EE5104 Adaptive Control Systems

AY2021/2022 Semester 2

Continuous Assessment 3

Philippe Andrin Brigger
A0248834U

Contents

1 Introduction

This *CA3* report is part of the spring 2022 module EE5104 Adaptive Control Systems at the National University of Singapore (NUS). Adaptive control system design is a design methodology which can be used to control systems with initially uncertain or varying parameters. The control law non-linearly adapts the controller gains to match the system output to a reference model output. The task of *CA3* is to design and simulate an adaptive angular position controller for a DC motor where all of the state variables are measurable. The first part of the report details the design process of the adaptive control system. As the adaptive control approach is non-linear, and this bears the risk of signals diverging at an arbitrary speed, the first part of the report focuses on rigorously showing that the tracking error of the system converges to zero and that all signals remain bounded. Next, the report discusses the simulation results of the proposed controller and examines the effects that different design choices have on the performance of the system. Finally, the appendix contains the code used to run the simulations. In general, all vectors are printed in bold while scalars and matrices are printed as normal text.

2 Adaptive Control System Design

2.1 Perfect Control Input

An adaptive angular position controller is designed for a DC motor where all states are measurable. The two states of the system are the angular velocity $\omega = \dot{\theta}$ and the angular position θ. A DC motor can be modeled as a first order system Σ_{DC}. A first order system is characterized by its steady-state gain k and time constant τ.

$$\Sigma_{DC} = \frac{k}{\tau s + 1} \tag{1}$$

The input control signal u of the DC motor is a voltage. Accordingly, after calibrating the DC motor, the angular velocity $\omega = \dot{\theta}$ of the DC motor is measurable with a tachogenerator while the angular position θ is measurable with a potentiometer. The motor is calibrated by measuring the angles and angular velocities of the motor for different input voltages. From this data, the linear gains k_ω, k_θ which map the output voltages to angular velocity and position can be calculated with linear regression. Figure 1 shows the DC motor model.

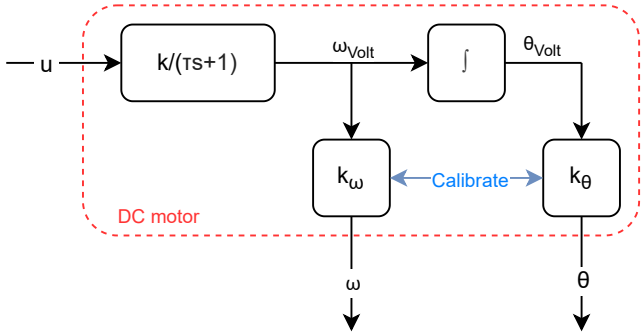


Figure 1: The DC motor is modeled as a first order system. The calibration gains k_ω, k_θ relate the output voltages to the system states angular velocity ω and angular position θ.

If the motor is calibrated correctly, the state vector $\boldsymbol{x_p} = [\theta_p, \omega_p]^T$ is measurable in real-time. The subscript p refers to properties belonging to the plant which in this case is the DC motor. With a motor model set up, a state space realization of the plant can be formulated:

$$\dot{\boldsymbol{x}}_{\boldsymbol{p}} = A_p \boldsymbol{x_p} + g\boldsymbol{b}u \tag{2}$$
$$\boldsymbol{y_p} = C_p \boldsymbol{x_p} \tag{3}$$

Plugging the parameters of eq. (1) into the state space realization gives:

$$\dot{\boldsymbol{x}}_{\boldsymbol{p}} = \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \frac{k}{\tau} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\boldsymbol{y_p} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \omega \end{bmatrix}$$

The goal of the adaptive control design is to make the DC motor response match a reference model. The reference model is chosen by the engineer and should be chosen

to meet the desired performance requirements. For the case of angular position control, the reference model should be a second order system since a second order derivative of the angular position θ_p with respect to time is present in the DC motor transfer function. The transfer function (which can be derived from fig. 1) of the DC motor for the state θ_p namely is:

$$\Sigma_{DC(\theta_p)} = \frac{\theta_p(s)}{U(s)} = \frac{1}{s}\frac{k}{\tau s + 1} = \frac{k}{\tau s^2 + s}$$

The transfer function of the second order reference model is:

$$H_m(s) = \frac{\theta_m(s)}{R(s)} = \frac{\omega_m^2}{s^2 + 2\zeta_m\omega_m s + \omega_m^2}$$

The subscript m denotes properties that belong to the reference model. The design parameters of the reference model are the natural frequency ω_m and damping coefficient ζ_m. The main differences between the reference model and actual plant are that the input to the reference model is the reference signal $r(t)$ and not the plant control input signal $u(t)$, and that the states $\boldsymbol{x_m}$ are the states of the reference model. By writing the reference model in state space form and comparing the state space form to eq. (2), the perfect DC motor control input u^* which will make the plant output match the reference model output can be derived.

$$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = A_m \boldsymbol{x_m} + g_m \boldsymbol{b} r \tag{4}$$
$$\boldsymbol{y_m} = C_m \boldsymbol{x_m} \tag{5}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = \begin{bmatrix} \dot{\theta_m} \\ \dot{\omega_m} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_m^2 & -2\zeta_m\omega_m \end{bmatrix} \cdot \begin{bmatrix} \theta_m \\ \omega_m \end{bmatrix} + \omega_m^2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\boldsymbol{y_m} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta_m \\ \omega_m \end{bmatrix}$$

State feedback is necessary to make the DC motor output match the reference model output. The DC motor voltage control input $u(t)$ has the following form where $\boldsymbol{\theta_x}$ and θ_r are the controller gains.

$$u(t) = \boldsymbol{\theta_x}^T \boldsymbol{x_p} + \theta_r r(t) \tag{6}$$

Plugging eq. (6) into eq. (2) gives:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{p}} &= A_p \boldsymbol{x_p} + g\boldsymbol{b}\big(\boldsymbol{\theta_x}^T \boldsymbol{x_p} + \theta_r r(t)\big) \\ &= \big(A_p + g\boldsymbol{b}\boldsymbol{\theta_x}^T\big)\boldsymbol{x_p} + g\boldsymbol{b}\theta_r r \end{aligned} \tag{7}$$

If the perfect control gains $\boldsymbol{\theta_x} = \boldsymbol{\theta_x^*}$ and $\theta_r = \theta_r^*$ are chosen, then eq. (7) must match eq. (4).

$$A_m = \big(A_p + g\boldsymbol{b}\boldsymbol{\theta_x^*}^T\big)$$
$$g_m = g\theta_r^*$$

The perfect control input thus would be:

$$\begin{aligned} u^*(t) &= \boldsymbol{\theta_x^*}^T \boldsymbol{x_p} + \theta_r^* r(t) \\ \boldsymbol{b}\boldsymbol{\theta_x^*}^T &= \frac{1}{g}\big(A_m - A_p\big) \\ \theta_r^* &= \frac{g_m}{g} \end{aligned} \tag{8}$$

This DC motor control input would make the DC motor response perfectly match the reference model response. Unfortunately, the perfect control gains depend on the plant state matrix A_p and plant gain g which depend on the steady-state gains k and time constant τ of the DC motor. These parameters are uncertain. They may change over time due to e.g. mechanical wear or temperature fluctuations. Thus, the perfect control gains cannot be calculated. To overcome this deficiency, an adaptive control approach may be adopted which starts with an initial guess of the control gains and then adapts the gains in an optimal way to make the output error between the DC motor and reference model converge to zero.

2.2 Adaptive Control Input

One way to control a system with uncertain parameters is to use adaptive control gains $\boldsymbol{\theta}(t)$. In the case of a DC motor, it makes sense to use an adaptive control approach because the DC motor parameters will change over time due to e.g. mechanical wear or temperature effects [3]. The adaptive control law will turn out to be nonlinear. Therefore, the design must guarantee that all the signals stay bounded and the error of the system converges to zero. This is because nonlinear systems bear the risk of having signals diverge arbitrarily quick if they become unstable. In summary, the adaptive control law must ensure:

- Output error $e_y = y_p - y_m$ converges to zero
- Bounded input $u(t)$ and bounded output $y(t)\ \forall\ t$

Since the output $y = C\boldsymbol{x}$ solely depends on the state vector $\boldsymbol{x}$ and C is the time-invariant system matrix, the output error e_y is guaranteed to converge if and only if the state vector error $\boldsymbol{e} = \boldsymbol{x_p} - \boldsymbol{x_m}$ converges to zero. Therefore, it is sufficient to design the adaptive control input $u(t)$ in a way that the state vector error $\boldsymbol{e}$ converges to zero.
Let $\boldsymbol{\Psi_x} = \boldsymbol{\theta_x}(t) - \boldsymbol{\theta}^*$ and $\Psi_r = \theta_r(t) - \theta_r^*$ define the control gain errors. Plugging the adaptive control gains $\boldsymbol{\theta_x}(t)$ and $\theta_r(t)$ into eq. (6) and eq. (2) gives a new state space description of the DC motor:

$$\begin{aligned}
\dot{\boldsymbol{x}}_{\boldsymbol{p}} &= \big(A_p + g\boldsymbol{b}\boldsymbol{\theta_x}^T\big)\boldsymbol{x_p} + g\boldsymbol{b}\theta_r r + g\boldsymbol{b}\boldsymbol{\theta_x^*}^T\boldsymbol{x_p} - g\boldsymbol{b}\boldsymbol{\theta_x^*}^T\boldsymbol{x_p} + g\boldsymbol{b}\theta_r^* r - g\boldsymbol{b}\theta_r^* r \\
&= \big(A_p + g\boldsymbol{b}\boldsymbol{\theta_x^*}^T\big)\boldsymbol{x_p} + g\boldsymbol{b}\theta_r^* r + g\boldsymbol{b}\Big[\big(\boldsymbol{\theta_x}^T - \boldsymbol{\theta_x^*}^T\big)\boldsymbol{x_p} + \big(\theta_r - \theta_r^*\big)r\Big] \\
&= A_m\boldsymbol{x_p} + g_m\boldsymbol{b}r + g\boldsymbol{b}\big[\boldsymbol{\Psi_x}^T\boldsymbol{x_p} + \Psi_r r\big]
\end{aligned}$$

It will be shown that one adaptive control law which makes the state error $\boldsymbol{e}$ converge to zero is:

$$\begin{aligned}
\dot{\boldsymbol{\theta}}_{\boldsymbol{x}} &= -sgn(g)\Gamma\boldsymbol{e}^T P\boldsymbol{b}\boldsymbol{x_p} \\
\dot{\theta}_r &= -sgn(g)\gamma\boldsymbol{e}^T P\boldsymbol{b}r
\end{aligned} \tag{9}$$

Considering that in eq. (9) $\boldsymbol{e} = \boldsymbol{x_p} - \boldsymbol{x_m}$, the adaptive control is nonlinear since the squared state vector $\boldsymbol{x_p}$ appears in the equation. In general, it is hard to prove global asymptotic stability of a nonlinear system $\dot{\boldsymbol{x}}(\boldsymbol{t}) = f(\boldsymbol{x}(t))$. One way to prove global asymptotic stability is with Lyapunov's Direct Method [2] and Barbalat's lemma [1].

2.2.1 Lyapunov's Direct Method

Lyapunov's Direct Method can be used to determine the stability of an equilibrium point of a nonlinear system. If a Lyapunov function $V(x)$ which depends on the state

vector of the system meets certain requirements, then the equilibrium point of the system x_e will be globally stable. In this case, a Lyapunov function $V(\boldsymbol{e})$ depending on the state vector error $\boldsymbol{e}$ must be found since then it would be ensured that the equilibrium point $\boldsymbol{e}_e = 0$ is globally stable (The goal is to have the error converge to zero making $\boldsymbol{e}_e = 0$ the desired equilibrium point). Barbalat's lemma then can be used to prove that the state error $\boldsymbol{e}$ converges to zero and that the DC motor output y_p matches the reference model output y. This would then make the overall system globally asymptotic stable. $V(\boldsymbol{e})$ must meet the following requirements to be a Lyapunov function to ensure that a system is globally stable:

- $V(\boldsymbol{e})$ is positive definite
- $V(\boldsymbol{e})$ is radially unbounded
- $\dot{V}(\boldsymbol{e})$ is negative semi-definite

If it can be shown that $\dot{V}(\boldsymbol{e})$ is negative definite then the equilibrium point would be globally asymptotically stable. The proposed function $V(\boldsymbol{e})$ is:

$$V(\boldsymbol{e}) = \frac{1}{2}\boldsymbol{e}^T P \boldsymbol{e} + \frac{1}{2}|g|\boldsymbol{\Psi_x}^T \Gamma^{-1} \boldsymbol{\Psi_x} + \frac{1}{2}|g|\frac{1}{\gamma}\Psi_r^2 \tag{10}$$

It is to be noted that the gain errors $\boldsymbol{\Psi_x}$ and Ψ_r depend on the state error $\boldsymbol{e}$ (see eq. (9)). Furthermore, Γ is a positive definite diagonal gain matrix and γ is a positive gain scalar. Both can be chosen by the engineer. P is a symmetric positive definite matrix which results from the Algebraic Riccati equation $A_m^T P + P A_m = -Q$. In the latter equation Q is a symmetric positive definite matrix which can be chosen by the engineer and A_m is the asymptotically stable system matrix from the reference model.
With these properties and the fact that $\boldsymbol{e}$, $\boldsymbol{\Psi_x}$, Ψ_r get squared in eq. (10), it is trivial that $V(\boldsymbol{e})$ is positive definite.

- $V(\boldsymbol{e})$ positive definite: $V(\boldsymbol{e}) > 0\,, \quad \forall \|\boldsymbol{e}\| > 0$

Additionally, it is obvious that $V(\boldsymbol{e})$ is radially unbounded since $V(\boldsymbol{e})$ will grow to infinity if $\boldsymbol{e}$ grows to infinity.

- $V(\boldsymbol{e})$ radially unbounded: $V(\boldsymbol{e}) \to \infty$ as $\|\boldsymbol{e}\| \to \infty$

The last point to prove is that $\dot{V}(\boldsymbol{e})$ is negative semi-definite. The time derivative of $V(\boldsymbol{e})$ is:

$$\dot{V}(\boldsymbol{e}) = \boldsymbol{e}^T P \dot{\boldsymbol{e}} + |g|\boldsymbol{\Psi_x}^T \Gamma^{-1} \dot{\boldsymbol{\Psi}}_x + |g|\frac{1}{\gamma}\Psi_r \dot{\Psi}_r \tag{11}$$

In the equation above, $\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_p - \dot{\boldsymbol{x}}_m$ which can be found from the state space descriptions.

$$\begin{aligned}
\dot{\boldsymbol{e}} &= \dot{\boldsymbol{x}}_p - \dot{\boldsymbol{x}}_m \\
&= A_m \boldsymbol{x_p} + g_m \boldsymbol{b} r + g\boldsymbol{b}\big[\boldsymbol{\Psi_x}^T \boldsymbol{x_p} + \Psi_r r\big] - \big(A_m \boldsymbol{x_m} + g_m \boldsymbol{b} r\big) \\
&= A_m \boldsymbol{e} + g\boldsymbol{b}\big[\boldsymbol{\Psi_x}^T \boldsymbol{x_p} + \Psi_r r\big]
\end{aligned}$$

Plugging $\dot{\boldsymbol{e}}$ into eq. (11) yields:

$$\dot{V} = \boldsymbol{e^T} P A_m \boldsymbol{e} + \boldsymbol{e}^T P g \boldsymbol{b} \boldsymbol{\Psi_x}^T \boldsymbol{x_p} + \boldsymbol{e}^T P g \boldsymbol{b} \Psi_r r + |g|\boldsymbol{\Psi_x}^T \Gamma^{-1} \dot{\boldsymbol{\Psi}}_x + |g|\frac{1}{\gamma}\Psi_r \dot{\Psi}_r$$

From the above equation, it can be seen that the fourth term cancels out the second term while the fifth term cancels out the third term if $\dot{\boldsymbol{\Psi}}_x$ and $\dot{\Psi}_r$ are defined as in the adaptive control law in eq. (9). This leaves $\dot{V} = \boldsymbol{e^T} P A_m \boldsymbol{e}$. It is to be noted that a square matrix A can be split up into a symmetric part A_s and an anti-symmetric part A_{as}:

$$A = A_s + A_{as} = \frac{1}{2}\big(A + A^T\big) + \frac{1}{2}\big(A - A^T\big)$$

Since PA_m is a square matrix, this relationship, together with the Algebraic Riccati equation, can be exploited:

$$\begin{aligned} PA_m &= \big([PA_m]_s + [PA_m]_{as}\big) = \frac{1}{2}\big(PA_m + [PA_m]^T\big) + \frac{1}{2}\big([PA_m - [PA_m]^T\big) \\ &= \frac{1}{2}\big(PA_m + A_m^T P\big), \text{ since } P = P^T, \text{ and } PA_m - A_m^T P = 0 \\ \Rightarrow PA_m &= -\frac{1}{2}Q \end{aligned}$$

Plugging this back into eq. (11) yields:

$$\dot{V}(\boldsymbol{e}) = -\frac{1}{2}\boldsymbol{e}^T Q \boldsymbol{e}$$

Since Q is a symmetric positive definite matrix and $\boldsymbol{e}$ gets squared in the equation above, $\dot{V}(\boldsymbol{e})$ must be negative semi-definite.

