

Satellite GNC Design Workshop

MathWorks

Satellite GNC Design Workshop

- Simulate an orbit with 2 slews
 - Orbit propagation
 - Pointing logic
 - Different visualization methods
- Design an attitude controller
- Improve controller with automated tuning tools
- Test a tuned controller back in the scenario



Today's Agenda

Time	Item
11AM – 12PM	Workshop Introduction and MATLAB Online Set-Up (15 mins) Exercise 1: Simulate an Ideal Scenario (15 mins) Begin Exercise 2: Controller Design and Tuning (30 mins)
12PM – 12:30PM	Lunch
12:30PM – 1:30PM	Continue Exercise 2: Controller Design and Tuning (45 mins) Exercise 3: Simulate a Scenario Including a Tuned Controller (15 mins)

Setting Up MATLAB Online

Launch your browser (Google Chrome recommended) and follow the instructions in the readme of this GitHub repository:

<https://github.com/juliabrupt-ML/SatelliteGNCDesign>

Conference Wifi
Beaver Run Meeting 1
conferencewifi

The screenshot shows the GitHub repository page for `juliabrupt-ML/SatelliteGNCDesign`. The repository is public and has 5 commits. The commit history shows the following changes:

Commit	Message	Time
2a2dcfe	Added conical sensor	5 minutes ago
	Moved files over from gitlab	1 hour ago
	Added conical sensor	5 minutes ago
	Moved files over from gitlab	1 hour ago
	fixed ml drive tags	23 minutes ago
	Update README.md	1 hour ago
	Moved files over from gitlab	1 hour ago

The repository contains the following files and folders:

- Exercise1
- Exercise2
- Exercise3
- Utilities
- resources/project
- README.md
- SatelliteScenarioAndControls.prj

The README file is titled "Satellite GNC Design".

Exercise 1: Simulate an Ideal Scenario

Purpose of this exercise:

- Familiarize yourself with the MATLAB and Simulink Online environment
- Get to know Aerospace Toolbox and Blockset features for mission simulation
 - Orbit Propagation
 - Pointing Logic
 - Visualization

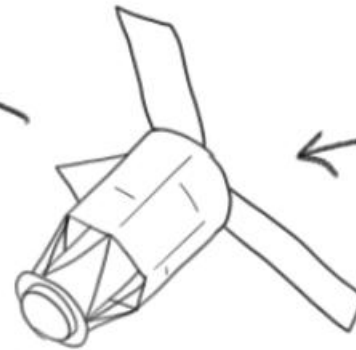


Exercise 2: Controller Design and Tuning



Do Not Point!
unless it's on purpose

Keep some
distance
-Important-



"Satellite"

Point here

"Planet"

Exercise 2: Controller Design and Tuning

- We're going to **develop a controller** that will follow an attitude profile
- We're going to use Simulink and the **Control System Tuner app**
- The controller we design is not as important as **the process** we follow
- Be thinking of how you can apply this workflow to your problems

