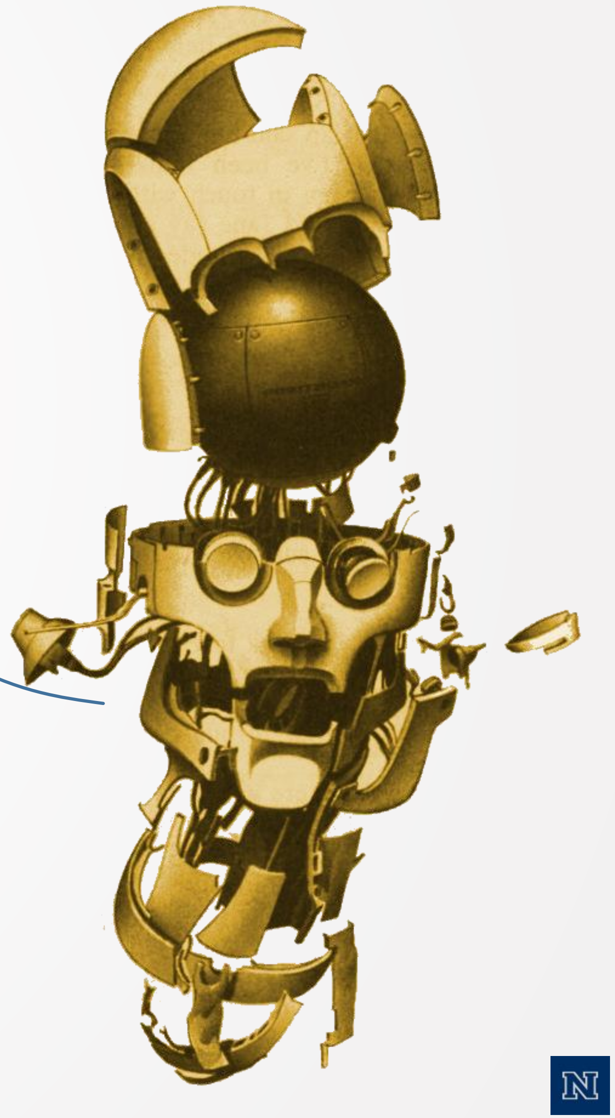
Autonomous Mobile Robot Design

Topic: Guidance and Control – Introduction and PID Loops

Dr. Kostas Alexis (CSE)

Autonomous Robot Challenges

How do I control where to go?



Autonomous Mobile Robot Design

Topic: Basic principles and prerequisites for control

Dr. Kostas Alexis (CSE)

What is it all about?

- **Control theory** is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems with inputs, and how their behavior is modified by feedback.

What is it all about?

- **Control theory** is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems with inputs, and how their behavior is modified by feedback.
- To do that in a systematic way, we should develop a **model-based** approach on control synthesis.

1. Modeling of Dynamic Systems

- Generic form of dynamic system:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

- **x** is the state vector, **u** is the input vector.
- Generic linear state space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

  - And given specific output vector:

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

State Space Representation

- Generic linear (time invariant) state space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

- **A** is the system matrix (nxn), **B** is the input matrix (nxp), **C** is the output matrix (qxn), **D** is the feedforward matrix (qxp).
- In general a MIMO system (Multi-Input, Multi-Output)
- In fact a Linear Time Invariant system (LTI). More complex representations exist.

Transfer Function Representation

- Linear time invariant can be represented in the frequency domain using the Laplace Transform.
- Laplace Transform:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt$$

- Transfer function form to represent Single Input Single Output (SISO) relationships:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}$$

  - And equivalently:

$$G(s) = K \frac{(s - z_1)(s - z_2)\ldots(s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2)\ldots(s - p_{n-1})(s - p_n)}$$

  - $p_1 \ldots p_n$ correspond to the poles and $z_1 \ldots z_m$ to the zeros.

State Space to Transfer Function

- Transformation from state space to transfer function:

$$G(s) = \frac{Y(s)}{U(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

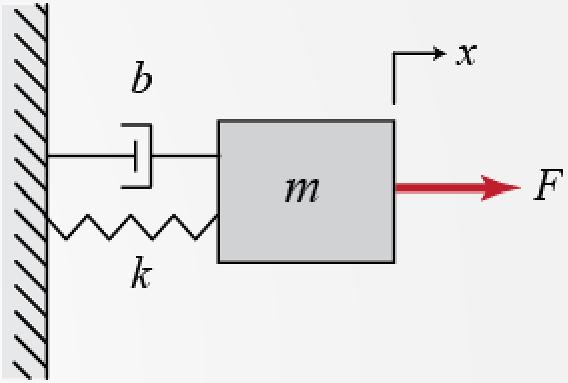
- Possibly leads to multiple transfer functions

Dynamic system example

- System ordinary differential equation:

$$m\ddot{x} = \sum F$$

$$F - b\dot{x} - kx = m\ddot{x}$$



- Or in state space form:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

- Or in transfer function form:

$$F(s) - bsX(s) - kX(s) = ms^2X(s) \Rightarrow$$

$$\frac{X(s)}{F(s)} = \frac{F(s)}{ms^2 + bs + k}$$

2. Analysis of Dynamic Systems

- Time-Domain Response of an LTI system
- Frequency-Domain Response of an LTI system
- Stability of an LTI system

Time-Domain Response

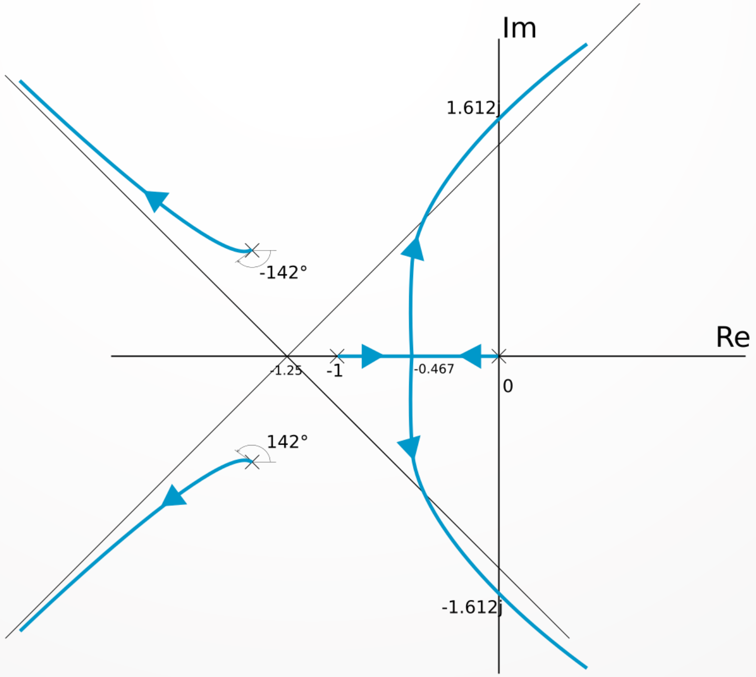
- The time response of a linear dynamic system consists of the **sum of the transient response** which depends on the initial conditions **and the steady-state response** which depends on the system input.
- These correspond to the free (homogeneous or zero input) and the forced (inhomogeneous or non-zero input) solutions of the governing differential equations respectively.

Frequency-Domain Response

- All the examples presented in this tutorial are modeled by linear constant coefficient differential equations and are thus linear time-invariant (LTI). LTI systems have the extremely important property that if the input to the system is sinusoidal, then the steady-state output will also be sinusoidal at the same frequency but in general with different magnitude and phase. These magnitude and phase differences as a function of frequency comprise the **frequency response** of the system.
- The frequency response of a system can be found from the transfer function in the following way: create a vector of frequencies (varying between zero or "DC" to infinity) and compute the value of the plant transfer function at those frequencies. If G(s) is the open-loop transfer function of a system and ω is the frequency vector, we then plot G(jω) versus ω. Since G(jω) is a complex number, we can plot both its magnitude and phase (the Bode Plot) or its position in the complex plane (the Nyquist Diagram). Both methods display the same information in different ways.

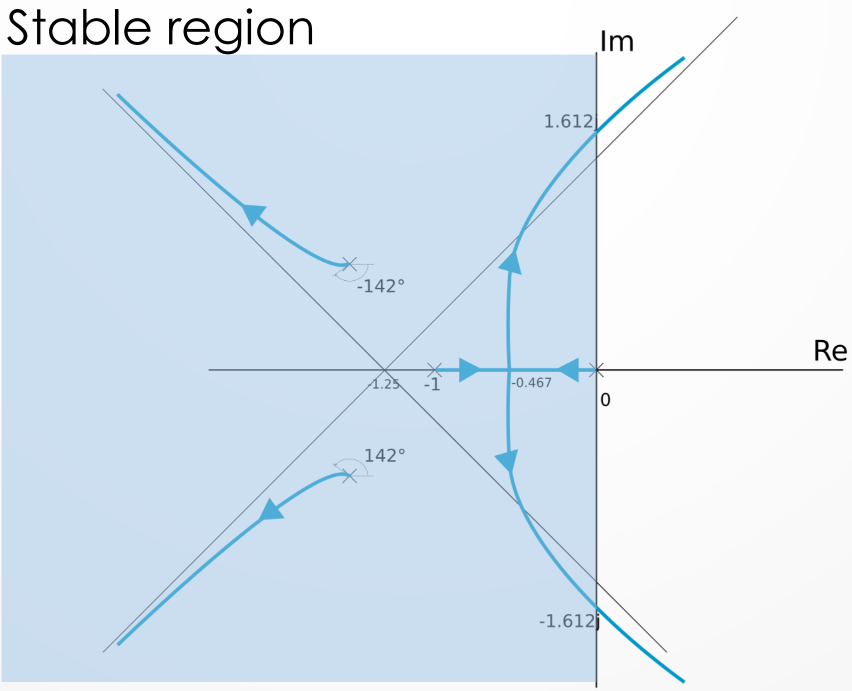
Stability

- **Bounded Input Bounded Output** (**BIBO**) sense: a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.
- **Using transfer functions:** the system is stable if ALL of the poles are in the left complex plane. If any of them is on the right then the system is unstable. If any of them has zero real component, the system is critically stable.



Stability

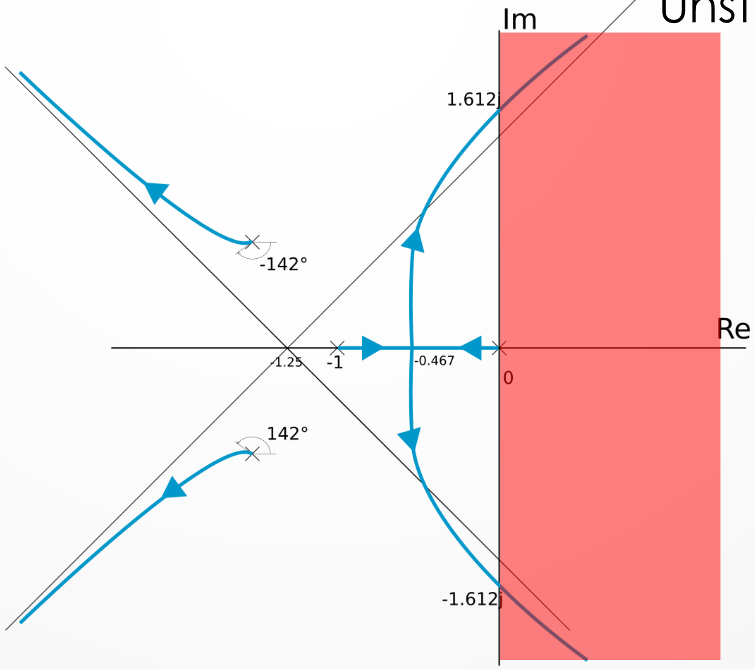
- **Bounded Input Bounded Output** (**BIBO**) sense: a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.
- **Using transfer functions:** the system is stable if ALL of the poles are in the left complex plane. If any of them is on the right then the system is unstable. If any of them has zero real component, the system is critically stable.



Stability

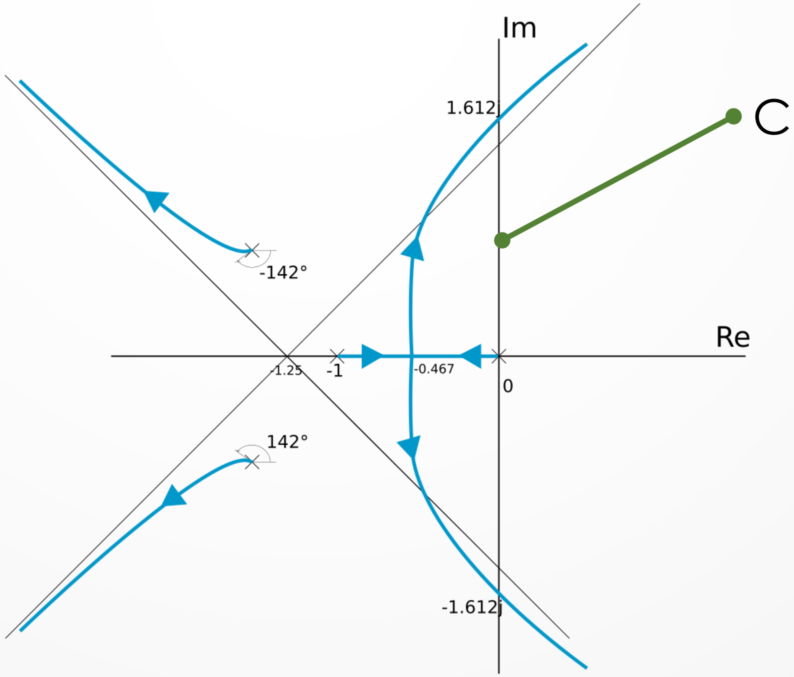
- **Bounded Input Bounded Output** (**BIBO**) sense: a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.
- **Using transfer functions:** the system is stable if ALL of the poles are in the left complex plane. If any of them is on the right then the system is unstable. If any of them has zero real component, the system is critically stable.

Unstable region



Stability

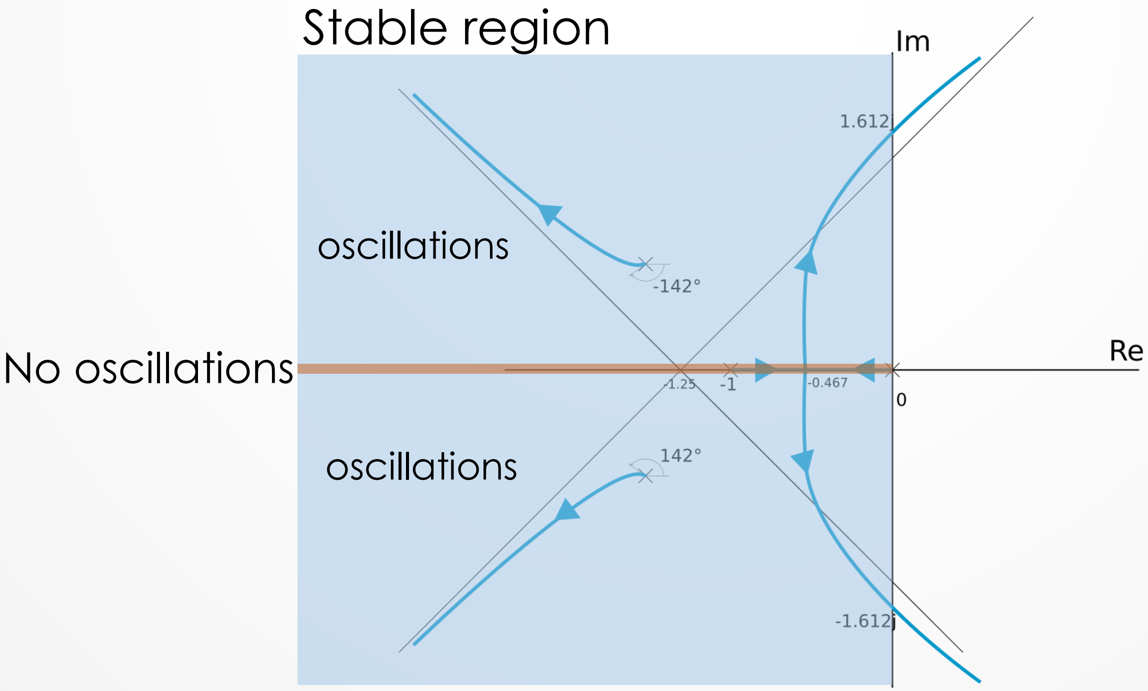
- **Bounded Input Bounded Output** (**BIBO**) sense: a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.
- **Using transfer functions:** the system is stable if ALL of the poles are in the left complex plane. If any of them is on the right then the system is unstable. If any of them has zero real component, the system is critically stable.



Critical stability

Stability

- **Bounded Input Bounded Output** (**BIBO**) sense: a system is stable if the output remains bounded for all bounded (finite) inputs. Practically, this means that the system will not "blow up" while in operation.
- **Using transfer functions:** the system is stable if ALL of the poles are in the left complex plane. If any of them is on the right then the system is unstable. If any of them has zero real component, the system is critically stable.



Stable region

oscillations

No oscillations

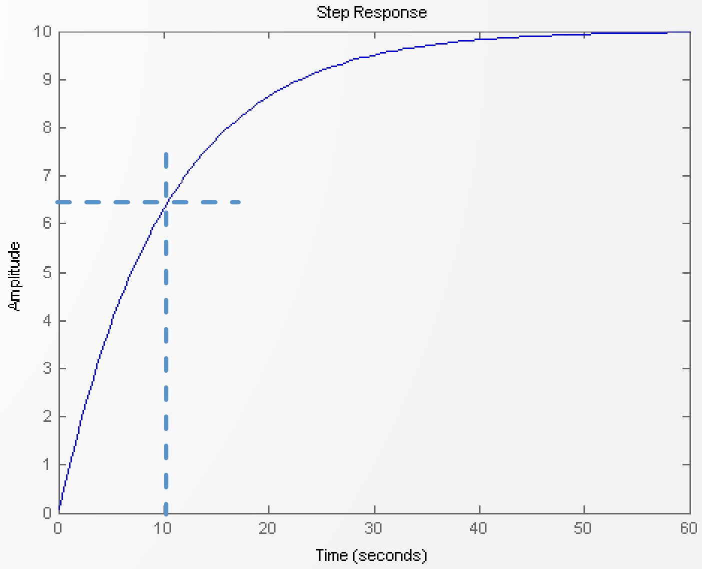
oscillations

1st Order Systems

- First order system representation:

$$\dot{y} + ay = bu \Rightarrow \tau\dot{y} + y = k_{dc}u$$
$$\Rightarrow G(s) = \frac{b}{s+a} = \frac{k_{dc}}{\tau s + 1}$$

- With **pole** – $\boldsymbol{a}$, **time constant** $\boldsymbol{\tau}$ and **dcgain** $\boldsymbol{k_{dc}}$
- Time constant is the value that the system reaches 63% of its steady-state value.