- $\dot{V}(\boldsymbol{e})$ negative semi-definite: $\dot{V}(\boldsymbol{e}) \leq 0\,, \quad \forall\, \boldsymbol{e}$

Since the function defined in eq. (10) fulfills the three requirements of being positive definite, being radially unbounded and having a negative semi-definite derivative, it can be said that $V(\boldsymbol{e})$ is a Lyapunov function and thus the solution $\boldsymbol{e}(t) = 0$ is globally Lyapunov stable. As a result, $\|\boldsymbol{e}\|$ must be bounded. However, it is not clear if $\boldsymbol{e}(t) = 0$ is asymptotically stable. Therefore, it also remains unclear if $\boldsymbol{e}(\boldsymbol{t})$ converges to zero under all starting conditions. To prove this, Barbalat's lemma may be used.

2.2.2 Barbalat's Lemma

Barbalat's lemma states that a function $f(t)$ converges to zero if the function is uniformly continuous and if the function has an integral with a finite limit. In other words, if it can be shown that $f(t) = \boldsymbol{e}^T Q \boldsymbol{e}$ is uniformly continuous and the integral $\int_{t_0}^{t} f(\tau)d\tau = \int_{t_0}^{t} \boldsymbol{e}^T(\tau) Q \boldsymbol{e}(\tau) d\tau$ has a finite limit, then $\boldsymbol{e}^T Q \boldsymbol{e} \to 0 \; as \; t \to \infty$. Since Q is a symmetric positive definite matrix, this immediately implies that $\boldsymbol{e}(t) \to 0$ $as \; t \to \infty$.
A function $f(t)$ is uniformly continuous if it fulfills the following condition:

$$\exists\, \epsilon > 0 \; \forall\, \delta > 0\,, \; \exists\, x_0 \,\epsilon\, \mathbb{R}\,, \; \exists\, x \,\epsilon\, \mathbb{R} \left[|x - x_0| < \delta \, and \, |f(x) - f(x_0)| \geq \epsilon\right]$$

Since $f(t)$ depends on $\boldsymbol{e}(t)$ which is bounded and Q which is constant, there will be a δ and ϵ which fulfill the definition of uniform continuity. Next, since the Lyapunov function $V(\boldsymbol{e})$ is positive definite and has a negative semi-definite derivative, it can be said that $V(\boldsymbol{e}(t = 0))$ imposes an upper bound on V. Since the first term of $V(\boldsymbol{e})$ depends on the squared state error $\boldsymbol{e}$ and the constant symmetric positive definite matrix P, integrating this term will have a finite limit. Furthermore, since P depends on Q it can be said that the integration of $\boldsymbol{e}^T Q \boldsymbol{e}$ has a finite limit.

$$\int_{t_0}^{t} f(\tau)d\tau = \int_{t_0}^{t} \boldsymbol{e}^T(\tau) Q \boldsymbol{e}(\tau) d\tau < \infty$$

Since $\boldsymbol{e}^T Q \boldsymbol{e}$ meets the two conditions of Barbalat's lemma and Q remains positive and greater than zero, it results that $\boldsymbol{e}$ must converge to zero.

- Barbalat's lemma: $\boldsymbol{e} \to 0 \; as \; t \to \infty$

Barbalat's lemma proves that the state error signal $\boldsymbol{e}$ converges to zero while the existence of a Lyapunov function $V(\boldsymbol{e})$ proves that the equilibrium point $\boldsymbol{e} = 0$ is globally stable. The combination of the both means that the equilibrium point $\boldsymbol{e} = 0$ is globally asymptotically stable. This means that regardless of the initial state error, the adaptive control law will manage to stabilize the DC motor and the state error will converge to zero. With the state error converging to zero, the output error inherently also converges to zero and the DC motor will match the desired reference model. Furthermore, since $\|\boldsymbol{e}\|$ is bounded, all the signals $\|\boldsymbol{x_p}\|$, $\|\boldsymbol{x_m}\|$, $\|y_p\|$, $\|u\|$, $\|\boldsymbol{\theta_x}\|$, and $\|\theta_r\|$ will stay bounded. This is thanks to the fact that the latter signals all depend on $\boldsymbol{e}$. Additionally, the reference signal $r(t)$ is chosen by the engineer and thus chosen to be bounded while the reference model is designed as an asymptotic stable system. Therefore the output y_m of the reference model will be bounded.

- $\boldsymbol{\theta_x}(t)$ bounded because: $\boldsymbol{\theta_x} = \boldsymbol{\theta_x}(\boldsymbol{e}) = -\int sgn(g) \Gamma \boldsymbol{e}^T P \boldsymbol{b} \boldsymbol{x_p} + \boldsymbol{\theta_x^0}$
- $\theta_r(t)$ bounded because: $\theta_r = \theta_r(\boldsymbol{e}) = -\int sgn(g) \gamma \boldsymbol{e}^T P \boldsymbol{b} r + \theta_r^0$
- $u(t)$ bounded because: $u(t) = \boldsymbol{\theta_x}^T \boldsymbol{x_p} + \theta_r r(t)$
- $y_p(t)$ bounded because: $y_p(t) = C_p \boldsymbol{x_p} = C_p(\boldsymbol{x_p} - \boldsymbol{x_m} + \boldsymbol{x_m}) = C_p \boldsymbol{e} + y_m$

To conclude, the adapative control law defined in eq. (9) makes the DC motor respond as desired. The steady-state error $e_y = y_p - y_m$ converges to zero and the control input $u(t)$ as well as plant output $y_p(t)$ remain bounded at all times. Furthermore, the state signals and control gains remain bounded at all times. This follows from the proof of the state error $\boldsymbol{e} = \boldsymbol{x_p} - \boldsymbol{x_m}$ being globally asymptotically stable with Lyapunov's Direct Method and Barbalat's lemma. To sum up, the DC motor controlled with the adaptive control in eq. (9) has the following properties:

- Output error $e_y = y_p - y_m$ converges to zero
- State error $\boldsymbol{e} = \boldsymbol{x_p} - \boldsymbol{x_m}$ converges to zero
- Bounded input $u(t)$ and bounded output $y(t)$ $\forall$ t
- Bounded system states $\boldsymbol{x_p}$ $\forall$ t
- Bounded control gains $\boldsymbol{\theta}(t)$ $\forall$ t
- Globally asymptotically stable

3 Simulation Results

3.1 DC Motor

3.1.1 Calibration

A DC motor takes a voltage as input. A tachogenerator is used to measure the output voltage of the angular velocity while a potentiometer is used to measure the output voltage of the angular position. It is hard to grasp how a certain measured output voltage physically translates to an angular velocity in rad/s or an angular position in rad. Therefore, the output voltages of the tachogenerator and potentiometer need to be mapped to an angular velocity and position in rad/s respectively rad. This is done with a calibration step. First, a dataset with a large range of angular velocities and positions for different output voltages is logged. From this dataset, the calibration parameters k_ω and k_θ can be found via linear regression. Since this project is only simulation based, datasets for the calibration were already given. The linear regression resulted in the calibration gains being:

- $k_\omega = 0.137$
- $k_\theta = 0.628$

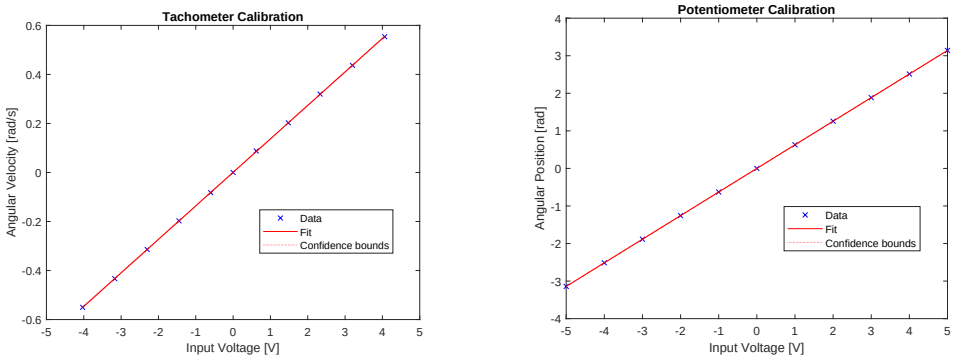


Figure 2: Calibrating tachometer and potentiometer via linear regression.

Figure 2 justifies calculating the calibration gains via linear regression as a linear relationship between the output voltages and angular velocity and position is clearly observable.
The adaptive control law defined in eq. (9) depends on the sign of g. Even though $g = k/\tau$ depends on the uncertain steady-state gain k and time constant τ of the DC motor, $sign(g)$ can be evaluated by simulating the DC motor open-loop with a positive input voltage. If the resulting output angular velocity and position are positive, then $sign(g)$ is positive too. The open-loop simulation showed that the DC motor starts to turn in the positive direction if a positive input voltage is chosen. Therefore it can be said that $sign(g)$ is positive, since the DC motor has a positive steady-state gain k (the time constant τ per definition is positive).

- $sign(g) = +1$

3.1.2 Parameters

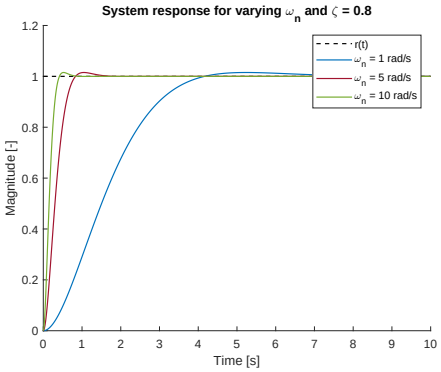
To be able to simulate the DC motor, a time constant τ and steady-state gain k had to be specified. The parameters were given in the project description as:

- $k = 6.2rad/(sV)$
- $\tau = 0.25s$

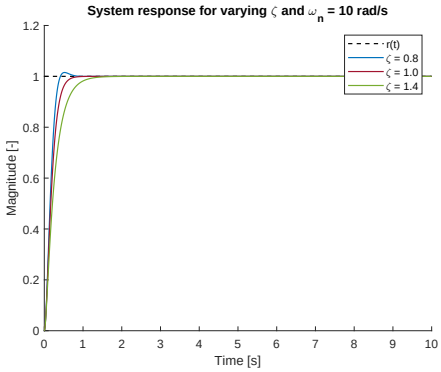
Additionally, the maximum input voltage was limited to $\pm 5V$.

3.2 Reference Model

The goal of the controller is to make the DC motor output match the reference model output. Therefore, the choice of reference model determines what the performance of the DC motor will look like. The two defining parameters are the damping coefficient ζ and natural frequency ω_n. A system responds quicker and more aggressively to input changes the higher the natural frequency ω_n is. As for the damping coefficient, the system either can be underdamped ($0 < \zeta < 1$), critically damped ($\zeta = 1$) or overdamped ($zeta > 1$). The damping coefficient will influence how quickly the system responds to a change in input and also whether the system response will have an overshoot.



(a) Reference model response with different natural frequencies ω_n.



(b) Underdamped ($0 < \zeta < 1$), critically damped ($\zeta = 1$) and overdamped ($zeta > 1$) reference model response.

Figure 3: The choice of natural frequency ω_n and damping coefficient ζ determine how the reference model response looks like.

From fig. 3 it was decided to use a reference model with $\omega_n = 10[rad/s]$ and $\zeta = 1.0$. This gives a reference model response which is quick and doesn't have any overshoot.

3.3 Perfect Control Input

If the DC motor parameters are known, then the perfect control gains can be calculated with the equations in eq. (8). With these gains, the DC motor matches the reference model output. Figure 4 shows the implementation of the perfect control input $u^*(t)$.
As is visible in fig. 5, the DC motor matches the reference model if perfect control gains are chosen for a square and sinusoidal reference signal. The reason for a short transient phase with the square reference signal has to do with the fact that the calibration gains of the DC motor don't perfectly map the output voltage to the angular velocity and position since they were calculated via linear regression from an experimental data set.

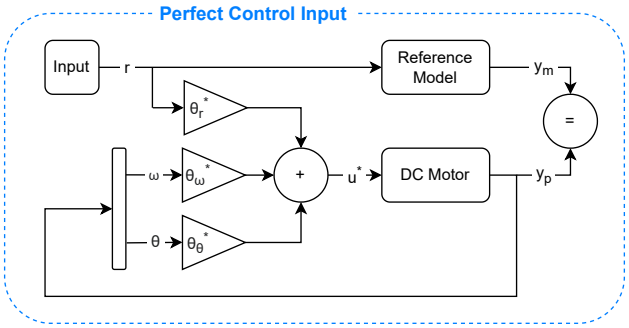


Figure 4: Perfect control input. DC motor output matches reference model output.

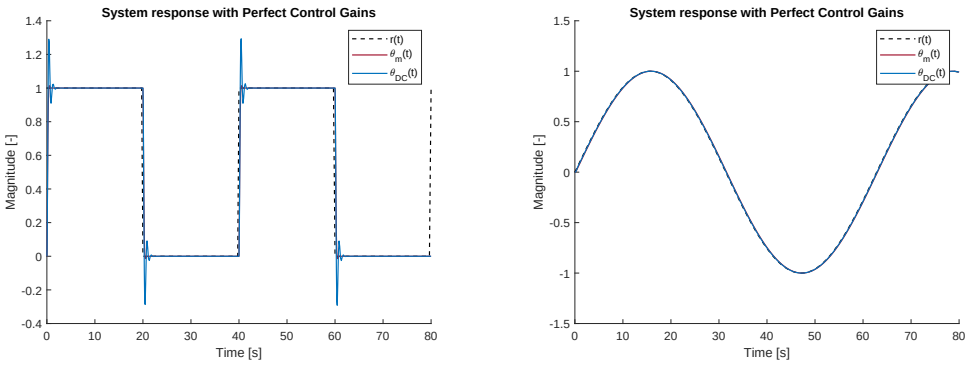


Figure 5: The DC motor response matches the reference model response.

3.4 Adaptive Control Input

Since the DC motor parameters are not known, the perfect control gains cannot be calculated in practice. Therefore, an adaptive control approach as illustrated in fig. 6 is used. The adaptive control gains with the dynamics defined in eq. (9) are used. The gains start with an initial guess. In this case all of the adaptive control gains started as zero and adapted over time to make the DC motor match the reference model. Finding the right design parameters requires an iterative approach. The first design parameter which was tuned was the value for the symmetric positive definite matrix Q.
Figure 7 shows that a larger Q gives a quicker and more aggressive system response. In other words, the system takes less time to converge to the reference model output. The response is however more oscillatory and the control input increases. The

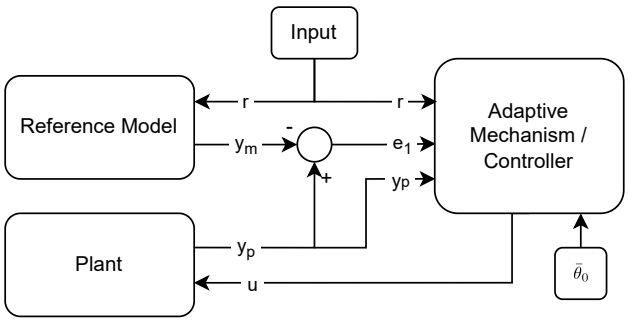


Figure 6: Adaptive control scheme.

