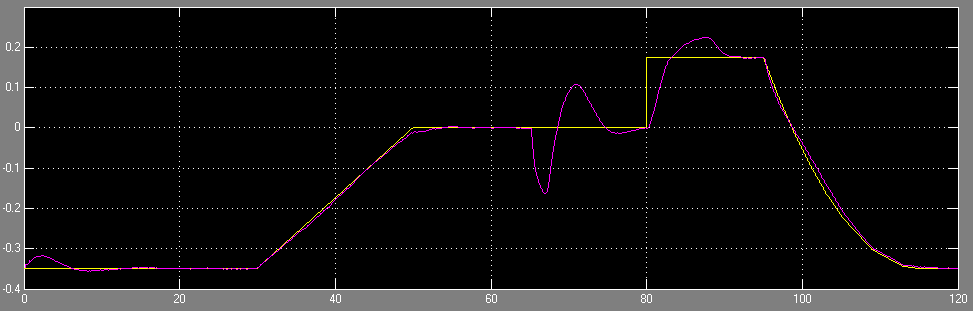


Figure 2: Best case controller performance

## Appendix: Simulated Cost at Boundary Conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **T** | **ωn** | **ζ** | **k** | **J** |
| .09 | 1.0 | .03 | .1 | 3.4049 |
| .09 | 1.0 | .03 | .15 | 3.9813 |
| .09 | 1.0 | .05 | .1 | 3.3939 |
| .09 | 1.0 | .05 | .15 | 3.9247 |
| .09 | 1.5 | .03 | .1 | 5.2782 |
| .09 | 1.5 | .03 | .15 | 8.2649 |
| .09 | 1.5 | .05 | .1 | 5.4248 |
| .09 | 1.5 | .05 | .15 | 7.0073 |
| .11 | 1.0 | .03 | .1 | 3.4804 |
| .11 | 1.0 | .03 | .15 | 4.7634 |
| .11 | 1.0 | .05 | .1 | 3.5423 |
| .11 | 1.0 | .05 | .15 | 4.4361 |
| .11 | 1.5 | .03 | .1 | 6.3267 |
| .11 | 1.5 | .03 | .15 | 14.4299 |
| .11 | 1.5 | .05 | .1 | 7.1175 |
| .11 | 1.5 | .05 | .15 | 9.8917 |

*Table 1: Simulated costs at boundary conditions*