aerospace

MDPI

Comparing commonly used controller algorithms to determine optimality with respect to controller quadratic cost, computational time, rise time, and rigidity

Philip Naumann, Timothy Sands

Abstract: In modern engineering, controllers are used as a brain-of-sorts to reach our targeted outcome by manipulating the inputs of the system. Many algorithms will claim optimality; however, it is not always clear what metric is being used to determine this. A controller can be optimal on a wide basis of design criteria. These include factors like hard resource cost to run the controller, the computational cost/ characteristic time of the processor, the rise time or settling time the control algorithm can achieve, or the rigidity of the controller under uncertainty. The appropriate criteria for optimality will range depending on application to which the controller must perform.

Classically, controllers like PID (proportional, integral, and derivative) or P+V (proportional plus velocity) controllers are used. These controllers feedback the targeted output into the controller. A decision is made based on the distance from the desired output, the rate of approach, and any oscillations. This algorithm is simpler as it is a numerical approach. The drawback is the inability to achieve exact results, especially for second order differential equations.

Consequently, an analytically based control algorithm solves this issue. For example, in a Real Time Optimal Controller (RTOC) the dynamics of the system are pre-programmed as a part of the algorithm. This makes it possible for the controller to track the reference exactly. The tradeoff is in the increased computational complexity.

Keywords: Controls, Rotational Kinematics, PID controllers, P+V controllers, Double Integral Quadratic Controller, Open Loop Controller, Real Time Optimal Controller, Monte Carlo Simulation, Pontryagin's method, Richard Bellman's Method

1. Introduction

In math, there are two approaches to solving any higher order problems: a numerical approximation or an exact analytical approach. In numerical methods, an approximation is generated based on linearizing around a small timestep. This solution is never exact, in fact, it would only converge to the exact solution as the timestep approached zero. On the contrary, the analytical result is the achieved by rigorously solving the problem. This result may however not always be attainable.

In a parallel way, classical control methods as taught Richard Bellman take on a numeric approach. Controllers like PID or P+V solve for the optimal control using linear feedback of tuned gains.

The analytical control solution was derived by Pontryagin. This method rigorously solves for the control based on the system dynamics. However, solving for the RTOC control is much more computationally expensive. This method can also go unstable under discontinuities or singularities.

1.1. Broad context and importance

Knowing the advantages and disadvantages of different control algorithms will help engineers be able to choose one that is right for their given application. The six control algorithms that will be discussed and tested in this paper include the following:

1) P + V controller,
2) P+V with a Double Integrator patching filter
3) Double Integrator patching filter and gain tuning
4) 2DOF Feed Forward controller
5) P+V with a Control Law Inversion patching filter
6) Open loop guidance (Double Integral Quadratic Controller)
7) RTOC

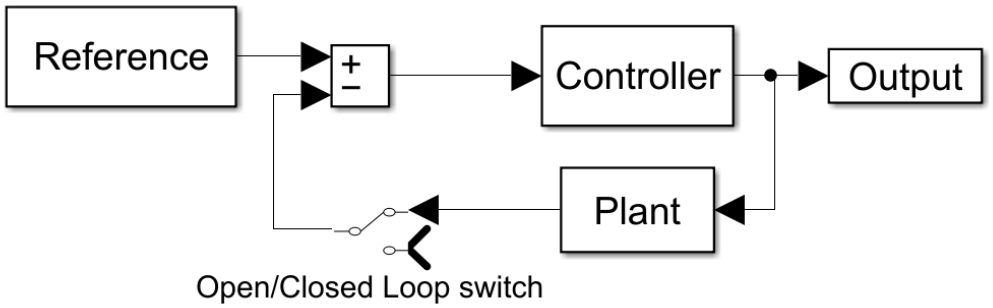
2. Set up of the controllers

These controllers will be compared using the same missile system. The missile will start at quiescent initial conditions and travel to a final normalized angle with zero final velocity. See Equations 1 and 2

$$\theta(0) = \omega(0) = 0 \quad (1)$$

$$\theta(1) = 0, \ \omega(1) = 1 \quad (2)$$

The systems will be set up to take a reference, that is feed into the controller. The controller output some torque that translates to some change in system dynamics in the plant block. The plant will return the kinematics of the missile which in the feedback control algorithms will be compared against the reference. Figure 1 shows these interactions in block diagram format.



2.1. *P+V controller architecture*

In the P+V controller, the output is calculated as a linear function of the error. A proportional gain, Kp, is used to translate the current position towards the desired position linearly. In addition, the velocity of the missile is also factored in with a velocity gain, Kv. This is used by the controller as an indicator of the rate at which the position is approaching the reference. Factoring in the velocity helps the controller avoid overshoot and oscillation. The input of the signal therefore could be written as described in Equation 3, with θ, ω, u *and* θ_d representing the angle, angular velocity, control, and reference respectively.

$$u = K_P(\theta_d - \theta) - K_V \cdot \omega \quad (3)$$

The plant system then uses the given input to activate an actuator. This incurs a rotational force on the system. The full dynamic equation of the system is written in Equation 4.

$$I\ddot{\theta} + K_V\dot{\theta} + K_P\theta = K_P\theta_d \quad (4)$$

As the force acts on the system, both the position and velocity change as an output, relative to a timestep. The new position and velocity can be determined by solving for the angular acceleration in Equation 4, and then integrating. These outputs are feed back into the P+V controller, thus completing the loop.

For the P+V controller, the gains were chosen based in the missiles settling time and damping ratio. Given a damping ratio of 0.7 and a settling time of 1 second, the natural frequency can be derived. See Equation 5.

$$\omega_n = \frac{4.6}{\zeta \cdot t_s} = 6.57\ s^{-1} \tag{5}$$

Having solved for the natural frequency, the gains Kp and Kv can be found by translating the equations for a damped oscillator to the control equation. The dynamics of a damped oscillator are shown in Equation 6.

$$F = -kx - c\frac{dx}{dt} = m\frac{d^2x}{dt^2} \tag{6}$$

$$\omega_n = \sqrt{\frac{k}{m}} \quad \text{and} \quad \zeta = \frac{c}{2\sqrt{mk}}$$

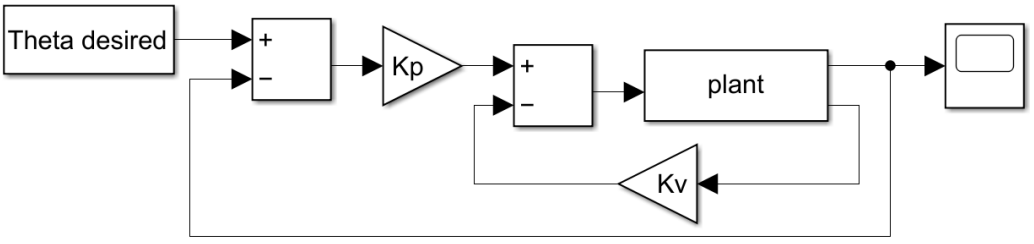
$$\frac{d^2x}{dt^2} + 2\zeta\omega_n\frac{dx}{dt} + \omega_n{}^2 x = 0 \tag{7}$$

The values of Kp and Kv now become apparent as they match the proportional and first order derivative in the dynamical equation. The optimal values for this system can be derived as

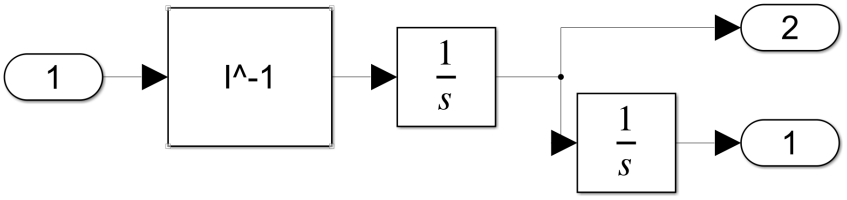
$$Kp = \omega_n{}^2 = 43.18 \quad \text{and} \quad Kv = 2\zeta\omega_n = 9.2 \tag{8}$$

2.1.1 Simulink P+V model

P+V controller can be modeled as a block diagram to represent the interactions present in the controller. The block controller was set up using Simulink. The system was mathematically structured by the dynamic governing equations (outlined in Equation 3 and 4). See Figure 2 for the result.



As mentioned before, the plant outputs both the angular position and velocity. This is done by integrating the input signal u. The internal block diagram of the plant is shown in Figure 3.



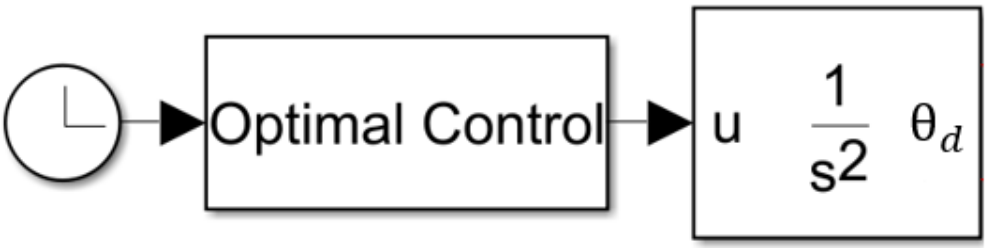
To initialize the system and choose appropriate gains, a MATLAB script containing the system parameters was written. This could also be used to iterate gains, or to graph and calculate the parameters discussed in the Results (section 4)

2.2.1 Adding a patching filter

In certain circumstances it may be less accurate to set a desired end point as a reference because the controller output could be is susceptible to offsets, overshoots, or oscillations. This is especially relevant when the gains cannot be tuned. In these cases, it may make more sense to have the controller follow a reference trajectory. By knowing the optimal control, this reference can be derived and substituted in place of θ_d.