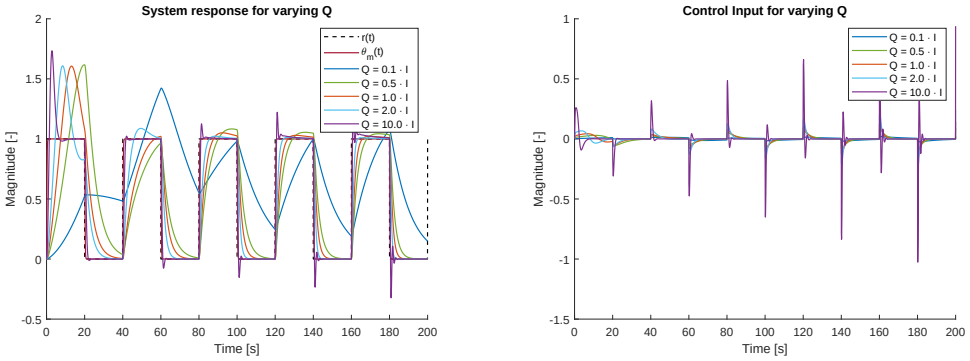


Figure 7: DC motor response and control input for varying Q matrix.

system response with $Q = 2 \cdot \mathbb{I}$ was not oscillatory but still quick. Therefore, $Q = 2 \cdot \mathbb{I}$ was kept for the further simulations. Next, the influence of varying the control gains Γ and γ was analyzed.

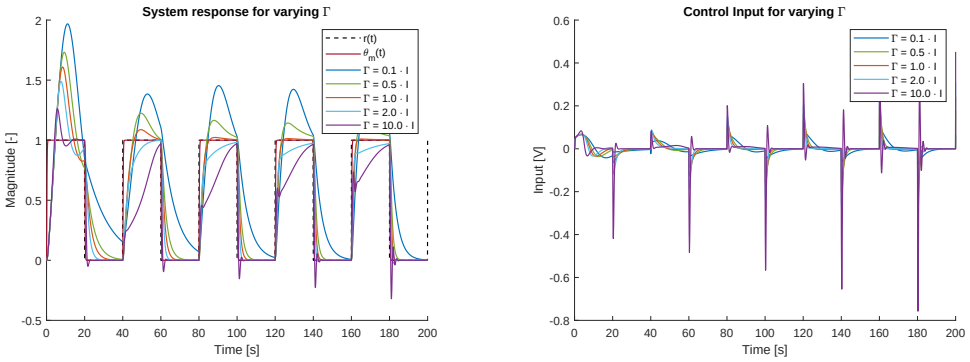


Figure 8: DC motor response and control input for varying Γ control gain matrix.

As can be seen in fig. 8 and fig. 9, the DC motor output converges to the reference model output quickest when $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$. Additionally, the control input remains lower than for other control gains and remains within the maximum input voltage bound.
Figure 10 shows that the DC motor output converges to the reference model output as described in section 2. In fig. 11 one can observe that the gains get adapted over time to get the desired system response. It is to be noted that the gains never converge to the perfect control gains. The signals remain bounded though.
Figure 12 furthermore shows that the adaptive controller can also track time-varying (here sinusoidal) reference signals well. Since the controller does not have any feed-forward elements the tracking error oscillates around zero instead of reaching a constant state of zero error. This is due to the fact that the reference signal is constantly varying but the controller cannot predict the changes. Thus, the controller is constantly reacting and adapting to the changing reference signal with a slight delay.

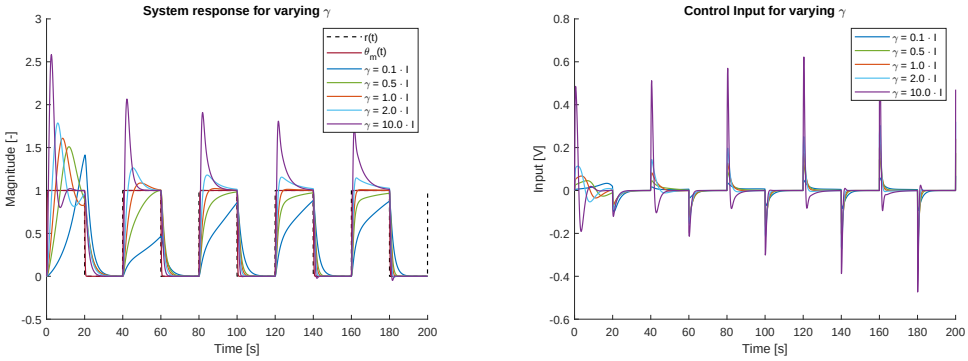


Figure 9: DC motor response and control input for varying γ control gain.

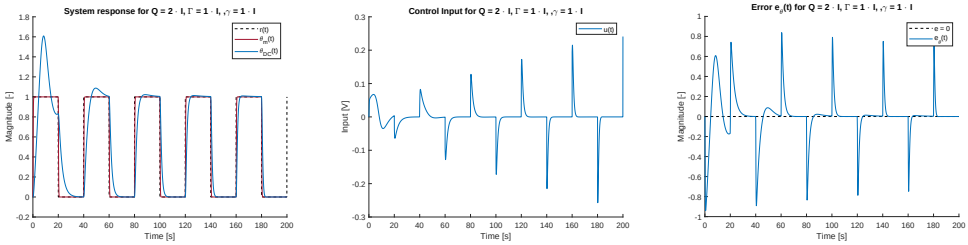


Figure 10: DC motor response, control input and output error for $Q = 2 \cdot \mathbb{I}$, $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$ and a square reference signal.

3.4.1 Simulation with Noise

To assess the robustness of the adaptive controller, additional simulations with band limited white noise acting on the input and output signals were run. The signals had a power spectral density of 0.01 and a correlation time of 0.01 s.
Figure 14 and fig. 16 show that the DC motor tracks the reference model much worse when noise acts on the input and output. The tracking especially decreases for a square wave reference signal. Additionally, fig. 15 and fig. 17 show that the adaptive controller adapts the gains quicker and greater than in the noise-free simulations.

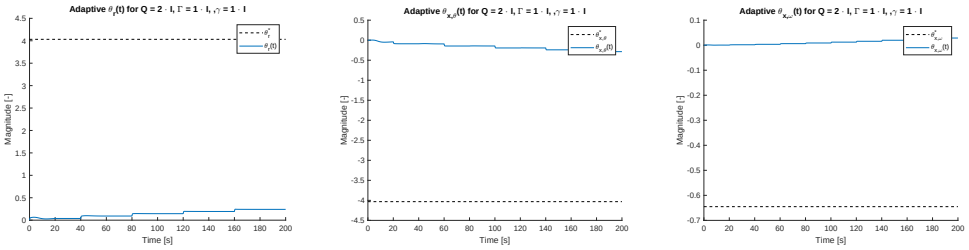


Figure 11: Controller gain adaptation for $Q = 2 \cdot \mathbb{I}$, $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$ and a square reference signal.

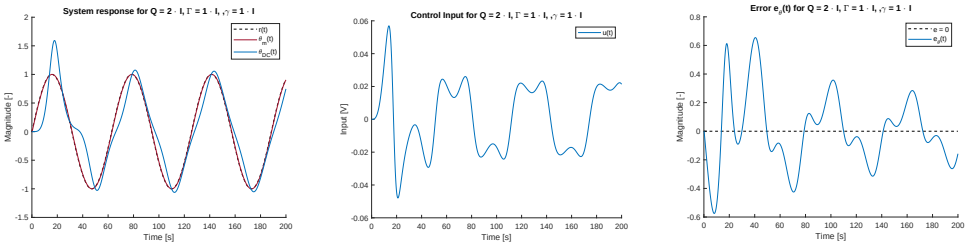


Figure 12: DC motor response, control input and output error for $Q = 2 \cdot \mathbb{I}$, $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$ and a sinusoidal reference signal.

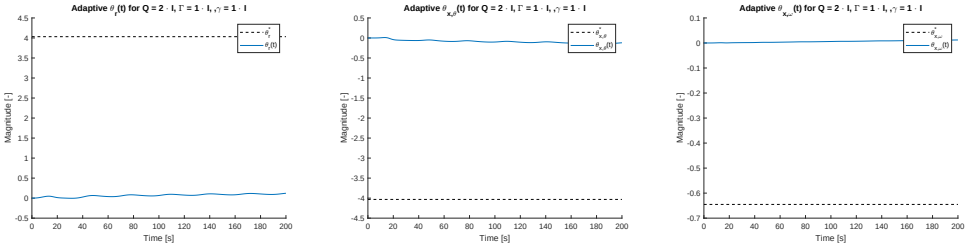


Figure 13: Controller gain adaptation for $Q = 2 \cdot \mathbb{I}$, $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$ and a sinusoidal reference signal.

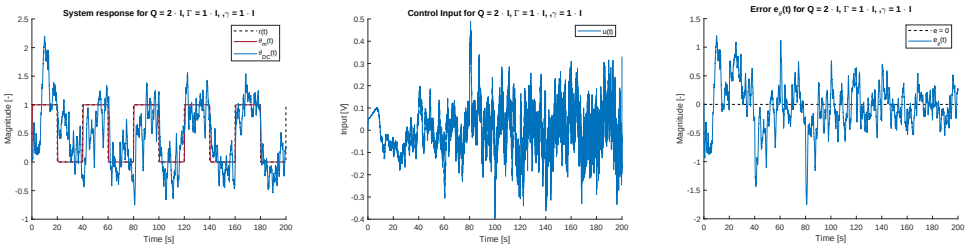


Figure 14: DC motor response, control input and output error for $Q = 2 \cdot \mathbb{I}$, $\Gamma = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$, white noise acting on input and output and a square reference signal.

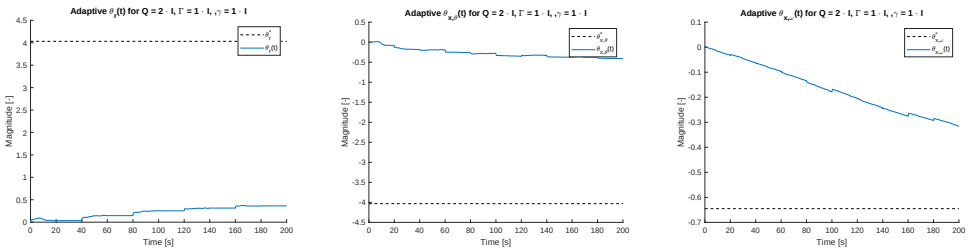


Figure 15: Controller gain adaptation for $Q = 2 \cdot \mathbb{I}$, $\mathit{\Gamma} = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$, white noise acting on input and output and a square reference signal.

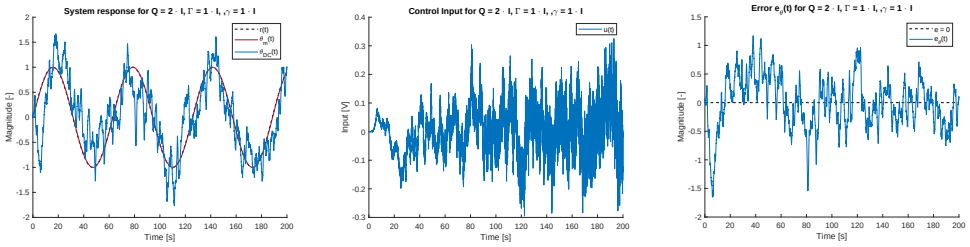


Figure 16: DC motor response, control input and output error for $Q = 2 \cdot \mathbb{I}$, $\mathit{\Gamma} = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$, white noise acting on input and output and a sinusoidal reference signal.

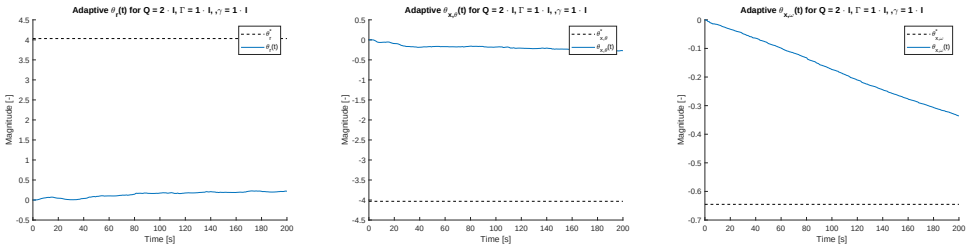


Figure 17: Controller gain adaptation for $Q = 2 \cdot \mathbb{I}$, $\mathit{\Gamma} = 1 \cdot \mathbb{I}$ and $\gamma = 1 \cdot \mathbb{I}$, white noise acting on input and output and a sinusoidal reference signal.

4 Conclusion

The adaptive control law proposed in section 2.2 makes a DC motor with all states measurable match a reference model. In particular, the steady-state output error of the DC motor converges to zero. The input and output signals remain bounded. The simulation results in section 3 show that the system performance strongly depends on the design choices of Q and the gain matrix Γ as well as gain scalar γ. Furthermore, the reference model must be chosen such that performance requirements are met. In this case, a second order model was chosen. The damping coefficient determines whether the system response has an overshoot while the natural frequency has a strong influence on the rise time of the system.
The system response can be quicker or slower and more or less oscillatory depending on the design choices. Having roots with a large negative real part and no imaginary part for the observer polynomial T results in a quick response without much oscillation or overshoot. The same can be said for a moderately chosen Γ. In this exercise, there are no constraints on the input signals. In a real, hardware application this wouldn't further apply as all physical systems have their limits and saturations. In such a case, the design choices would need to be re-evaluated and adjusted to ensure that the control input doesn't exceed the physical limits of the system at hand.

References

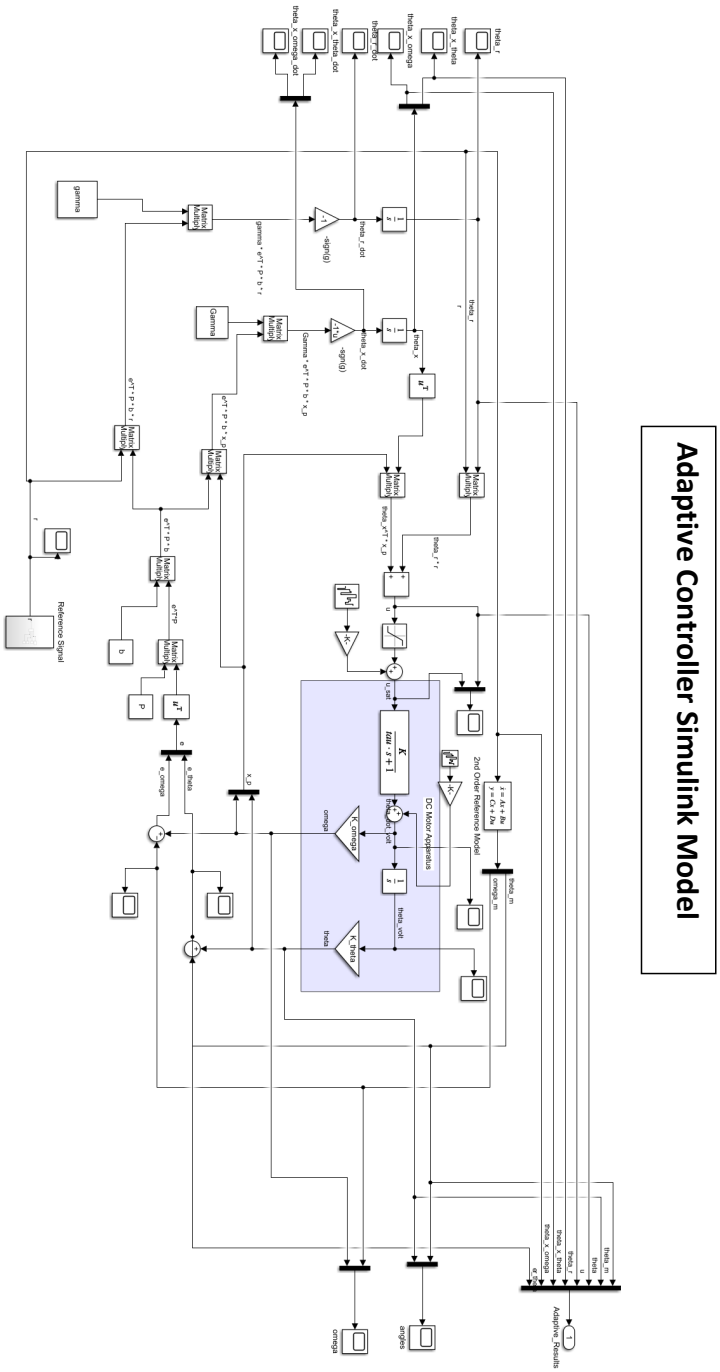
[1] HOU, M., Duan, G., and Guo, M. (2010). New versions of barbalat's lemma with applications. *Journal of Control Theory and Applications*, 8:545–547.