2nd Order Systems

- First order system representation:

$$m\ddot{y} + b\dot{y} + ky = f(t) \quad \Rightarrow \quad \ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = k_{dc}\omega_n^2 u$$

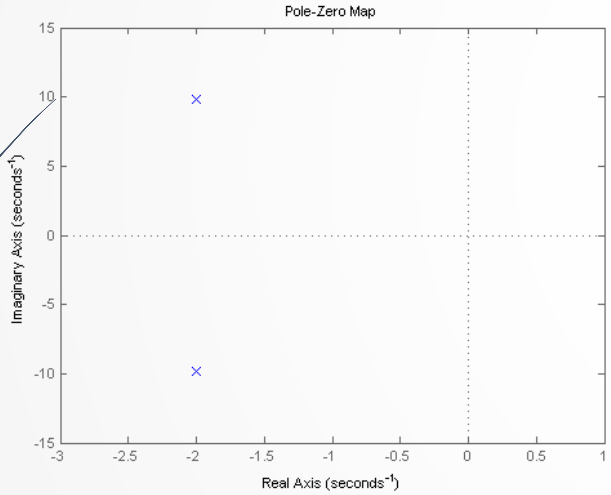
- Or in transfer function form:

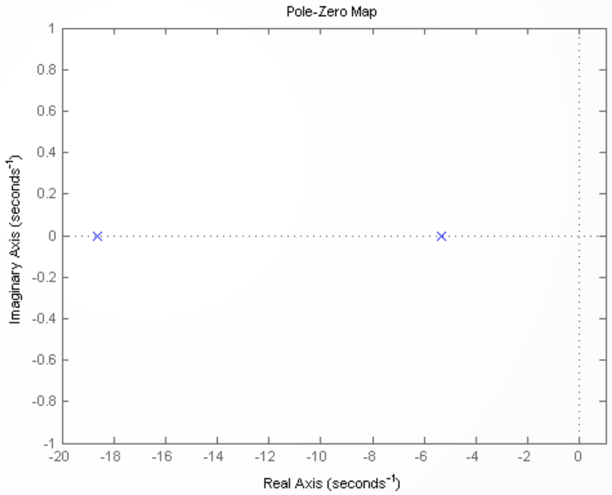
$$G(s) = \frac{1}{ms^2 + bs + k} = \frac{k_{dc}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

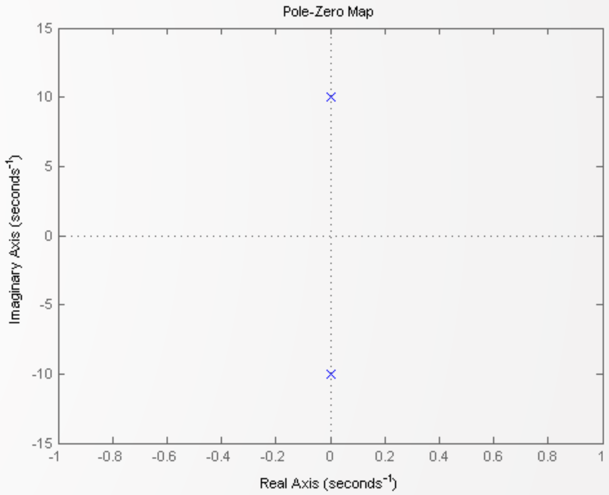
- **dcgain**: $k_{dc} = \frac{1}{k}$
- **damping ratio**: $\zeta = \frac{b}{2\sqrt{k/m}}$ (if $\zeta < 1$ system is underdamped)
- **natural frequency**: $\omega_n = \sqrt{\frac{k}{m}}$
- **poles/zeros**: $s_p = -\zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2}$

2nd Order Systems

- **Pole map**

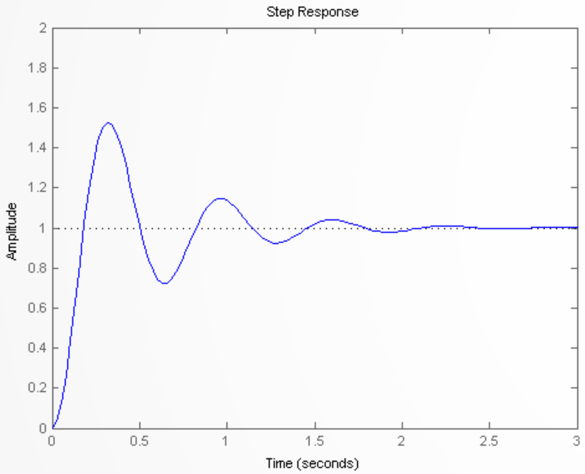


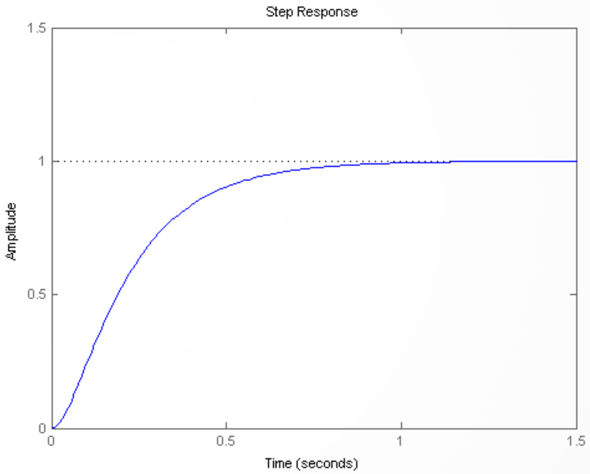


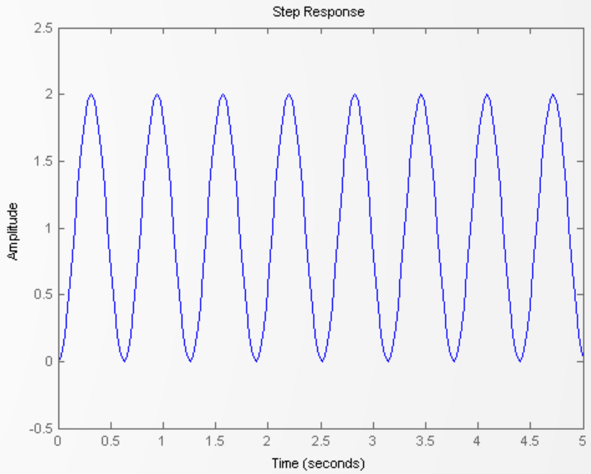


2nd Order Systems

- Time response

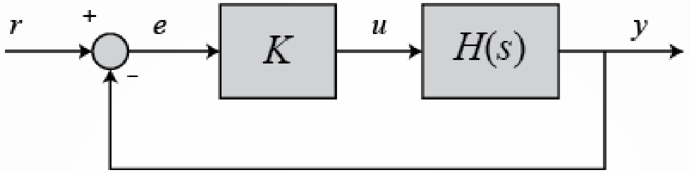






3. The Root Locus Method

- Consider a transfer function and a gain K:



- The closed-loop transfer function takes the form:

$$\frac{Y(s)}{R(s)} = \frac{KH(s)}{1+KH(s)}$$

- Thus the closed-loop poles are the solutions of:

$$1 + KH(s) = 0$$

- And if we write:

$$H(s) = b(s)/a(s)$$

- Then the poles are the solutions of:

$$\frac{a(s)}{K} + b(s) = 0$$

Root Locus Method example

- Consider the transfer function:

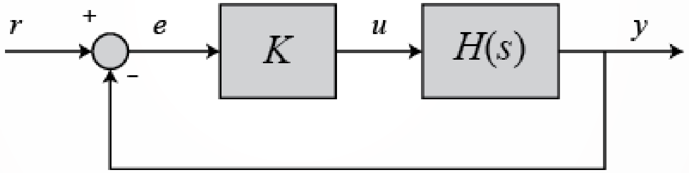
$$H(s) = \frac{Y(s)}{U(s)} = \frac{s+7}{s(s+5)(s+15)(s+20)}$$

Root Locus Method example

- Consider the transfer function:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s+7}{s(s+5)(s+15)(s+20)}$$

- Use a single-gain Controller (K):

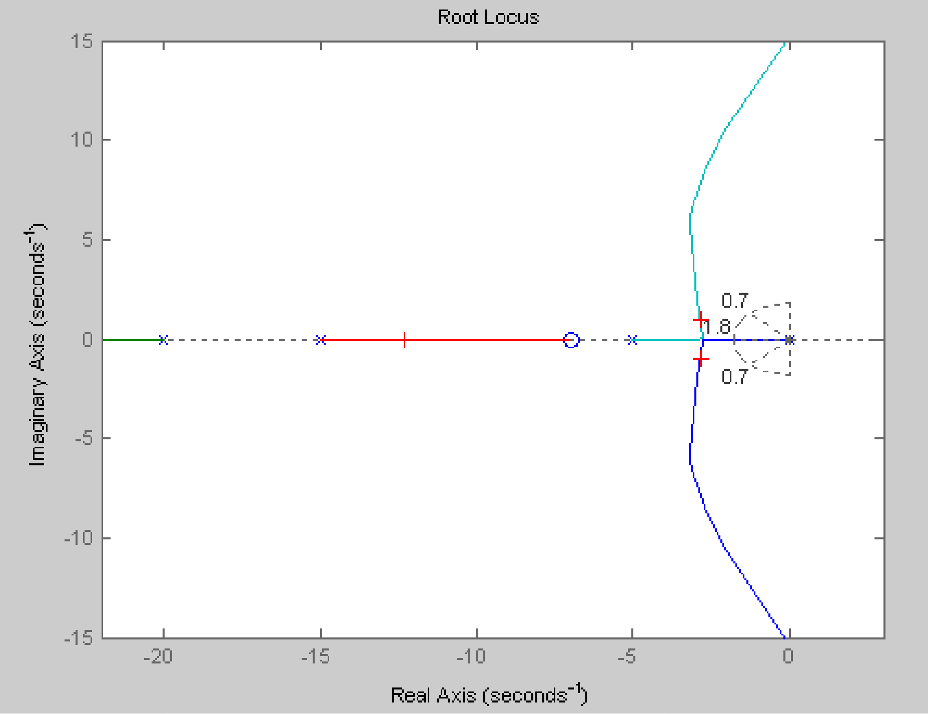


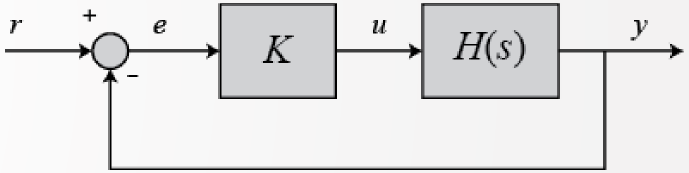
Root Locus Method example

- Consider the transfer function:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s+7}{s(s+5)(s+15)(s+20)}$$

- Root locus for a gain K:





5. State Space Control

- Consider the system:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

- How do we define **observability**?
- How do we define **controllability**?
- How do we **design a simple controller** to tune the dynamic response as desired? To place the poles where desired?

State Space system Stability

- Consider the system:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

- **Stability:** The system is stable if the **eigenvalues** of matrix A are all with negative real part.

State Space system Observability

- Consider the system:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

- **Observability:** The system is observable if the initial state x_0 can be determined from the system output, $y(t)$, over some finite time $t_0 < t < t_f$. For LTI systems, a system is observable if the observability matrix is of full rank.

$$OB = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

Rank of a Matrix

- The rank of a matrix **A** is the dimension of the vector space generated (or spanned) by its columns.
- This is the same as the dimension of the space spanned by its rows.
  - Rank row = Rank column
- It is a measure of the "nondegenerateness" of the system of linear equations and linear transformation encoded by A.

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

Rank of a Matrix

- The rank of a matrix **A** is the dimension of the vector space generated (or spanned) by its columns.
- This is the same as the dimension of the space spanned by its rows.
  - Rank row = Rank column
- It is a measure of the "nondegenerateness" of the system of linear equations and linear transformation encoded by A.

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \quad \text{Rank} = 2$$

State Space system Controllability

- Consider the system:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

- **Controllability:** A system is controllable if there exists a control input, $u(t)$, that transfers any state of the system to zero in finite time. For LTI systems, a system is controllable if the controllability matrix is of full rank.

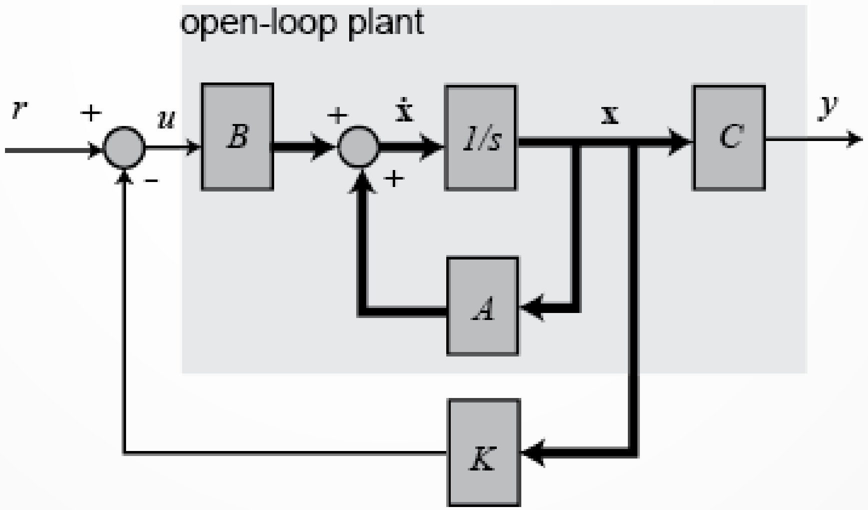
$$CO = [B|AB|A^2B|...|A^{n-1}B]$$

Pole Placement

- Consider the system:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$
$$y = C\mathbf{x} + Du$$

- Then the following structure is correct:



Pole Placement

- Assuming zero reference:

$$u = -Kx$$

- The state space equations of the **closed loop** system take the form:

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x}$$
$$\mathbf{y} = (\mathbf{C} - \mathbf{DK})\mathbf{x}$$

- The pole placement techniques is about how to find the gain matrix K such that the poles of the closed-loop system go to desired locations. This is under the assumption of an LTI system and infinite actuator dynamic range and bandwidth.

Autonomous Mobile Robot Design

Topic: PID Flight Control for Multirotors

Dr. Kostas Alexis (CSE)

MAV Dynamics

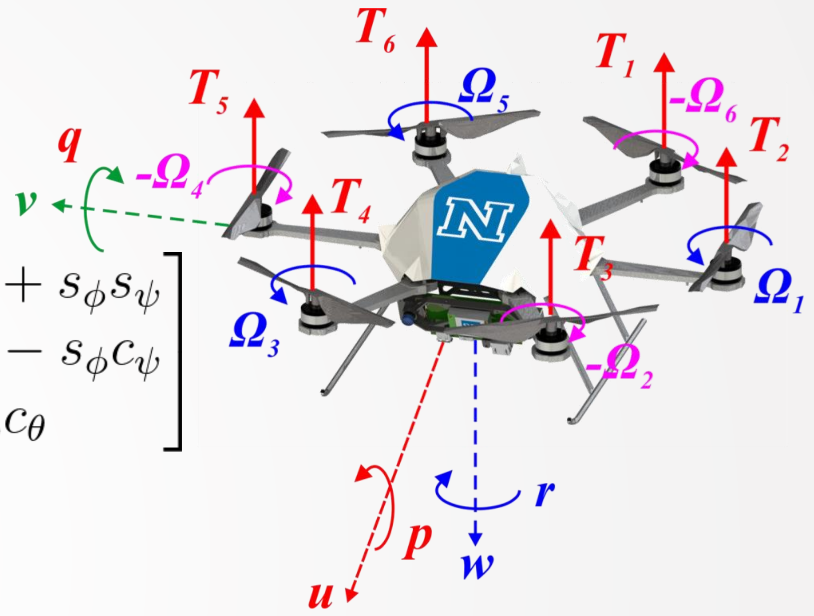
- To append the forces and moments we need to combine their formulation with

$$\begin{bmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{bmatrix} = \mathcal{R}_b^v \begin{bmatrix}u\\ v\\ w\end{bmatrix}, \quad \mathcal{R}_b^v = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{bmatrix}rv - qw\\ pw - ru\\ qu - pv\end{bmatrix} + \frac{1}{m}\begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{J_y - J_z}{J_x} qr\\ \frac{J_z - J_x}{J_y} pr\\ \frac{J_x - J_y}{J_z}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}\frac{1}{J_x} M_x\\ \frac{1}{J_y} M_y\\ \frac{1}{J_z} M_z\end{bmatrix}$$

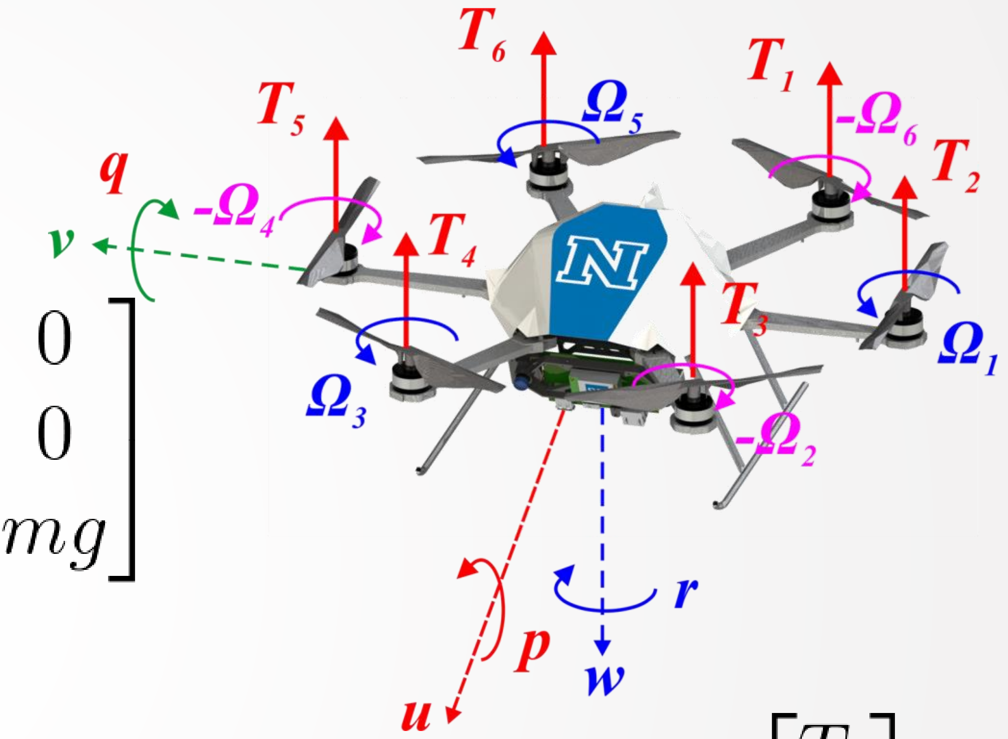


- ***Next step: append the MAV forces and moments***

MAV Dynamics

- MAV forces in the body frame:

$$\mathbf{f}_b = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{6} T_i \end{bmatrix} - \mathcal{R}_v^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$



