



# Drones Demystified!


K. Alexis, C. Papachristos, Autonomous Robots Lab, University of Nevada, Reno

A. Tzes, Autonomous Robots & Intelligent Systems Lab, NYU Abu Dhabi

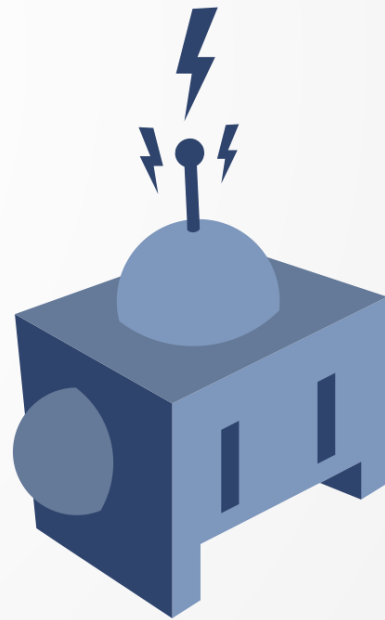
A decorative graphic on the left side of the slide, featuring a blue arrow pointing right and several thin, curved lines in shades of blue and grey.

# Drones Demystified!

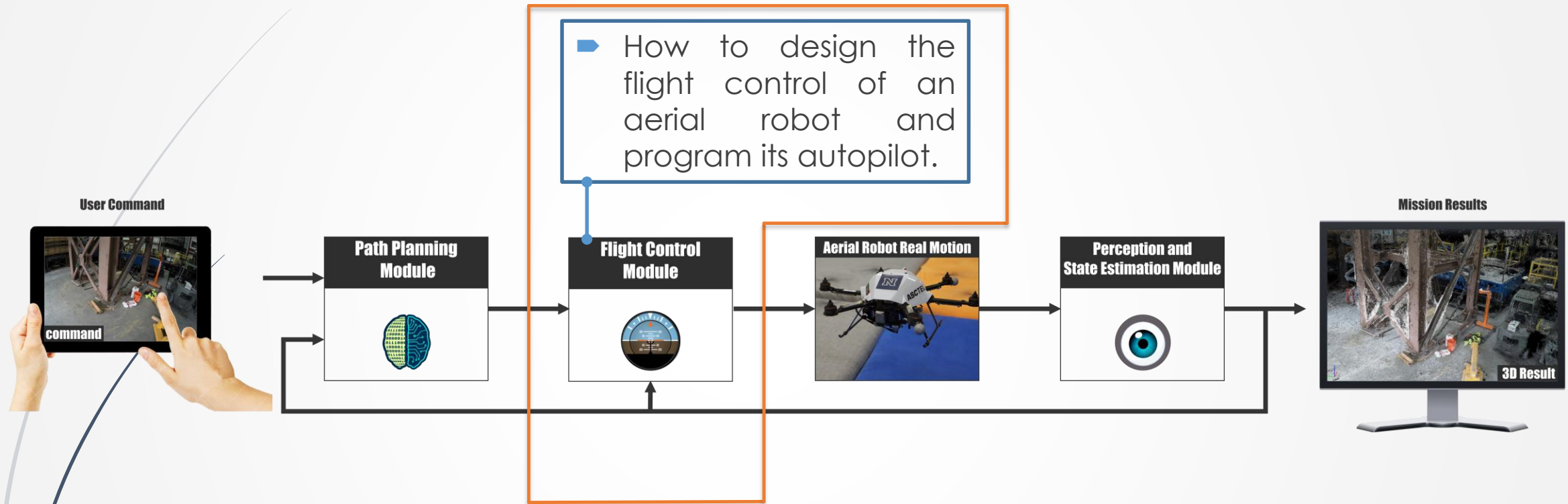
## Topic: Flight Controls Introduction



How do I  
control my  
motion?