2.2.2 Double Integrator patching filter

In a Double Integrator Patching Filter, the dynamic reference trajectory is found directly though integrating the optimal control. This will be the first filtered controller tested. See Figure 4.



2.1.2 Gain Tuned double integrator patching Filter

In this manuscript, the Double Integrator Patching Filter with gain tuning was also analyzed. This differed from the Double integrator filter, because the state and velocity trajectories were used to tune the controller. This was done by subtracting the current state from the ideal states calculated by the double integrator filter. The control is therefore defined by equation 9.

$$u = u^* + K_p(\theta^* - \theta) + K_V(\omega^* - \omega) \quad (9)$$

The asterisks in Equation 9 represents the optimal states that are calculated analytically before noise or disturbances are introduced into the system.
Gain Tuning allows the coefficients Kp and Kv to tune the error of the system.

2.1.3 Control Law Inversion patching filter

The control law patching filter works similarly to the Double Integrator Patching Filter, but a transfer function replaces the double PV filter. The transfer function relates the optimal control to the optimal state. The equation for the optimal state can be derived from Equation 3. Equation 10 below shows the solution for the optimal angular trajectory.

$$\theta_d^* = \frac{u^*}{K_p} + \theta^* + \frac{K_V}{K_p} \bullet \omega^* \quad (10)$$

The states can be rewritten as a function of the optimal control. See Equation 11

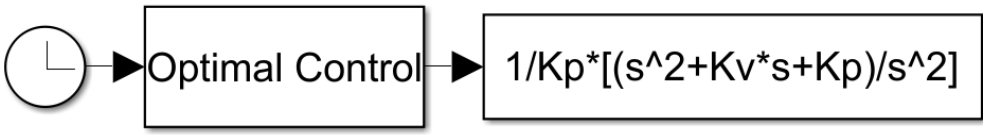
$$\theta_d^* = \frac{1}{K_p}(u^* + K_V \int u^* dt + K_p \iint \left(u^* dt\right) dt) \quad (11)$$

Applying the Laplace transform, we get the results shown in equations 12 and 13

$$X_d(s) = \frac{1}{K_p}\left(U(s) + \frac{K_V}{s} U(s) + \frac{K_p}{s^2} U(s)\right) \quad (12)$$

$$= \frac{1}{K_p}\left(\frac{s^2+K_V \cdot s+K_p \bullet s^2}{s^2}\right)\cdot U(s) \quad (13)$$

This transfer function can be used as a filter between the optimal control and the P+V Controller. Since it was derived from the control equation, the results should be more accurate than the Double Integrator Filter. See Figure 6.



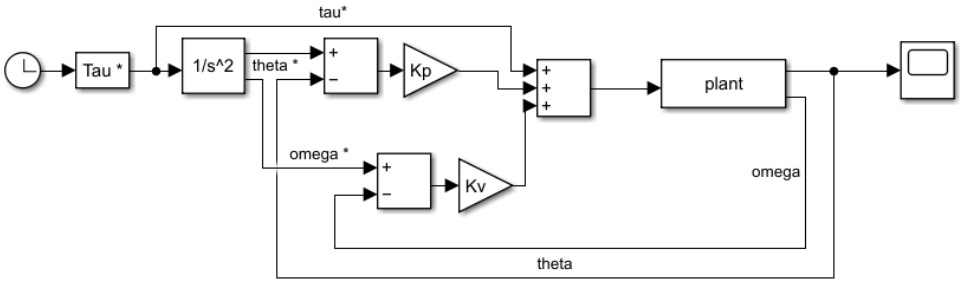
2.2 2DOF Feed Forward controller

Another control solution that was attempted, was to create a 2DOF controller that would reference the optimal control.

Using Equation 10 for the optimal trajectory θ_d, this solution can be substituted into the control equation, Equation 3. See Equation 13.

$$u = u^* + K_p(\theta^* - \theta) + K_V(\omega^* - \omega) \quad (13)$$

The solution for this filter is built differently than the P+V filters. See Figure 5 for the block diagram representation of the Gain Tuned Double Integrator Filter.



2.3. Open Loop/*RTOC controller architecture*

For the setup for the RTOC and Open Loop DQC, the dynamics are defined along with a cost function and end point conditions.

To minimize fuel usage, the cost function was built around keeping the torque at a minimum. Using the quadratic cost model, the cost function was written as seen in Equation 14.

$$Minimize\ J[x(\bullet), u(\bullet)] = \frac{1}{2}\int_{t_0}^{t_f} \tau^2 dt \quad (14)$$

Torque was squared to measure magnitude. A multiple of ½ was introduced to simplify the derivative calculation.

The state equations for a dynamic system follow Euler's equation as is shown in Equation 15.

$$\tau = I \cdot \ddot{\theta} \quad (15)$$

Using the state space form, Equation 15 can be linearized using a system of equations. The result is shown in Equation 16.

$$\begin{pmatrix} \dot{\theta} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} \omega \\ \frac{\tau}{I} \end{pmatrix} \tag{16}$$

Setting up the states, the MATH method can be applied to solve for the optimal control, u, and the optimal state trajectories. The derivation of this solution can be found in the Appendix (section 6.1). The optimal control and trajectories can be seen in Equations 17-19.

$$u = \dot{\omega}(t) = 6 - 12 \cdot t \tag{17}$$

$$\omega(t) = 6 \cdot t - 6 \cdot t^2 \tag{18}$$

$$\theta(t) = 3 \cdot t^2 - 2 \cdot t^3 \tag{19}$$

With DQC, the solution for the control will always be linear. In addition, we can rule out any constants of integration for omega or theta due to static initial conditions. This makes it so that with a DQC we only must solve for the slope and intercept of the controller.

In closed loop, the controller will have to re-evaluate the dynamic equations with every time step. With disturbances or noise to the system, the solution is constantly subject to change. To be able to recalculate the coefficients for the control by solving for equation 20.

$$[T][\bar{p}] = [\bar{b}] \ldots \begin{pmatrix} \frac{t_0^3}{6} & \frac{t_0^2}{2} t_0 & 1 & \frac{t_0^2}{2} t_0 & 1 & 0 & \frac{1}{6} & \frac{1}{2} & 1 & 1 & \frac{1}{2} & 1 & 1 & 0 \end{pmatrix} (a\ b\ c\ d) = \begin{pmatrix} \theta_0 & \omega_0 & \theta_f & \omega_f \end{pmatrix} \tag{20}$$

Considering the initial conditions and solving for the constants, equation 9 can be re-written. See equation 21.

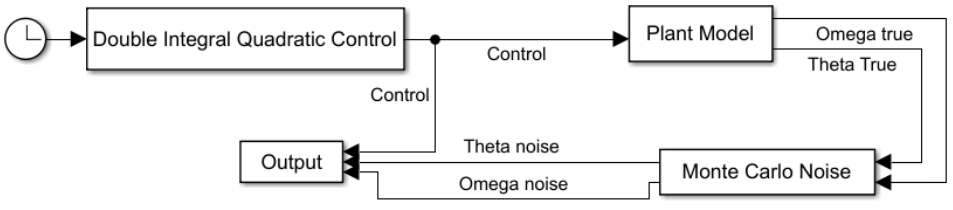
$$(a\ b\ 0\ 0) = \begin{pmatrix} \theta_0 & \omega_0 & \theta_f & \omega_f \end{pmatrix} \begin{pmatrix} \frac{t_0^3}{6} & \frac{t_0^2}{2} t_0 & 1 & \frac{t_0^2}{2} t_0 & 1 & 0 & \frac{1}{6} & \frac{1}{2} & 1 & 1 & \frac{1}{2} & 1 & 1 & 0 \end{pmatrix}^{-1} \tag{21}$$

Equation 10 is used for the feedback controller. The T matrix is a function of real time. The $\bar{p}$ vector can be determined by using the final conditions for theta and omega final. The initial conditions are reset with every feedback iteration. This yields a unique solution for elements a and b of (vector $\bar{b}$) with every loop of the controller.

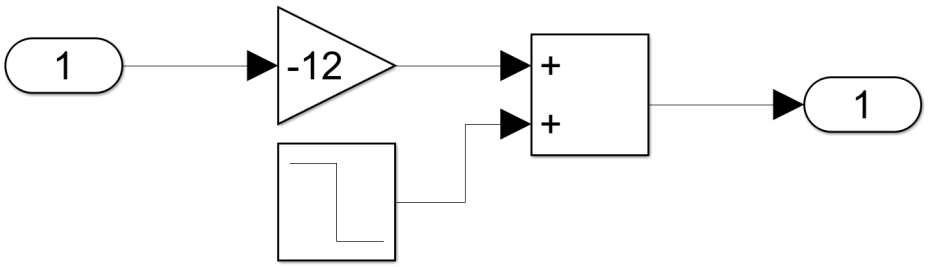
To solve for gains a and b (of vector $\bar{b}$), the inverse must be taken of matrix T. The $1 \backslash T$ command was used in MATLAB, because this method had the lowest quadratic cost. See Appendix section 7.1 for comparisons of different MATLAB inverse strategies is discussed in section 3.

2.2.1. Setting up the Open Loop DQC in Simulink

The Open Loop DQC controller can be modeled as a block diagram to represent the interactions present in the controller. The block controller was set up using Simulink. The system was mathematically structured by the dynamic governing equations (see Equation 5). The open loop control system is shown in Figure 7.