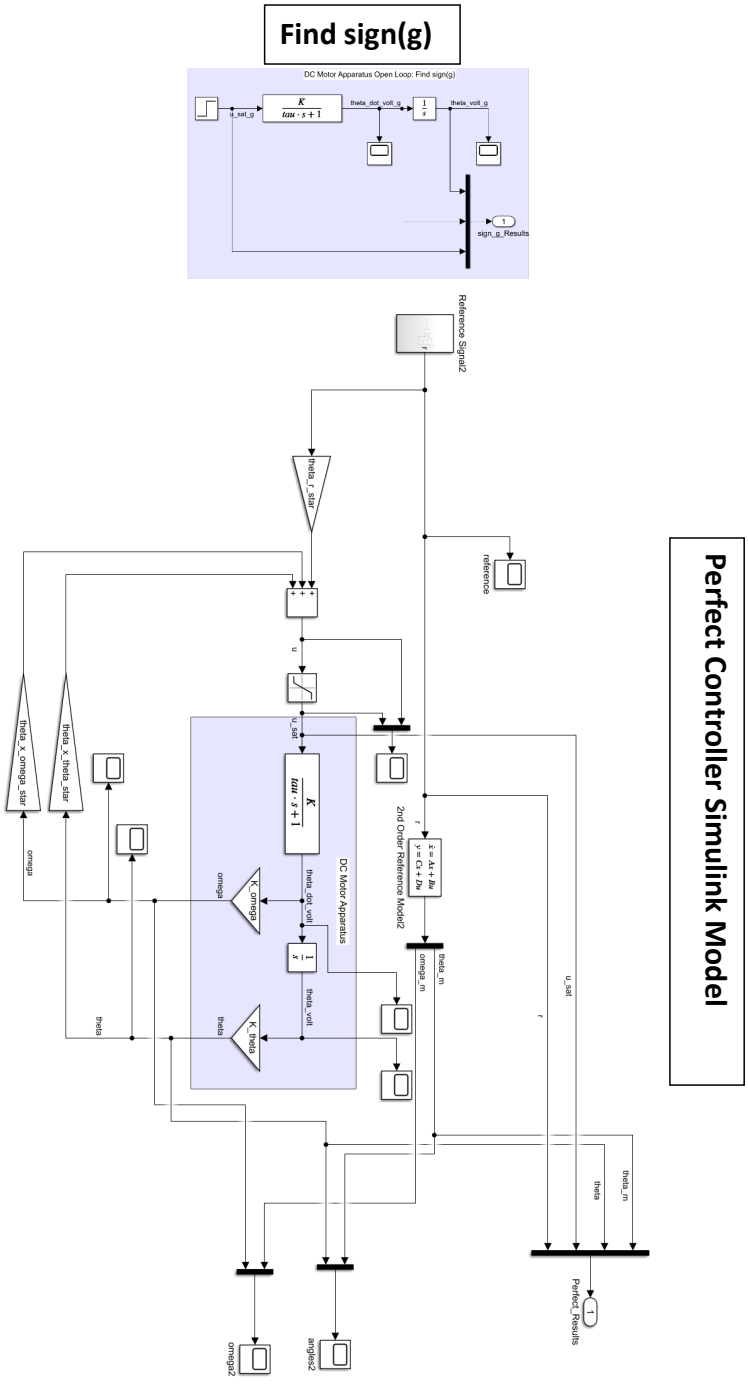
[2] Khalil, H. (2002). *Nonlinear Systems*. Pearson Education. Prentice Hall.

[3] Åström, K. (1983). Theory and applications of adaptive control—a survey. *Automatica*, 19(5):471–486.

A Appendix

The simulations were conducted with Simulink and Matlab. The adaptive controller was designed with Simulink and enables the DC motor output to converge to the reference model output. The parameters were set and calculated with Matlab scripts. Three Simulink models were setup. One was used to simulate the adaptive controller. Another one represented the perfect control law while the third was used to determine the sign of the steady state gain of the DC motor. The CA3 Matlab live script defined all the simulation parameters. To automate the simulation parameter calculations and result plotting, two functions were defined in the *functionsContainer* script.


Adaptive Controller Simulink Model
DC Motor Apparatus
2nd Order Reference Model
Reference Signal
Adaptive_Results
omega
angles


Find sign(g)
DC Motor Apparatus Open Loop: Find sign(g)
u_set_g
theta_dot_volt_g
theta_volt_g
sign_g_Results
Perfect Controller Simulink Model
Reference Signal2
theta_r_star
reference
u
u_set
theta_x_omega_star
theta_x_theta_star
omega
theta
K_omega
K_theta
DC Motor Apparatus
theta_dot_volt
theta_volt
2nd Order Reference Model2
omega_m
theta_m
Perfect_Results
omega2
angles2

CA3 Adaptive Control of a DC Motor Apparatus

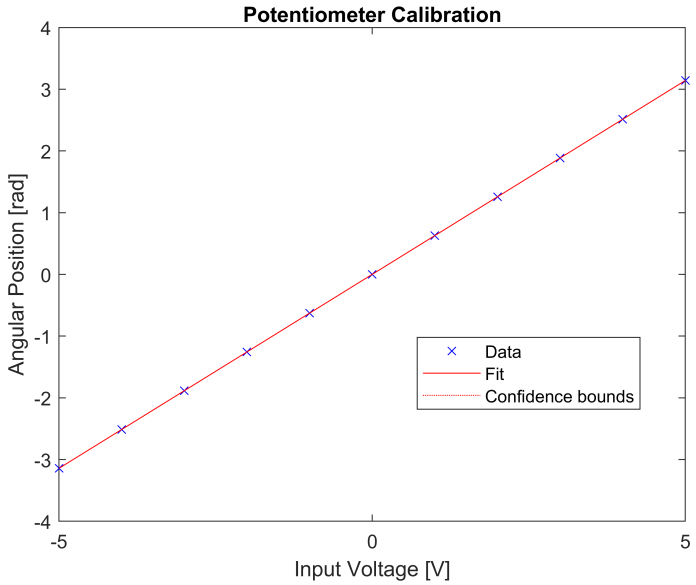
In CA3 an adaptive angular position controller is designed for a DC motor.

```
clear all; close all; clc
```

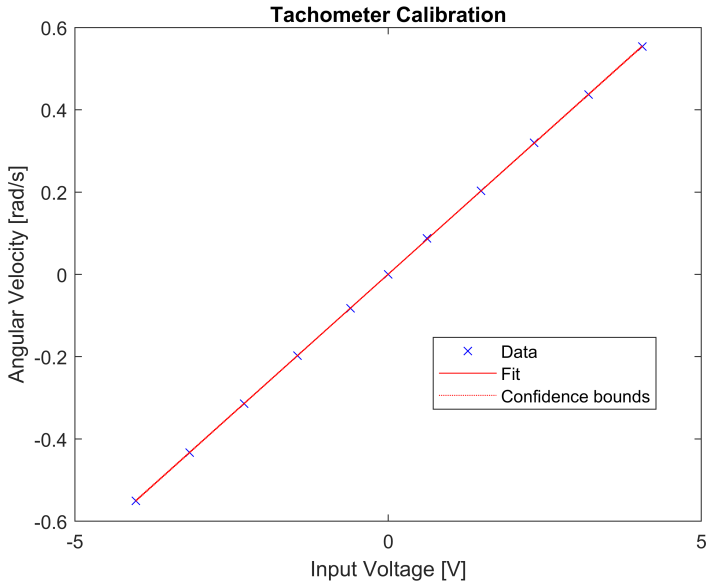
DC Motor parameters for simulation

Here all the DC motor parameters are set and the DC motor is calibrated.

```
% Parameters
K = 6.2; % [rad/s/V]
tau = 0.25; % [s]
g = K/tau; Ap = [0 1; 0 -1/tau];
saturation = 5; % [V]

% Calibration
% Potentiometer
pot_volts_output = [-5 -4 -3 -2 -1 0 1 2 3 4 5]';
ang_rad = [-180 -144 -108 -72 -36 0 36 72 108 144 180]'*pi/180;
K_theta = pot_volts_output\ang_rad; % [rad]
% Linear regression plot
mdl = fitlm(pot_volts_output,ang_rad);
fig_num = 1;
figure(fig_num); fig_num = fig_num+1;
plot(mdl)
xlabel('Input Voltage [V]'); ylabel('Angular Position [rad]');
title('Potentiometer Calibration'); saveas(gcf,'pot_cal.pdf');
```




```
% Tachogenerator
tach_volts_output = [-4.03 -3.17 -2.3 -1.45 -0.6 0 0.62 1.48 2.33 3.2 4.06]';
ang_vel_rad_s = [-31.52 -24.82 -18.01 -11.31 -4.71 0 5.03 11.62 18.33 25.03 31.73]';
ang_vel_rad_s = ang_vel_rad_s*pi/180;
K_omega = tach_volts_output\ang_vel_rad_s; % [rad/s]
% Linear regression plot
mdl = fitlm(tach_volts_output,ang_vel_rad_s);
figure(fig_num); fig_num = fig_num+1;
plot(mdl)
xlabel('Input Voltage [V]'); ylabel('Angular Velocity [rad/s]');
title('Tachometer Calibration'); saveas(gcf,'tach_cal.pdf')
```



Run Simulations and Plot Results

The function 'prepareSimulation' calculates all the necessary parameters for a simulation run. The reference parameters and control gains can be set when calling the function to vary the simulation results. The function 'plotMe' conveniently plots and saves the results of the simulation.

For the 'prepareSimulation' function, the user must define the refernce model parameters (natural frequency and damping coefficient), the control gains Γ and γ, as well as the symmetric positive matrix Q which is used to calculate P.

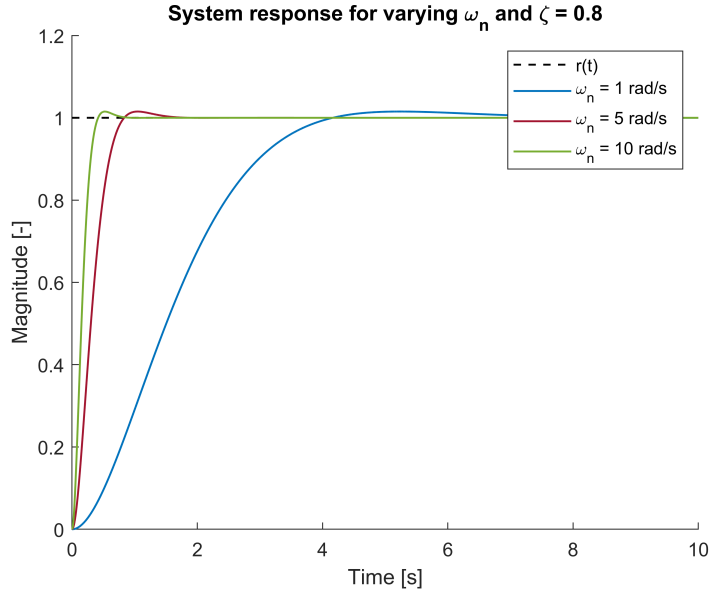
For the 'plotMe' function, the user must specify the time vector and data which should be plotted. Furthermore, plot parameters such as color, line style and line width must be defined. Also, the name of the x and y axis, the plot title, the legend entries, the figure title and figure number must be passed to the function for it to function properly.

```
% Reference Model Parameters
wm = 10; % natural frequency [rad/s]
zetam = 0.8; % damping coefficient [-]

noise = false; % Run simulation without noise;

% Plot colors
blue = [0 0.4470 0.7410]; red = [0.6350 0.0780 0.1840]; orange = [0.8500 0.3250 0.0980];
black = [0 0 0]; green = [0.4660 0.6740 0.1880];
lblue = [0.3010 0.7450 0.9330]; purple = [0.4940 0.1840 0.5560];
```

Reference Model

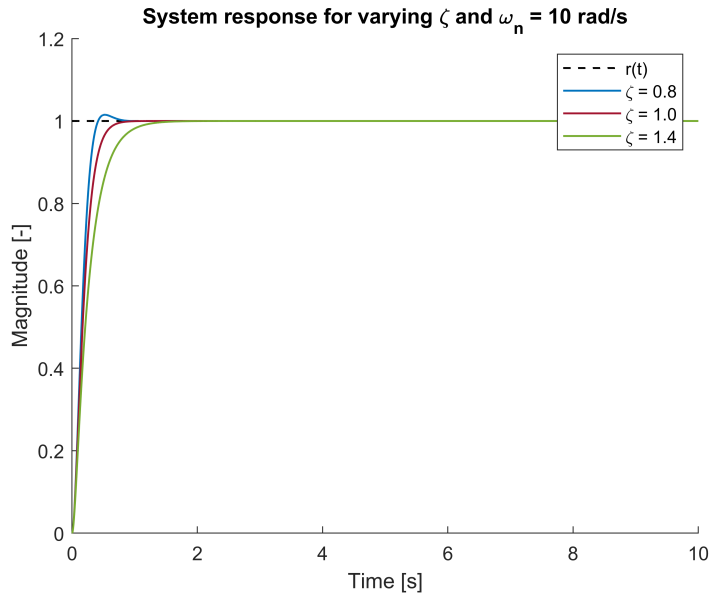
Influence of different natural frequencies and damping coefficients on reference model response.

```
% Simulate reference model with different natural frequencies and damping
% coefficients. Analyze step response behavior.
% Various natural frequencies. Constant damping coefficient ζ = 0.8;
myObj = functionsContainer;
wm = [1, 5, 10]; zetam = [0.8, 1.0, 1.4]; ref_signal = 3; simTime = 10.0;
Q_gain = 1; gamma_gain = 1; Gamma_gain = 1; % Default gains. Not used for reference model simul
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(1),zetam(1),Gamma_gain,gamma_gain,g,Ap);
out_wm_1 = sim("CA3_model.slx");
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(2),zetam(1),Gamma_gain,gamma_gain,g,Ap);
out_wm_5 = sim("CA3_model.slx");
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(3),zetam(1),Gamma_gain,gamma_gain,g,Ap);
out_wm_10 = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(out_wm_1.tout,[out_wm_1.yout{1}.Values.Data(:,7), ...
    out_wm_1.yout{1}.Values.Data(:,1),out_wm_5.yout{1}.Values.Data(:,1),...
    out_wm_10.yout{1}.Values.Data(:,1)],'Vary_wm_zetam_const.pdf',[{'r(t)'},...
    {'\omega_n = 1 rad/s'},{'\omega_n = 5 rad/s'},{'\omega_n = 10 rad/s'}],...
    [black, blue, red, green],['--','- ','- ', '- '],[1 1 1 1], 'Time [s]',...
    'Magnitude [-]', 'System response for varying \omega_n and \zeta = 0.8',fig_num);
```




```
% Various damping coefficients. Constant natural frequency ω = 10 rad/s;
```

```
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(3),zetam(1),Gamma_gain,gamma_gain,g,Ap);
out_zm_08 = sim("CA3_model.slx");
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(3),zetam(2),Gamma_gain,gamma_gain,g,Ap);
out_zm_10 = sim("CA3_model.slx");
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain,wm(3),zetam(3),Gamma_gain,gamma_gain,g,Ap);
out_zm_14 = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(out_wm_1.tout,[out_zm_08.yout{1}.Values.Data(:,7), ...
    out_zm_08.yout{1}.Values.Data(:,1),out_zm_10.yout{1}.Values.Data(:,1),...
    out_zm_14.yout{1}.Values.Data(:,1)],'Vary_zetam_wm_const.pdf',[{'r(t)'},...
    {'\zeta = 0.8'},{'\zeta = 1.0'},{'\zeta = 1.4'}],[black, blue, red, green],...
    ['--','- ','- ', '- '],[1 1 1 1],'Time [s]','Magnitude [-]', ...
    'System response for varying \zeta and \omega_n = 10 rad/s',fig_num);
```