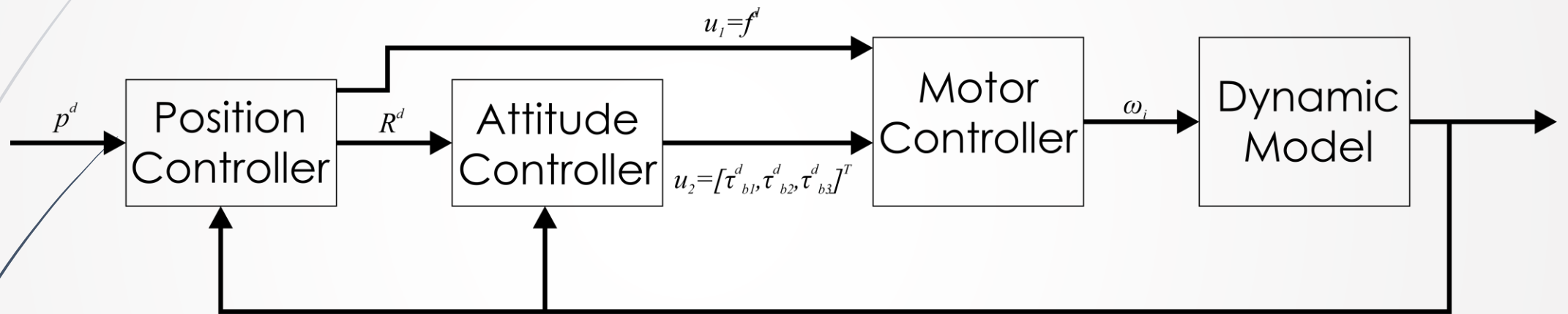


# The Aerial Robot Loop



Section 3 of our course

# Control System Block Diagram



➡ Simplified loop

# Controlling a Multicopter 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 Multicopter 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 Multicopter along the x-axis

- Simplified linear dynamics

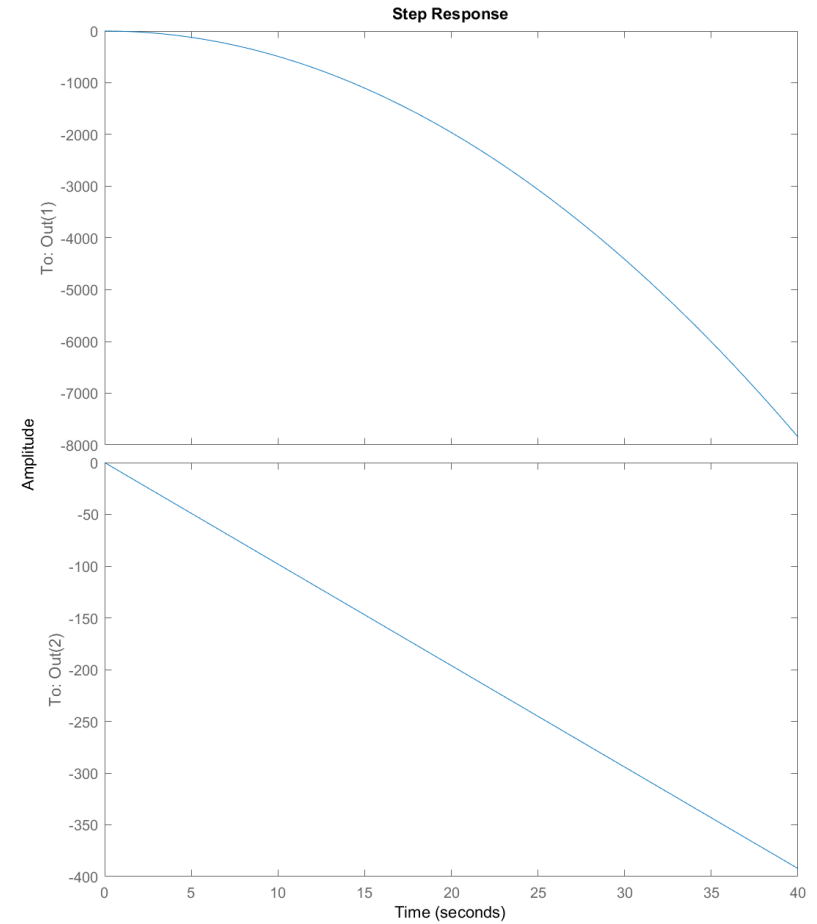
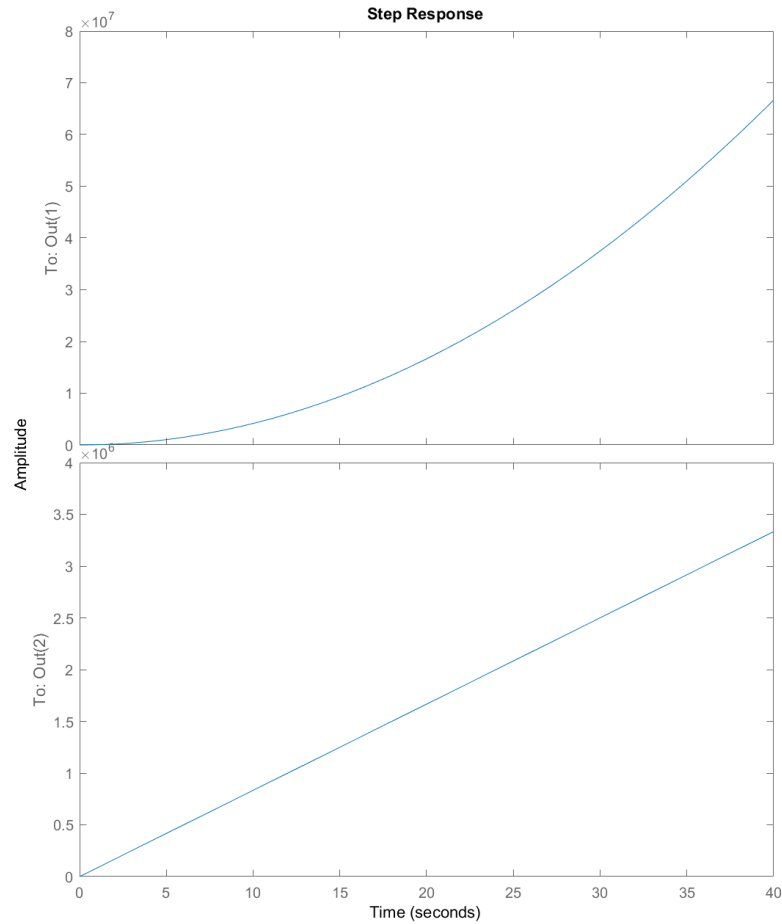
$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/J_y \end{bmatrix} M_y$$