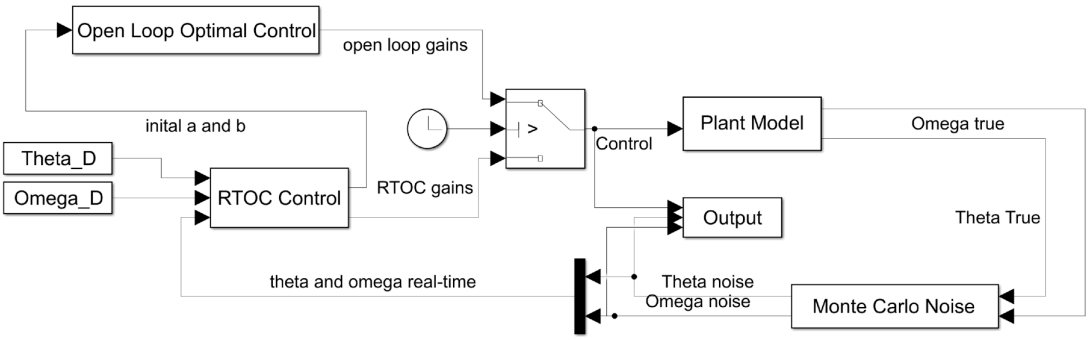


The equation for the optimal control was contained within the Quadratic Control Computation block. This block was expanded to reveal the internal block diagram. See Figure 8.

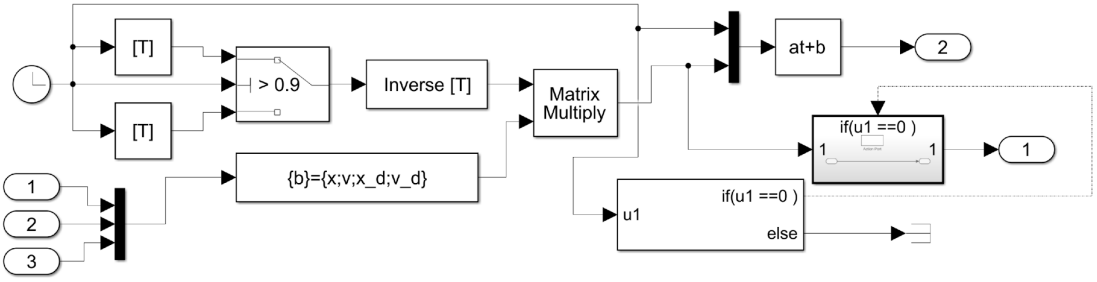


2.2.2. Setting up the RTOC in Simulink

The RTOC solution can be modeled as a block diagram to represent the interactions present in the controller. The system was mathematically structured by the dynamic governing equations (see Equation 16). The closed loop control system is shown in Figure 9. A switch exists in order to turn the closed loop solution to open loop at t~1. This is because the one over the determinant of the T matrix goes to infinity and drives the controller to become unstable.

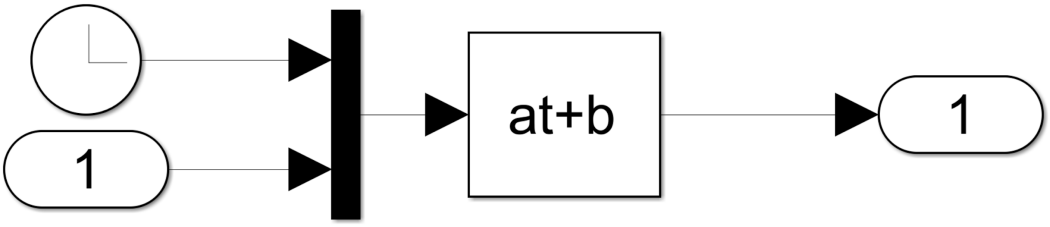


The controller takes the feedback values theta and omega and process them into new gains using Equation 21. The block diagram for this can be seen in Figure 10. This once again uses a switch to shut stop computation after the determinant approaches zero.



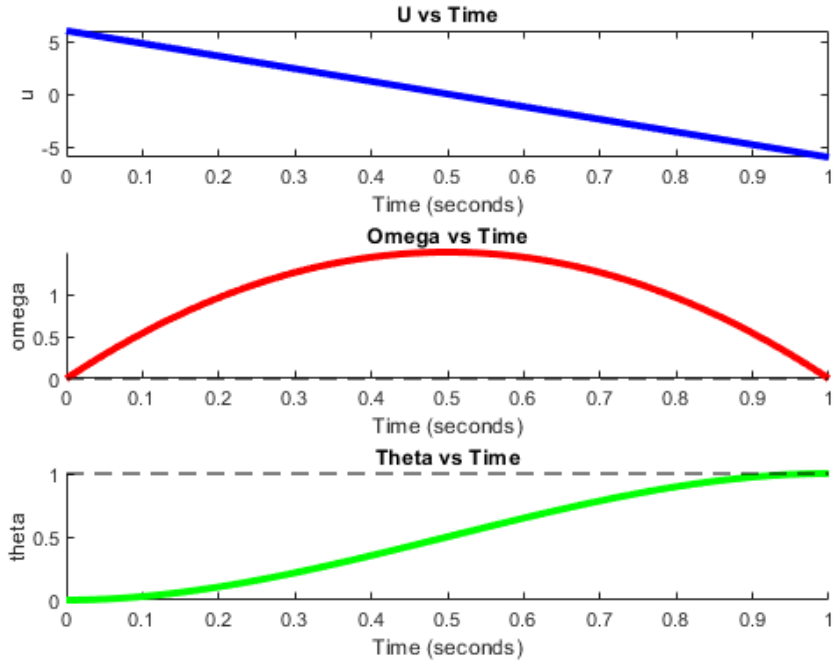
The open loop controller can be seen in Figure 10. The controller is given the initial coefficients a and b. These are the conditions of the optimal controller. The open loop

form of the RTOC controller is switched on as the controller gets reaches a singularity. Alternative gain values could be pulled from the last iteration of RTOC, before controller switches to open loop. This could have yielded better results. However initial values were chosen to guarantee rigidity of the controller.



2.2. Position and velocity plots

The optimal controller solution (u*, omega* and theta*) was graphed for angilar acceleration, velocity and position. This was done by using a Runge Kutta solver with a 0.01 second timstep from 0 to 1 seconds. See Figure 11.



As can be seen in Figure 4, bot end point conditions are exactly meet.

2.3 Quadratic cost

The equation used to derive the quadratic cost is shown in Equation 22.

$$J = \frac{1}{2}\int_{t_0}^{t_f} u(t)^2 dt \quad (22)$$

Using this method, the quadratic cost can be solved for analytically by substituting in the solution for the optimal control from Equation 6. Integrating over the time interval $t_0 = 0$ and $t_f = 1$ the quadratic cost is equal to 2.

The quadratic cost was also calculated as a function of the outputs of the Simulink model. This was done numerically since the states in the Simulink model were numerically calculated with a Euler's method ordinary differential equation (ODE) solver (delta t = 0.01). For this solution, Equation 22 was re-written in discrete form. The revised formula can be seen in Equation 23 where n represents the end point of the time vector.

$$J = \frac{1}{2} \sum_{i=2}^{n} u\left(t_i\right)^2 \bullet \left(t_i - t_{i-1}\right) \tag{23}$$

MATLAB was used to calculate the quadratic cost based on the angular acceleration and timesteps output from the Simulink simulation. The quadratic cost of the open loop DQC was calculated for each of the inverting techniques. The results can be seen in Table 1.

2.4 Rise time

The rise time was calculated by finding the time it took for the angular position, theta, to get to 90% of its target value.

2.5 Testing the robustness using a Monte Carlo Simulation

The robustness of the optimized controller was tested using a Monte Carlo Simulation in MATLAB. The Monto Carlo Simulation works to test the controller by inputing random numerical values for the parameters of the simulation. These random numbers are generated based on their predicted statistical distributions.

The Monte Carlo simulations were run 1000 times to make sure that the random variables would not case random noise.

Based on the charicteristic equations, the parameters that are subject to variability are the true moment of inertia, and the sensor feedback (theta and omega).

The actual moment of interia was detmined to be within $\pm 10\%$ of the calculated value. To represent this in the simulation, a uniform random number was generated within the tolrancing range.

3. Results

2.4. Robustness Measures of the Optimal Controller

The results of the Monte Carlo analysis on the different controllers are shown in Table 1 below. For all of these controllers, 1000 samples where averaged. The controllers where defined in MATLAB and built Simulink. A Runge Kutta solver was used at a timestep of 0.001s. This timestep was chosen becase it was determined to be optimal with rewpect to the Quadratic Cost (see section 6.2.4 of the Apendix). Each of these controllers was evaluated based on Quadratic cost, end point errors, standard diviations, and rise time.

	Quadratic Cost	*End Point Error - Theta*	*End Point Error -Omega*	*Standard diviation -Theta*	*Standard diviation -Omega*	*Rise time*

P+V Controller	*55.709082*	*8.239661e-03*	*9.394143e-02*	*1.061509e-02*	*2.301743e-02*	*3.90e-01*
P+V Double Integrator	*2.544054*	*1.216665e-01*	*1.077242e+00*	*1.044603e-02*	*2.084530e-02*	*N/A*
Double Integrator gain tuning	*6.579885*	*1.087292e-01*	*3.527516e-01*	*1.085613e-02*	*2.024489e-02*	*7.80e-01*
2DOF feed forward	*6.147382*	*2.911532e-02*	*5.954429e-02*	*1.115621e-02*	*1.699173e-02*	*8.00e-01*
P+V Control Law Inversion	*6.127791*	*2.929454e-02*	*5.920005e-02*	*1.106685e-02*	*1.935900e-02*	*8.00e-01*
Open loop guidance (DQC)	*6.181200*	*5.775071e-03*	*3.505268e-04*	*5.761962e-02*	*1.034162e-02*	*8.00e-01*
RTOC	*6.169038*	*1.865604e-05*	*7.128359e-04*	*2.710996e-03*	*4.798987e-02*	*8.10e-01*

For each of these controllers, a scatter plot was made showing each of the 1000 results along with the standard diviations. Refer to the Apendix section 6.3. In addition the trajectories where graphed with the mean value and the upper and lower saturation limits defined. Refer to the Apendix section 6.4.