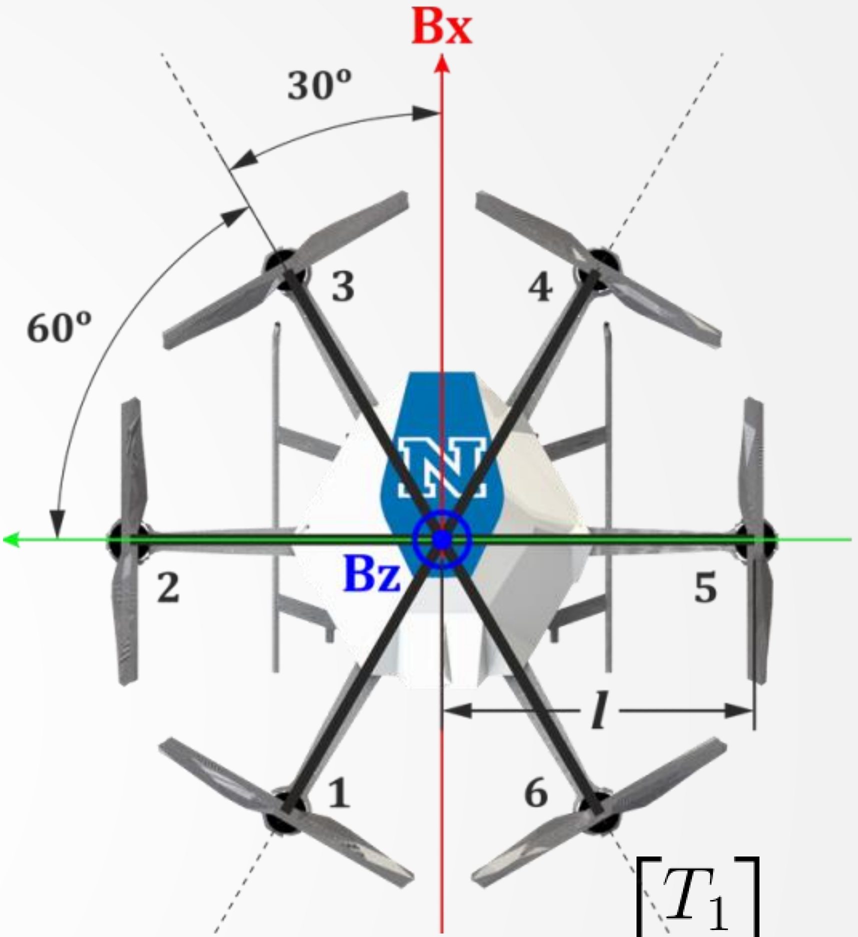
- Moments in the body frame:

$$\mathbf{m}_b = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} ls_{30} & l & ls_{30} & -ls_{30} & -l & ls_{30} \\ -lc_{60} & 0 & lc_{60} & lc_{60} & 0 & -lc_{60} \\ -k_m & k_m & -k_m & k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix}$$

MAV Dynamics

- MAV forces in the body frame:

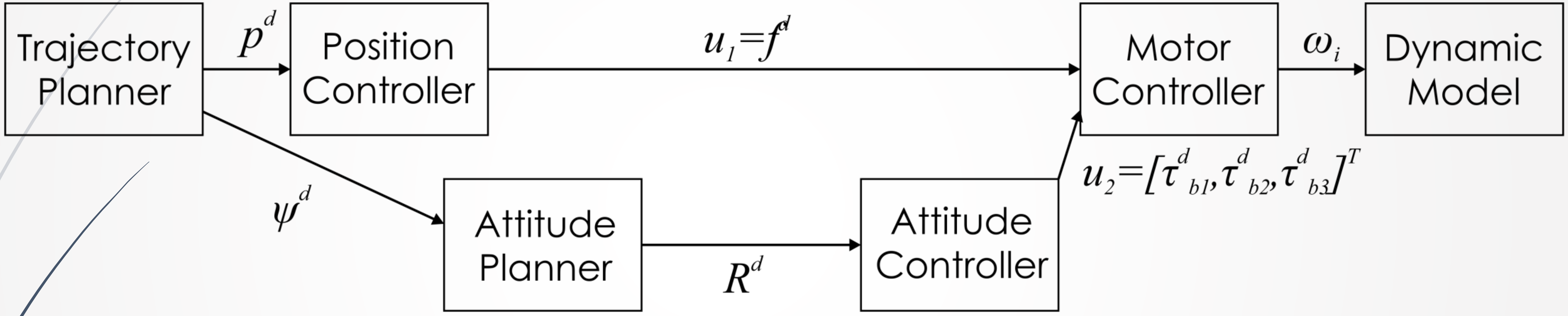
$$\mathbf{f}_b = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{6} T_i \end{bmatrix} - \mathcal{R}_v^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$



- Moments in the body frame:

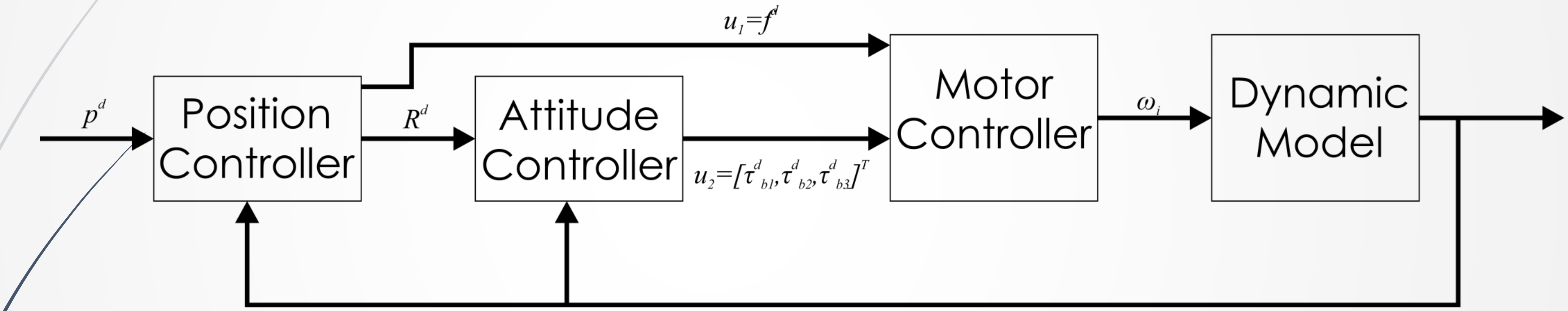
$$\mathbf{m}_b = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} ls_{30} & l & ls_{30} & -ls_{30} & -l & ls_{30} \\ -lc_{60} & 0 & lc_{60} & lc_{60} & 0 & -lc_{60} \\ -k_m & k_m & -k_m & k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix}$$

Control System Block Diagram



- There are simpler

Control System Block Diagram



- **Simplified loop**

Controlling a Multirotor along the x-axis

- Assume a single-axis case.
  - The system has to coordinate its pitching motion and thrust to move to the desired point ahead of its axis.
  - Roll is considered to be zero, yaw is considered to be constant. No initial velocity. No motion is expressed in any other axis.
  - A system of only two degrees of freedom.

Controlling a Multirotor along the x-axis

- Simplified linear dynamics

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 1/J_y\end{bmatrix} M_y$$

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ -g\end{bmatrix} \theta$$

Controlling a Multirotor along the x-axis

- Explanation of translation model

$$ma = F_x \Rightarrow ma = T_{TOT}\cos(-\theta) \Rightarrow ma \approx -\theta T_{TOT}$$

$$ma = -\theta mg$$

$$\text{s.t.} T_{TOT} = mg$$

$$a = \ddot{x}$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \theta$$

Controlling a Multirotor along the x-axis

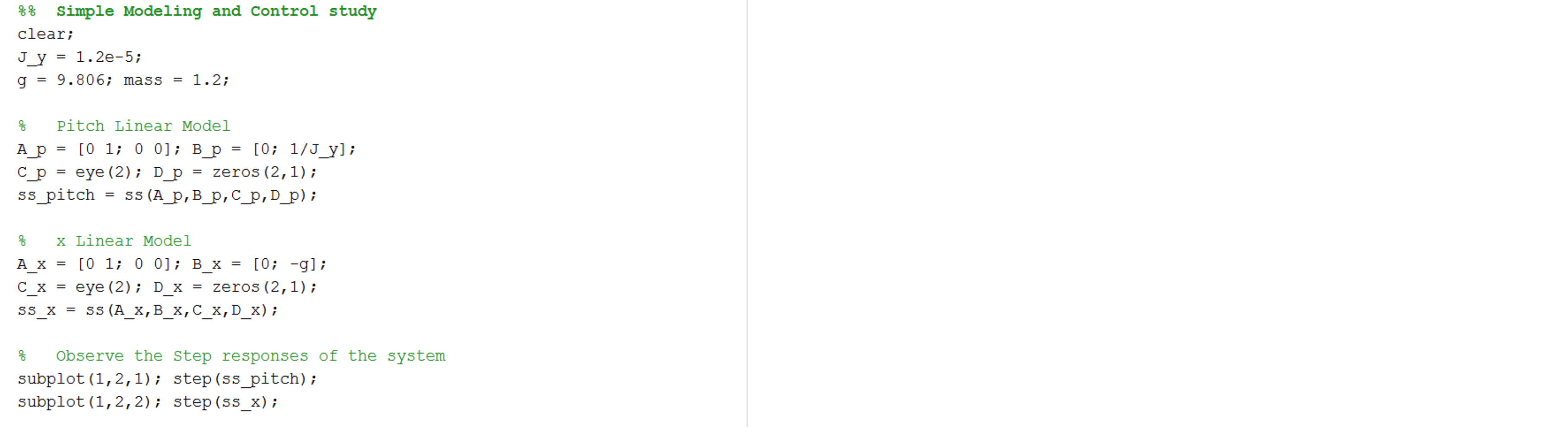
- Simplified linear dynamics

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 1/J_y\end{bmatrix} M_y$$

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ -g\end{bmatrix} \theta$$

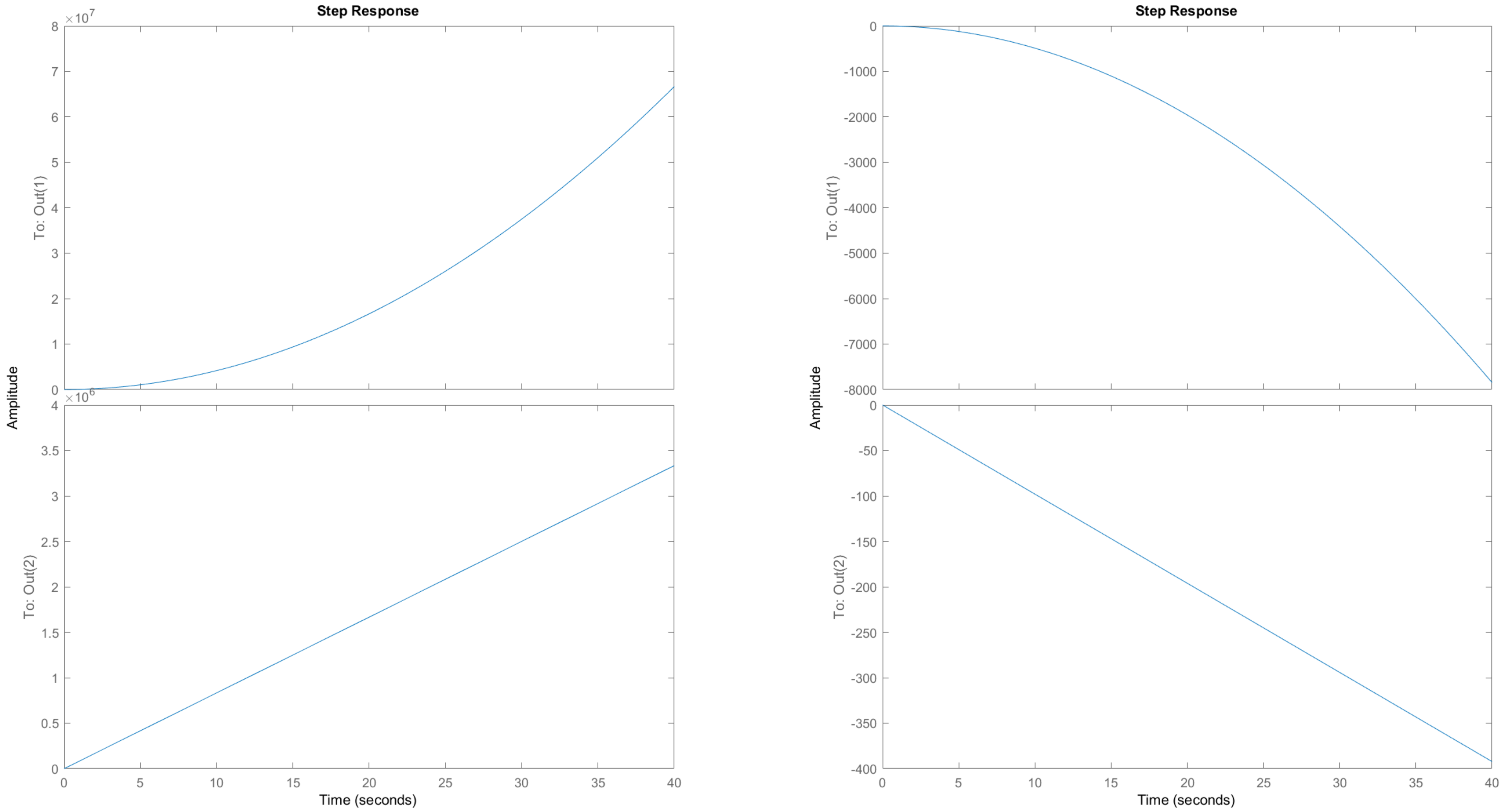
How does this system behave?

Controlling a Multirotor along the x-axis



```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);
```

Controlling a Multirotor along the x-axis



Controlling a Multirotor along the x-axis

- Simplified linear dynamics

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 1/J_y\end{bmatrix} M_y$$

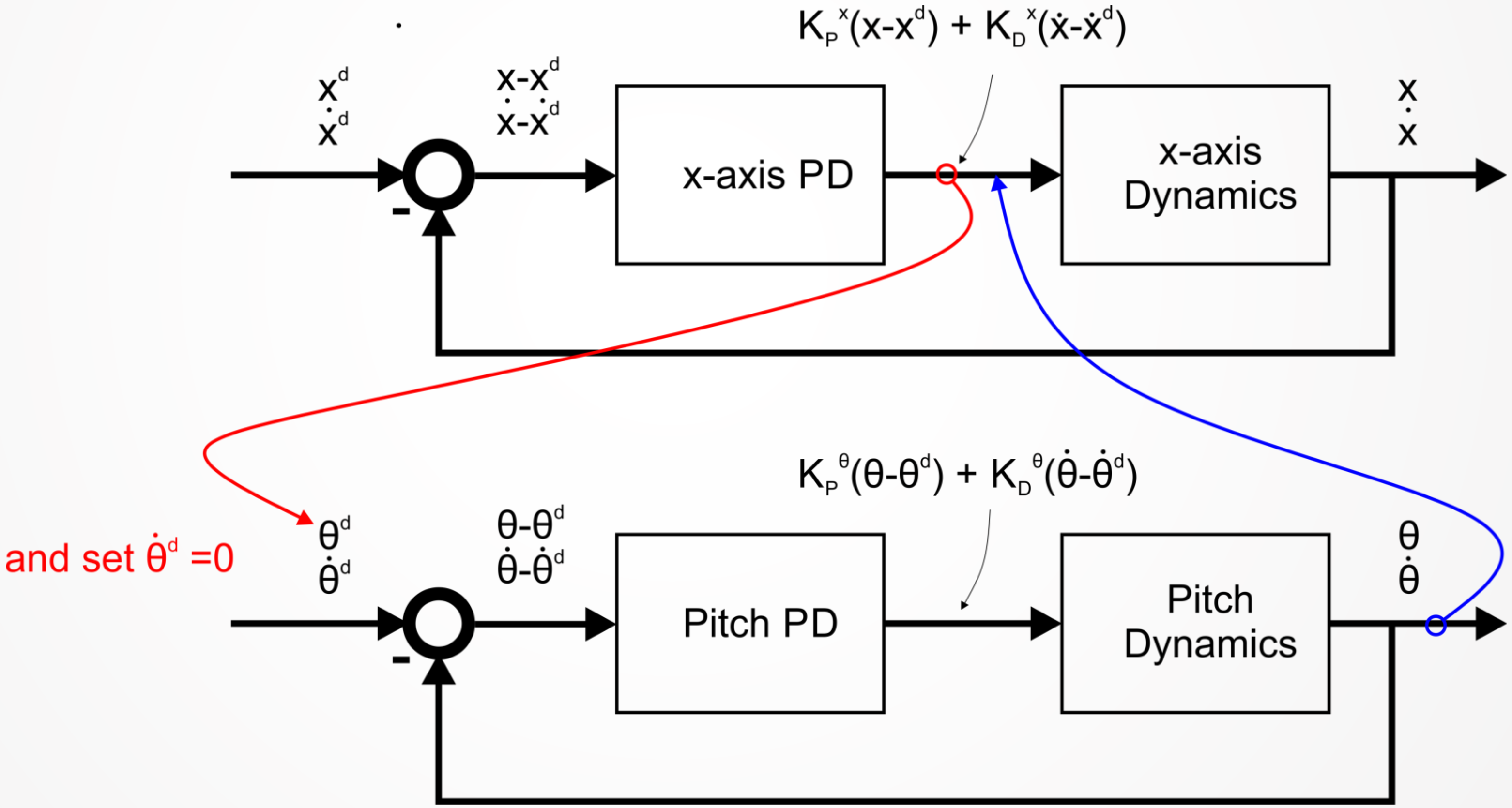
$$\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ -g\end{bmatrix}\theta$$

How to control this system?