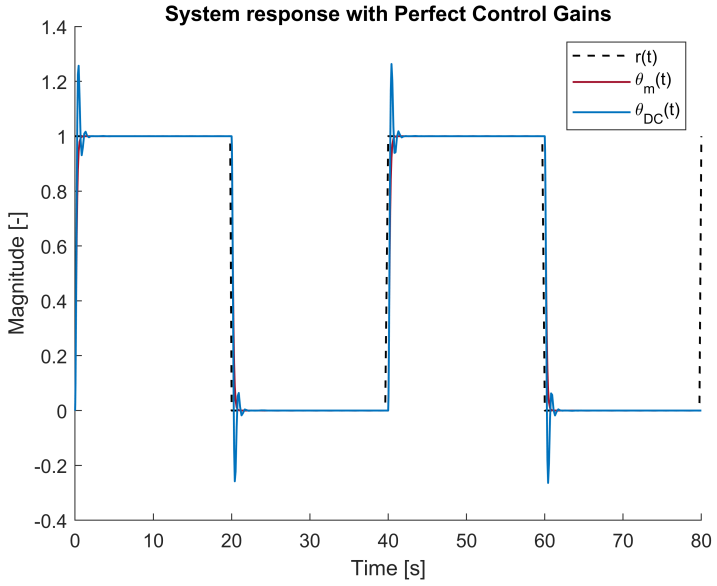


Perfect Control Input

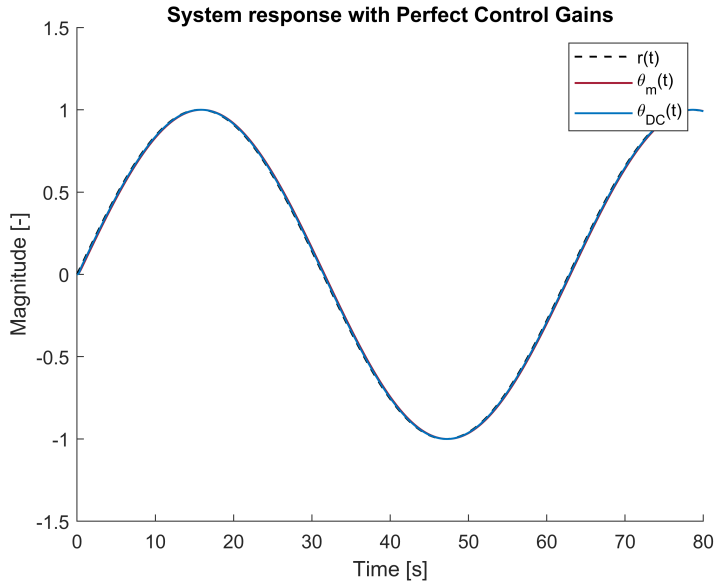
With the perfect control input the DC motor matches the reference model. The reason for a small error at the beginning is because the calibration parameters of the DC motor come from linear regression and therefore don't perfectly relate the output voltage to the actual angular velocity and position.

```
simTime = 80.0; ref_signal = 3;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain, wm(3), zetam(2), Gamma_gain, gamma_gain,g,Ap);
out_perfect_square = sim("Perfect_Control_Input.slx"); fig_num = fig_num+1;
```

```
myObj.plotMe(out_perfect_square.tout,[out_perfect_square.yout{1}.Values.Data(:,4),...
    out_perfect_square.yout{1}.Values.Data(:,1),out_perfect_square.yout{1}.Values.Data(:,2)],..
    'perfect_square_output.pdf',[{'r(t)'},{'\theta_m(t)'},{'\theta_{DC}(t)'}],...
    [black, red, blue],['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',...
    'System response with Perfect Control Gains',fig_num);
```




```
ref_signal = 1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain, wm(3), zetam(2), Gamma_gain, gamma_gain,g,Ap);
out_perfect_sin = sim("Perfect_Control_Input.slx"); fig_num = fig_num+1;
myObj.plotMe(out_perfect_sin.tout,[out_perfect_sin.yout{1}.Values.Data(:,4),...
    out_perfect_sin.yout{1}.Values.Data(:,1),out_perfect_sin.yout{1}.Values.Data(:,2)],...
    'perfect_sin_output.pdf',[{'r(t)'},{'\theta_m(t)'},{'\theta_{DC}(t)'}],[black, red, blue],.
    ['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',...
    'System response with Perfect Control Gains',fig_num);
```

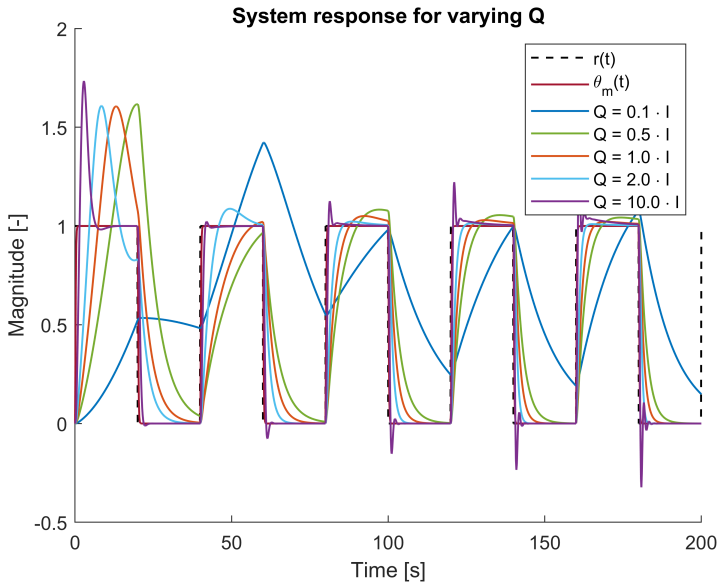


Influence of Varying Gains on DC Motor Response

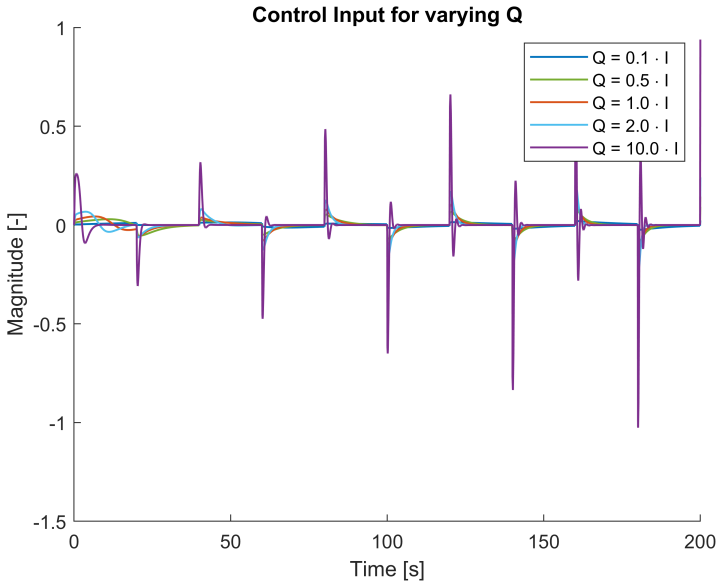
The symmetric positive definite matrix Q is a design choice. Together with the reference model state matrix Am, it determines what value P takes. Since P is present in the adaptive control law, Q in the end influences how the gains are adapted and thus influences the motor response. The same can be said for the gain matrices Γ and γ. The gains are varied and tuned to find an optimal motor respone in this section.

```
Q_gain = [0.1, 0.5, 1, 2, 10]; simTime = 200; ref_signal = 3;
Gamma_gain = [0.1, 0.5, 1, 2, 10]; gamma_gain = [0.1, 0.5, 1, 2, 10];

% Vary Q and plot results
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(1), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Q_gain_01 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(2), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Q_gain_05 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(3), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Q_gain_1 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Q_gain_2 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(5), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Q_gain_10 = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(Q_gain_01.tout,[Q_gain_01.yout{1}.Values.Data(:,7),...
    Q_gain_01.yout{1}.Values.Data(:,1),Q_gain_01.yout{1}.Values.Data(:,2),...
```

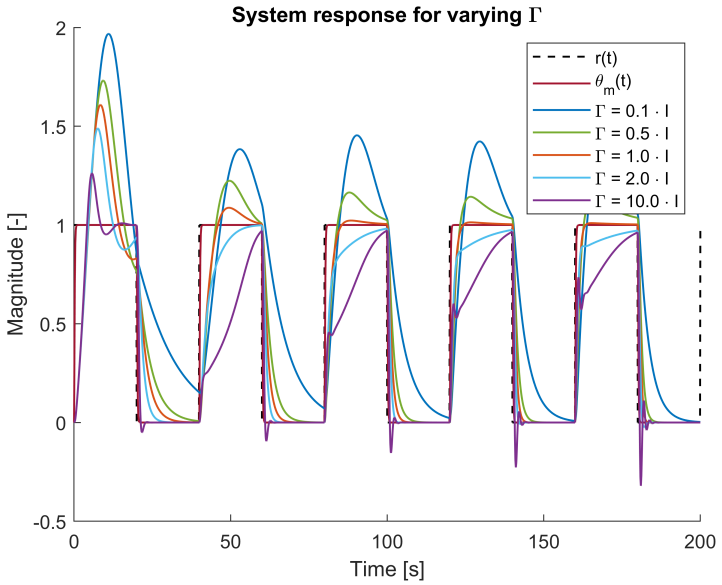
```
    Q_gain_05.yout{1}.Values.Data(:,2),Q_gain_1.yout{1}.Values.Data(:,2),...
    Q_gain_2.yout{1}.Values.Data(:,2),Q_gain_10.yout{1}.Values.Data(:,2)], ...
    'Vary_Q_gain.pdf',[{'r(t)'},{'\theta_m(t)'},{'Q = 0.1 \cdot I'},{'Q = 0.5 \cdot I'},...
    {'Q = 1.0 \cdot I'},{'Q = 2.0 \cdot I'},{'Q = 10.0 \cdot I'}],[black, red, blue, green,...
    orange,lblue,purple],['--','- ','- ','- ','- ','- ','- '],[1 1 1 1 1 1 1],'Time [s]',...
    'Magnitude [-]', 'System response for varying Q',fig_num); fig_num = fig_num+1;
```



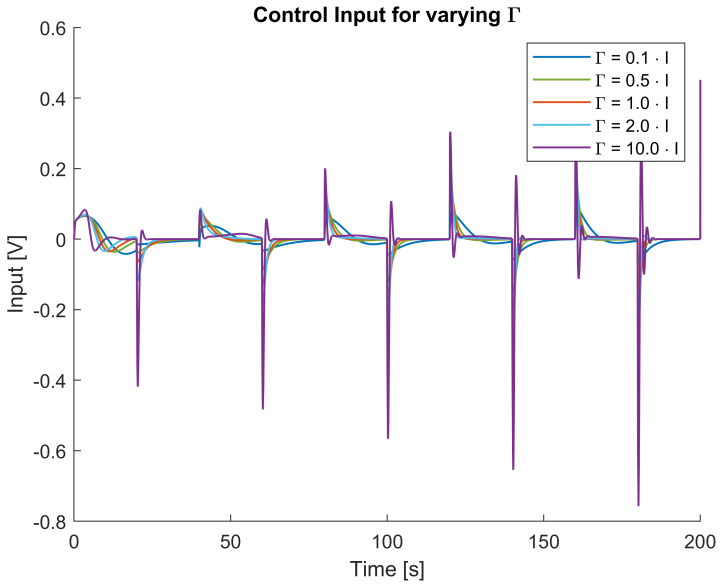

```
myObj.plotMe(Q_gain_01.tout,[Q_gain_01.yout{1}.Values.Data(:,3),...
    Q_gain_05.yout{1}.Values.Data(:,3),Q_gain_1.yout{1}.Values.Data(:,3),...
    Q_gain_2.yout{1}.Values.Data(:,3),Q_gain_10.yout{1}.Values.Data(:,3)],...
    'u_Vary_Q_gain.pdf',[{'Q = 0.1 \cdot I'},{'Q = 0.5 \cdot I'},{'Q = 1.0 \cdot I'},...
    {'Q = 2.0 \cdot I'},{'Q = 10.0 \cdot I'}],[blue, green,orange,lblue,purple],...
    ['- ','- ','- ','- ','- '],[1 1 1 1 1 1 1],'Time [s]','Magnitude [-]',...
    'Control Input for varying Q',fig_num);
```



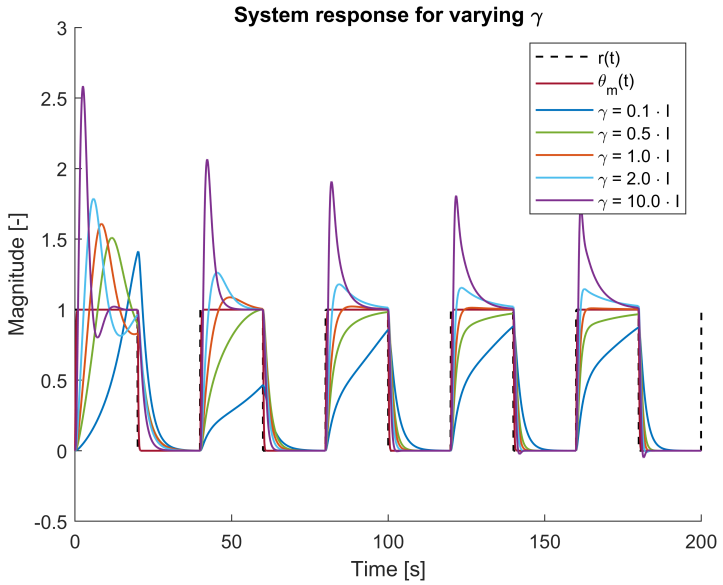

```
% Vary Γ and plot results with Q = 2*I
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(1), gamma_gain(3),g,Ap);
Gamma_gain_01 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(2), gamma_gain(3),g,Ap);
Gamma_gain_05 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
Gamma_gain_1 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(4), gamma_gain(3),g,Ap);
Gamma_gain_2 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(5), gamma_gain(3),g,Ap);
Gamma_gain_10 = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(Gamma_gain_01.tout,[Gamma_gain_01.yout{1}.Values.Data(:,7),...
    Gamma_gain_01.yout{1}.Values.Data(:,1),Gamma_gain_01.yout{1}.Values.Data(:,2),...
    Gamma_gain_05.yout{1}.Values.Data(:,2),Gamma_gain_1.yout{1}.Values.Data(:,2),...
    Gamma_gain_2.yout{1}.Values.Data(:,2),Gamma_gain_10.yout{1}.Values.Data(:,2)],...
    'Vary_Gamma_gain.pdf',[{'r(t)'},{'\theta_m(t)'},{'\Gamma = 0.1 \cdot I'},...
    {'\Gamma = 0.5 \cdot I'},{'\Gamma = 1.0 \cdot I'},{'\Gamma = 2.0 \cdot I'},...
    {'\Gamma = 10.0 \cdot I'}],[black, red, blue, green,orange,lblue,purple],...
    ['--','- ','- ','- ','- ','- ','- '],[1 1 1 1 1 1 1],'Time [s]','Magnitude [-]',...
    'System response for varying \Gamma',fig_num); fig_num = fig_num+1;
```



