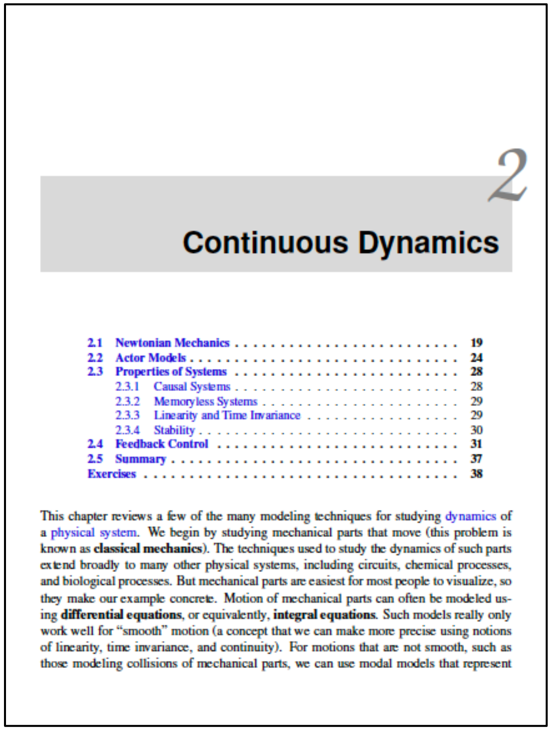
Lecture 8 : Continuous Dynamics

Slides were originally developed by Profs. Edward Lee and Sanjit Seshia, and subsequently updated by Profs. Gavin Buskes and Iman Shames.

Outline

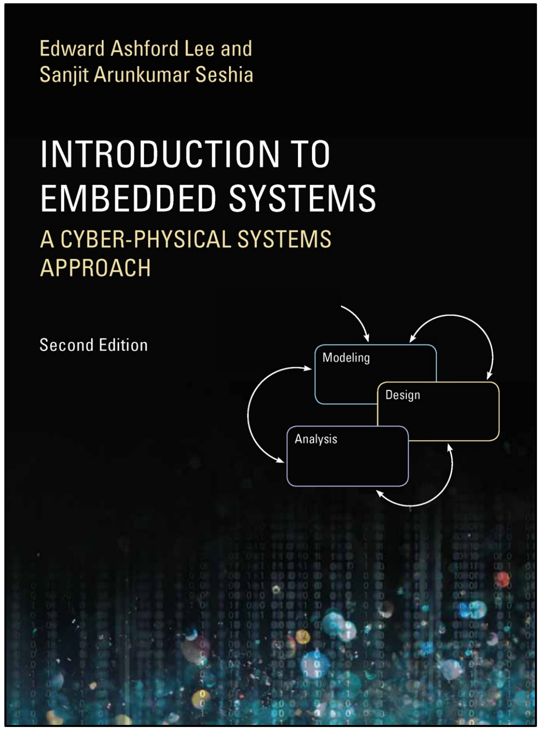
- Modelling and its value
- Actor model of systems



2

Continuous Dynamics

This chapter reviews a few of the many modeling techniques for studying dynamics of a physical system. We begin by studying mechanical parts that move (this problem is known as **classical mechanics**). The techniques used to study the dynamics of such parts extend broadly to many other physical systems, including circuits, chemical processes, and biological processes. But mechanical parts are easiest for most people to visualize, so they make our example concrete. Motion of mechanical parts can often be modeled using **differential equations**, or equivalently, **integral equations**. Such models really only work well for "smooth" motion (a concept that we can make more precise using notions of linearity, time invariance, and continuity). For motions that are not smooth, such as those modeling collisions of mechanical parts, we can use modal models that represent