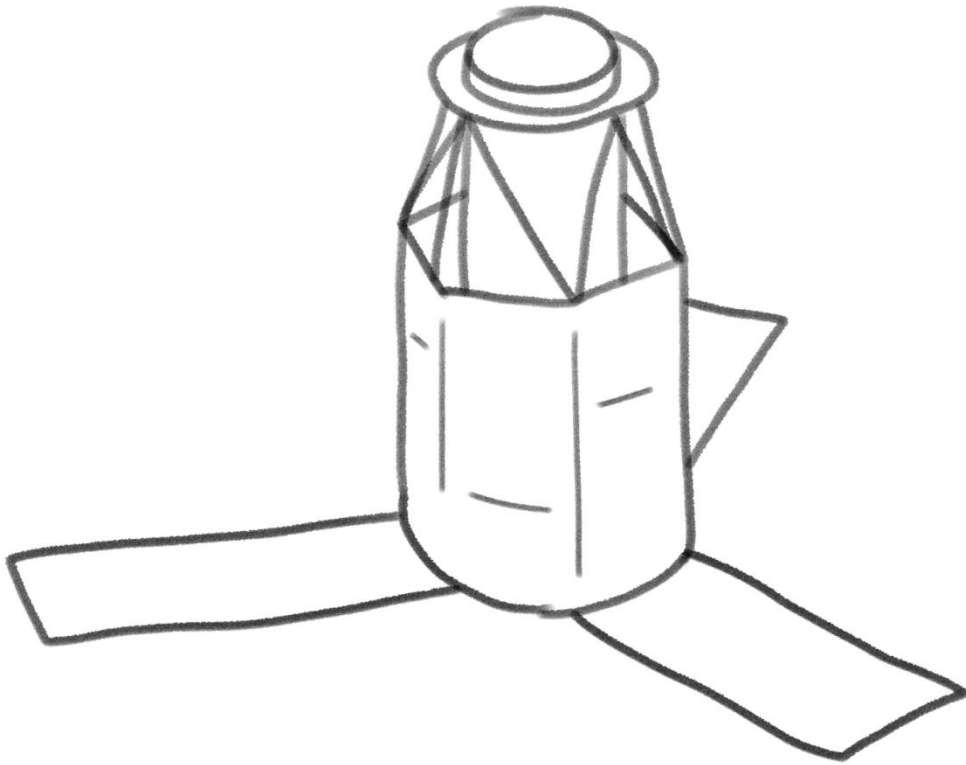


Take away

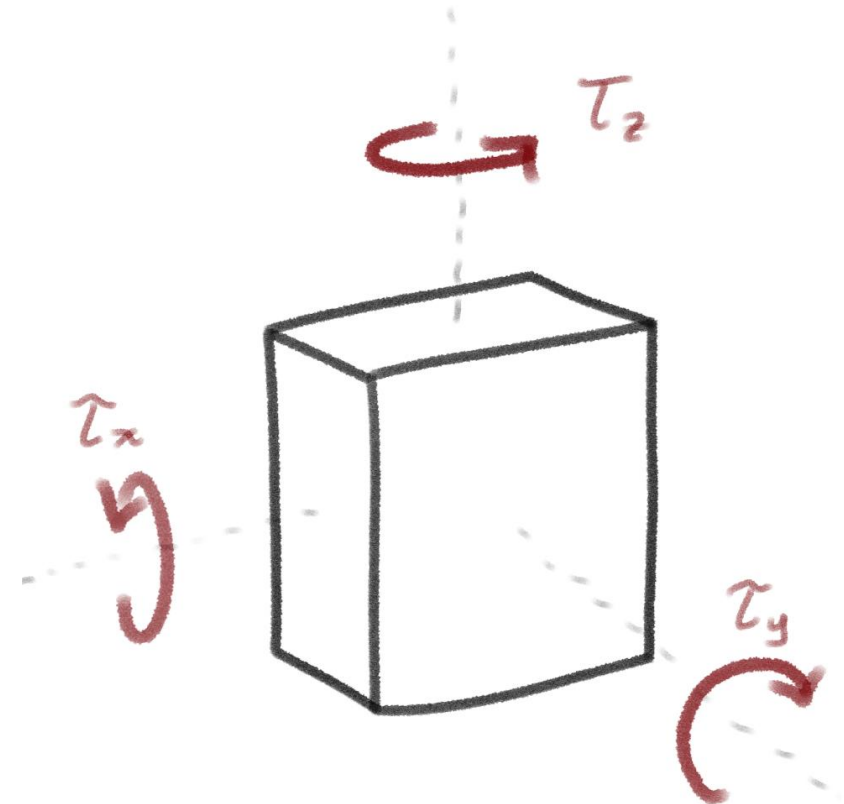
“Wow! There are some powerful tuning tools that I haven't been using!”