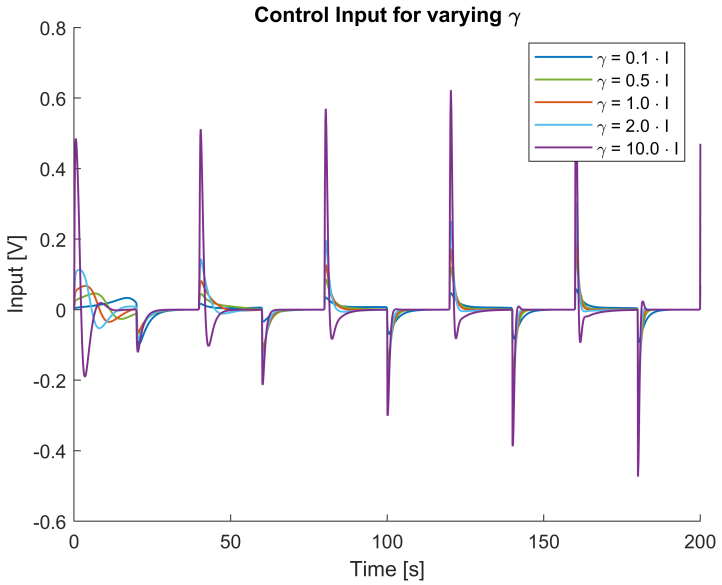

```
myObj.plotMe(Gamma_gain_01.tout,[Gamma_gain_01.yout{1}.Values.Data(:,3),...
    Gamma_gain_05.yout{1}.Values.Data(:,3),Gamma_gain_1.yout{1}.Values.Data(:,3),...
    Gamma_gain_2.yout{1}.Values.Data(:,3),Gamma_gain_10.yout{1}.Values.Data(:,3)], ...
    'u_Vary_Gamma_gain.pdf',[{'\Gamma = 0.1 \cdot I'},{'\Gamma = 0.5 \cdot I'},...
    {'\Gamma = 1.0 \cdot I'},{'\Gamma = 2.0 \cdot I'},{'\Gamma = 10.0 \cdot I'}],...
    [blue, green,orange,lblue,purple],['- ','- ','- ','- ','- '],[1 1 1 1 1 1 1], ...
    'Time [s]','Input [V]', 'Control Input for varying \Gamma',fig_num);
```



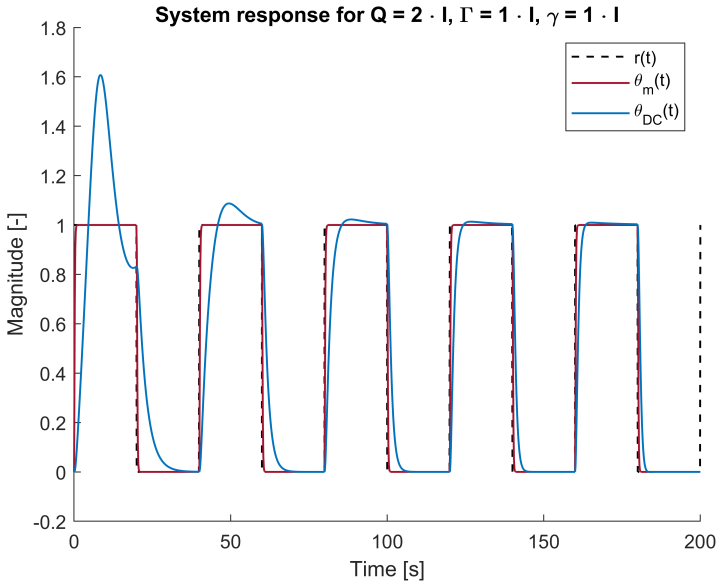

```
% Vary γ and plot results with Q = 2*I
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(1),g,Ap);
gamma_gain_01 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(2),g,Ap);
gamma_gain_05 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
gamma_gain_1 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(4),g,Ap);
gamma_gain_2 = sim("CA3_model.slx"); fig_num = fig_num+1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(5),g,Ap);
gamma_gain_10 = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(gamma_gain_01.tout,[gamma_gain_01.yout{1}.Values.Data(:,7),...
    gamma_gain_01.yout{1}.Values.Data(:,1),gamma_gain_01.yout{1}.Values.Data(:,2),...
    gamma_gain_05.yout{1}.Values.Data(:,2),gamma_gain_1.yout{1}.Values.Data(:,2),...
    gamma_gain_2.yout{1}.Values.Data(:,2),gamma_gain_10.yout{1}.Values.Data(:,2)],...
    'Vary_gamma_r_gain.pdf',[{'r(t)'},{'\theta_m(t)'},{'\gamma = 0.1 \cdot I'},...
    {'\gamma = 0.5 \cdot I'},{'\gamma = 1.0 \cdot I'},{'\gamma = 2.0 \cdot I'},...
    {'\gamma = 10.0 \cdot I'}],[black, red, blue, green,orange,lblue,purple],...
    ['--','- ','- ','- ','- ','- ','- '],[1 1 1 1 1 1 1],'Time [s]','Magnitude [-]',...
    'System response for varying \gamma',fig_num); fig_num = fig_num+1;
```



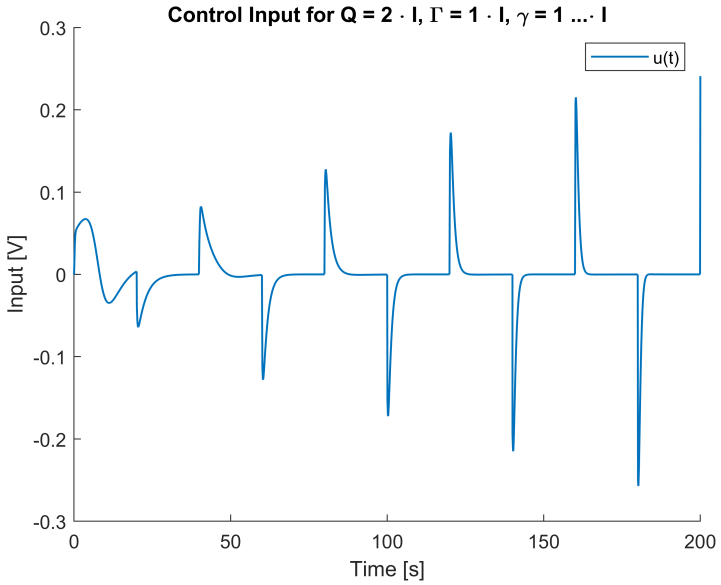

```
myObj.plotMe(Gamma_gain_01.tout,[gamma_gain_01.yout{1}.Values.Data(:,3),...
    gamma_gain_05.yout{1}.Values.Data(:,3),gamma_gain_1.yout{1}.Values.Data(:,3),...
    gamma_gain_2.yout{1}.Values.Data(:,3),gamma_gain_10.yout{1}.Values.Data(:,3)], ...
    'u_Vary_gamma_r_gain.pdf',[{'\gamma = 0.1 \cdot I'},{'\gamma = 0.5 \cdot I'},...
    {'\gamma = 1.0 \cdot I'},{'\gamma = 2.0 \cdot I'},{'\gamma = 10.0 \cdot I'}],...
    [blue, green,orange,lblue,purple],['- ','- ','- ','- ','- '],[1 1 1 1 1], ...
    'Time [s]','Input [V]', 'Control Input for varying \gamma',fig_num);
```



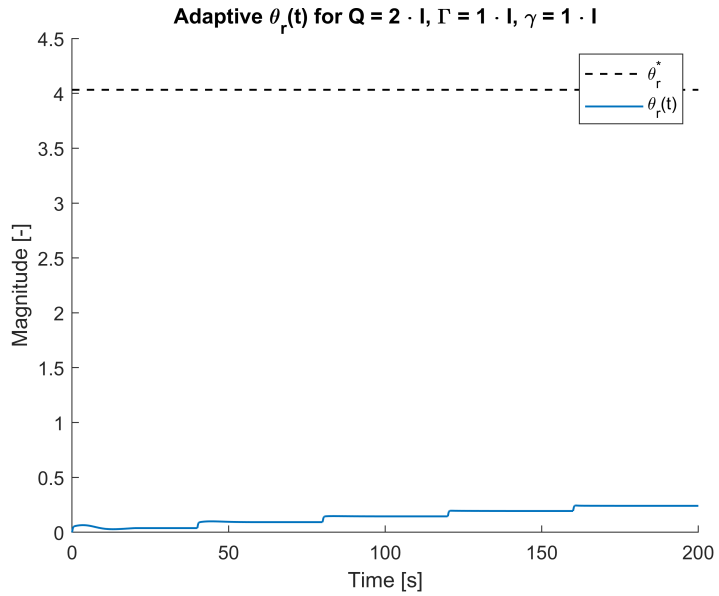

```
% "Optimal" Q = 2*I, Gamma =  1*I, gamma = 1*I
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
optimal = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[optimal.yout{1}.Values.Data(:,7), optimal.yout{1}.Values.Data(:,1),...
    optimal.yout{1}.Values.Data(:,2)],'Optimal_Output.pdf',[{'r(t)'},{'\theta_m(t)'},...
    {'\theta_{DC}(t)'}],[black, red, blue],['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',...
    'System response for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



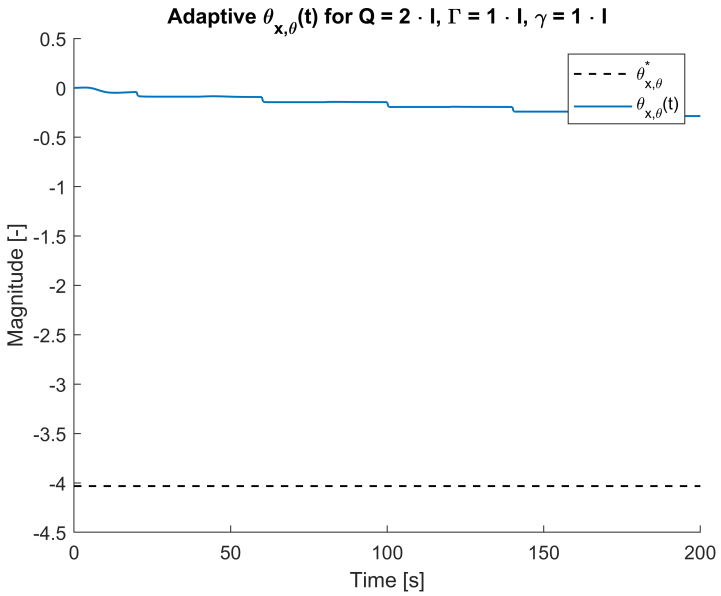

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,optimal.yout{1}.Values.Data(:,3),'u_Optimal.pdf',{'u(t)'},blue,'- ',1
    'Time [s]','Input [V]', ['Control Input for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1
    '\cdot I'],fig_num);fig_num = fig_num+1;
```



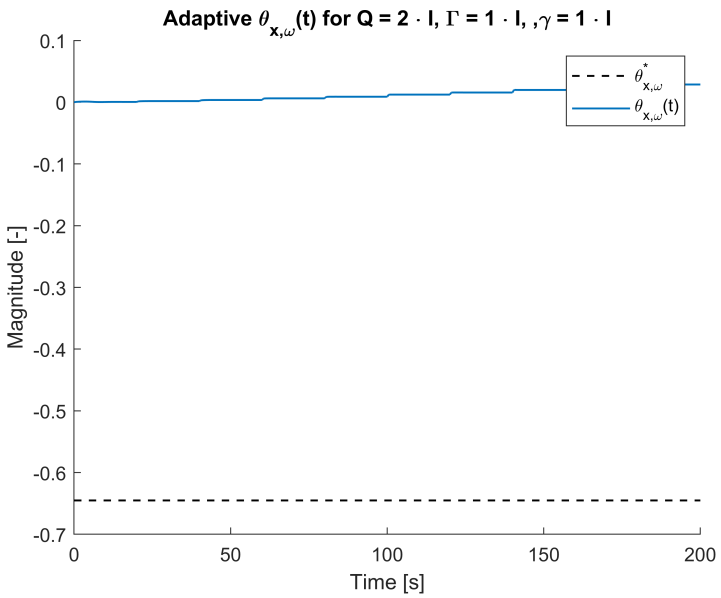

```
myObj.plotMe(optimal.tout,[theta_r_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,4)],'theta_r_Optimal.pdf',[{'\theta_r^*'},{'\theta_r(t)'}],..
    [black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_r(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



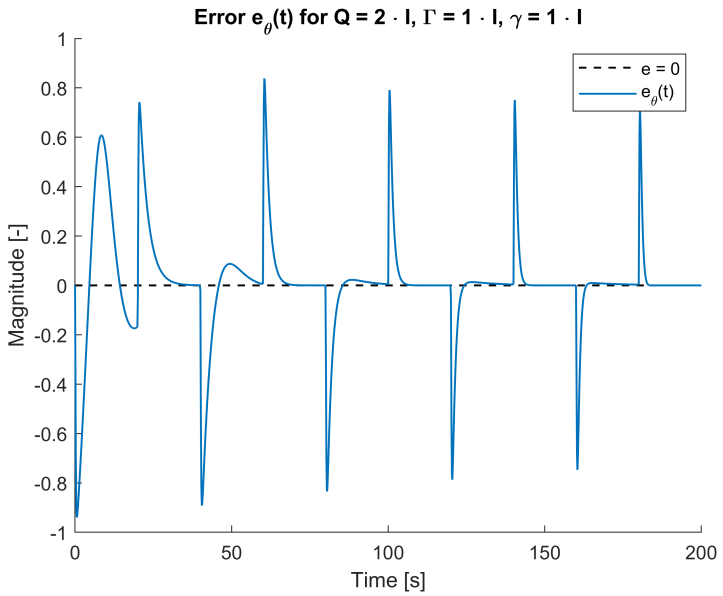

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_x_theta_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,5)],'theta_x_theta_Optimal.pdf',[{'\theta_{x,\theta}^*'},...
    {'\theta_{x,\theta}(t)'}],[black, blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_{x,\theta}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',.
    fig_num); fig_num = fig_num+1;
```



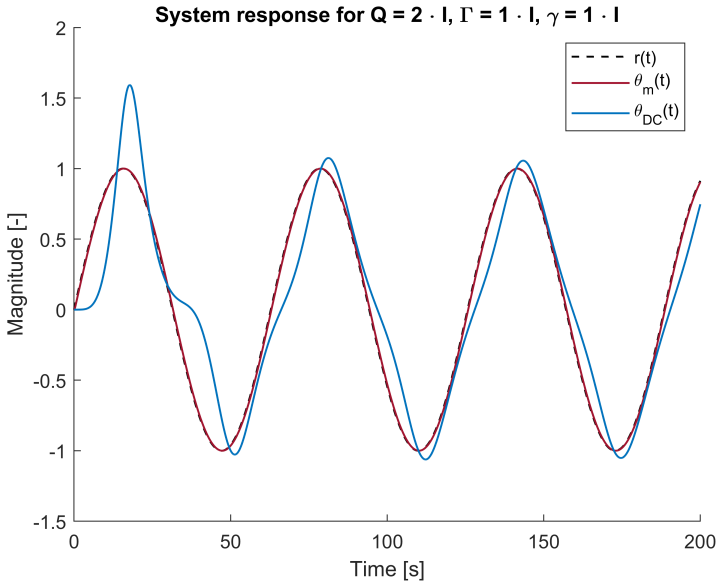

```
myObj.plotMe(optimal.tout,[theta_x_omega_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,6)],'theta_x_omega_Optimal.pdf',[{'\theta_{x,\omega}^*'},...
    {'\theta_{x,\omega}(t)'}],[black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_{x,\omega}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, ,\gamma = 1 \cdot I',
    fig_num); fig_num = fig_num+1;
```