Most common way: use of PD controllers

Controlling a Multirotor along the x-axis

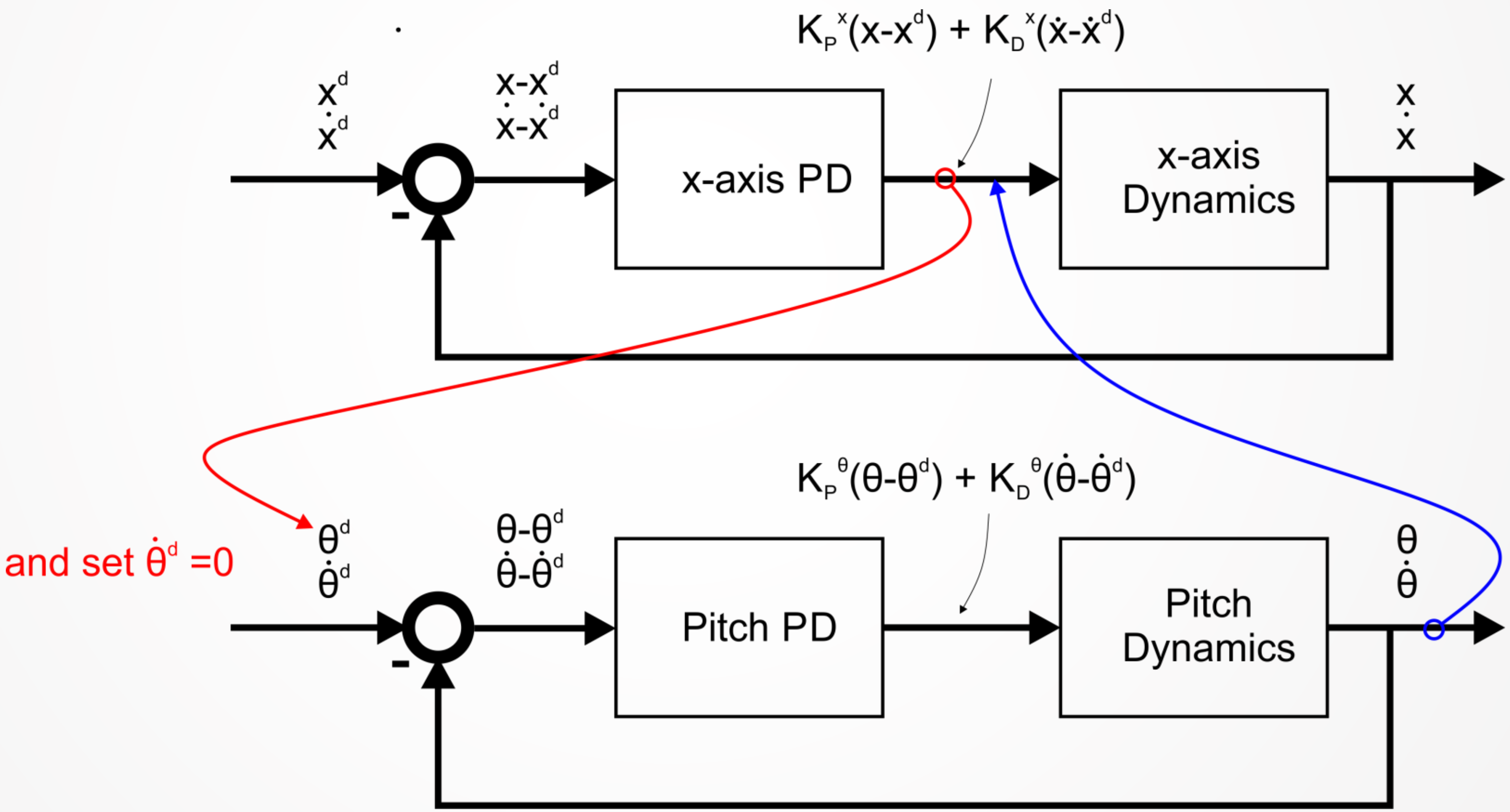
- Decoupled Control Structure



$K_P^x(x-x^d) + K_D^x(\dot{x}-\dot{x}^d)$
x^d
$\dot{x}^d$
$x-x^d$
$\dot{x}-\dot{x}^d$
x-axis PD
x-axis Dynamics
x
$\dot{x}$
-
$K_P^\theta(\theta-\theta^d) + K_D^\theta(\dot{\theta}-\dot{\theta}^d)$
and set $\dot{\theta}^d = 0$
θ^d
$\dot{\theta}^d$
$\theta-\theta^d$
$\dot{\theta}-\dot{\theta}^d$
Pitch PD
Pitch Dynamics
θ
$\dot{\theta}$
-

Controlling a Multirotor along the x-axis

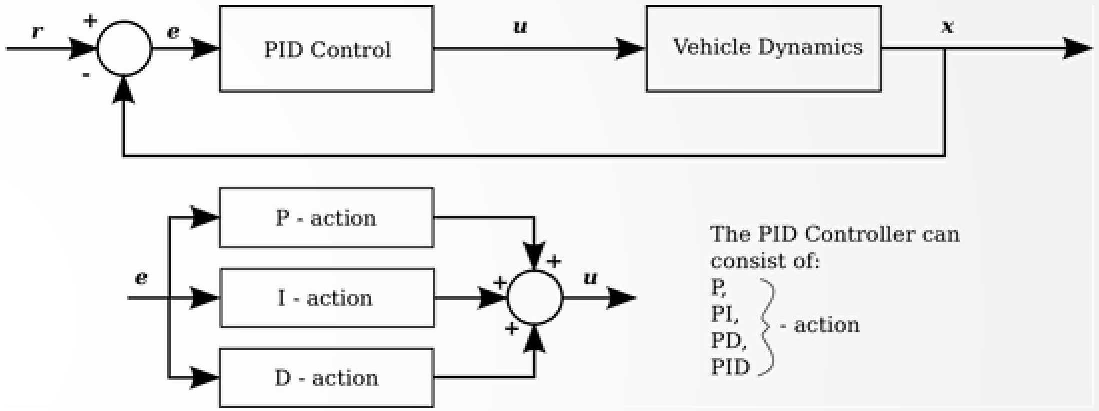
- Decoupled Control Structure


$K_P^x(x-x^d) + K_D^x(\dot{x}-\dot{x}^d)$
x^d
$\dot{x}^d$
$x-x^d$
$\dot{x}-\dot{x}^d$
x-axis PD
x-axis Dynamics
x
$\dot{x}$
-
$K_P^\theta(\theta-\theta^d) + K_D^\theta(\dot{\theta}-\dot{\theta}^d)$
and set $\dot{\theta}^d$ =0
θ^d
$\dot{\theta}^d$
$\theta-\theta^d$
$\dot{\theta}-\dot{\theta}^d$
Pitch PD
Pitch Dynamics
θ
$\dot{\theta}$
-

How to select the PD gains?

A mini introduction to PID Control

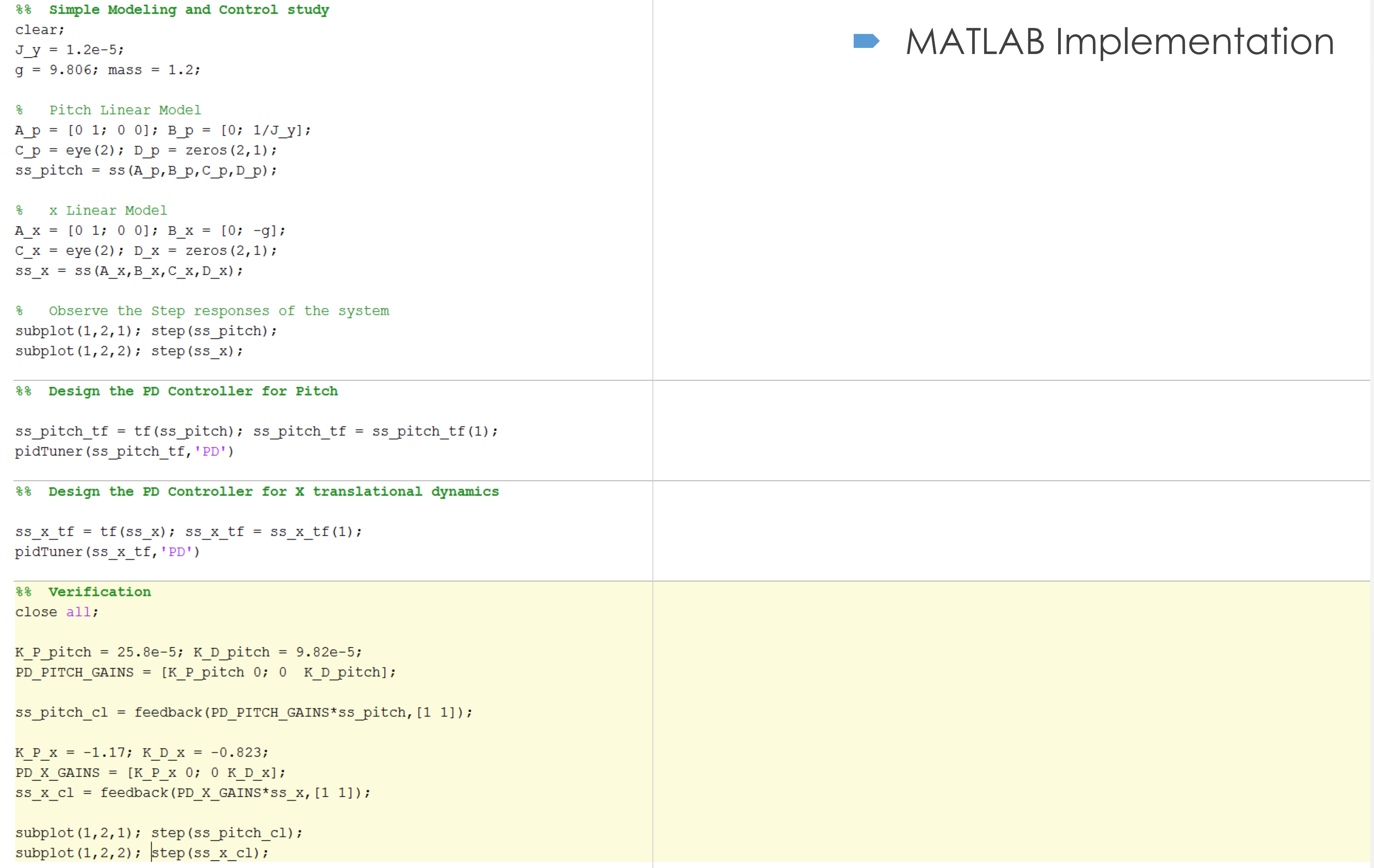
- **PID = Proportional-Integral-Derivative** feedback control
- It corresponds to one of the most commonly used controllers used in industry.
- It's success is based on its capacity to efficiently and robustly control a variety of processes and dynamic systems, while having an extremely simple structure and intuitive tuning procedures.
- Not comparable in performance with modern control strategies, but still the most common starting point



How to select the PD gains?

Controlling a Multirotor along the x-axis

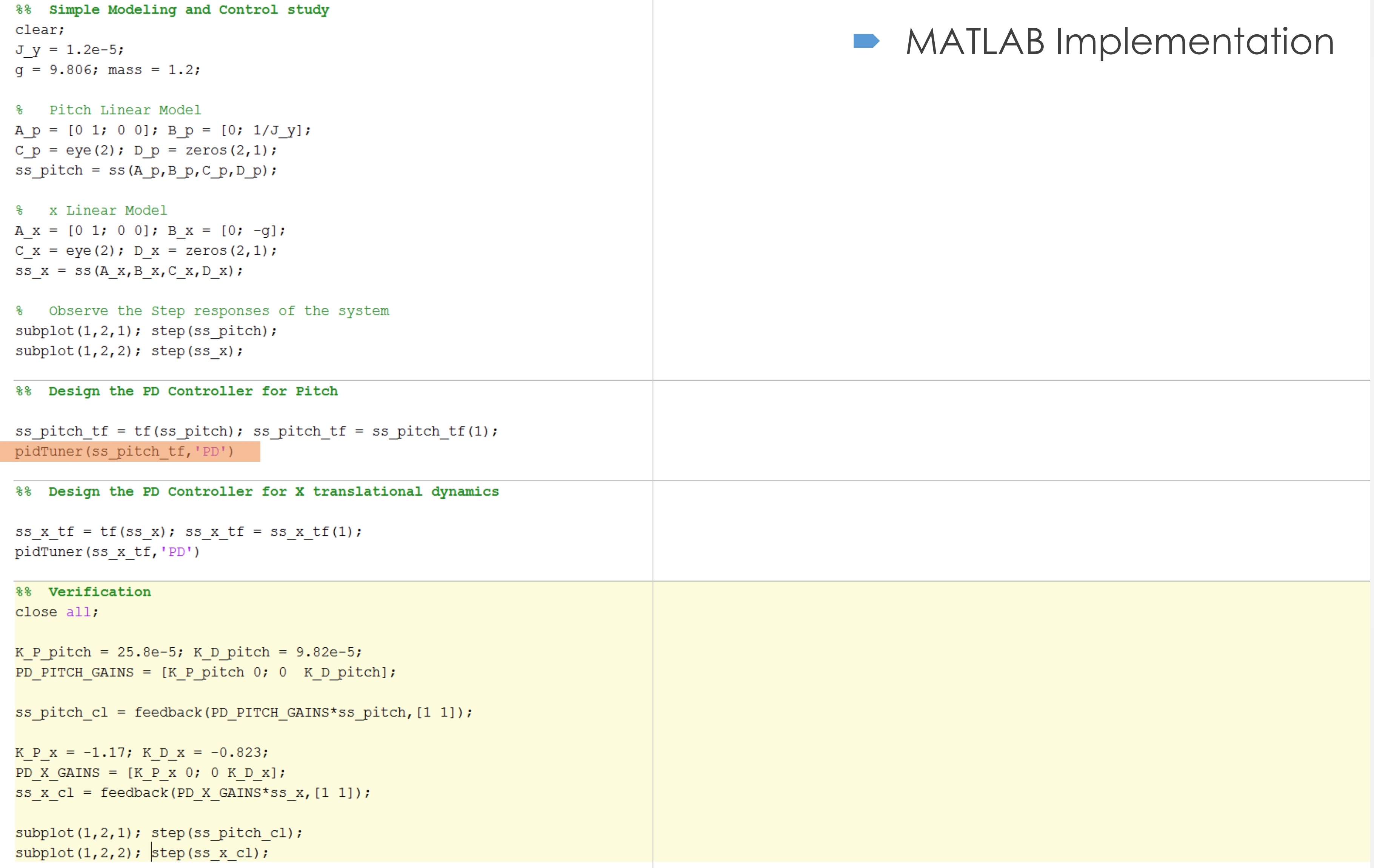
- MATLAB Implementation



```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);

%%  Design the PD Controller for Pitch

ss_pitch_tf = tf(ss_pitch); ss_pitch_tf = ss_pitch_tf(1);
pidTuner(ss_pitch_tf,'PD')

%%  Design the PD Controller for X translational dynamics

ss_x_tf = tf(ss_x); ss_x_tf = ss_x_tf(1);
pidTuner(ss_x_tf,'PD')

%%  Verification
close all;

K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0  K_D_pitch];

ss_pitch_cl = feedback(PD_PITCH_GAINS*ss_pitch,[1 1]);

K_P_x = -1.17; K_D_x = -0.823;
PD_X_GAINS = [K_P_x 0; 0 K_D_x];
ss_x_cl = feedback(PD_X_GAINS*ss_x,[1 1]);

subplot(1,2,1); step(ss_pitch_cl);
subplot(1,2,2); step(ss_x_cl);
```

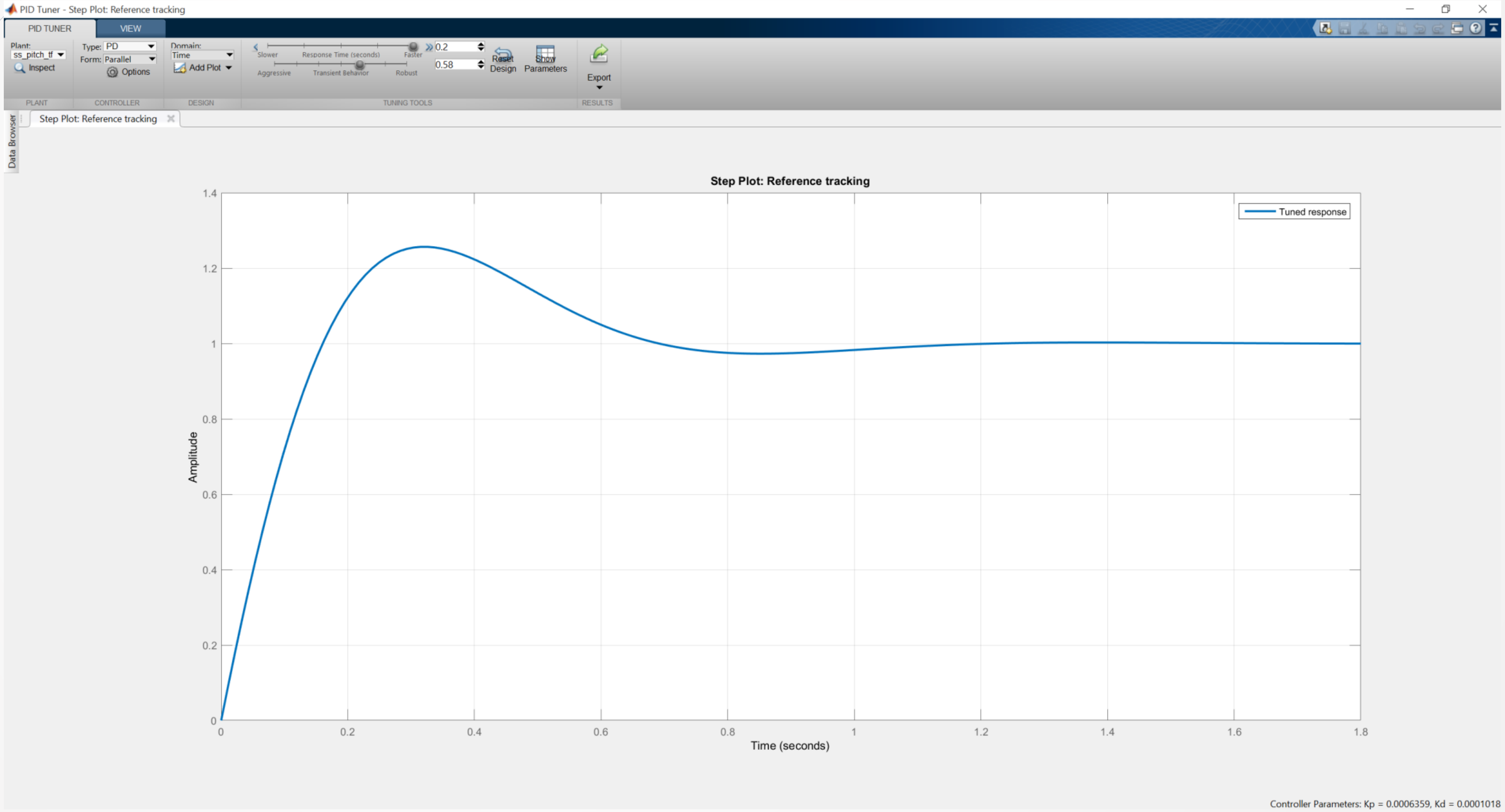
Controlling a Multirotor along the x-axis