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

How does this system behave?



# Controlling a Multicopter along the x-axis



# Controlling a Multicopter along the x-axis

- Simplified linear dynamics

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

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

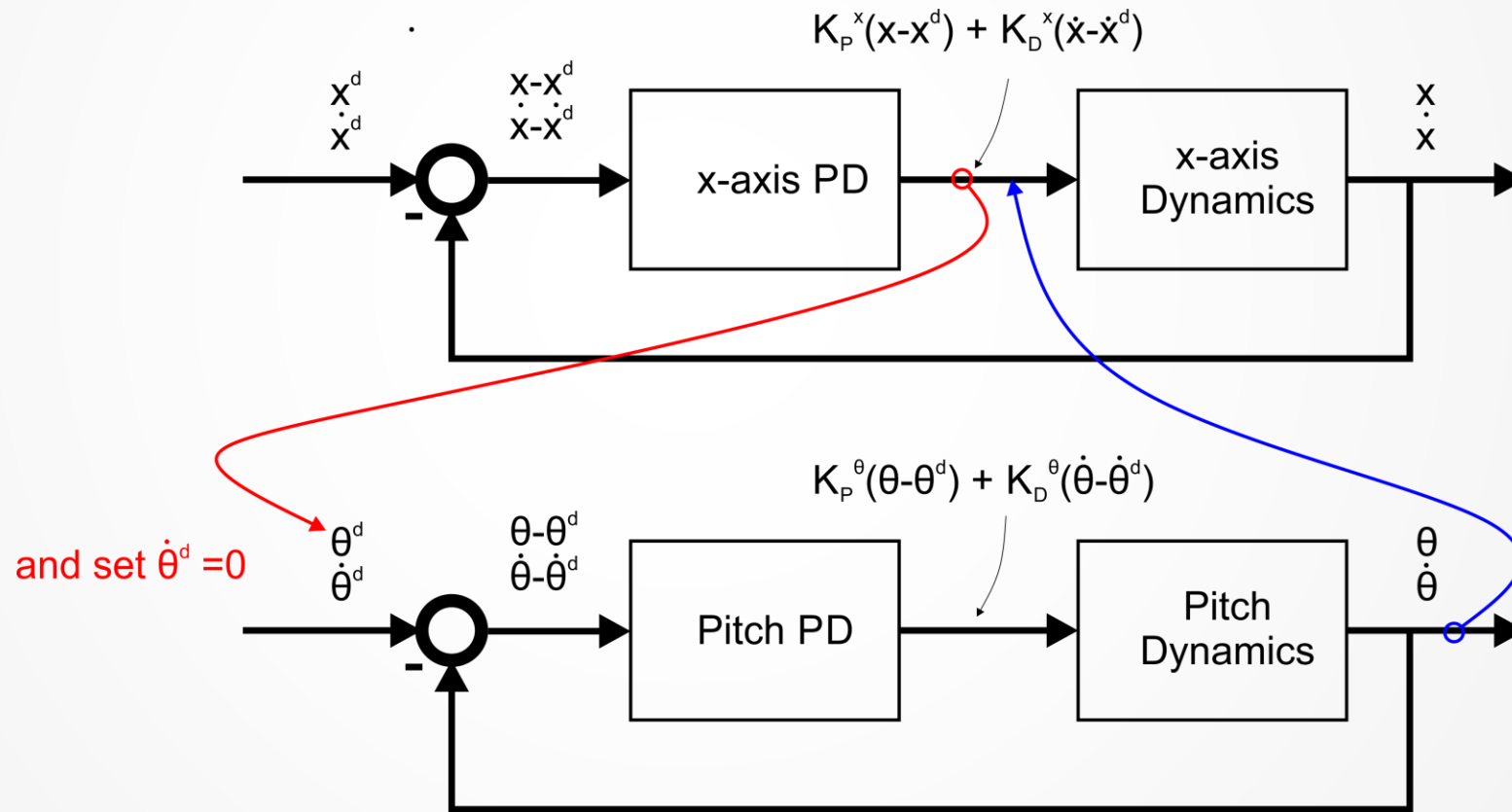
**How to control this system?**

Most common way: use of PD controllers



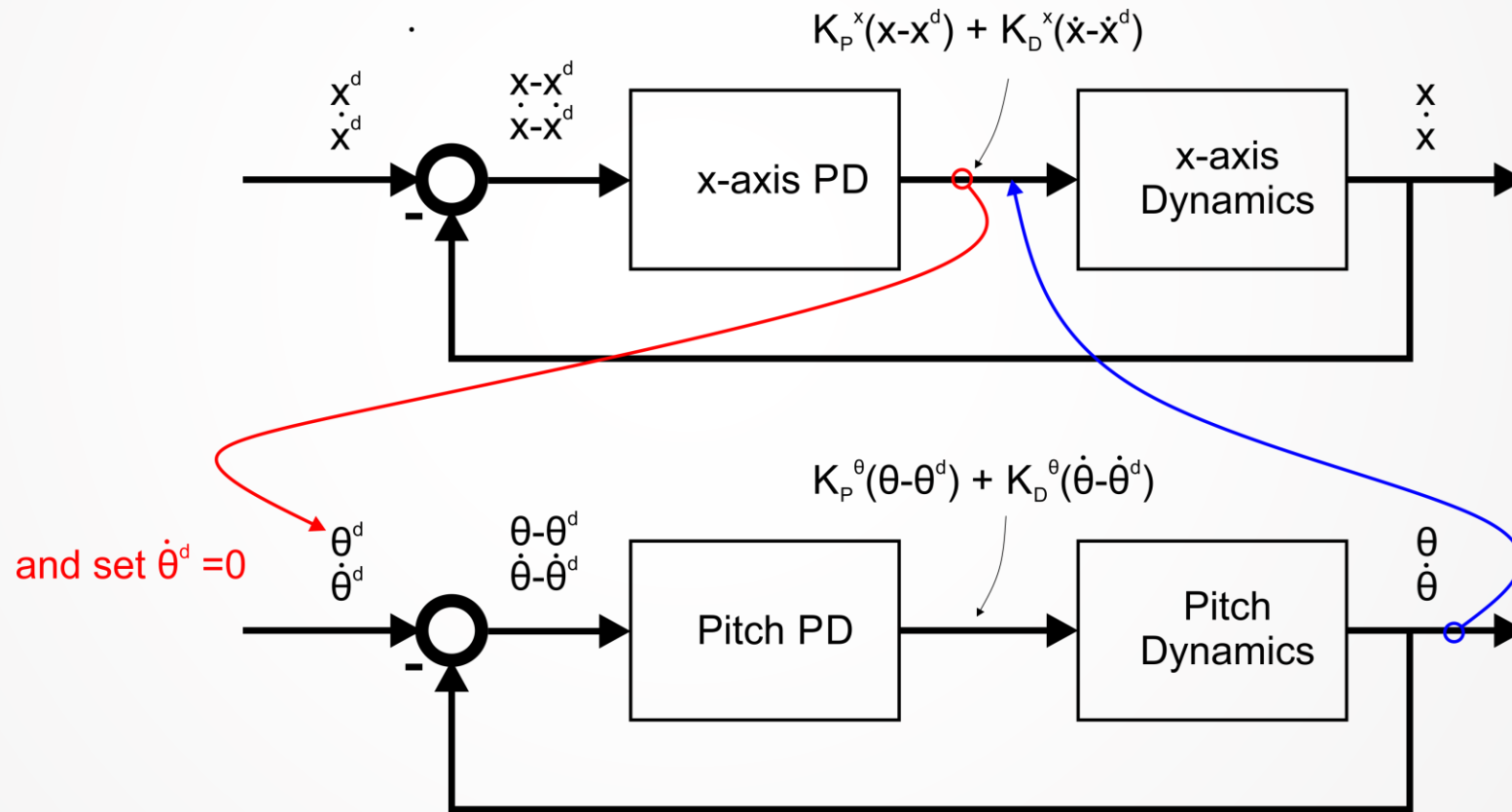
# Controlling a Multirotor along the x-axis