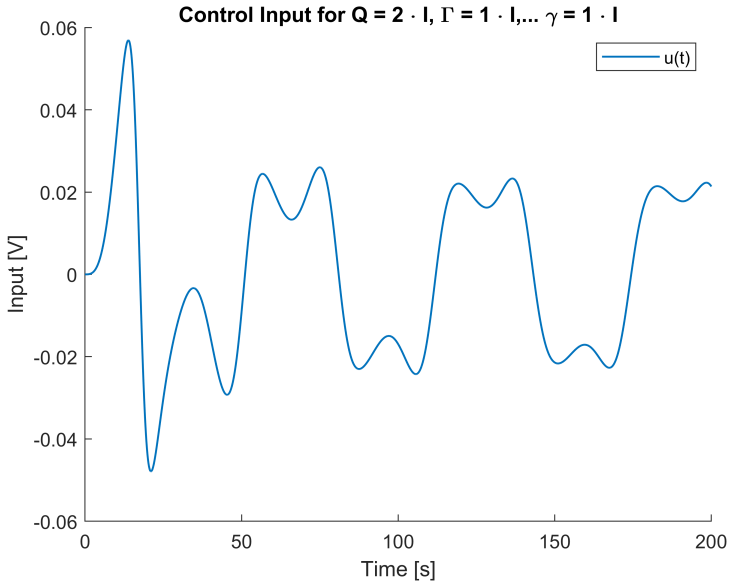

```
myObj.plotMe(optimal.tout,[zeros(length(optimal.tout),1), optimal.yout{1}.Values.Data(:,8)],...
    'error_Optimal.pdf',['e = 0',{'e_\theta(t)'}],[black,blue],['--','- '],[1 1],'Time [s]',...
    'Magnitude [-]', 'Error e_\theta(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot
    fig_num); fig_num = fig_num+1;
```



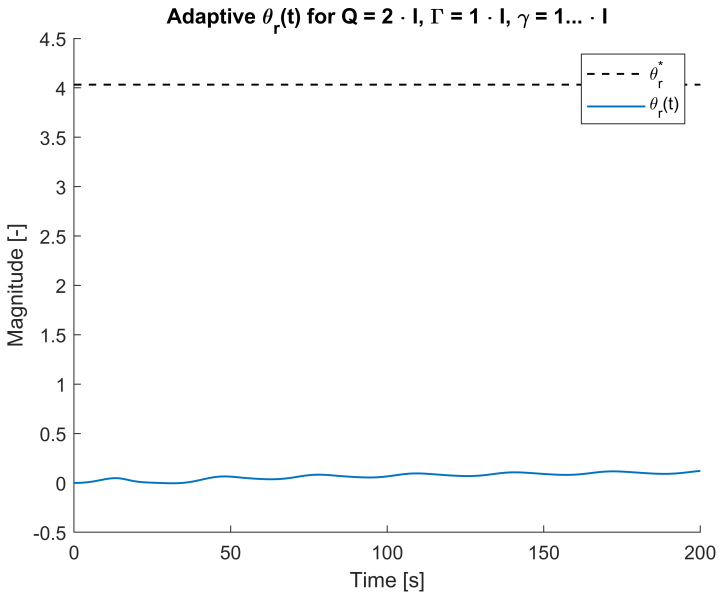

```
% "Optimal" Q = 2*I, Gamma =  1*I, gamma = 1*I. Sinusoidal reference signal
ref_signal = 1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4),wm(3),zetam(2),Gamma_gain(3),gamma_gain(3),g,Ap);
optimal = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[optimal.yout{1}.Values.Data(:,7),optimal.yout{1}.Values.Data(:,1),..
    optimal.yout{1}.Values.Data(:,2)],'sin_Optimal_Output.pdf',[{'r(t)'},{'\theta_m(t)'},...
    {'\theta_{DC}(t)'}],[black, red, blue],['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',
    'System response for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num); fig_
```



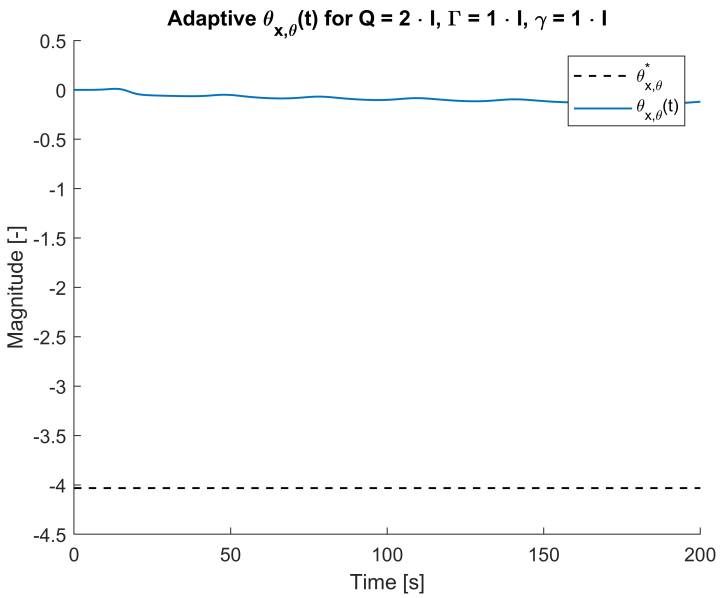

```
myObj.plotMe(optimal.tout,optimal.yout{1}.Values.Data(:,3),'sin_u_Optimal.pdf',{'u(t)'},...
    blue,'- ',1,'Time [s]','Input [V]', ['Control Input for Q = 2 \cdot I, \Gamma = 1 \cdot I,
    ' \gamma = 1 \cdot I'],fig_num);  fig_num = fig_num+1;
```



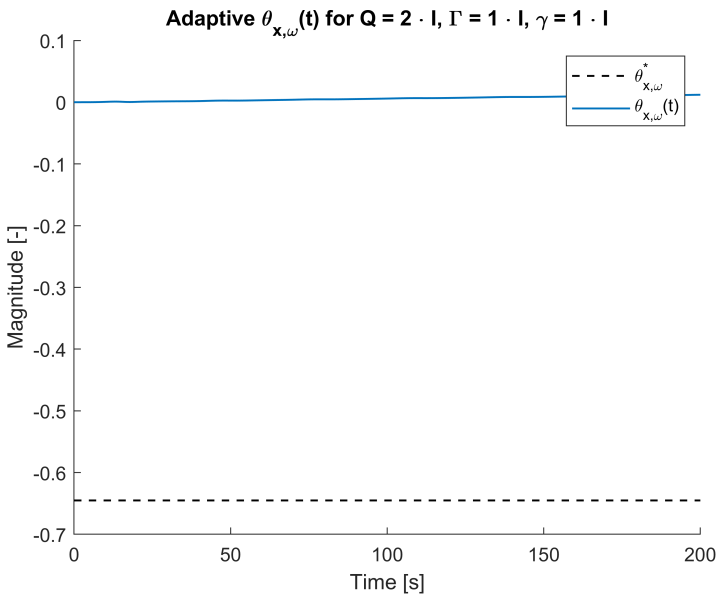

```
myObj.plotMe(optimal.tout,[theta_r_star*ones(length(optimal.tout),1),optimal.yout{1}.Values.Dat
    'sin_theta_r_Optimal.pdf',[{'\theta_r^*'},{'\theta_r(t)'}],[black,blue],['--','- '],[1 1],
    'Time [s]','Magnitude [-]', ['Adaptive \theta_r(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \
    ' \cdot I'],fig_num); fig_num = fig_num+1;
```



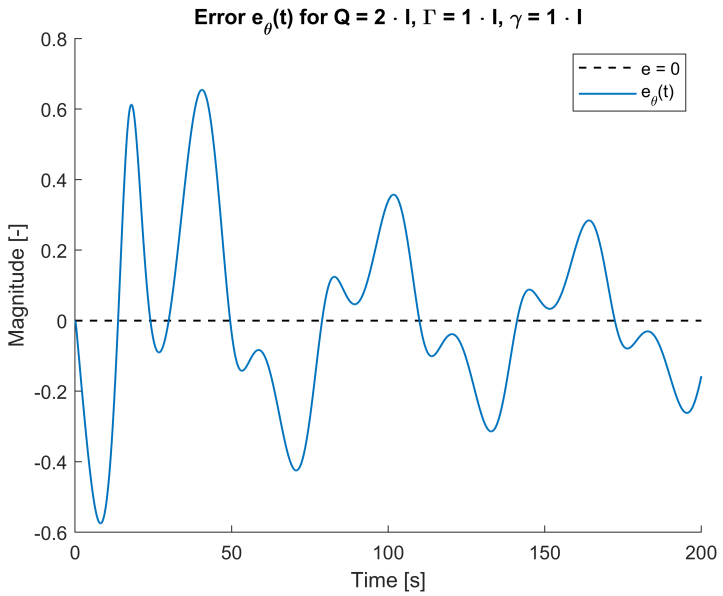

```
myObj.plotMe(optimal.tout,[theta_x_theta_star*ones(length(optimal.tout),1),optimal.yout{1}.Valu
    'sin_theta_x_theta_Optimal.pdf',[{'\theta_{x,\theta}^*'},{'\theta_{x,\theta}(t)'}],[black,
    ['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_{x,\theta}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',
```



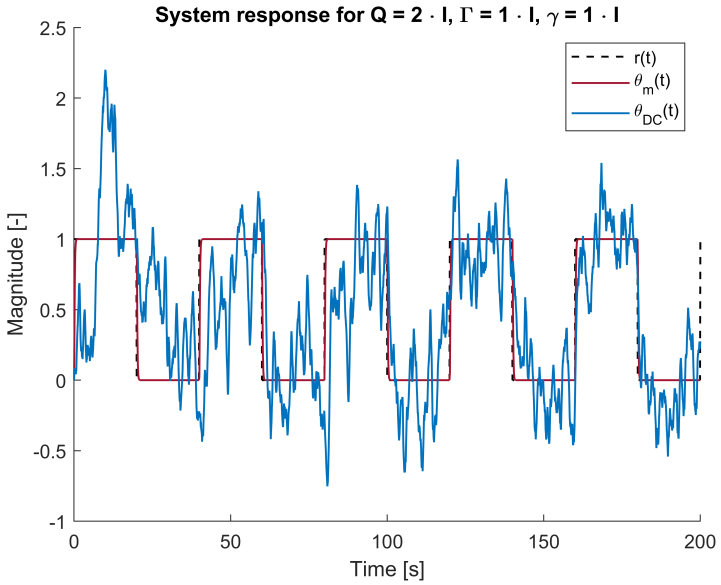

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_x_omega_star*ones(length(optimal.tout),1),optimal.yout{1}.Valu
    'sin_theta_x_omega_Optimal.pdf',[{'\theta_{x,\omega}^*'},{'\theta_{x,\omega}(t)'}],[black,b
    'Time [s]','Magnitude [-]', 'Adaptive \theta_{x,\omega}(t) for Q = 2 \cdot I, \Gamma = 1 \c
    fig_num); fig_num = fig_num+1;
```



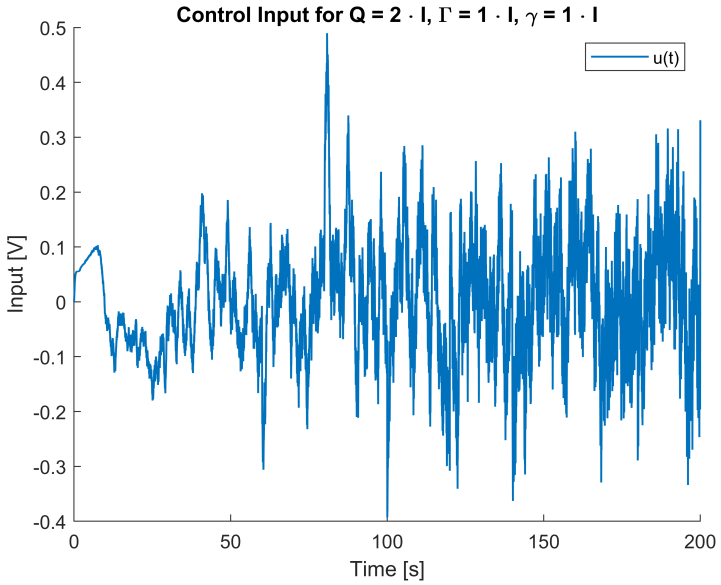

```
myObj.plotMe(optimal.tout,[zeros(length(optimal.tout),1), optimal.yout{1}.Values.Data(:,8)],'s
    ['e = 0',{'e_\theta(t)'}],[black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]', ...
    'Error e_\theta(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num); fi
```



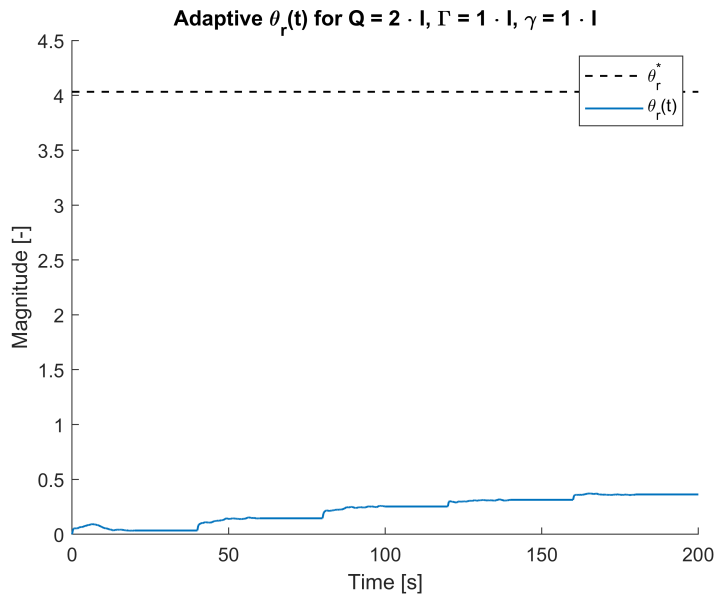

```
noise = true; % Simulations with noise

% "Optimal" Q = 2*I, Gamma =  1*I, gamma = 1*I
ref_signal = 3;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4),wm(3),zetam(2),Gamma_gain(3),gamma_gain(3),g,Ap);
optimal = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[optimal.yout{1}.Values.Data(:,7),...
    optimal.yout{1}.Values.Data(:,1),optimal.yout{1}.Values.Data(:,2)], ...
    'noise_Optimal_Output.pdf',[{'r(t)'},{'\theta_m(t)'},{'\theta_{DC}(t)'}],...
    [black, red, blue],['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',...
    'System response for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