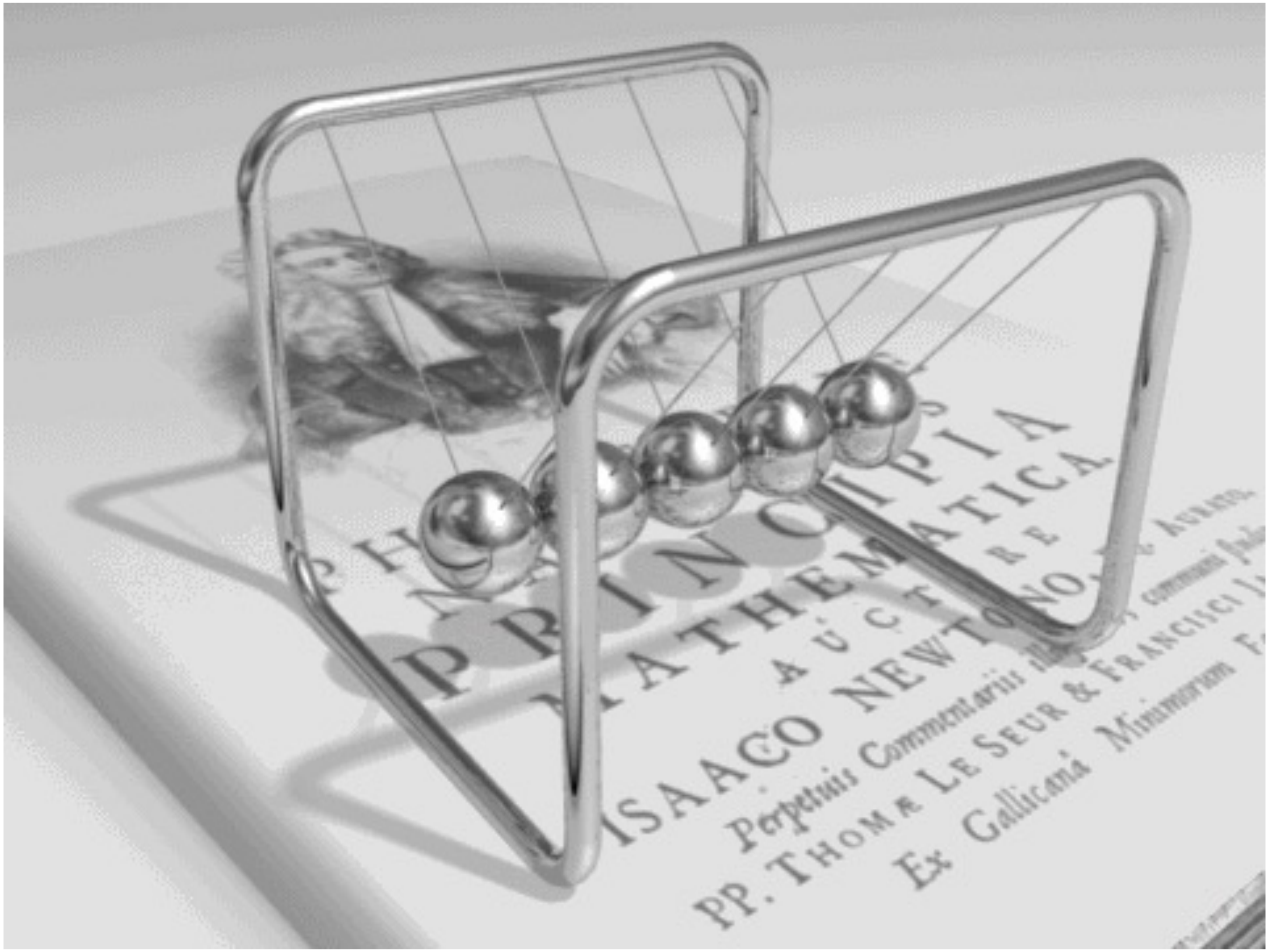
The Value of Models

- In *science*, the value of a *model* lies in how well its behaviour matches that of the physical system.
- In *engineering*, the value of the *physical system* lies in how well its behaviour matches that of the model.