## Decoupled Control Structure



# Controlling a Multicopter along the x-axis

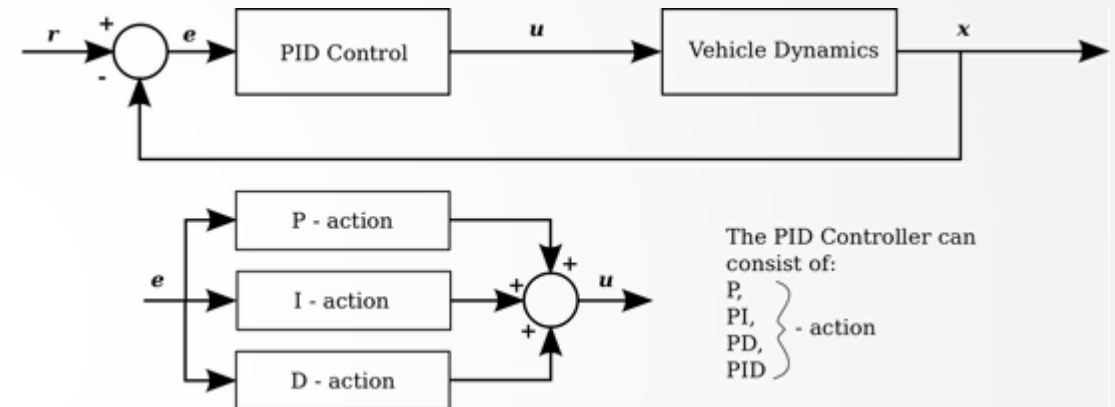
## Decoupled Control Structure



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.
- Its 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 Multicopter 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 Multicopter 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;

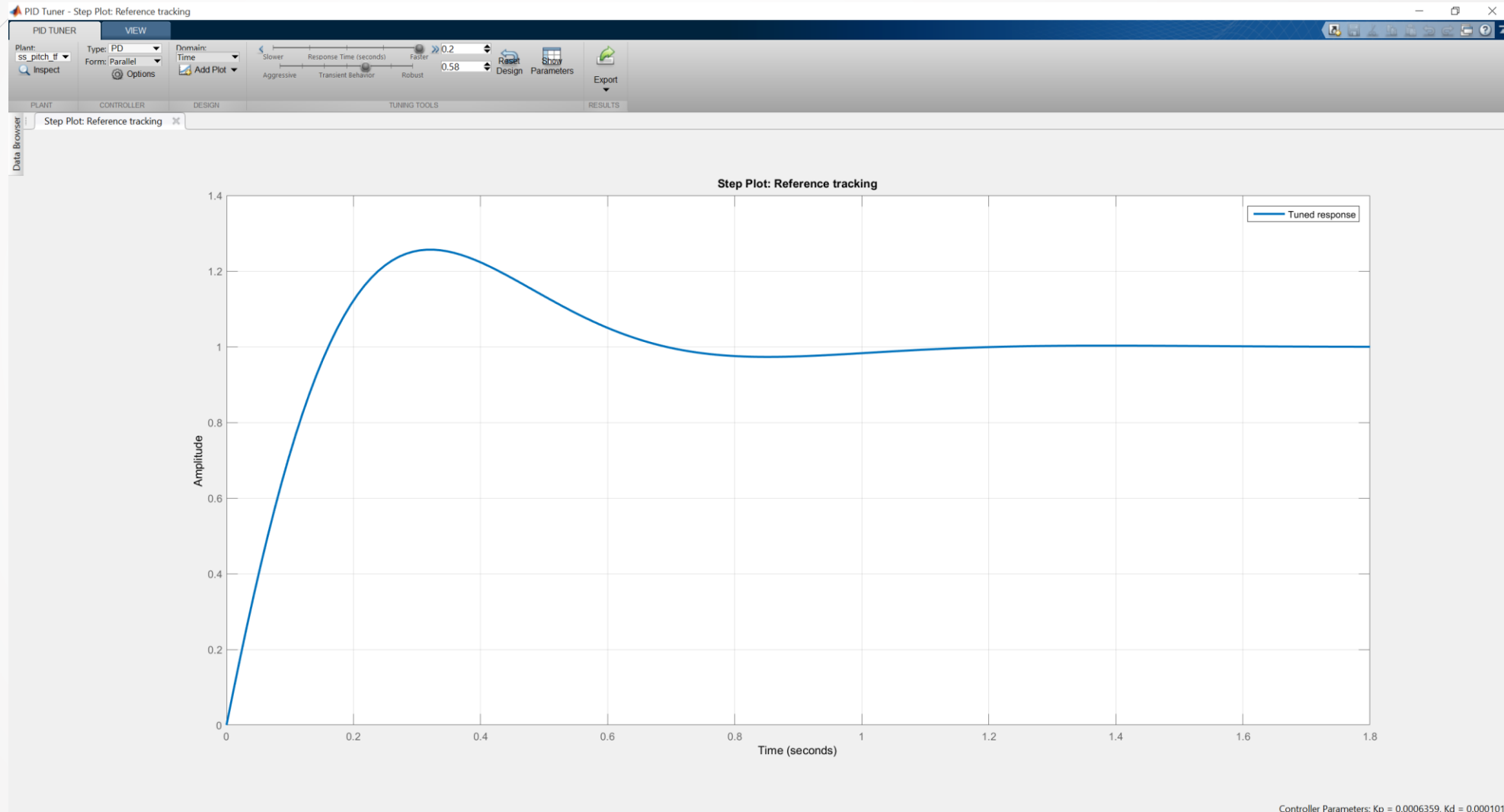
K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0 K_D_pitch];