4. Conclusion

When looking at these controllers, certain methods will never be competitive with respect to others. However, it is not necessarily possible to assign an optimal controller. The double integrator filter on the P+V has the lowest computational cost, but this controller struggles to get to the end point. The controller with the quickest rise time was the P+V controller, however the tradeoff is in the cost. The RTOC controller had the lowest error, but in the case of a fast characteristic time, this option may be too computationally expensive.

Many of the other options are also effective control methods and were balanced across cost, computational power/time, and endpoint error. The Open Loop DQC controller had one of the lowest run times, along with minimal error and cost. However, because this controller does not account for feedback, it would be much less robust under greater uncertainties.

5. Acknowledgements

Special thanks to Dr. Sands (Toolman) for his help on this homework.

6. Appendix

6.1. *Derivation of the Optimal Open Loop Solution using HMAT (MATH)*
Below is the derivation of the equations used the DQC methods.

$$F(t) = \frac{1}{2}\tau^2$$

$$f(\omega, u, p) = \omega$$

$$f(\theta, u, p) = \frac{\tau}{I} = \frac{1}{I}\left(K_p\left(\theta_d - \theta\right) - K_v\omega\right)$$

$$E\left(\theta(t_f)\right) = 0$$

$$E\left(\omega(t_f)\right) = 0$$

$$e\left(\theta(t_f)\right) = \theta(t_f)$$

$$e\left(\omega(t_f)\right) = \omega(t_f) - 1$$

H: $H = F(t) + \lambda_\theta(t)\cdot f(\theta, u, p) + \lambda_\omega(t)\cdot f(\omega, u, p)$

$$= \frac{1}{2}\tau^2 + \lambda_\theta(t)\cdot u + \lambda_\omega(t)\cdot\omega$$

$$\tau = uI$$

M: $\frac{\delta H}{\delta u} = 0 = \frac{\delta}{\delta u}\left(\frac{1}{2}(uI)^2 + \lambda_\omega(t)\cdot u + \lambda_\theta(t)\cdot\omega\right) = \tau I + \lambda_{w(t)} \ldots$

$$\tau = -\frac{\lambda_w(t)}{I}\ldots u = -\frac{\lambda_w(t)}{I^2} = \dot{\omega}(t)$$

A: $\frac{\delta H}{\delta\theta} = -\dot{\lambda}_\theta(t) = 0; \quad \frac{\delta H}{\delta\omega} = -\dot{\lambda}_\omega(t) = \lambda_\theta(t) = \alpha\ldots - \lambda_\theta(t) = \alpha_0\cdot t + \beta_0$

T: $\bar{E} = E\left(\theta\left(t_f\right)\right) + E\left(\omega\left(t_f\right)\right) + \nu_\theta\cdot e\left(\theta\left(t_f\right)\right) + \nu_\omega\cdot e\left(\omega\left(t_f\right)\right)$

$$= \nu_\theta\cdot\theta\left(t_f\right) + \nu_\omega\cdot\left(\omega\left(t_f\right) - 1\right)$$

$$\frac{\delta\bar{E}}{\delta\theta\left(t_f\right)} = \lambda_\theta\left(t_f\right) = \nu_\theta\,; \frac{\delta\bar{E}}{\delta\omega\left(t_f\right)} = \lambda_\omega\left(t_f\right) = \nu_\omega;$$

$$-\lambda_\theta(t) = \alpha_0\cdot t + \beta_0; \ \left(\alpha = \frac{\alpha_0}{I}, \beta = \frac{\beta_0}{I}\right) and\ u = \dot{\omega}(t) = -\frac{\lambda_\theta(t)}{I^2} = \alpha\cdot t + \beta$$

$$\omega(t) = \frac{\alpha}{2}\bullet t^2 + \beta\cdot t + \gamma$$

$$\omega(0) = 0; \ \omega\left(t_f\right) = \omega(1) = 0$$

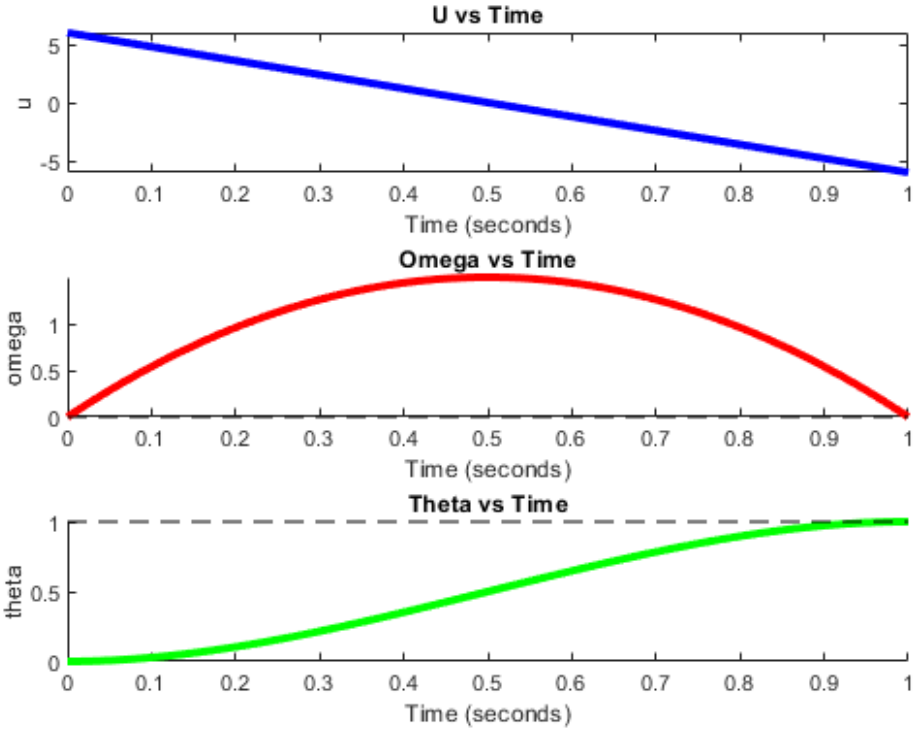
$$\gamma = 0; \ \beta = -\frac{\alpha}{2}$$

$$\theta(t) = \frac{\alpha}{6}\bullet t^3 - \frac{\alpha}{4}\bullet t^2 + \delta$$

$$\theta(0) = 0; \ \theta\left(t_f\right) = \theta(1) = 1$$

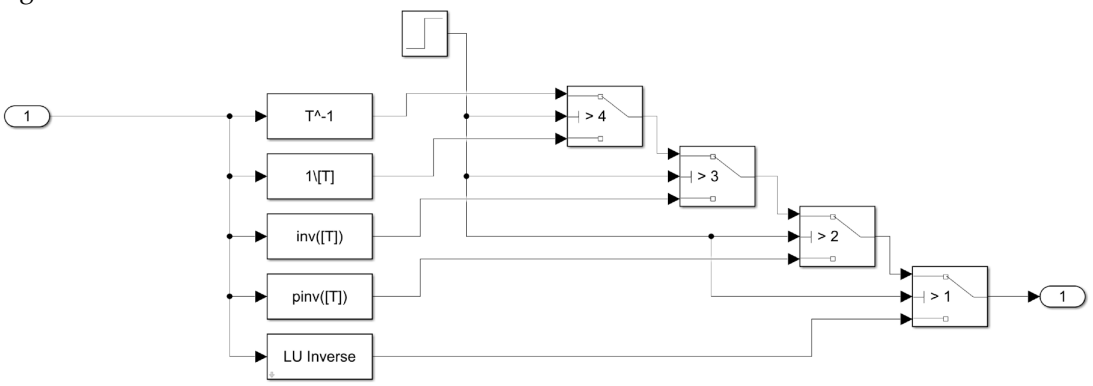
$$\delta = 0; \ \frac{\alpha}{12} = -1; \ \alpha = -12; \ \beta = 6$$

Figure 11 shows the graphical representation of the optimal solution.



6.2.1 Data for best inverstion technique with respect to Quadratic cost

Five different strategies were used to invert the matrix T. The in MATLAB the following inverting techniques were tested: T^-1, 1\T, inv(T), pinv(T), and LU Inverse. The switch cases for can be seen below in Figure 3. An iterative variable is used in a step function to change the method used. This is done in the MATLAB script though a for loop. See Figure 12.

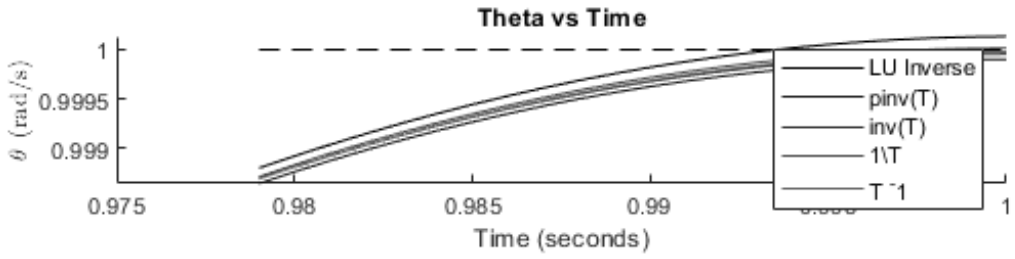


6.2.2. Finding the end point error for each of the methods

The end point value was calculated by taking the instantaneous output at t=1.

For most of the controller algorighems, the endpoint condition was closely reached, with mnimal end point error.

Plots of the angular position and velocity as a function of time where generated based on the mean after running the Monte Carlo simulation. Because the results for each inverse method was so similar, only the end points are shown when t > 0.98. Once again, the Runge Kutta solver was used along with a 0.001 timestep. The results can be seen in Figure 13.