- MATLAB Implementation



```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);

%%  Design the PD Controller for Pitch

ss_pitch_tf = tf(ss_pitch); ss_pitch_tf = ss_pitch_tf(1);
pidTuner(ss_pitch_tf,'PD')

%%  Design the PD Controller for X translational dynamics

ss_x_tf = tf(ss_x); ss_x_tf = ss_x_tf(1);
pidTuner(ss_x_tf,'PD')

%%  Verification
close all;

K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0  K_D_pitch];

ss_pitch_cl = feedback(PD_PITCH_GAINS*ss_pitch,[1 1]);

K_P_x = -1.17; K_D_x = -0.823;
PD_X_GAINS = [K_P_x 0; 0 K_D_x];
ss_x_cl = feedback(PD_X_GAINS*ss_x,[1 1]);

subplot(1,2,1); step(ss_pitch_cl);
subplot(1,2,2); step(ss_x_cl);
```

Controlling a Multirotor along the x-axis



PID Tuner - Step Plot: Reference tracking

Plant: ss_pitch_tf

Type: PD

Form: Parallel

Domain: Time

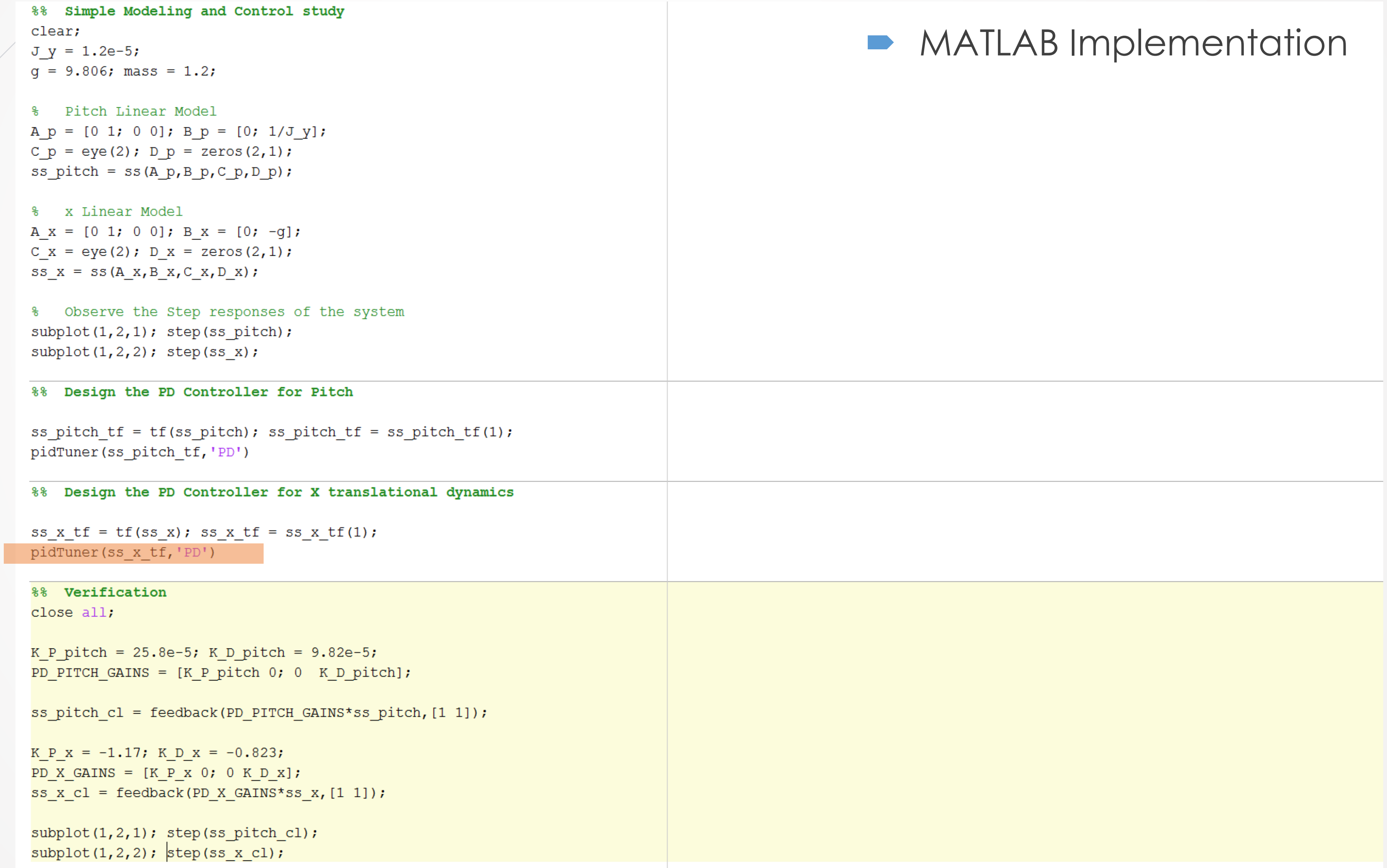
Response Time (seconds): 0.2

Transient Behavior: 0.58

Controller Parameters: Kp = 0.0006359, Kd = 0.0001018

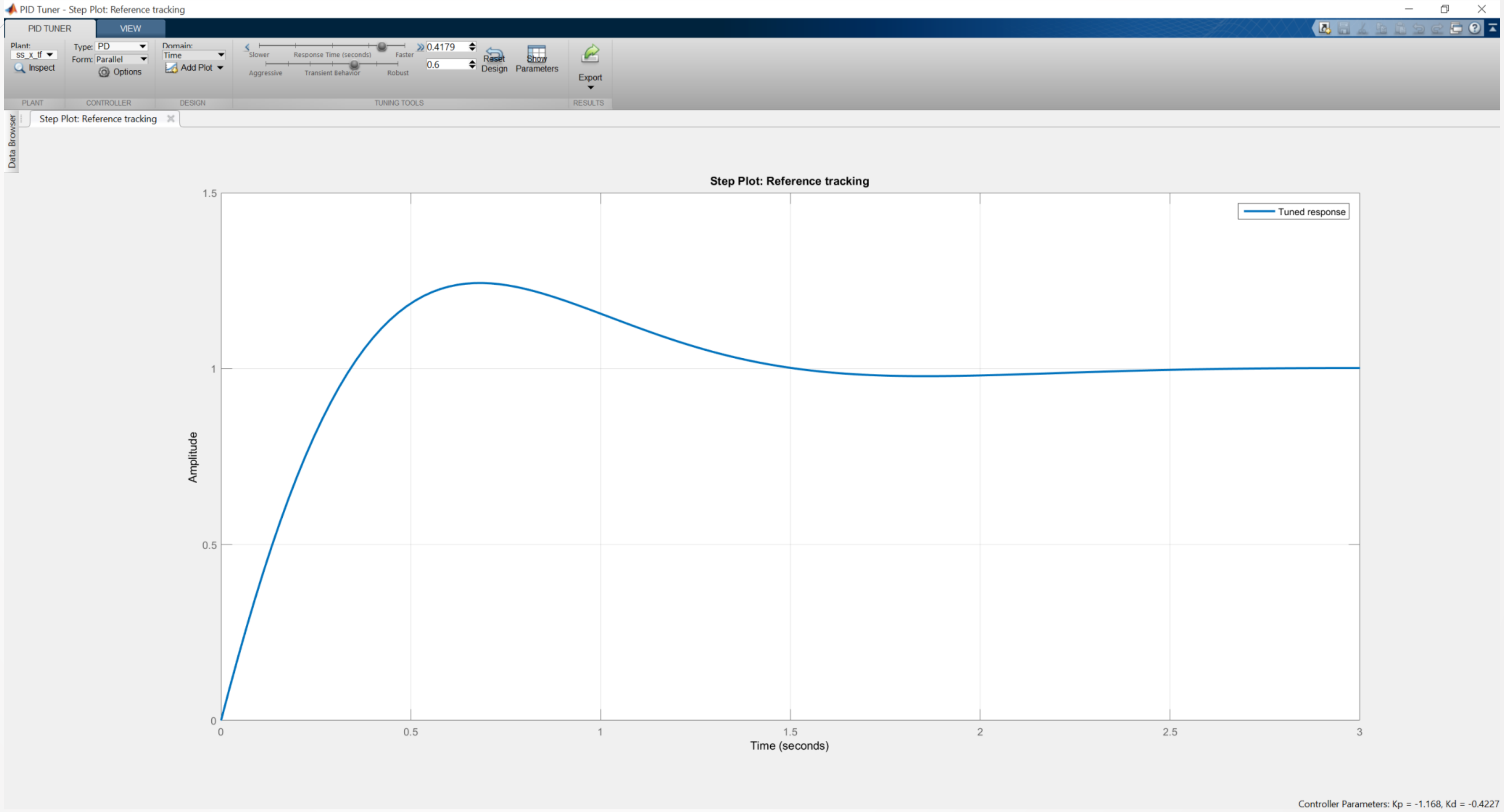
Controlling a Multirotor along the x-axis

- MATLAB Implementation



```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);

%%  Design the PD Controller for Pitch

ss_pitch_tf = tf(ss_pitch); ss_pitch_tf = ss_pitch_tf(1);
pidTuner(ss_pitch_tf,'PD')

%%  Design the PD Controller for X translational dynamics

ss_x_tf = tf(ss_x); ss_x_tf = ss_x_tf(1);
pidTuner(ss_x_tf,'PD')

%%  Verification
close all;

K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0  K_D_pitch];

ss_pitch_cl = feedback(PD_PITCH_GAINS*ss_pitch,[1 1]);

K_P_x = -1.17; K_D_x = -0.823;
PD_X_GAINS = [K_P_x 0; 0 K_D_x];
ss_x_cl = feedback(PD_X_GAINS*ss_x,[1 1]);

subplot(1,2,1); step(ss_pitch_cl);
subplot(1,2,2); step(ss_x_cl);
```

Controlling a Multirotor along the x-axis



PID Tuner - Step Plot: Reference tracking

Plant: ss_x_tf

Type: PD

Form: Parallel

Domain: Time

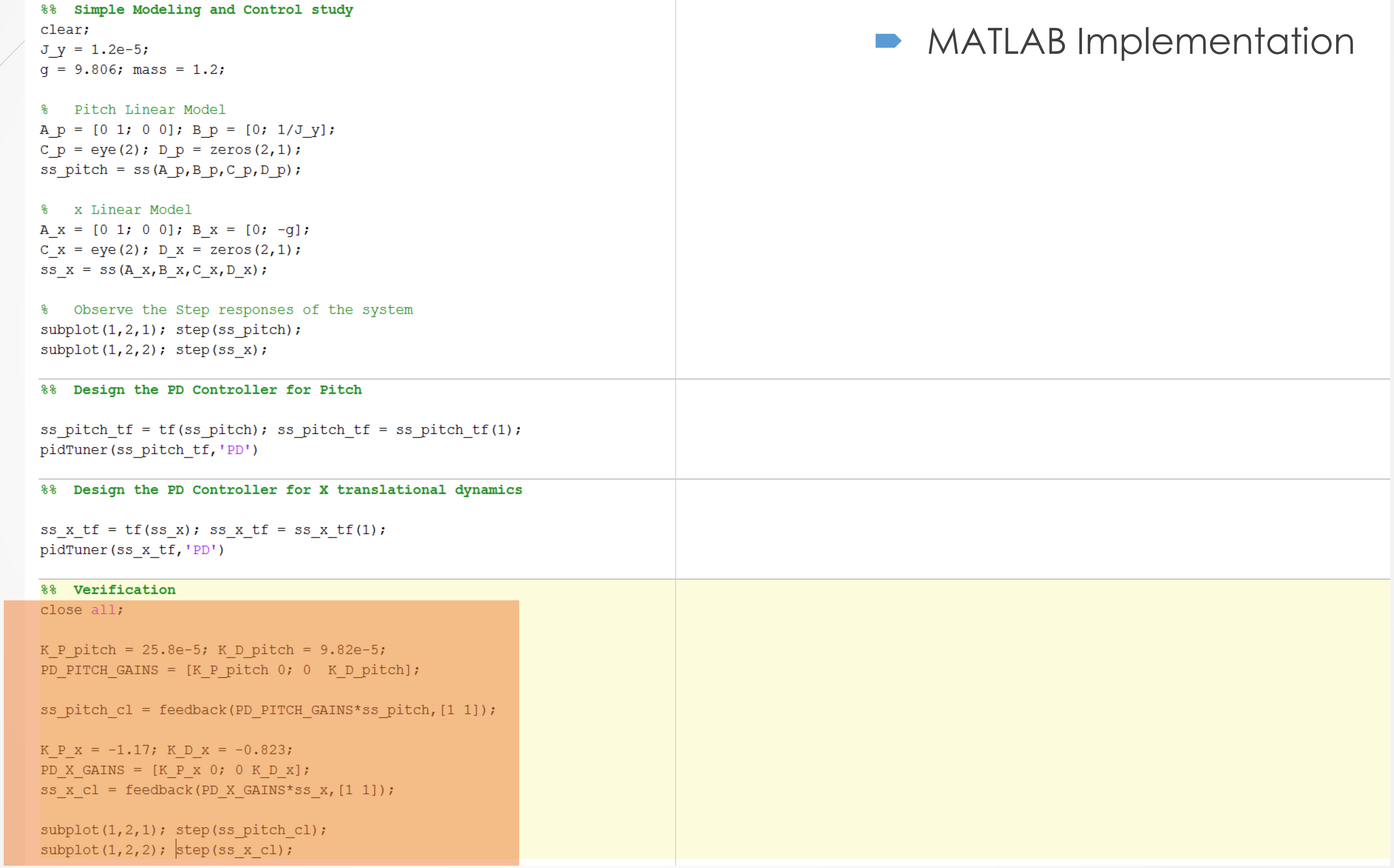
Response Time (seconds): 0.4179

Transient Behavior: 0.6

Controller Parameters: Kp = -1.168, Kd = -0.4227

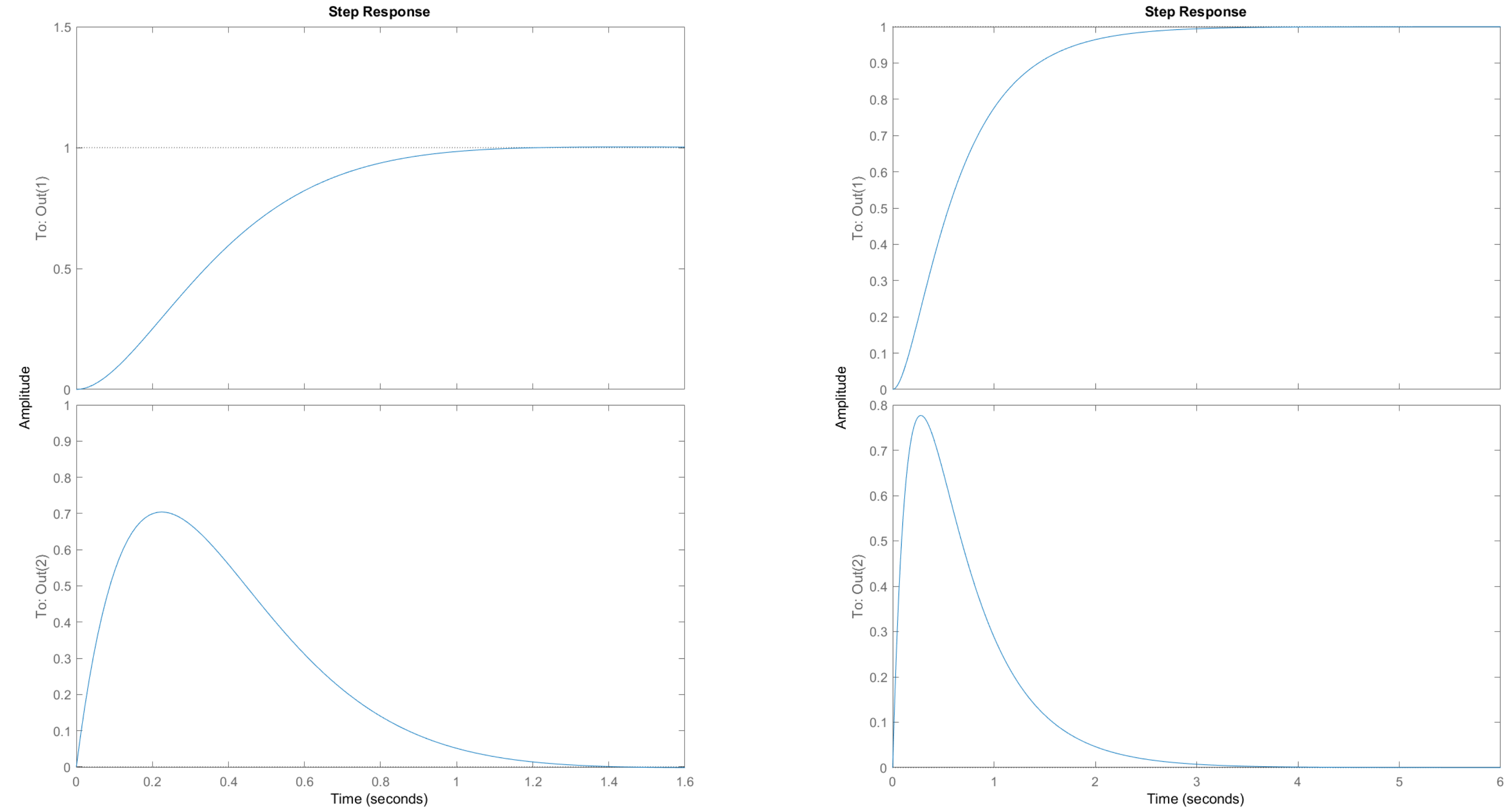
Controlling a Multirotor along the x-axis

- MATLAB Implementation



```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);

%%  Design the PD Controller for Pitch

ss_pitch_tf = tf(ss_pitch); ss_pitch_tf = ss_pitch_tf(1);
pidTuner(ss_pitch_tf,'PD')

%%  Design the PD Controller for X translational dynamics

ss_x_tf = tf(ss_x); ss_x_tf = ss_x_tf(1);
pidTuner(ss_x_tf,'PD')

%%  Verification
close all;

K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0  K_D_pitch];

ss_pitch_cl = feedback(PD_PITCH_GAINS*ss_pitch,[1 1]);

K_P_x = -1.17; K_D_x = -0.823;
PD_X_GAINS = [K_P_x 0; 0 K_D_x];
ss_x_cl = feedback(PD_X_GAINS*ss_x,[1 1]);

subplot(1,2,1); step(ss_pitch_cl);
subplot(1,2,2); step(ss_x_cl);
```

Controlling a Multirotor along the x-axis



Real-life Limitations

- The control margins of the aerial vehicle have limits and therefore the PID controller has to be designed account for these constraints.
- The integral term needs special caution due to the often critically stable or unstable characteristics expressed by unmanned aircraft.
- With the exception of hover/or trimmed-flight, an aerial vehicle is a nonlinear system. As the PID is controller, it naturally cannot maintain an equally good behavior for the full flight envelope of the system. A variety of techniques such as Gain scheduling are employed to deal with this fact.