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



# Controlling a Multicopter 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;

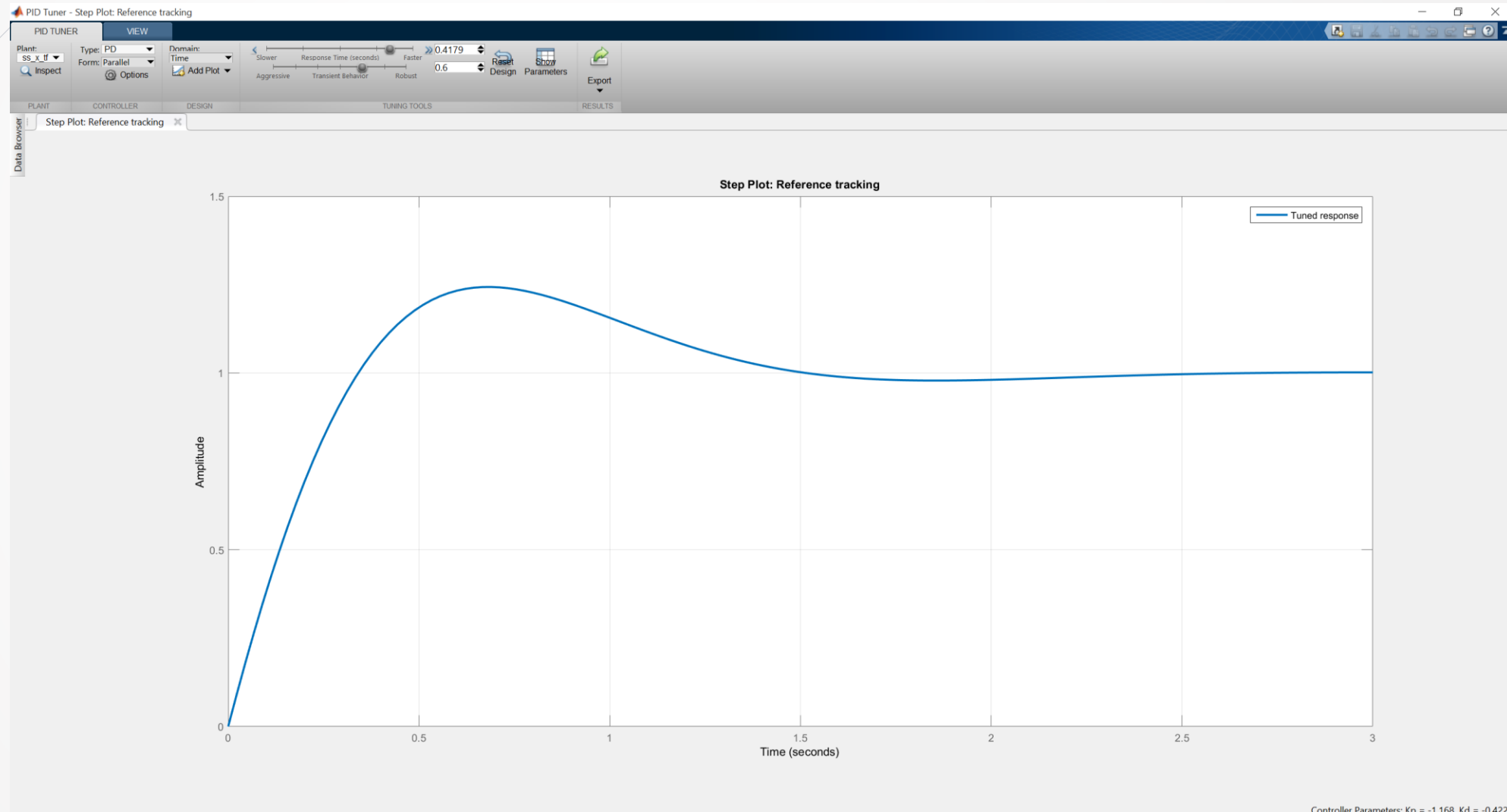
K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0 K_D_pitch];