We're using a simplified model of the spacecraft

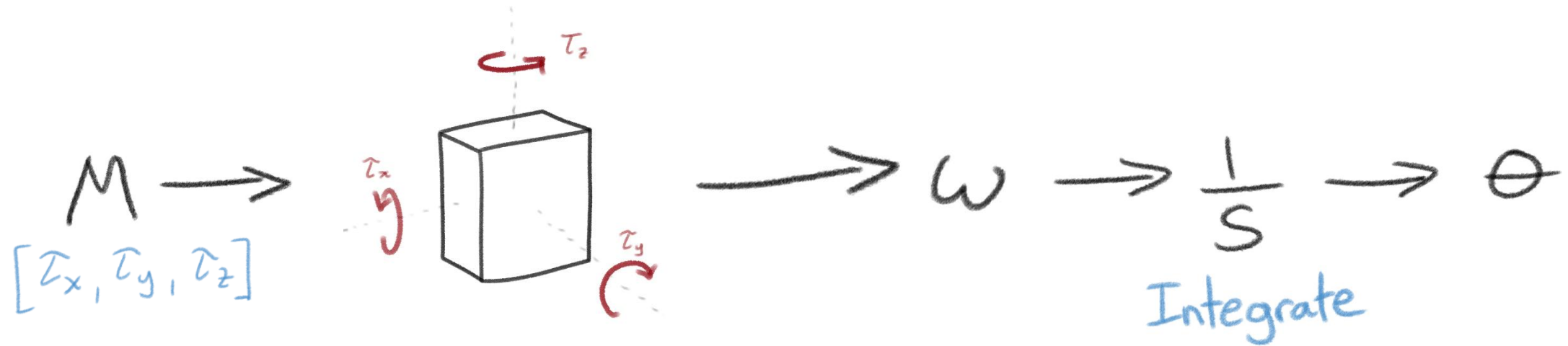
The spacecraft



The model



The governing dynamics are Euler's rigid body equations



$$M = I\dot{\omega} + \omega \times (I\omega)$$

Applied torque \uparrow \uparrow \uparrow

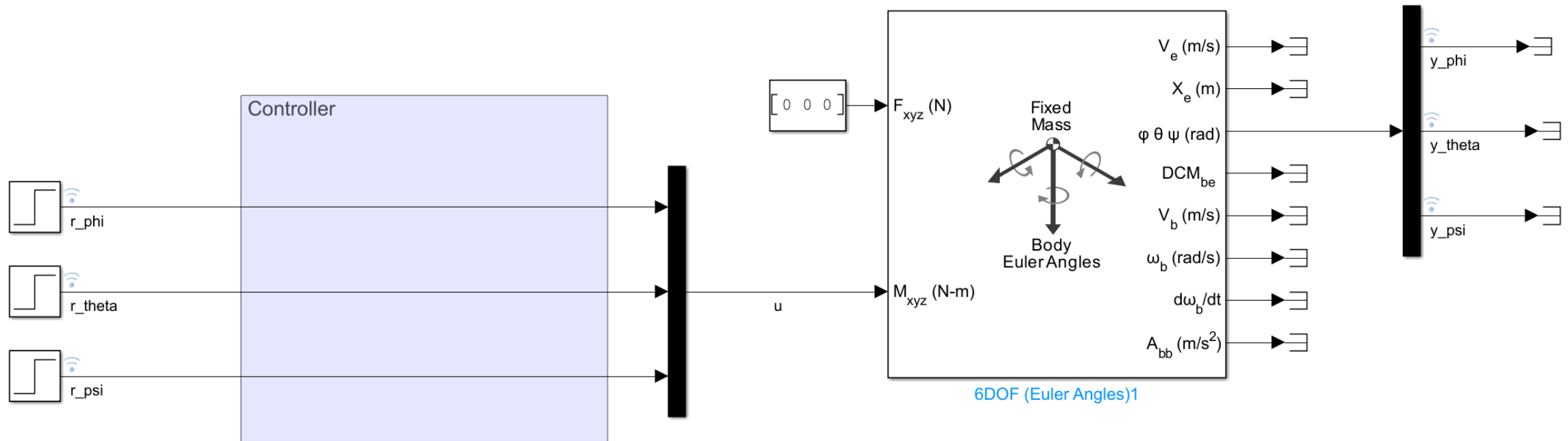
Directly affects angular acceleration

Plus this cross-coupling term



Discussion

How would you approach this problem?