Autonomous Mobile Robot Design

Topic: LQR Flight Control for Multirotors

Dr. Kostas Alexis (CSE)

Controlling a Multirotor along the x-axis

- Assume a single-axis multirotor.
  - The system has to coordinate its pitching motion and thrust to move to the desired point ahead of its axis.
  - Roll is considered to be zero, yaw is considered to be constant. No initial velocity. No motion is expressed in any other axis.
  - A system of only two degrees of freedom.

Controlling a Multirotor along the x-axis

- Simplified linear dynamics

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 1/J_y\end{bmatrix} M_y$$

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ -g\end{bmatrix} \theta$$

How does this system behave?

Optimal Model-Based Control

- Use model knowledge to design optimal control behaviors.
- The employed model must be simultaneously sufficiently accurate but also simple enough to enable efficient control computation.
- Optimal control can support linear and nonlinear systems as well as systems subject to state, output and input constraints. Further extensions (e.g. for hybrid systems) also exist.
- Established method for unconstrained linear systems regulation:
  - Linear Quadratic Regulator (LQR)
  - Generalization: Linear Quadratic Gaussian (LQG) control

Linear Quadratic Regulator

- Consider the system

$$\dot{x} = Ax + Bu$$

- and suppose we want to design state feedback control ***u*=*Fx*** to stabilize the system. The design of ***F*** is a trade-off between the transit response and the control effort. The optimal control approach to this design trade-off is to define the performance index (cost functional):

$$J = \int_0^\infty \left[x^T(t)Qx(t) + u^T(t)Ru(t)\right] dt$$

- and search for the control ***u*=*Fx*** that minimizes this index. ***Q*** is an ***n*x*m*** symmetric positive semidefinite matrix and ***R*** is an ***m*x*m*** symmetric positive definite matrix.

Linear Quadratic Regulator

- The matrix ***Q*** can be written as ***Q*=*M***$Q=M^TM$, where ***M*** is a pxn matrix, with $p \leq n$. With this representation:

$$x^T Q x = x^T M^T M x = z^T z$$

- Where $z=Mx$ can be viewed as a controlled input.

- Optimal Control Problem: Find $u(t)=Fx(t)$ to maximize J subject to the model:

$$\dot{x} = Ax + Bu$$

- Since J is defined by an integral over $[0, \infty)$, the first question we need to address is: Under what conditions will J exist and be finite?

Linear Quadratic Regulator

- Write J as:

$$J = \lim_{t_f \to \infty} \bar{J}(t_f)$$

$$\bar{J}(t_f) = \int_0^{t_f} \left[x^T(t)Qx(t) + u^T(t)Ru(t)\right] dt$$

- $\bar{J}(t_f)$ is a monotonically increasing function of t_f. Hence, as $t_f \to \infty$, $\bar{J}(t_f)$ either converges to a finite limit or diverges to infinity.

- Under what conditions will $J = \lim_{t_f \to \infty} \bar{J}(t_f)$ be finite?

Linear Quadratic Regulator

- Recall that ***(A,B)*** is stabilizable if the uncontrollable eigenvalues of ***A***, if any, have negative real parts.
- Notice that ***(A,B)*** is stabilizable if ***(A,B)*** is controllable or $\boldsymbol{R_e[\lambda(A)] < 0}$
- Definition: ***(A,C)*** is detectable if the observable eigenvalues of ***A***, if any, have negative real parts.
- Lemma 1: Suppose ***(A,B)*** is stabilizable, ***(A,M)*** is detectable, where $\boldsymbol{Q=M^TM}$, and $\boldsymbol{u(t)=Fx(t)}$. Then, ***J*** is finite for every $\boldsymbol{x}(0) \in \boldsymbol{R}^n$ if and only if:

$$\boldsymbol{R_e[\lambda(A+BF)] < 0}$$

Linear Quadratic Regulator

- **Remarks:**
  - The need for ***(A,B)*** to be stabilizable is clear, for otherwise there would be no ***F*** such that:
  - To see why detectability of ***(A,M)*** is needed, consider:

$$\dot{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{u}, \quad \boldsymbol{J} = \int_0^\infty \boldsymbol{u}^2(\boldsymbol{t})\, \boldsymbol{dt}$$

$$\boldsymbol{A} = 1, \;\; \boldsymbol{B} = 1, \;\; \boldsymbol{M} = 0, \;\; \boldsymbol{R} = 1$$

  - ***(A,B)*** is controllable, but ***(A,M)*** is not detectable

$$\boldsymbol{F} = 0 \;\Rightarrow\; \boldsymbol{u}(\boldsymbol{t}) = 0 \;\Rightarrow\; \boldsymbol{J} = 0$$

  - The control is clearly optimal and results in a finite ***J*** but it does not stabilize the system because $\boldsymbol{A} + \boldsymbol{BF} = \boldsymbol{A} = 1$

Linear Quadratic Regulator

- **Lemma 2:** For any stabilizing control ***u(t)=Fx(t)***, the cost given by:

$$J = x(0)^T W x(0)$$

- where ***W*** is a symmetric positive semidefinite matrix that satisfies the Lyapunov equation:

$$W(A+BF) + (A+BF)^T W + Q + F^T R F = 0$$

- Remark: The control ***u(t)=Fx(t)*** is stabilizing if:

$$R_e[\lambda(A+BF)] < 0$$

Linear Quadratic Regulator

- **Theorem:** Consider the system: $\dot{x} = Ax + Bu$ and the performance index:

$$J = \int_0^\infty \left[x^T(t)Qx(t) + u^T(t)Ru(t)\right] dt$$

- where $Q=M^TM$, R is symmetric and positive definite, (A,B) is stabilizable, and (A,M) is detectable. The optimal control is:

$$u(t) = -R^{-1}B^TPx$$

- where P is the symmetric positive semidefinite solution of the Algebraic Riccati Equation (ARE):

$$0 = PA + A^TP + Q - PBR^{-1}B^TP$$

Linear Quadratic Regulator

- **Remarks:**
  - Since the control is stabilizing:

$$R_e[\lambda(A - BR^{-1}B^T P)] < 0$$

  - The control is optimal among all square integratable signals ***u(t)***, not just among ***u(t)=Fx(t)***

  - The Ricatti equation can have multiple solutions, but only one of them is positive semidefinite.

Linear Quadratic Regulator

- **Typical penalty matrices selection:**

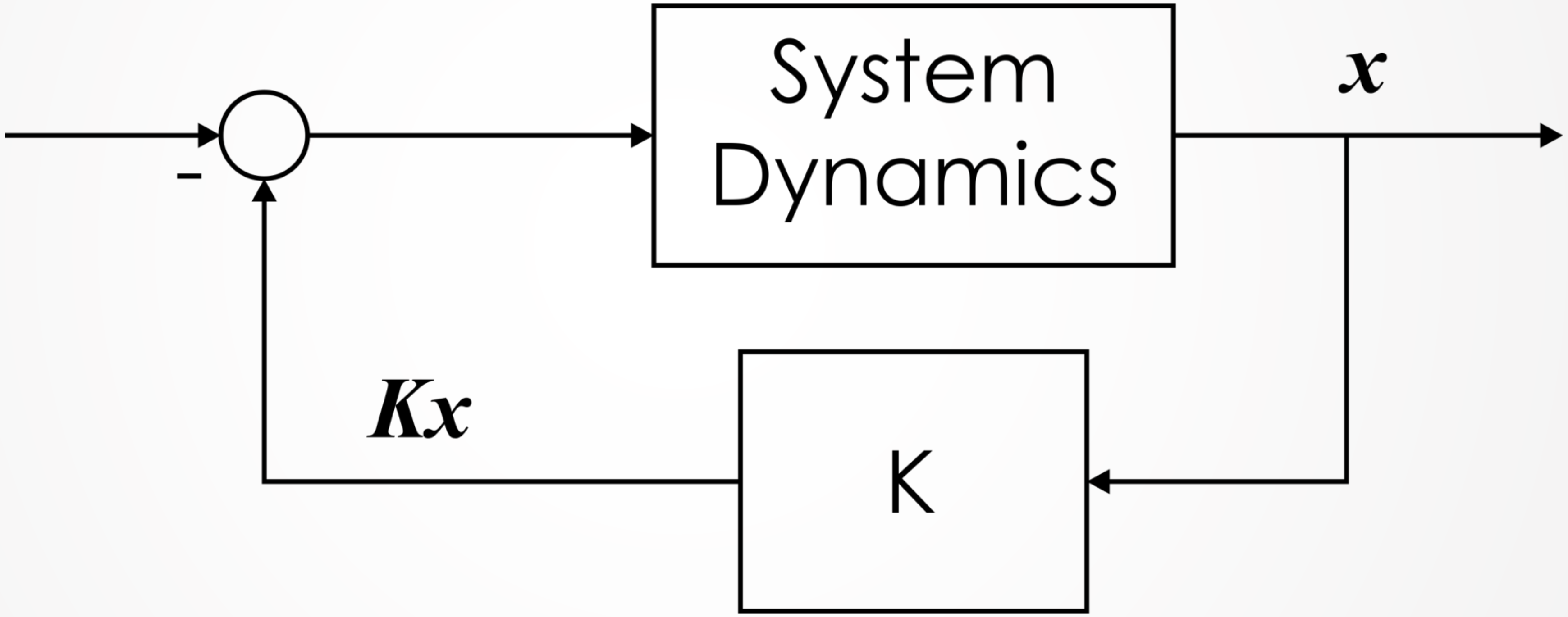
$$Q = \begin{bmatrix} q_1 & & & \\ & q_2 & & \\ & & \ddots & \\ & & & q_n \end{bmatrix}, \quad R = \rho \begin{bmatrix} r_1 & & & \\ & r_2 & & \\ & & \ddots & \\ & & & r_m \end{bmatrix}$$

$$q_i = \frac{1}{t_{si}(x_{imax})^2}, \quad r_i = \frac{1}{(u_{imax})^2}, \quad \rho > 0$$

- t_{si} is the desired settling time of xi
- x_{imax} is a constraint on $|x_i|$
- u_{imax} is a constraint on $|u_i|$
- ρ is chosen to tradeoff regulation versus control effort

Linear Quadratic Regulator

- **LQR Loop (F=K)**



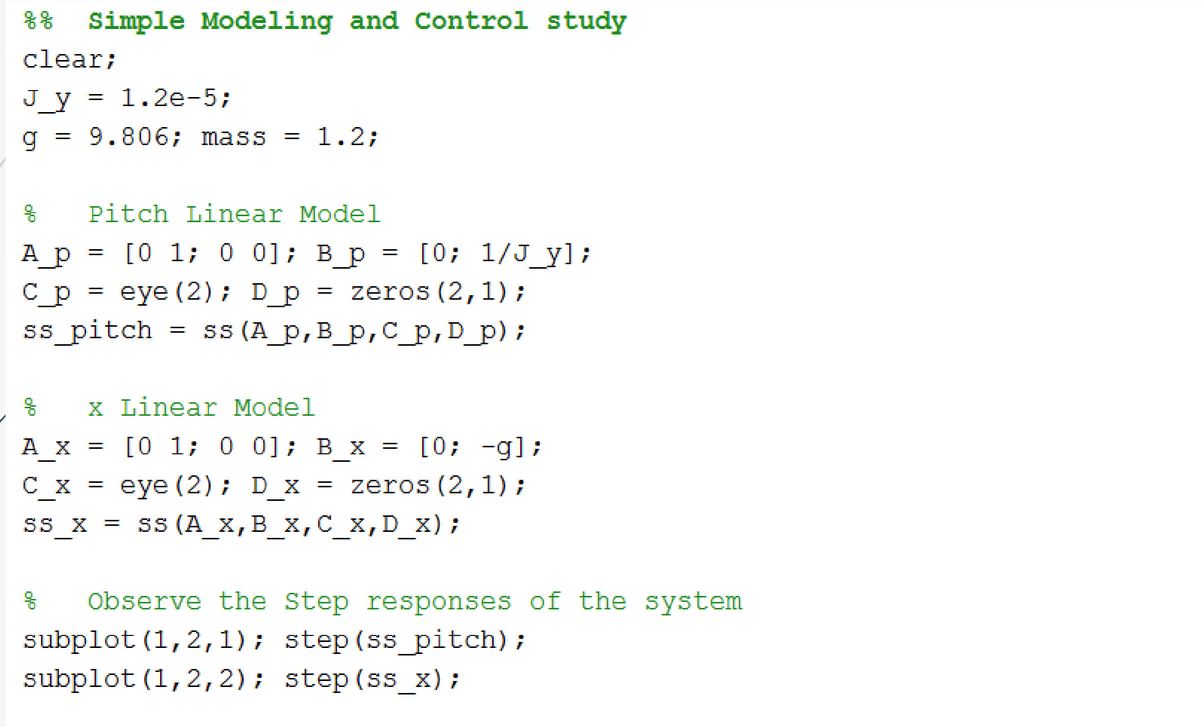
MATLAB Design Example

- Recall:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/J_y \end{bmatrix} M_y$$

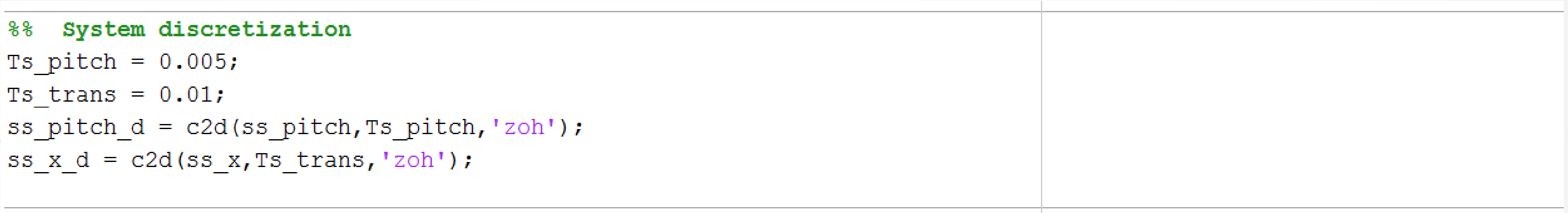
$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \theta$$

MATLAB Design Example



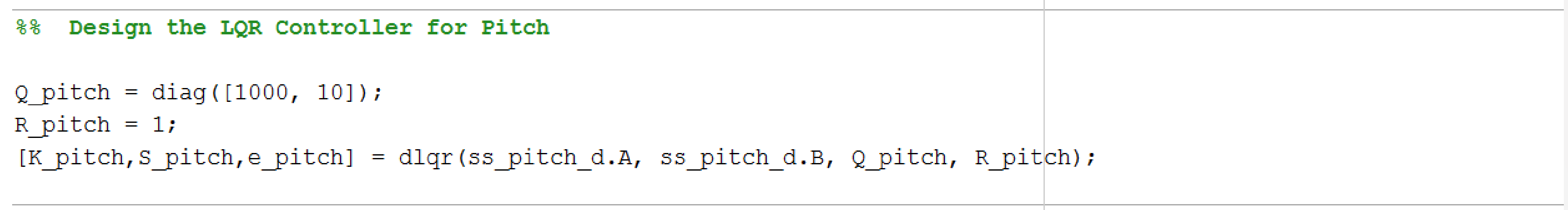
```
%%  Simple Modeling and Control study
clear;
J_y = 1.2e-5;
g = 9.806; mass = 1.2;

%   Pitch Linear Model
A_p = [0 1; 0 0]; B_p = [0; 1/J_y];
C_p = eye(2); D_p = zeros(2,1);
ss_pitch = ss(A_p,B_p,C_p,D_p);

%   x Linear Model
A_x = [0 1; 0 0]; B_x = [0; -g];
C_x = eye(2); D_x = zeros(2,1);
ss_x = ss(A_x,B_x,C_x,D_x);

%   Observe the Step responses of the system
subplot(1,2,1); step(ss_pitch);
subplot(1,2,2); step(ss_x);
```

MATLAB Design Example



```
%%  System discretization
Ts_pitch = 0.005;
Ts_trans = 0.01;
ss_pitch_d = c2d(ss_pitch,Ts_pitch,'zoh');
ss_x_d = c2d(ss_x,Ts_trans,'zoh');
```