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



Controller Parameters:  $K_p = -1.168$ ,  $K_d = -0.4227$

# Controlling a Multicopter 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;

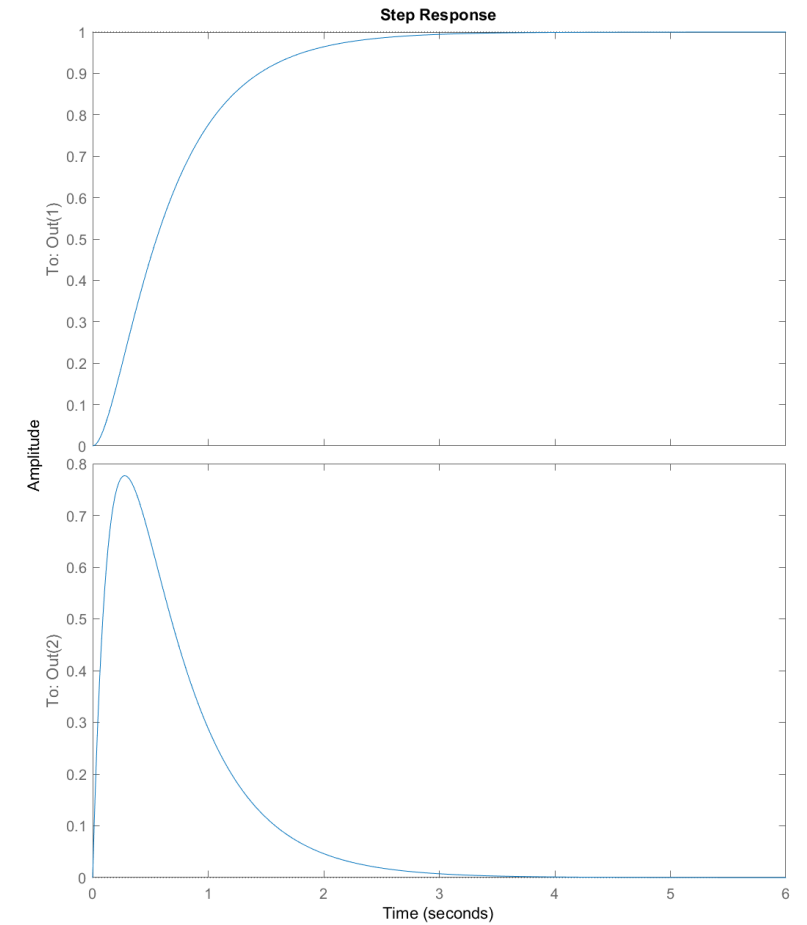
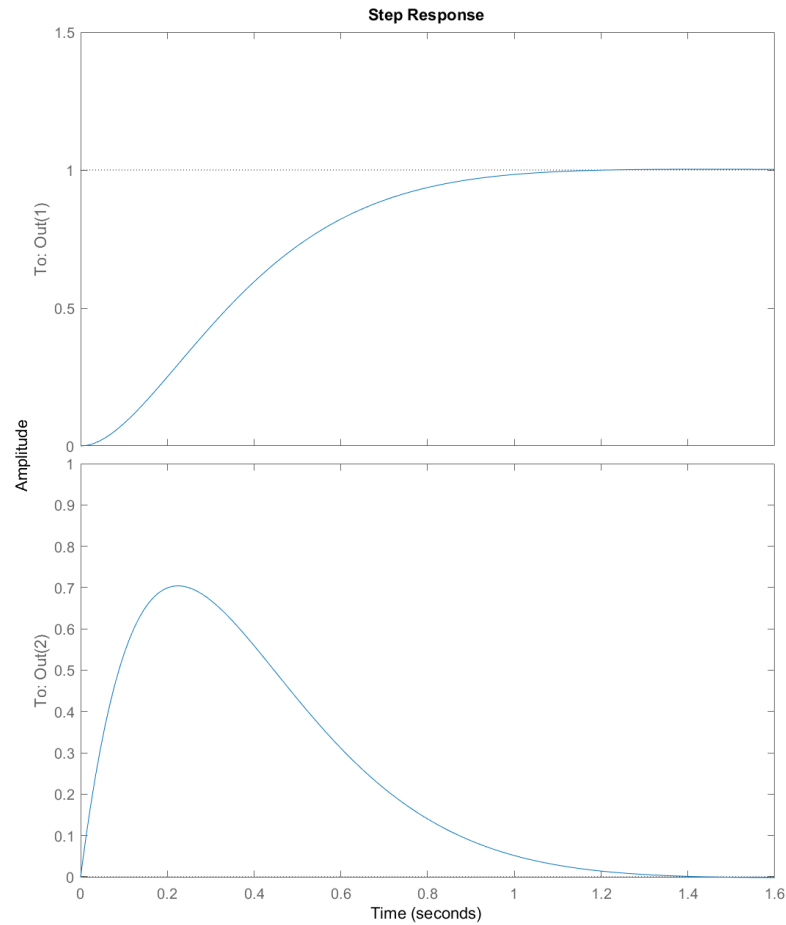
K_P_pitch = 25.8e-5; K_D_pitch = 9.82e-5;
PD_PITCH_GAINS = [K_P_pitch 0; 0 K_D_pitch];