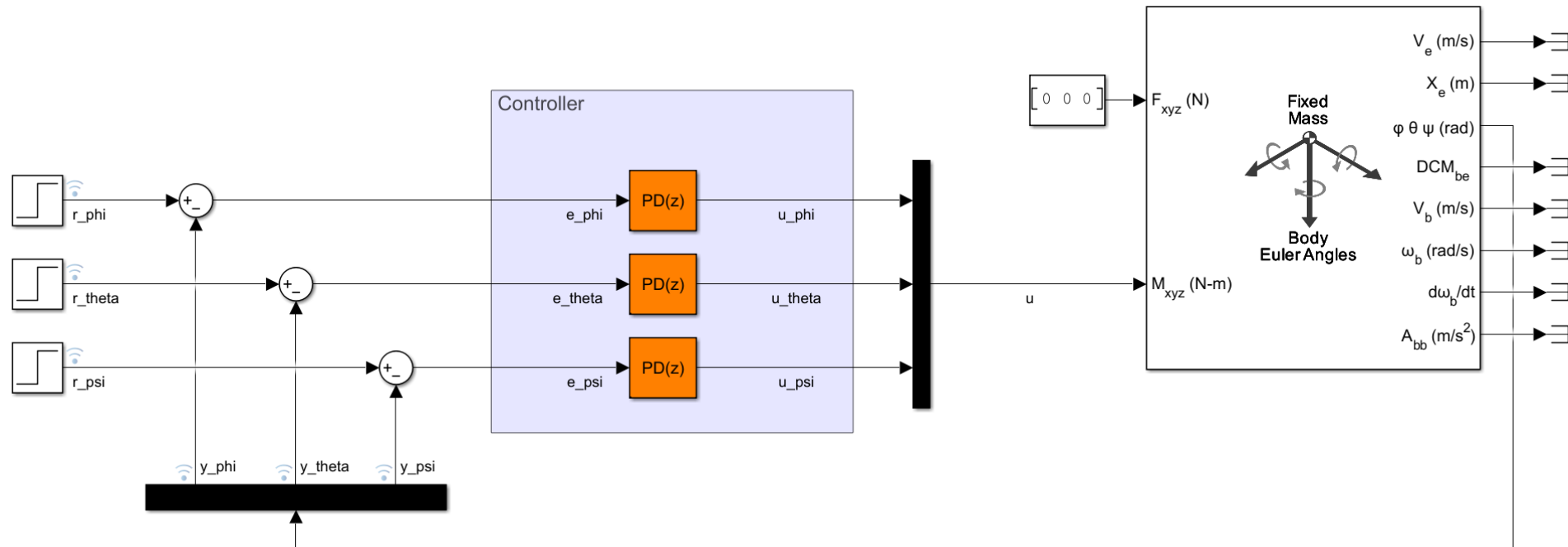


We're going start with three PD controllers

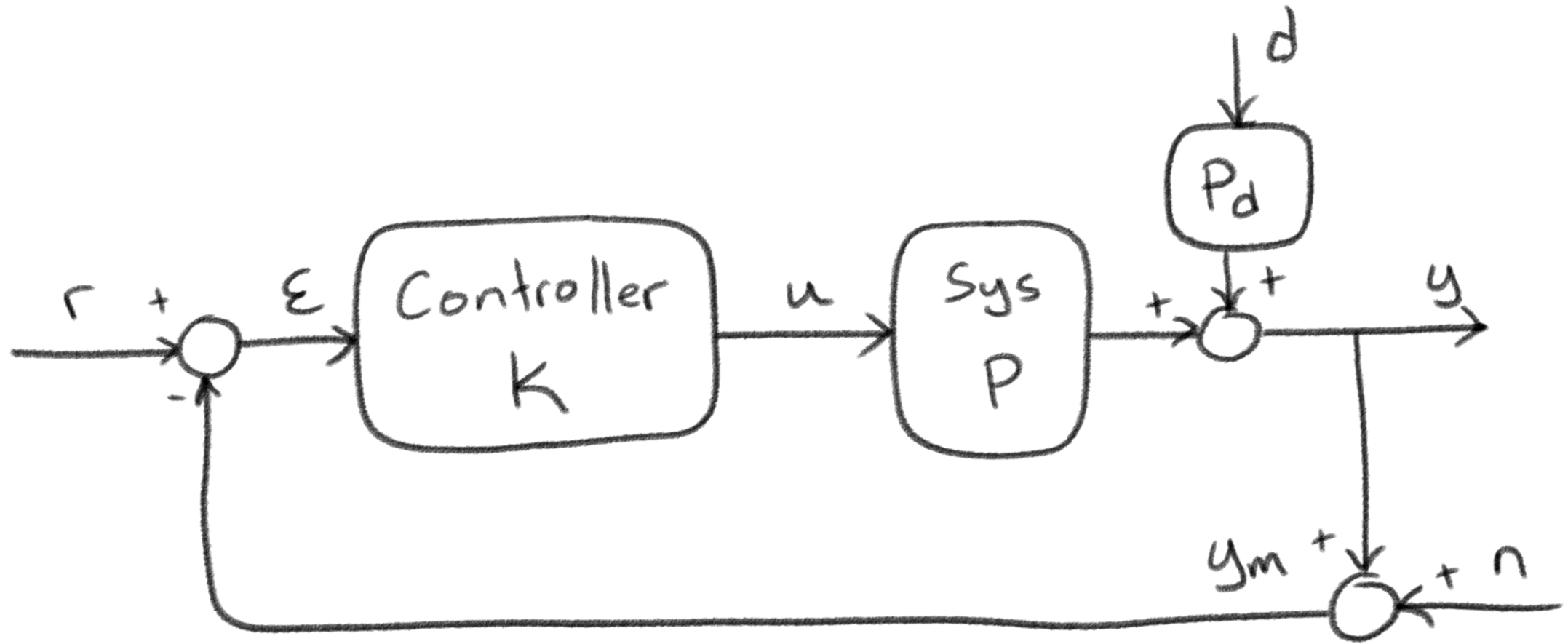


Run Part 1

Tune controller with Control System Tuner App



A fuller picture of the problem



Loop shaping overview

$$y = \underbrace{(I + PK)^{-1} PK}_T r + \underbrace{(I + PK)^{-1} P_d}_S d - \underbrace{(I + PK)^{-1} PK}_T n$$

follow all
frequencies

reject low
frequencies

attenuate high
frequencies

We want K to be more than PD control

PD + Nonlinear feedback + decoupler

$$M = I\dot{\omega} + \omega \times (I\omega)$$

$$K_{\text{mix}}[\omega, \theta] + \text{PD}_{\text{control}} + \omega \times (I\omega) = I\dot{\omega} + \omega \times (I\omega)$$