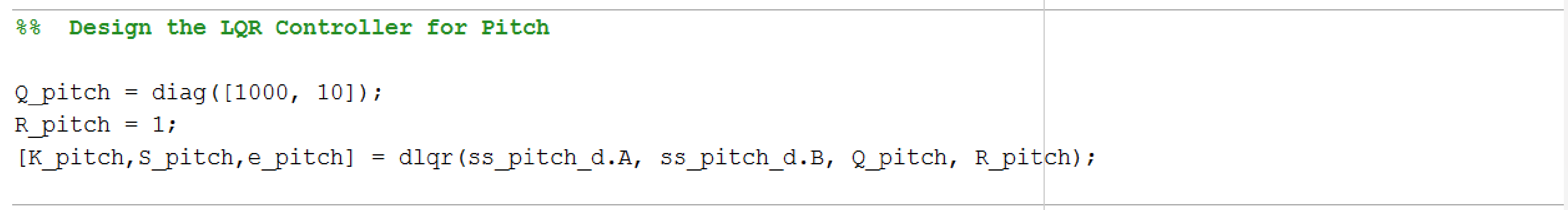
MATLAB Design Example



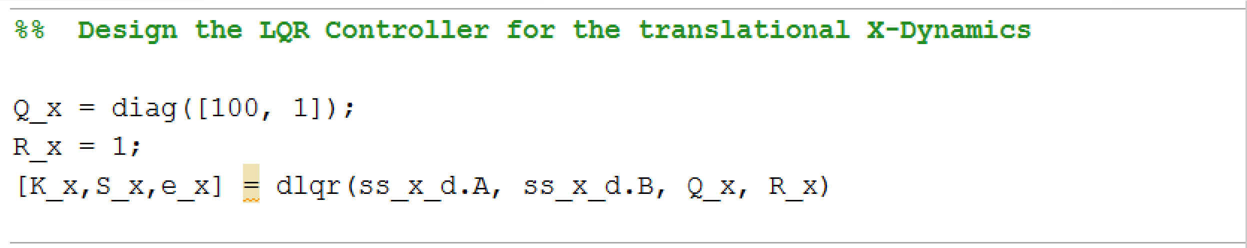
```
%%  Design the LQR Controller for Pitch

Q_pitch = diag([1000, 10]);
R_pitch = 1;
[K_pitch,S_pitch,e_pitch] = dlqr(ss_pitch_d.A, ss_pitch_d.B, Q_pitch, R_pitch);
```

```
[K,S,e] = dlqr(A,B,Q,R,N)
```

MATLAB Design Example

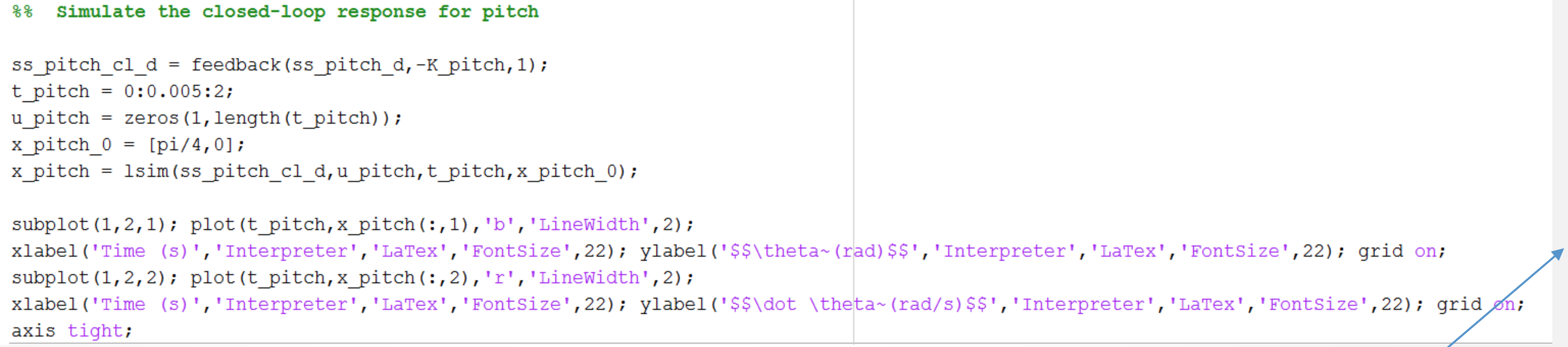


```
%%  Design the LQR Controller for Pitch

Q_pitch = diag([1000, 10]);
R_pitch = 1;
[K_pitch,S_pitch,e_pitch] = dlqr(ss_pitch_d.A, ss_pitch_d.B, Q_pitch, R_pitch);
```



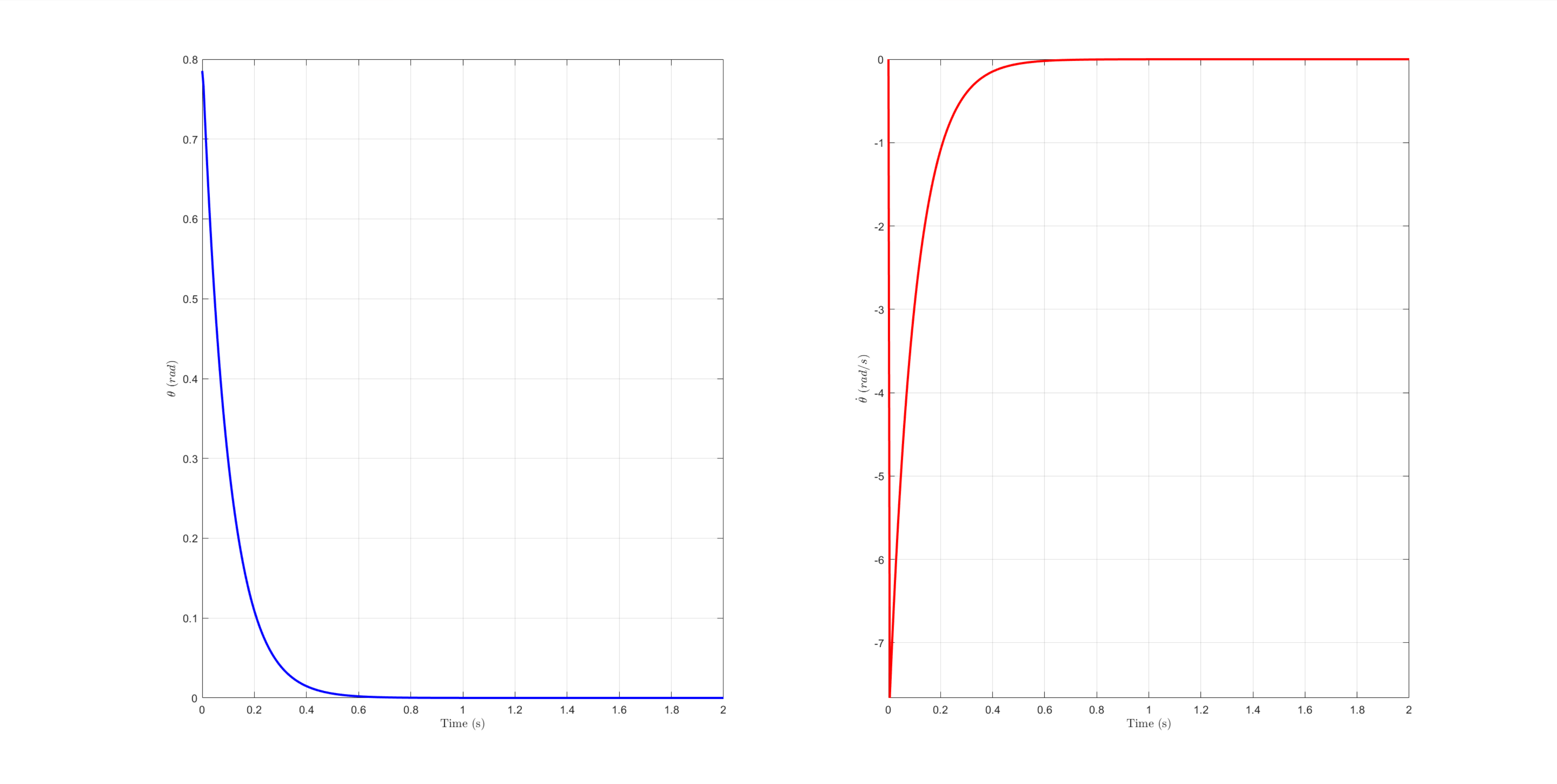
```
%%  Design the LQR Controller for the translational X-Dynamics

Q_x = diag([100, 1]);
R_x = 1;
[K_x,S_x,e_x] = dlqr(ss_x_d.A, ss_x_d.B, Q_x, R_x)
```

MATLAB Design Example

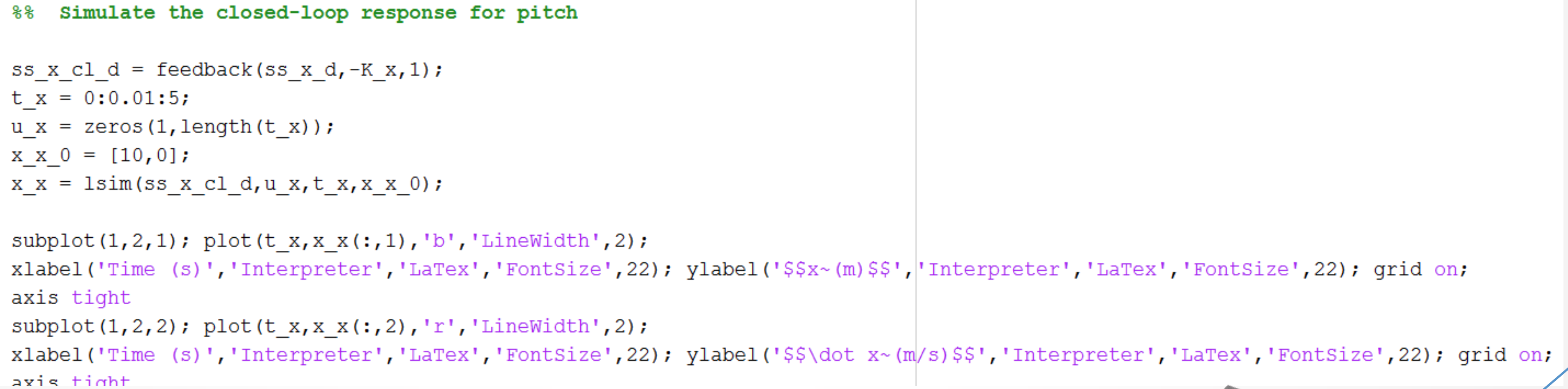


```
%%  Simulate the closed-loop response for pitch

ss_pitch_cl_d = feedback(ss_pitch_d,-K_pitch,1);
t_pitch = 0:0.005:2;
u_pitch = zeros(1,length(t_pitch));
x_pitch_0 = [pi/4,0];
x_pitch = lsim(ss_pitch_cl_d,u_pitch,t_pitch,x_pitch_0);

subplot(1,2,1); plot(t_pitch,x_pitch(:,1),'b','LineWidth',2);
xlabel('Time (s)','Interpreter','LaTex','FontSize',22); ylabel('$$\theta~(rad)$$','Interpreter','LaTex','FontSize',22); grid on;
subplot(1,2,2); plot(t_pitch,x_pitch(:,2),'r','LineWidth',2);
xlabel('Time (s)','Interpreter','LaTex','FontSize',22); ylabel('$$\dot \theta~(rad/s)$$','Interpreter','LaTex','FontSize',22); grid on;
axis tight;
```

MATLAB Design Example

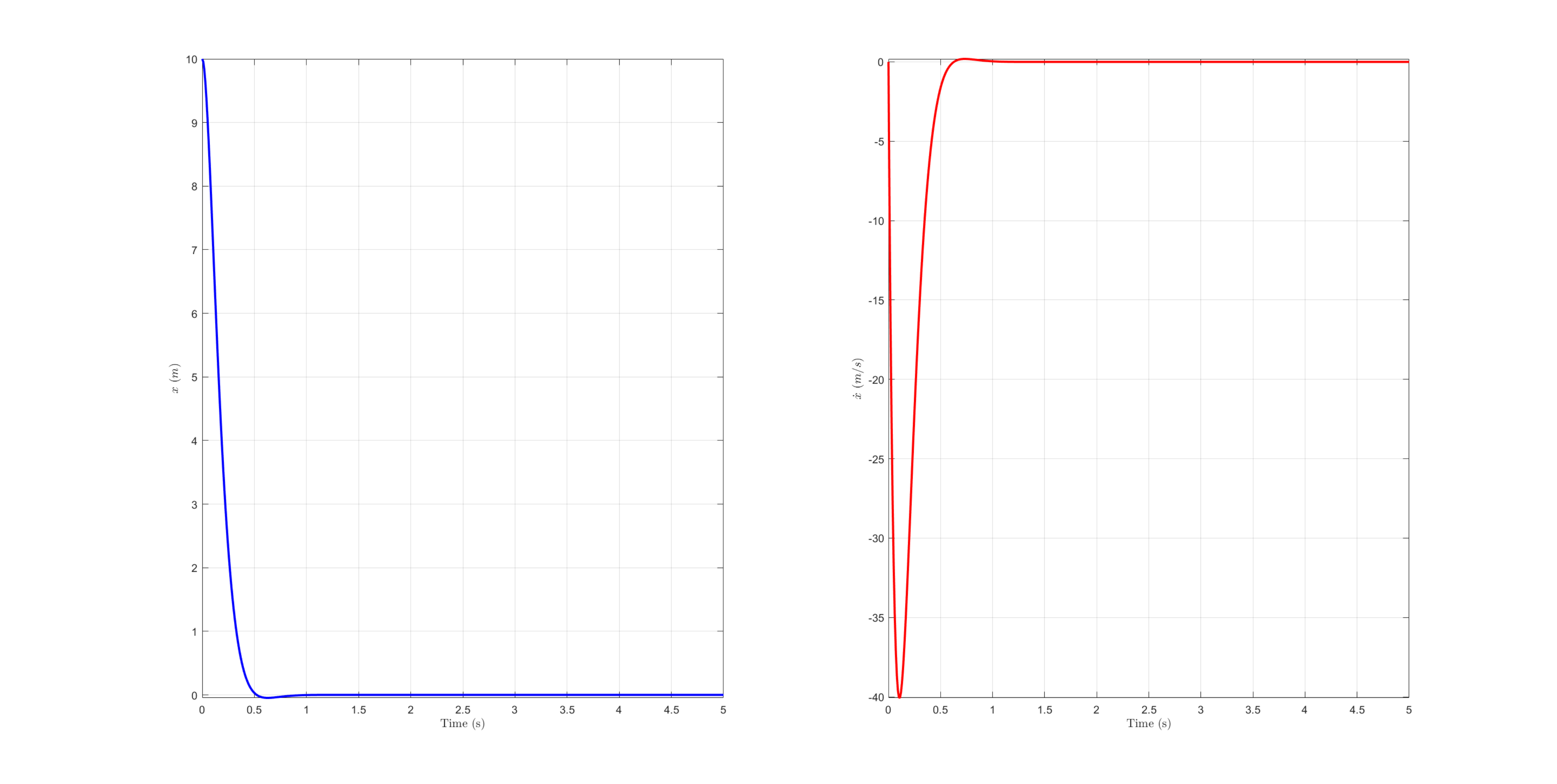


MATLAB Design Example



```
%%  Simulate the closed-loop response for pitch

ss_x_cl_d = feedback(ss_x_d,-K_x,1);
t_x = 0:0.01:5;
u_x = zeros(1,length(t_x));
x_x_0 = [10,0];
x_x = lsim(ss_x_cl_d,u_x,t_x,x_x_0);

subplot(1,2,1); plot(t_x,x_x(:,1),'b','LineWidth',2);
xlabel('Time (s)','Interpreter','LaTex','FontSize',22); ylabel('$$x~(m)$$','Interpreter','LaTex','FontSize',22); grid on;
axis tight
subplot(1,2,2); plot(t_x,x_x(:,2),'r','LineWidth',2);
xlabel('Time (s)','Interpreter','LaTex','FontSize',22); ylabel('$$\dot x~(m/s)$$','Interpreter','LaTex','FontSize',22); grid on;
axis tight
```

MATLAB Design Example



MATLAB Aircraft Pitch Example

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} u_{elev}$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

For a step reference of 0.2 radians, the design criteria are the following.

1. Overshoot less than 10%
2. Rise time less than 2 seconds
3. Settling time less than 10 seconds
4. Steady-state error less than 2%

MATLAB Aircraft Pitch Example

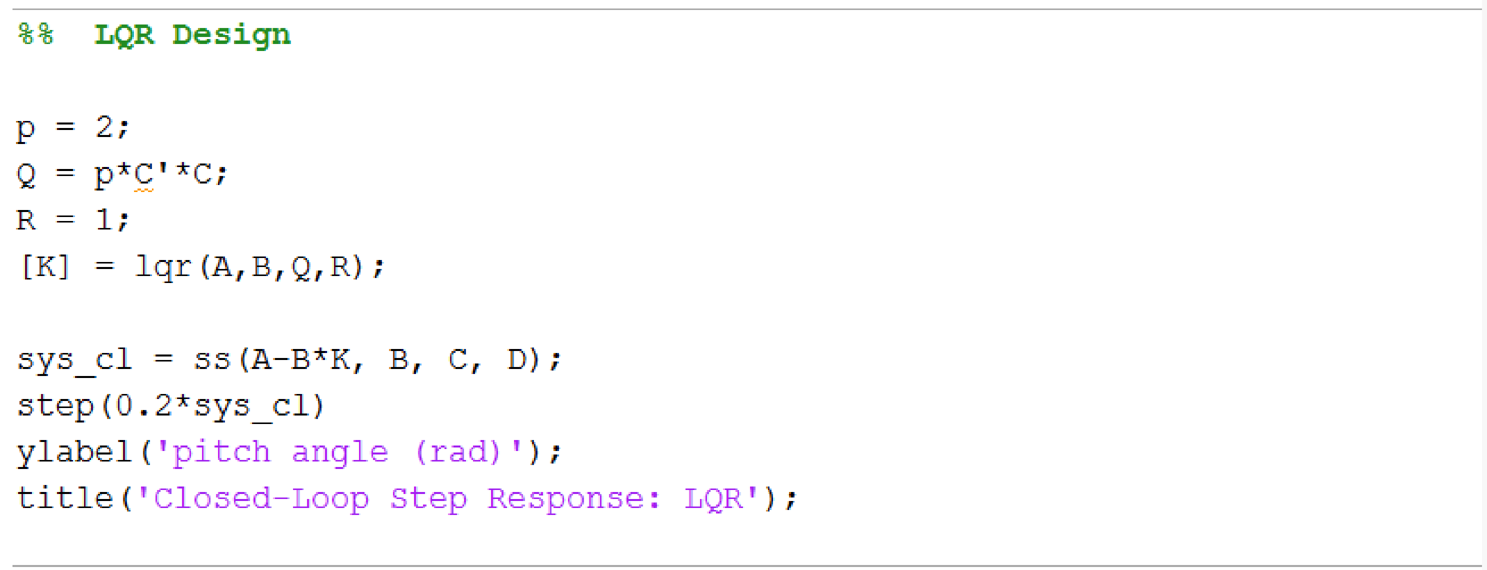
$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} u_{elev}$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

For a step reference of 0.2 radians, the design criteria are the following.