The error at the end point conditions was calculated by taking the mean value at the final time (t=1) and subtracting the set point value. The angular position and velocity errors can be found in Table 2.

6.2.3. Results table

	Quadratic Cost	*End Point Error - Theta*	*End Point Error - Omega*
T^-1	*6.028903*	*0.000076*	*0.001585*
1\T	*5.992644*	*0.000092*	*0.001903*
inv(T)	*6.012007*	*0.000029*	*0.001884*
pinv(T)	*5.996595*	*0.000087*	*0.001760*
LU Inverse	*6.031820*	*0.000065*	*0.001248*

6.2.4 Best step time

The RTOC feedback controller was also evaluted based on the optimal step time. The Matlab method $1\backslash T$ was used since it was evaluated to be the method with the lowest quadratic cost. In this case, the step size was iterated from 0.1 to 0.0001 seconds for the RTOC and open loop contoller blocks. The timestep for the plant and Monte Carlo systems was kept at the fastest timestep (0.0001s) since in practice, these should refresh instantanioulsy. The results are shown in Table 3.

Step Size (s)	*Quadratic Cost*	*End Point Error - Theta*	*End Point Error - Omega*
0.1	*6.049908*	*0.000032*	*0.000558*
0.01	*5.999301*	*0.000073*	*0.000458*

0.001	*5.998285*	*0.000080*	*0.000718*
0.0001	*6.0142*	*-6.7499e-05*	*0. 000377*

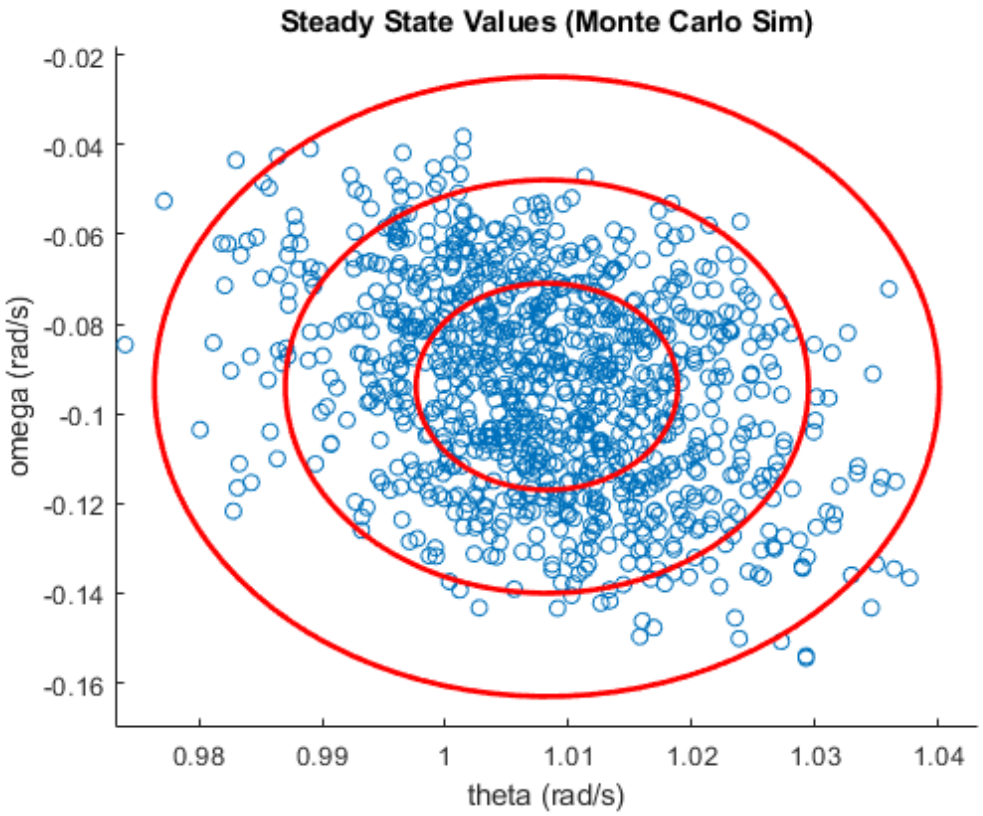
No timestep smaller than 1e-4 was attempted. This is becasuse not only did the quadratic cost increase again (so an optimal cost timstep was found), but the simulation took almost 25 hours to run.

As could be expected, when the timestep decreased the error tended to decrease. However, if optimizing for the lowest quadratic cost, a timestep of 0.001 seconds was the most efficient for the controller

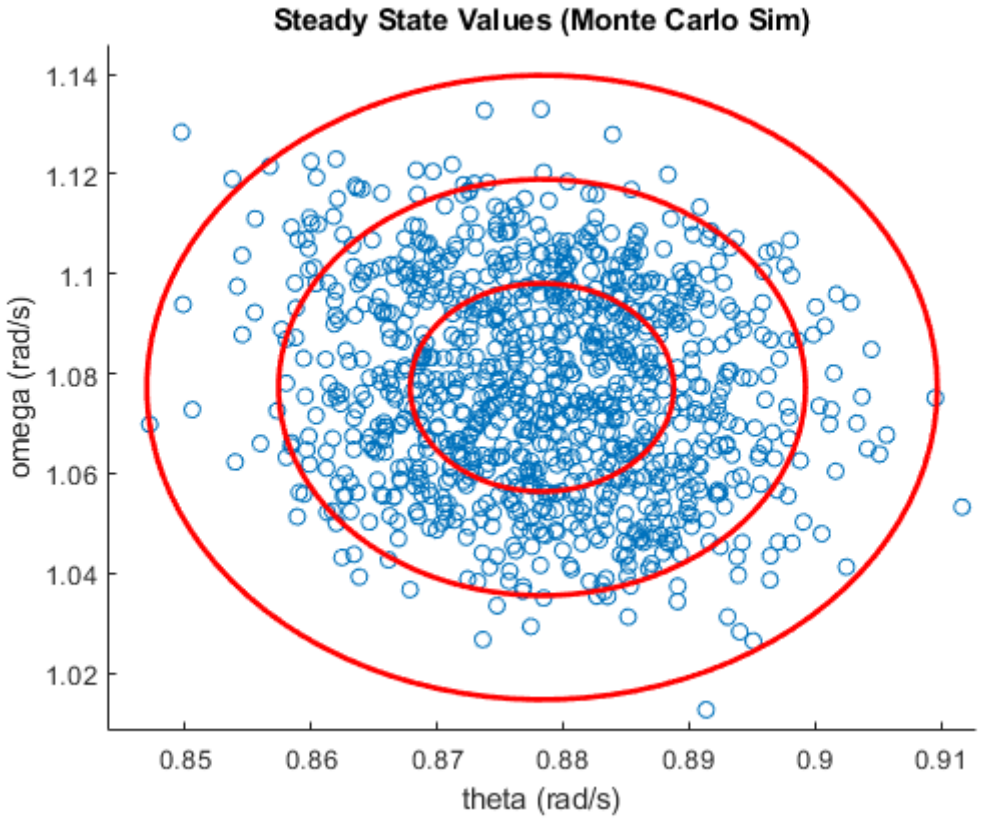
6.3 Scatter plots of the controllers with n=1000

The following plots are the result of plotting each of the 1000 runs of done in the Monte Caro Simulation for each of the controllers. For each of these, they were run with a Runge Kutta solver at a fixed timestep of 0.001