↑
Decoupler to
remove disturbances

↑
Track
reference

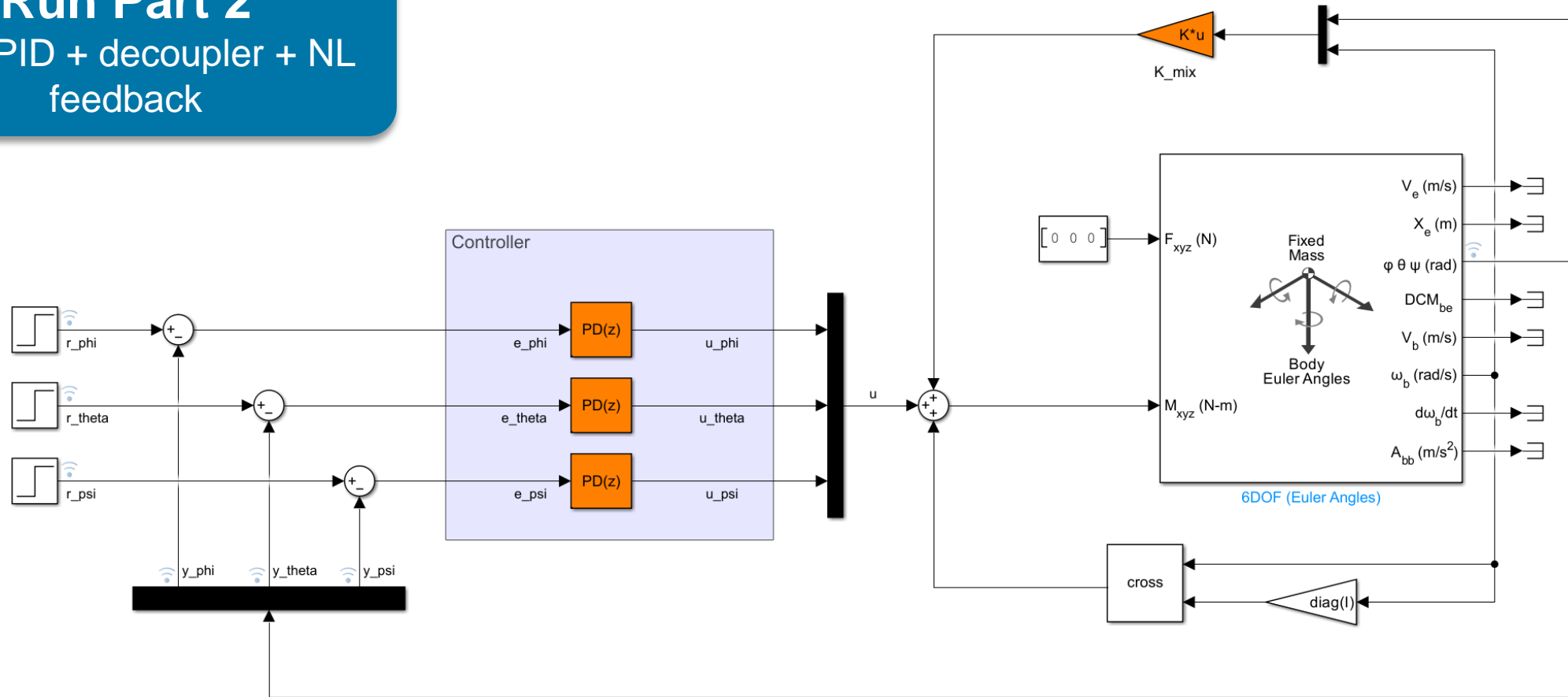
↑
known
nonlinearities

Controller Design and Tuning



Run Part 2

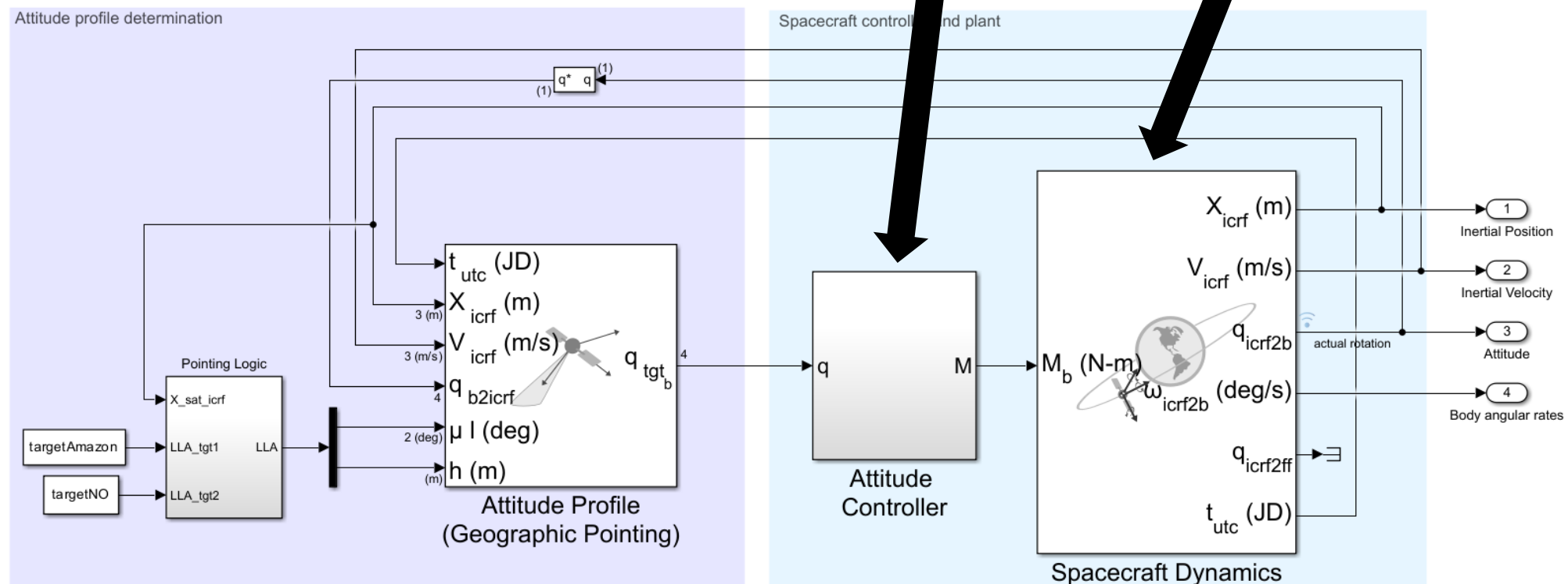
Tune PID + decoupler + NL feedback



Exercise 3: Simulate a Scenario Including a Tuned Controller

Purpose of this exercise:

- Show how the fidelity of Exercise 1 (ideal pointing, no simulation of dynamics) can be improved by adding a controller (controls and dynamics both simulated)



Resources for Aerospace Toolbox & Blockset

- **Product Pages**

- <https://www.mathworks.com/products/aerospace-toolbox.html>
- <https://www.mathworks.com/products/aerospace-blockset.html>

- **Product Overview Videos**

- <https://www.mathworks.com/videos/what-is-aerospace-toolbox--1539774600779.html>
- <https://www.mathworks.com/videos/what-is-aerospace-blockset--1539869697000.html>

- **Documentation**

- <https://www.mathworks.com/help/aerotbx/index.html>
- <https://www.mathworks.com/help/aeroblks/index.html>

- **Examples**

- <https://www.mathworks.com/help/aerotbx/examples.html>
- <https://www.mathworks.com/help/aeroblks/examples.html>

Resources for Controls

Tech Talk Videos

Explore the Control Systems Video Series

System Identification

Watch videos (4 videos)

Fuzzy Logic

Watch videos (4 videos)

Learning-Based Control