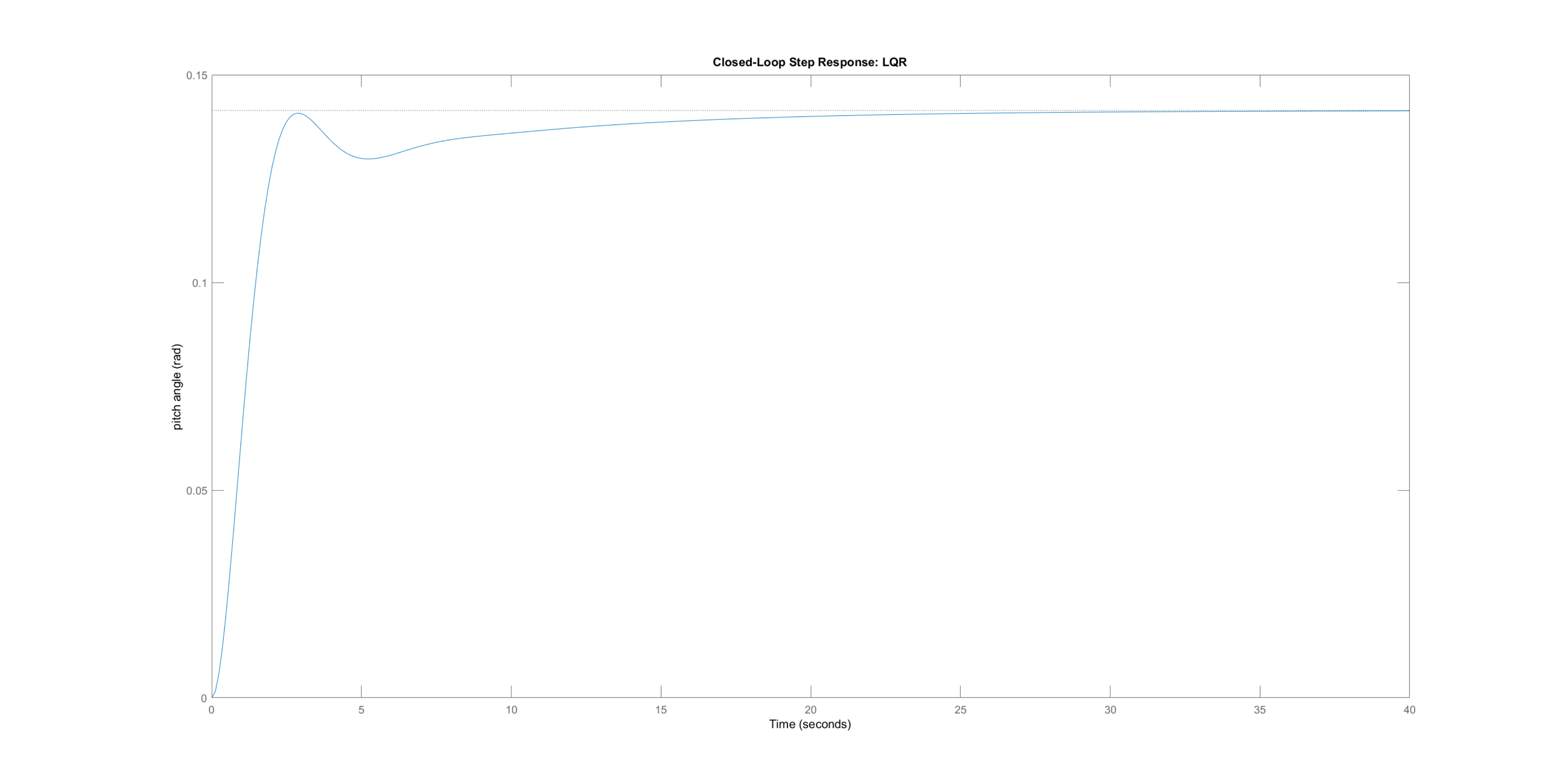
1. Overshoot less than 10%
2. Rise time less than 2 seconds
3. Settling time less than 10 seconds
4. Steady-state error less than 2%

MATLAB Aircraft Pitch Example

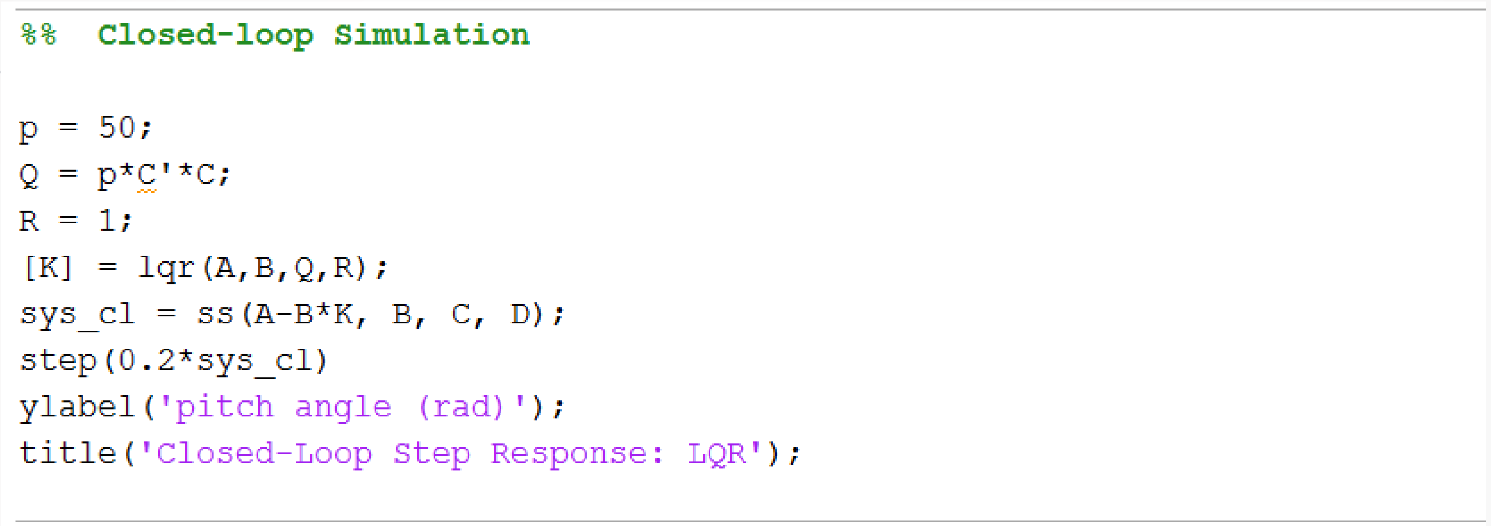


```
%%  LQR Design

p = 2;
Q = p*C'*C;
R = 1;
[K] = lqr(A,B,Q,R);

sys_cl = ss(A-B*K, B, C, D);
step(0.2*sys_cl)
ylabel('pitch angle (rad)');
title('Closed-Loop Step Response: LQR');
```

MATLAB Aircraft Pitch Example

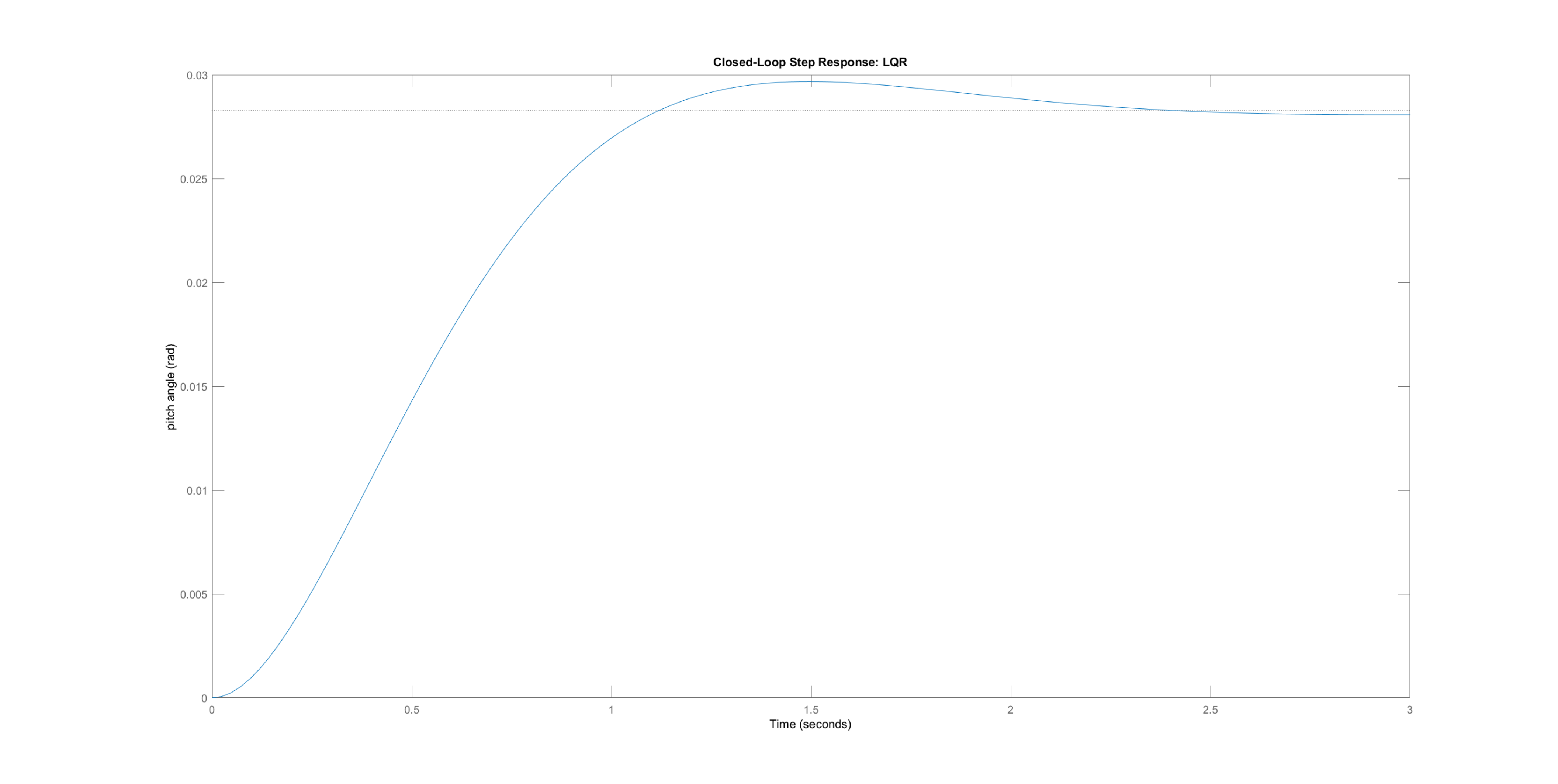


MATLAB Aircraft Pitch Example

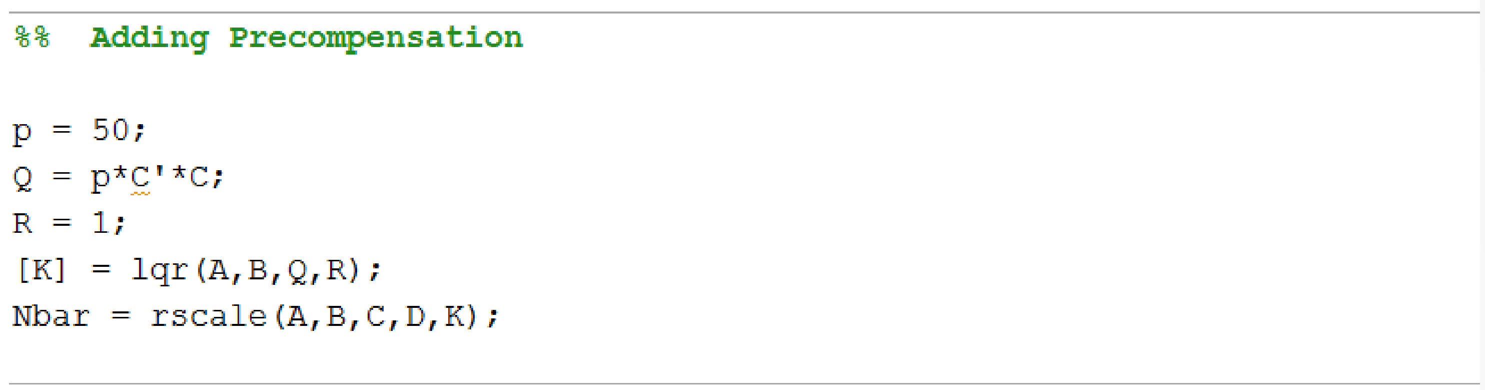


```
%%  Closed-loop Simulation

p = 50;
Q = p*C'*C;
R = 1;
[K] = lqr(A,B,Q,R);
sys_cl = ss(A-B*K, B, C, D);
step(0.2*sys_cl)
ylabel('pitch angle (rad)');
title('Closed-Loop Step Response: LQR');
```

MATLAB Aircraft Pitch Example

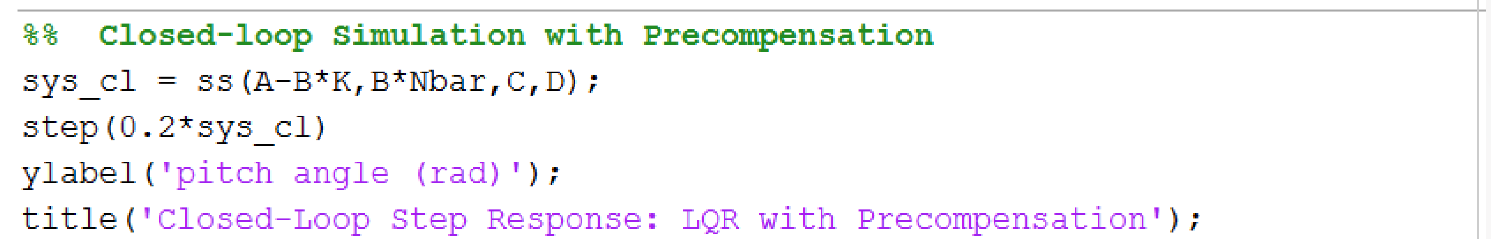


MATLAB Aircraft Pitch Example



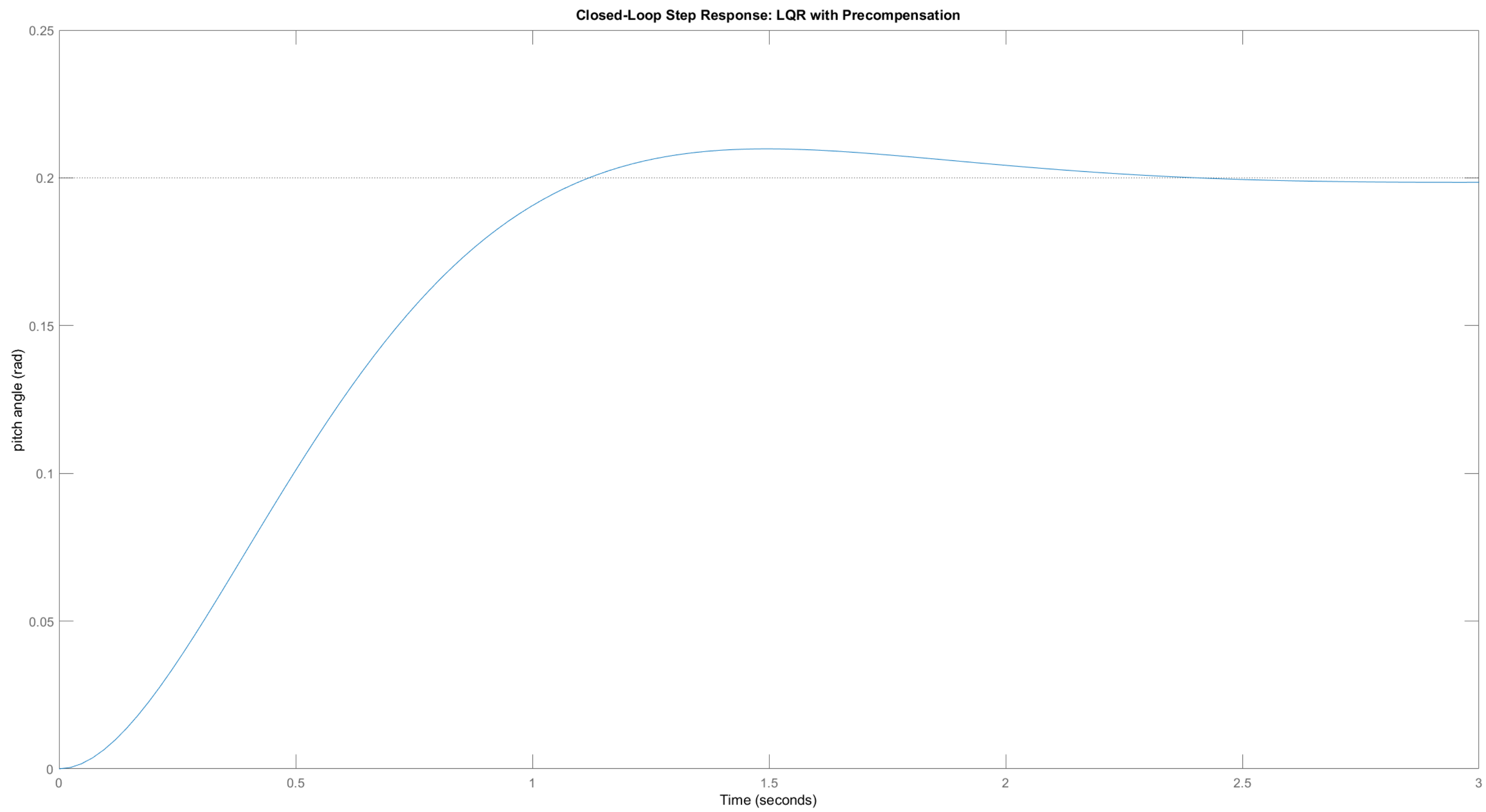
```
%%  Adding Precompensation

p = 50;
Q = p*C'*C;
R = 1;
[K] = lqr(A,B,Q,R);
Nbar = rscale(A,B,C,D,K);
```

MATLAB Aircraft Pitch Example



```
%%  Closed-loop Simulation with Precompensation
sys_cl = ss(A-B*K,B*Nbar,C,D);
step(0.2*sys_cl)
ylabel('pitch angle (rad)');
title('Closed-Loop Step Response: LQR with Precompensation');
```

MATLAB Aircraft Pitch Example



Code Examples and Tasks

- https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/control-systems/gain-scheduled-three-loop-aircraft-autopilot
- https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/control-systems/lqr
- https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/control-systems/pid-cruise-control
- https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/control-systems/pid

How does this apply to my project?

- To control the state of the robot and make it follow the desired trajectory.

Find out more

- http://www.kostasalexis.com/pid-control.html
- http://www.kostasalexis.com/lqr-control.html
- http://www.kostasalexis.com/linear-model-predictive-control.html
- http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace
- http://www.kostasalexis.com/literature-and-links.html
-

Thank you!

Please ask your question!