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 Multicopter along the x-axis

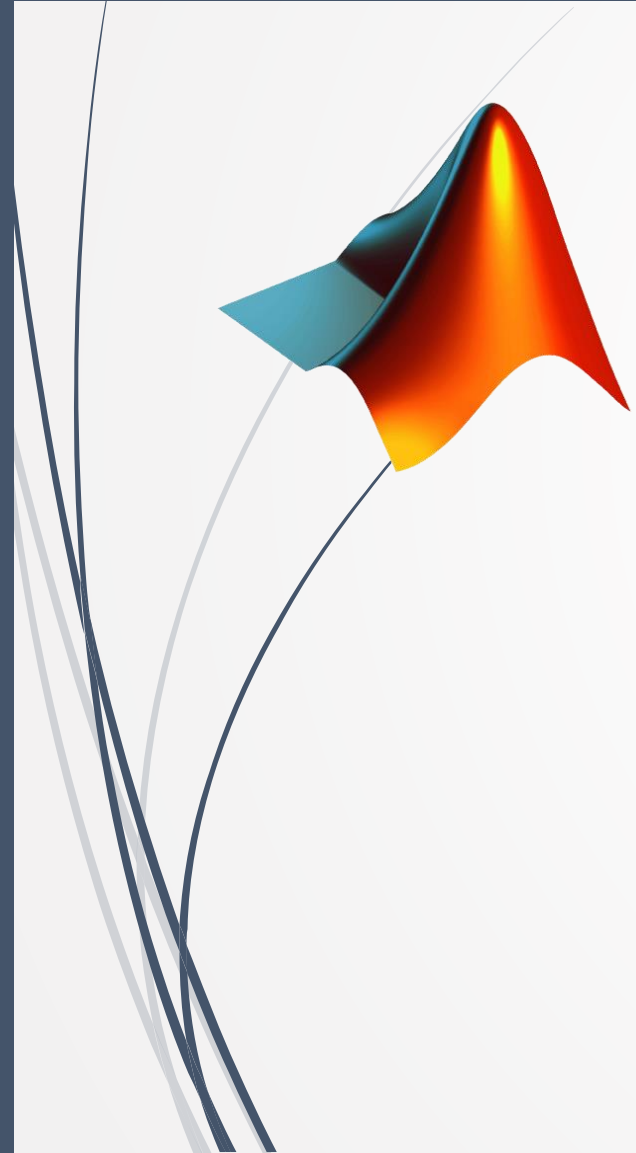




# Real-life Limitations

- ▶ 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.

# Code Examples and Tasks



- [https://github.com/unr-arl/drones\\_demystified/tree/master/matlab/control-systems/gain-scheduled-three-loop-aircraft-autopilot](https://github.com/unr-arl/drones_demystified/tree/master/matlab/control-systems/gain-scheduled-three-loop-aircraft-autopilot)
- [https://github.com/unr-arl/drones\\_demystified/tree/master/matlab/control-systems/lqr](https://github.com/unr-arl/drones_demystified/tree/master/matlab/control-systems/lqr)
- [https://github.com/unr-arl/drones\\_demystified/tree/master/matlab/control-systems/pid-cruise-control](https://github.com/unr-arl/drones_demystified/tree/master/matlab/control-systems/pid-cruise-control)
- [https://github.com/unr-arl/drones\\_demystified/tree/master/matlab/control-systems/pid](https://github.com/unr-arl/drones_demystified/tree/master/matlab/control-systems/pid)

# Find out more

- <http://www.autonomousrobotslab.com/pid-control.html>
- <http://www.autonomousrobotslab.com/lqr-control.html>
- <http://www.autonomousrobotslab.com/linear-model-predictive-control.html>
- <http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace>
- <http://www.autonomousrobotslab.com/literature-and-links1.html>

A black and white photograph of a drone flying in front of a construction site. The drone is in the foreground, slightly out of focus, with its four rotors visible. In the background, several large construction cranes are visible, also out of focus, against a bright sky. The overall scene is a construction site.

**Thank you!**

Please ask your question!