Watch videos (3 videos)

State Space

Watch videos (5 videos)

Reinforcement Learning

Watch videos (7 videos)

Trimming and Linearization

Watch videos (2 videos)

Control Systems in Practice

Watch videos (16 videos)

Understanding Model Predictive Control

Watch videos (9 videos)

Understanding PID Control

Watch videos (7 videos)

Reference Posters

Feedback Control Systems
with MATLAB and Simulink

Explore more capabilities for designing and tuning controllers

Traditional **Data-Driven and AI**

Controller Method

PID CONTROL MPC DESIGNER MRAC DATA-DRIVEN MPC IDENTIFIED PREDICTION MODELS

ADRC FUZZY INFERENCE SYSTEMS

REINFORCEMENT LEARNING EXTREMUM SEEKING

Tuning Algorithm

LQR / LQG SYNTHESIS POLE PLACEMENT REINFORCEMENT LEARNING DESIGNER

GRAPHICAL TUNING MULTI-LOOP / MULTI-OBJECTIVE TUNING CLOSED-LOOP PID AUTOTUNER

CO-OPTIMIZATION OF PLANT AND CONTROLLER PARAMETERS CONTROL SYSTEM TUNER SYSTEM IDENTIFICATION WITH MODEL-BASED TUNING

PID CONTROLLER TUNING FUZZY INFERENCE SYSTEM TUNING

RESPONSE OPTIMIZER PID TUNER

Onramps



Control Design Onramp with Simulink

7 modules | 1 hour | Languages

Get started quickly with the basics of feedback control design in Simulink.



Reinforcement Learning Onramp

5 modules | 2.5 hours | Languages

Master the basics of creating intelligent controllers that learn from experience.