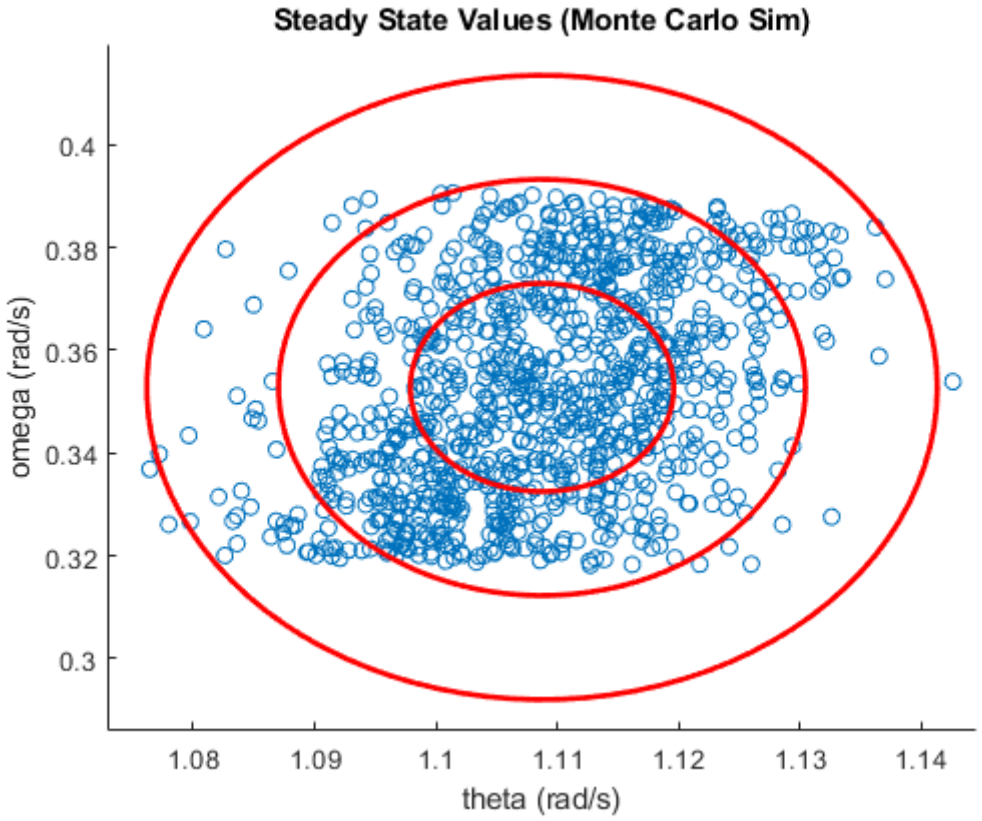
1) P + V controller



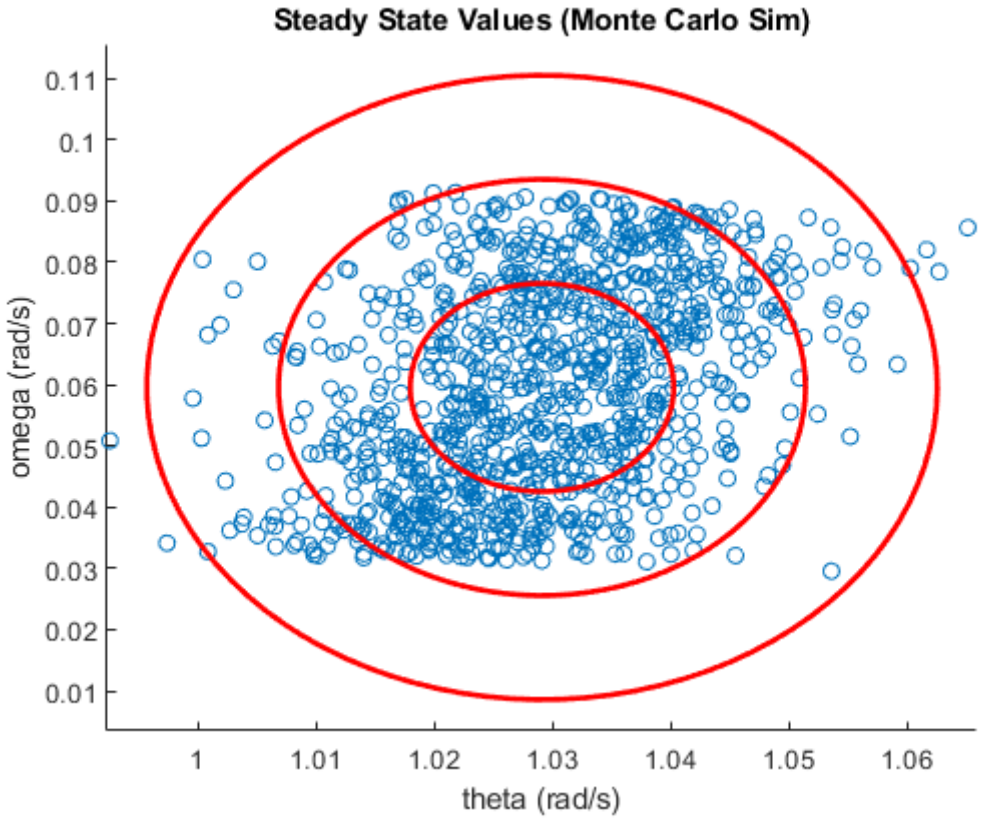
2) P+V with a Double Integrator patching filter



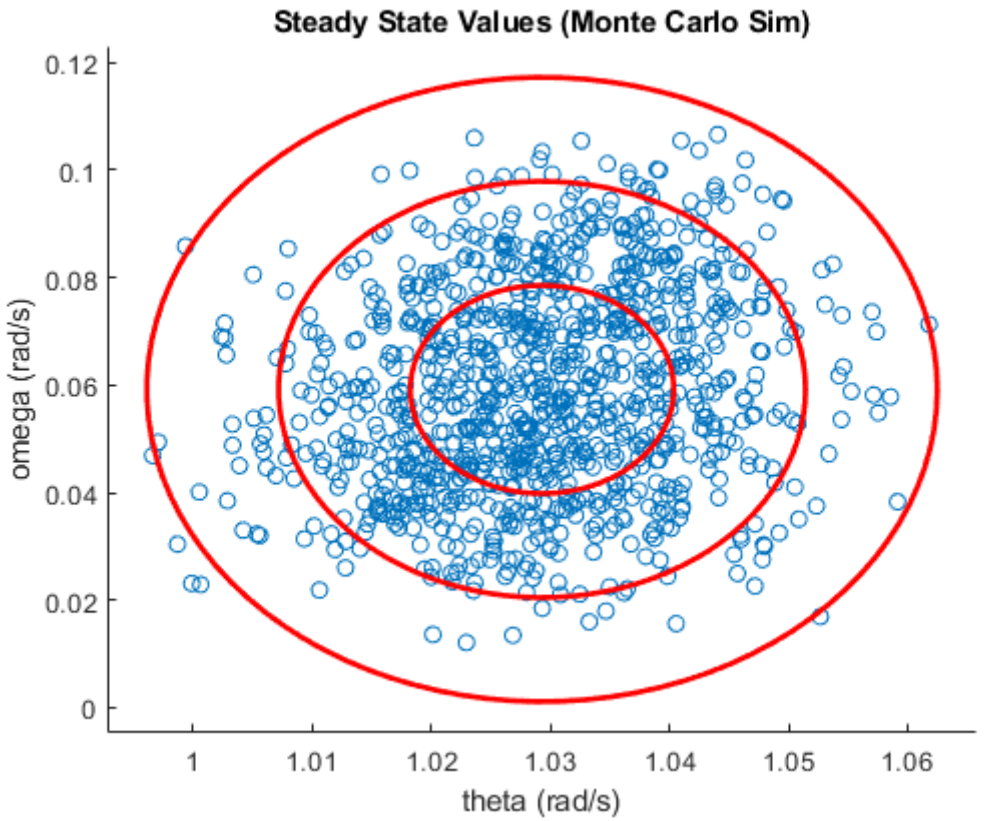
3) Double Integrator patching filter and gain tuning



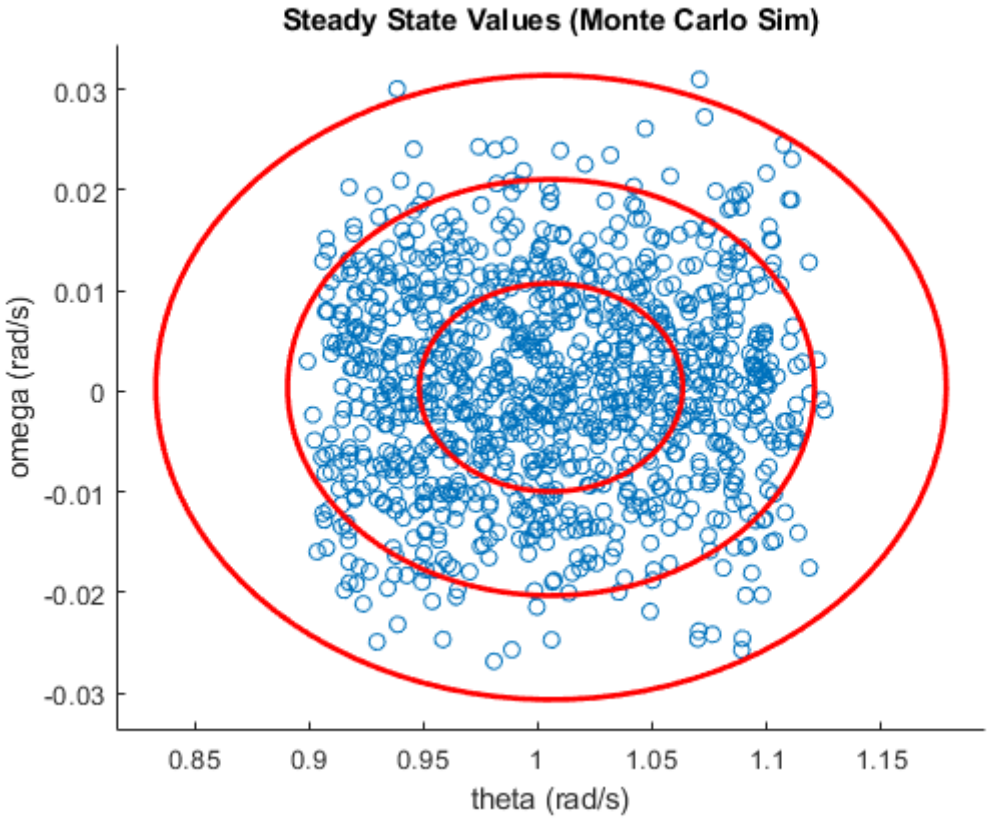
4) 2DOF Feed Forward controller



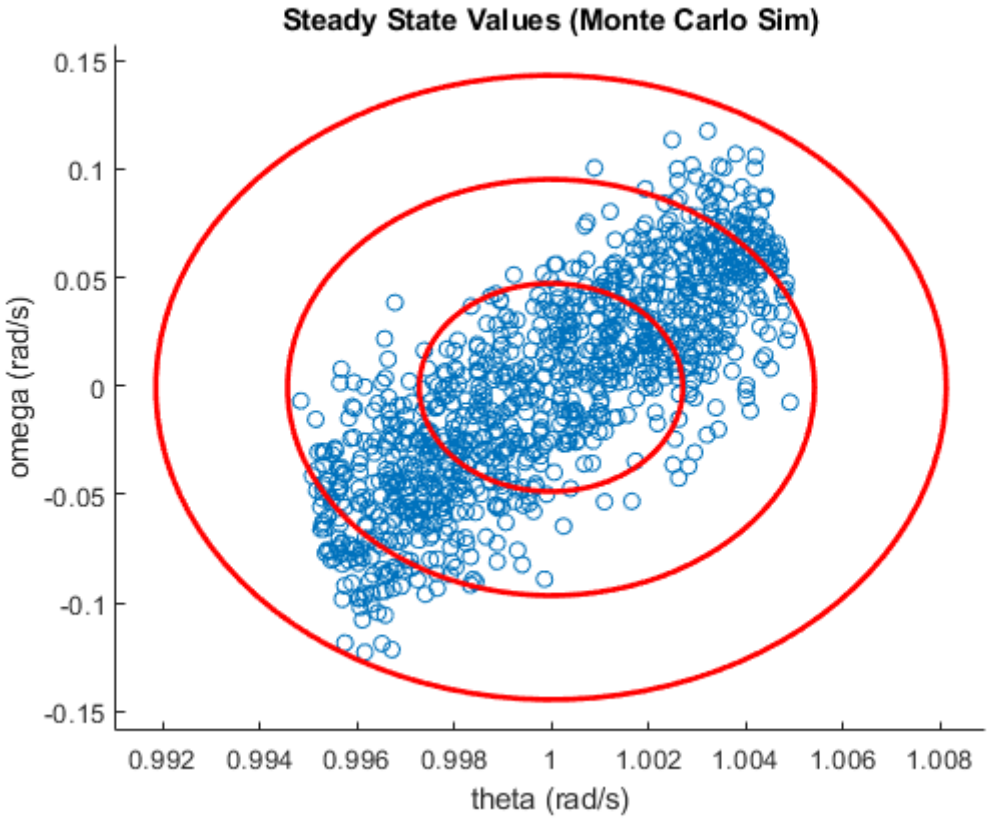
5) P+V with a Control Law Inversion patching filter