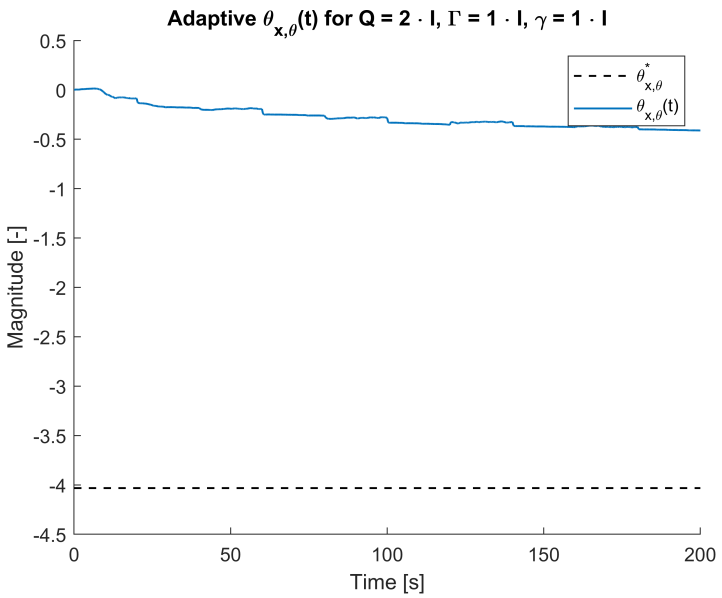

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,optimal.yout{1}.Values.Data(:,3),'noise_u_Optimal.pdf',{'u(t)'},...
    blue,'- ',1,'Time [s]','Input [V]',...
    ['Control Input for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I'],fig_num);
```



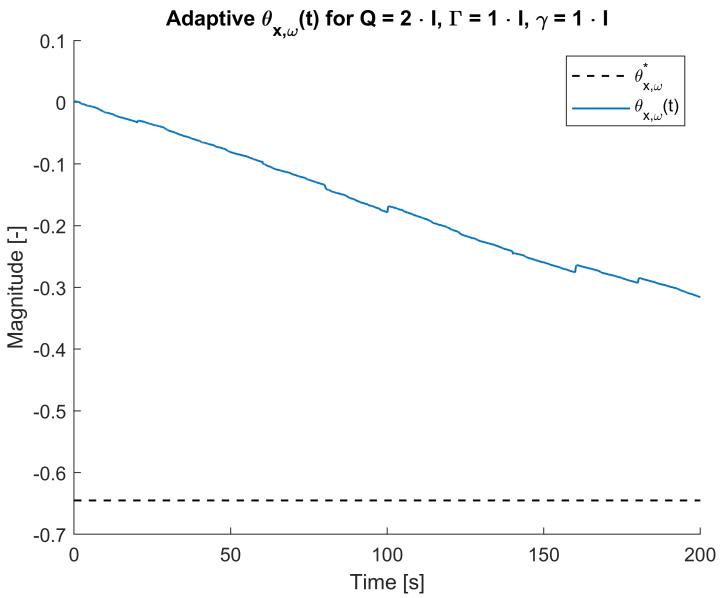

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_r_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,4)],'noise_theta_r_Optimal.pdf',[{'\theta_r^*'},...
    {'\theta_r(t)'}],[black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_r(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



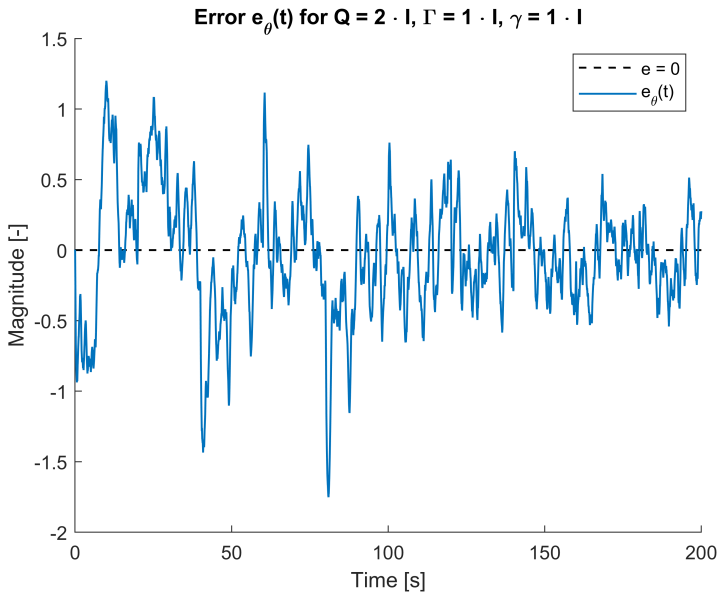

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_x_theta_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,5)],'noise_theta_x_theta_Optimal.pdf',[{'\theta_{x,\theta}^*'
    {'\theta_{x,\theta}(t)'}],[black, blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_{x,\theta}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',.
    fig_num); fig_num = fig_num+1;
```



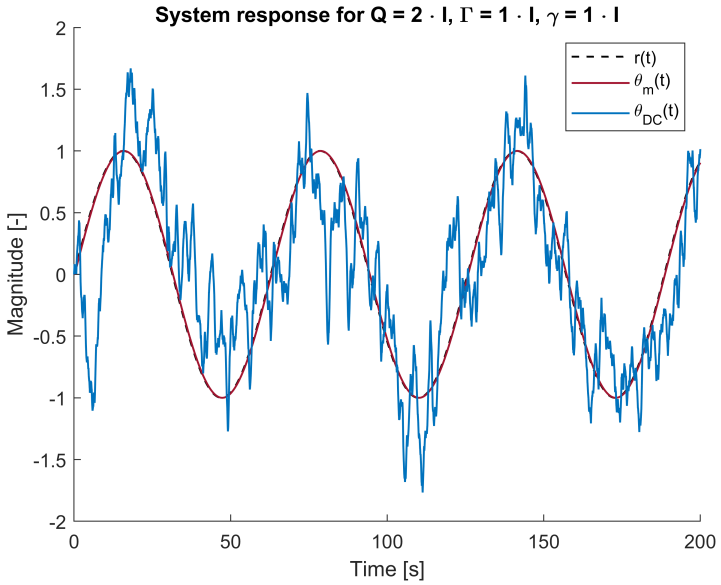

```
myObj.plotMe(optimal.tout,[theta_x_omega_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,6)],'noise_theta_x_omega_Optimal.pdf',...
    [{'\theta_{x,\omega}^*'},{'\theta_{x,\omega}(t)'}],[black,blue],['--','- '],[1 1],...
    'Time [s]','Magnitude [-]',...
    ['Adaptive \theta_{x,\omega}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I']
```



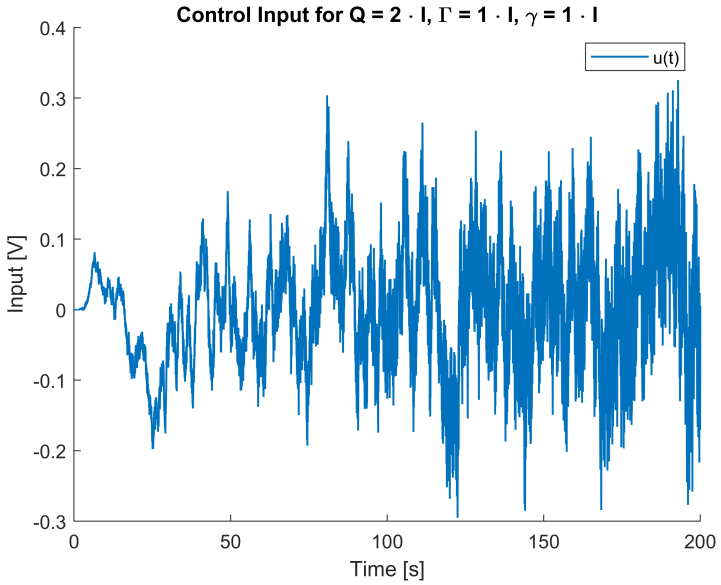

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[zeros(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,8)],'noise_error_Optimal.pdf',['e = 0',{'e_\theta(t)'}],...
    [black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Error e_\theta(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



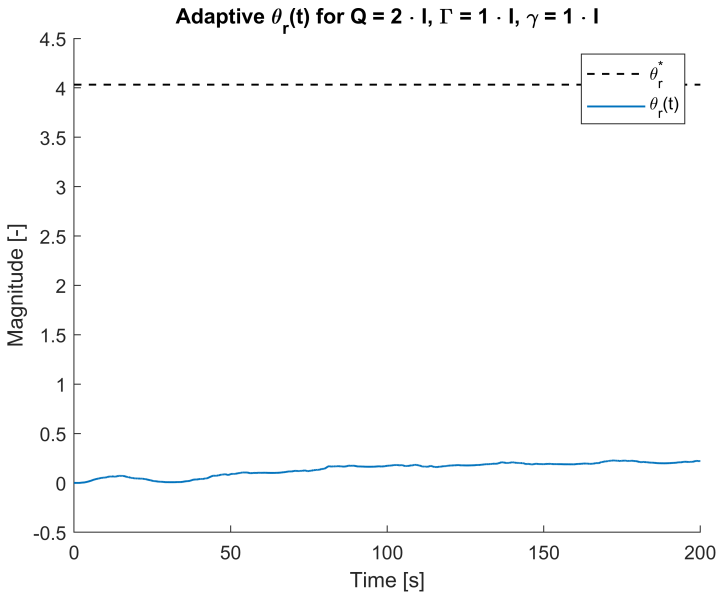

```
fig_num = fig_num+1;

% "Optimal" Q = 2*I, Gamma =  1*I, gamma = 1*I. Sinusoidal reference signal
ref_signal = 1;
[Am, gm, Bm, P, b, Gamma, gamma,theta_r_star, theta_x_theta_star, theta_x_omega_star]=...
    myObj.prepareSimulation(Q_gain(4), wm(3), zetam(2), Gamma_gain(3), gamma_gain(3),g,Ap);
optimal = sim("CA3_model.slx"); fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[optimal.yout{1}.Values.Data(:,7), optimal.yout{1}.Values.Data(:,1),
    optimal.yout{1}.Values.Data(:,2)],'noise_sin_Optimal_Output.pdf',[{'r(t)'},{'\theta_m(t)'},
    {'\theta_{DC}(t)'}],[black, red, blue],['--','- ','- '],[1 1 1],'Time [s]','Magnitude [-]',
    'System response for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



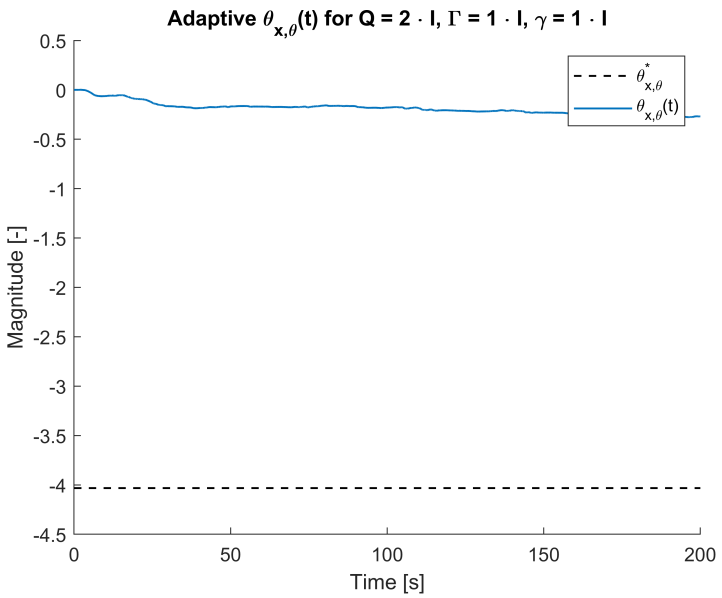

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,optimal.yout{1}.Values.Data(:,3),'noise_sin_u_Optimal.pdf',...
    {'u(t)'},blue,'- ',1,'Time [s]','Input [V]',...
    ['Control Input for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I'],fig_num);
```



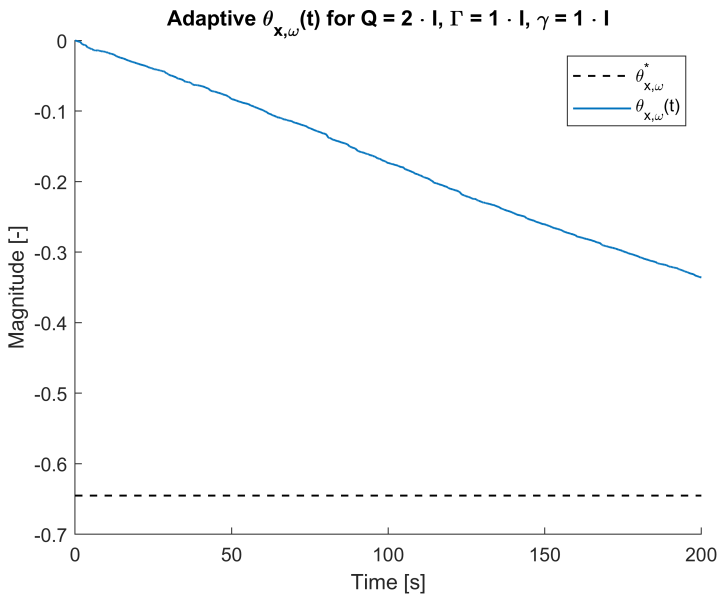

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_r_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,4)],'noise_sin_theta_r_Optimal.pdf',[{'\theta_r^*'},...
    {'\theta_r(t)'}],[black,blue],['--','- '],[1 1],'Time [s]','Magnitude [-]',...
    'Adaptive \theta_r(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I',fig_num);
```



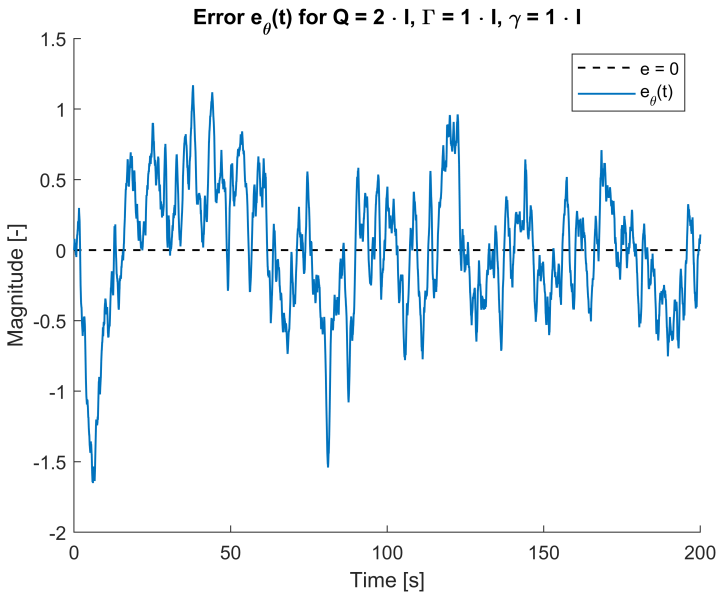

```
fig_num = fig_num+1;
myObj.plotMe(optimal.tout,[theta_x_theta_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,5)],'noise_sin_theta_x_theta_Optimal.pdf',...
    [{'\theta_{x,\theta}^*'},{'\theta_{x,\theta}(t)'}],[black, blue],['--','- '],[1 1],...
    'Time [s]','Magnitude [-]',...
    ['Adaptive \theta_{x,\theta}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I'],...
    fig_num); fig_num = fig_num+1;
```




```
myObj.plotMe(optimal.tout,[theta_x_omega_star*ones(length(optimal.tout),1),...
    optimal.yout{1}.Values.Data(:,6)],'noise_sin_theta_x_omega_Optimal.pdf',...
    [{'\theta_{x,\omega}^*'},{'\theta_{x,\omega}(t)'}],[black,blue],['--','- '],[1 1],...
    'Time [s]','Magnitude [-]',...
    ['Adaptive \theta_{x,\omega}(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I'],...
    fig_num); fig_num = fig_num+1;
```




```
myObj.plotMe(optimal.tout,[zeros(length(optimal.tout),1),optimal.yout{1}.Values.Data(:,8)],...
    'noise_sin_error_Optimal.pdf',['e = 0',{'e_\theta(t)'}],[black,blue],['--','- '],[1 1],...
    'Time [s]','Magnitude [-]',...
    ['Error e_\theta(t) for Q = 2 \cdot I, \Gamma = 1 \cdot I, \gamma = 1 \cdot I'],fig_num);
```




```
fig_num = fig_num+1;
```

```
%% Functions Container
% Function plotMe() to automate the plotting of simulation results.
% Function () to calculate parameters for simulation;

classdef functionsContainer
   methods (Static)
        function plotMe(timeData,yData,figureName,legendEntry,colorLine,...
                styleLine, widthLine, labelX, labelY, titlePlot, fig_num)
            max = length(yData(1,:));
            figure(fig_num);
            hold on;
            for i = 1:max
               plot(timeData,yData(:,i),'Color',colorLine((3*i-2):(3*i)),...
                   'LineStyle',styleLine((2*i-1):(2*i)),'LineWidth',widthLine(i));
            end
            title(titlePlot);
            legend(legendEntry);
            xlabel(labelX);
            ylabel(labelY);
            hold off;
            exportgraphics(gcf,figureName);
        end
        function [Am,gm,Bm,P,b,Gamma,gamma,theta_r_star,theta_x_theta_star,...
                theta_x_omega_star]=prepareSimulation...
                (Q_gain, wm, zetam, Gamma_gain, gamma_gain,g,Ap)
            % Reference Model State Space Description
            Am = [0 1; -wm^2 -2*zetam*wm];
            gm = wm^2;
            b = [0; 1];
            Bm = gm*b;

            % Perfect control input
            % u^* = (theta_x^*)^T x_p + theta_r^* r
            % Perfect control gains
            theta_r_star = gm/g;
            th_x_star = 1/g*(Am-Ap);
            theta_x_theta_star = th_x_star(2,1);
            theta_x_omega_star = th_x_star(2,2);

            % Choose Q for Lyapunov equation
            Q = Q_gain*eye(2);

            % Solve for P
            P = lyap(Am', Q);

            % Choose Gamma and gamma
            Gamma = Gamma_gain*eye(2);
            gamma = gamma_gain;
        end
   end
end
```