6) Open loop guidance (Double Integral Quadratic Controller)

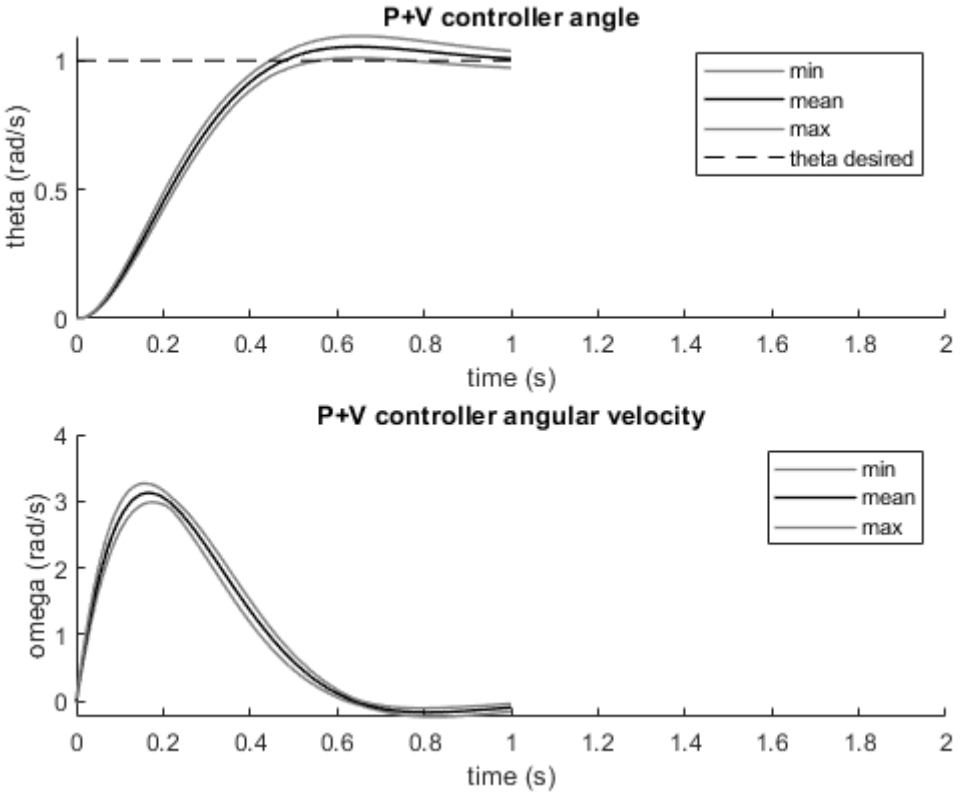


7) RTOC

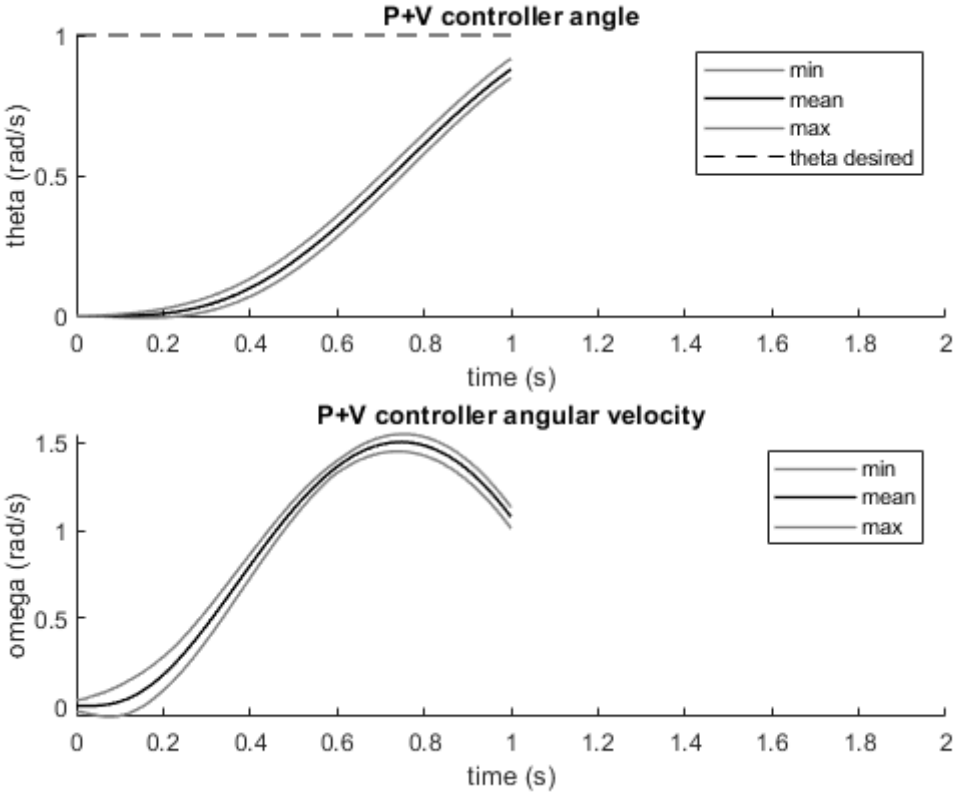


6.4 Trajectories of the controllers with n=1000

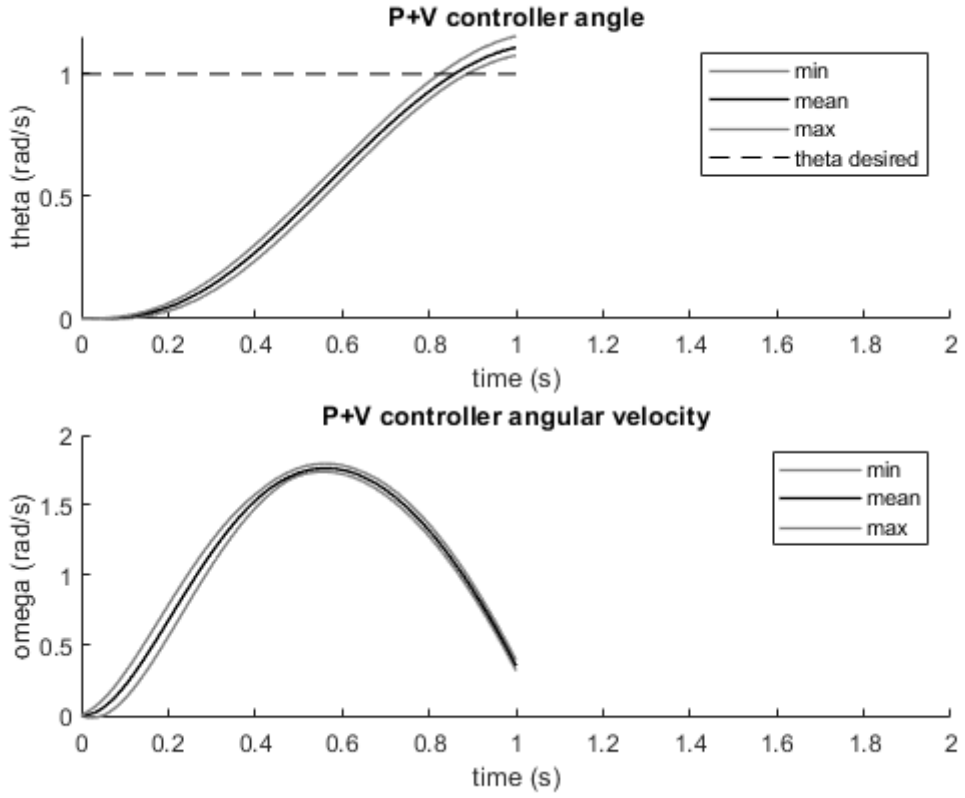
1) P + V controller



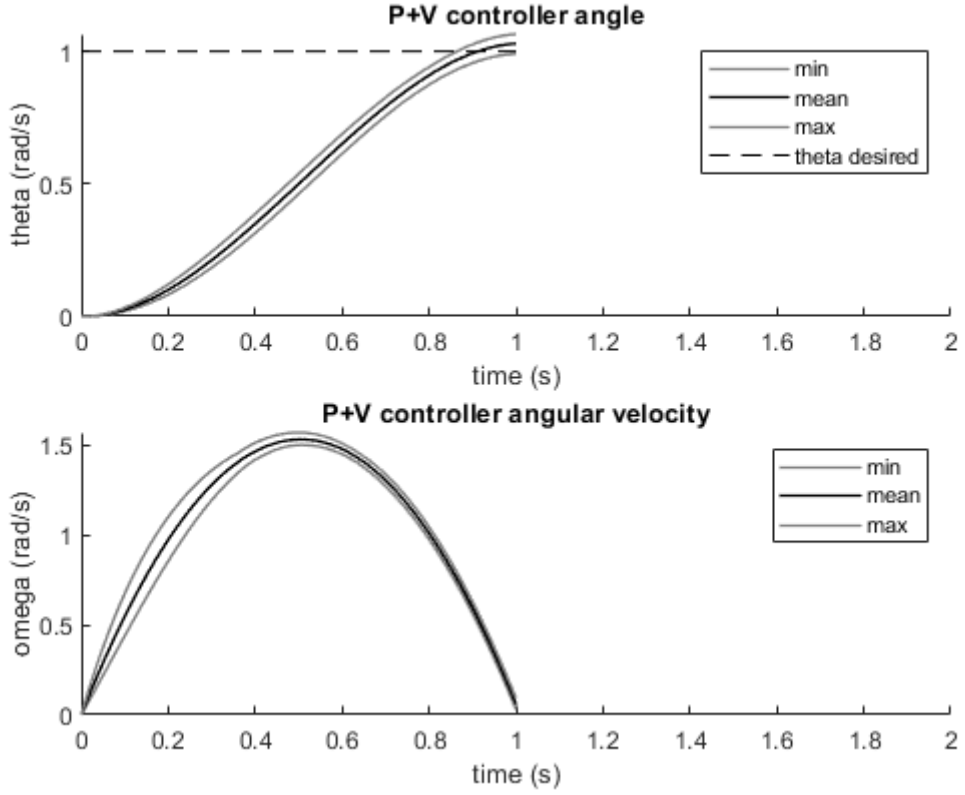
2) P+V with a Double Integrator patching filter



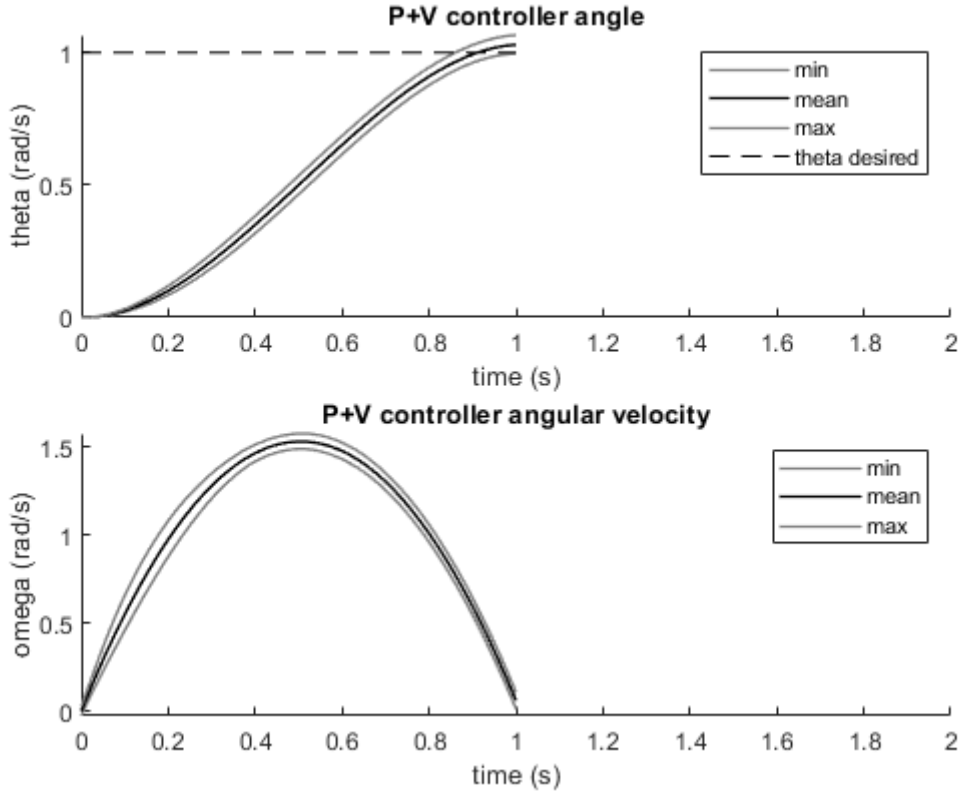
3) Double Integrator patching filter and gain tuning



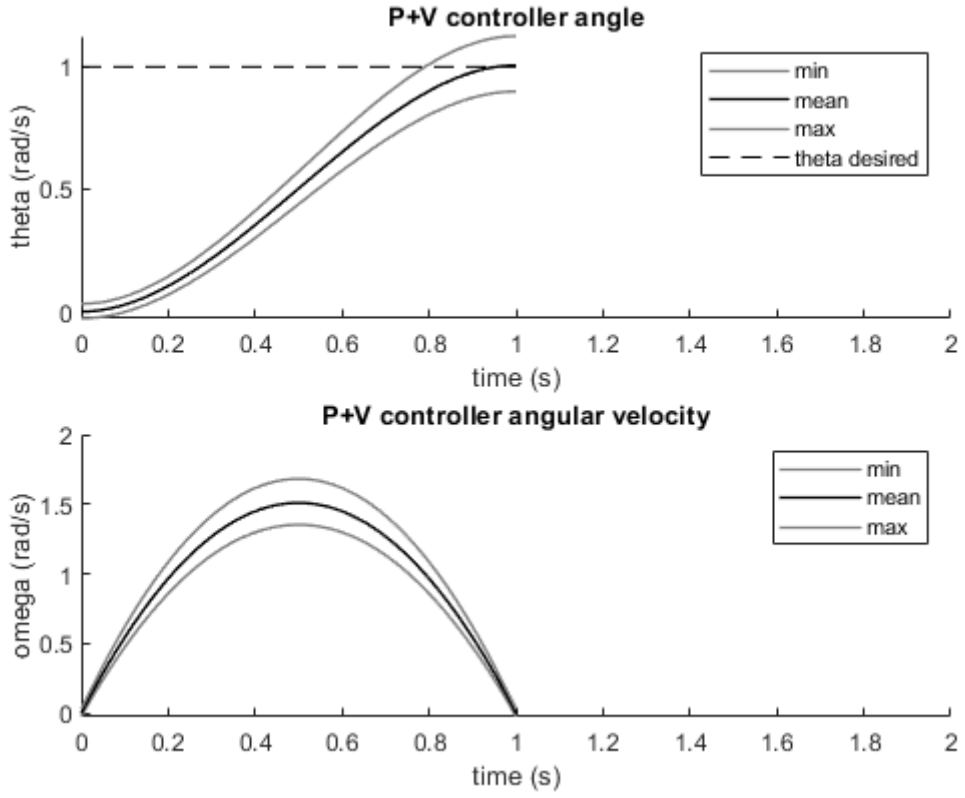
4) 2DOF Feed Forward controller



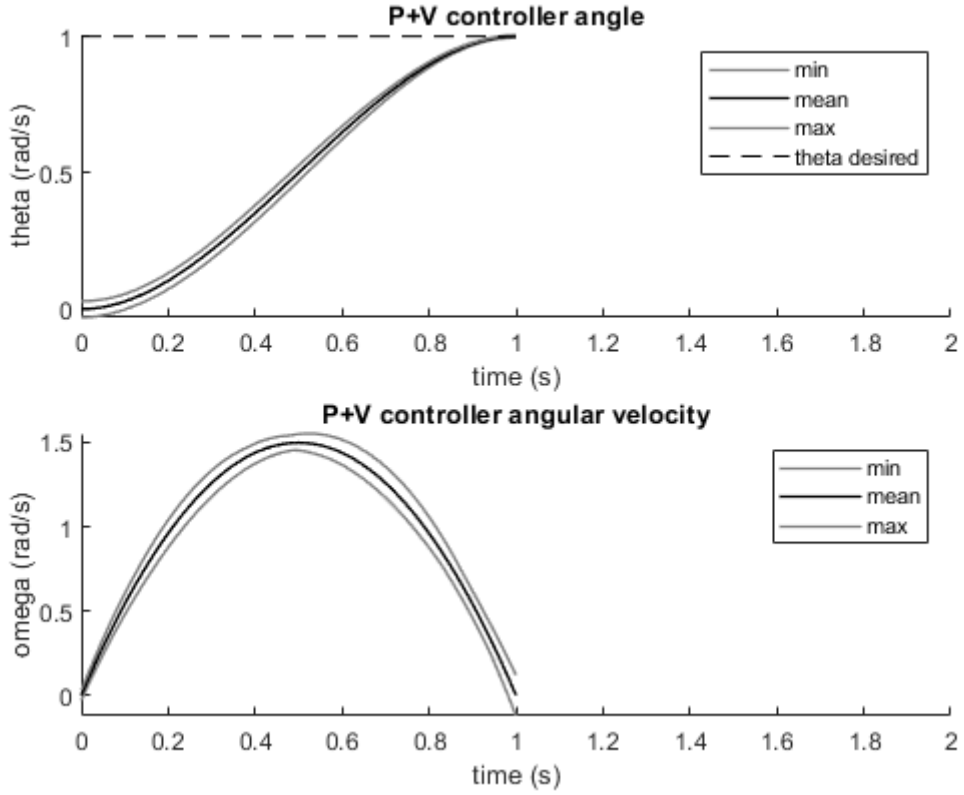
5) P+V with a Control Law Inversion patching filter



6) Open loop guidance (Double Integral Quadratic Controller)



7) RTOC


P+V controller angle
theta (rad/s)
time (s)
min
mean
max
theta desired
P+V controller angular velocity
omega (rad/s)
time (s)
min
mean
max