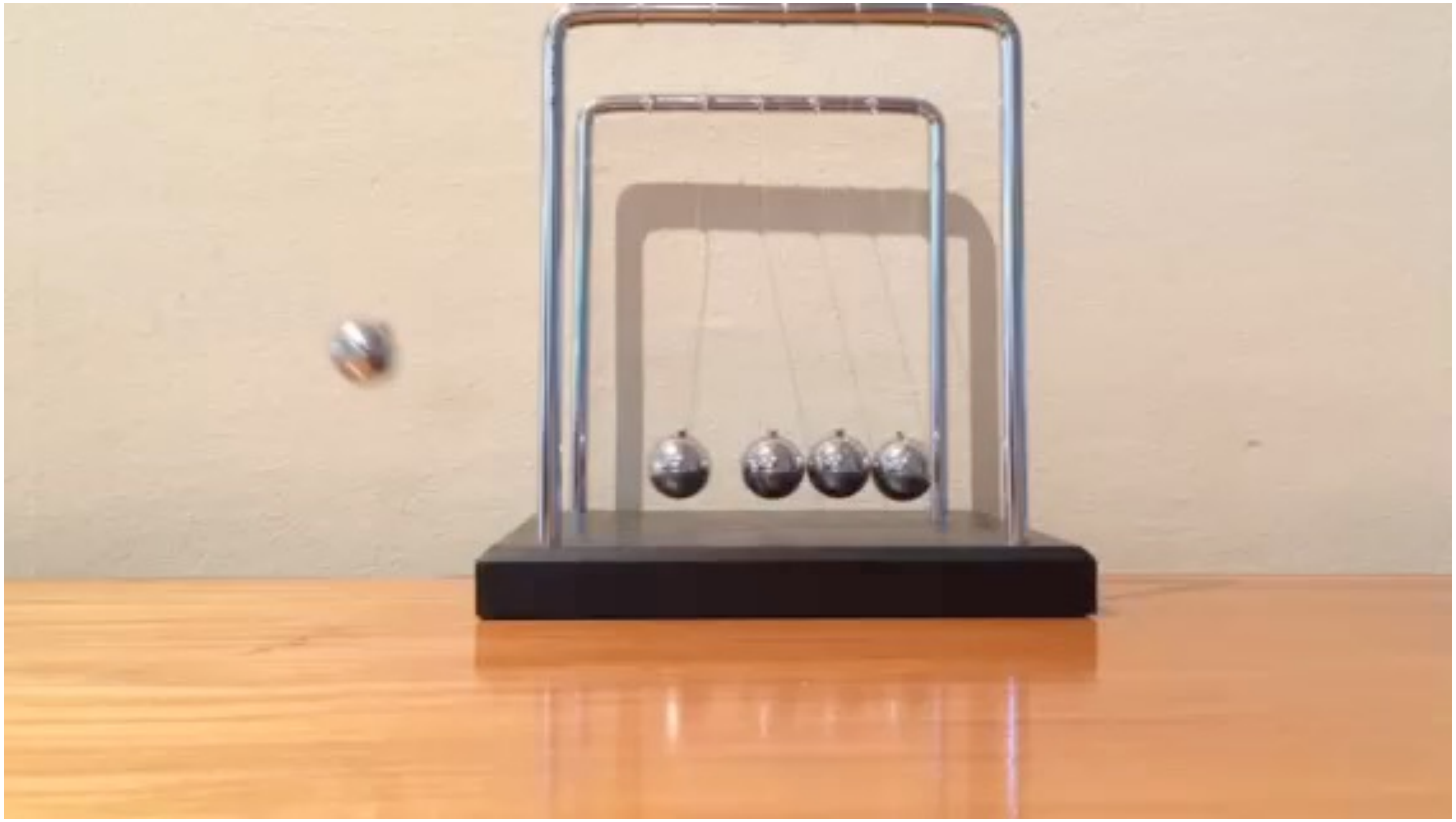
In engineering, model fidelity is a two-way street!

For a model to be useful, it is necessary (but not sufficient) to be able to be able to construct a faithful physical realisation.

A Model


PRINCIPIA
MATHEMATICA
AUCTORE
ISAACO NEWTONO
Perpetuis Commentariis
PP. THOMÆ LE SEUR & FRANCISCI
Ex Gallicanâ Minimorum

A Physical Realisation



Model Fidelity

- To a *scientist*, the model is flawed.

- To an *engineer*, the realisation is flawed.

I'm an engineer…

For CPS, we need to Change the Question

The question is *not* whether deterministic models can describe the behaviour of cyber-physical systems (with high fidelity).

The question is whether we can build cyber-physical systems whose behaviour matches that of a deterministic model (with high probability).

Deterministic models do not eliminate the need for robust, fault-tolerant designs.

In fact, they *enable* such designs, because they make it much clearer what it means to have a fault!

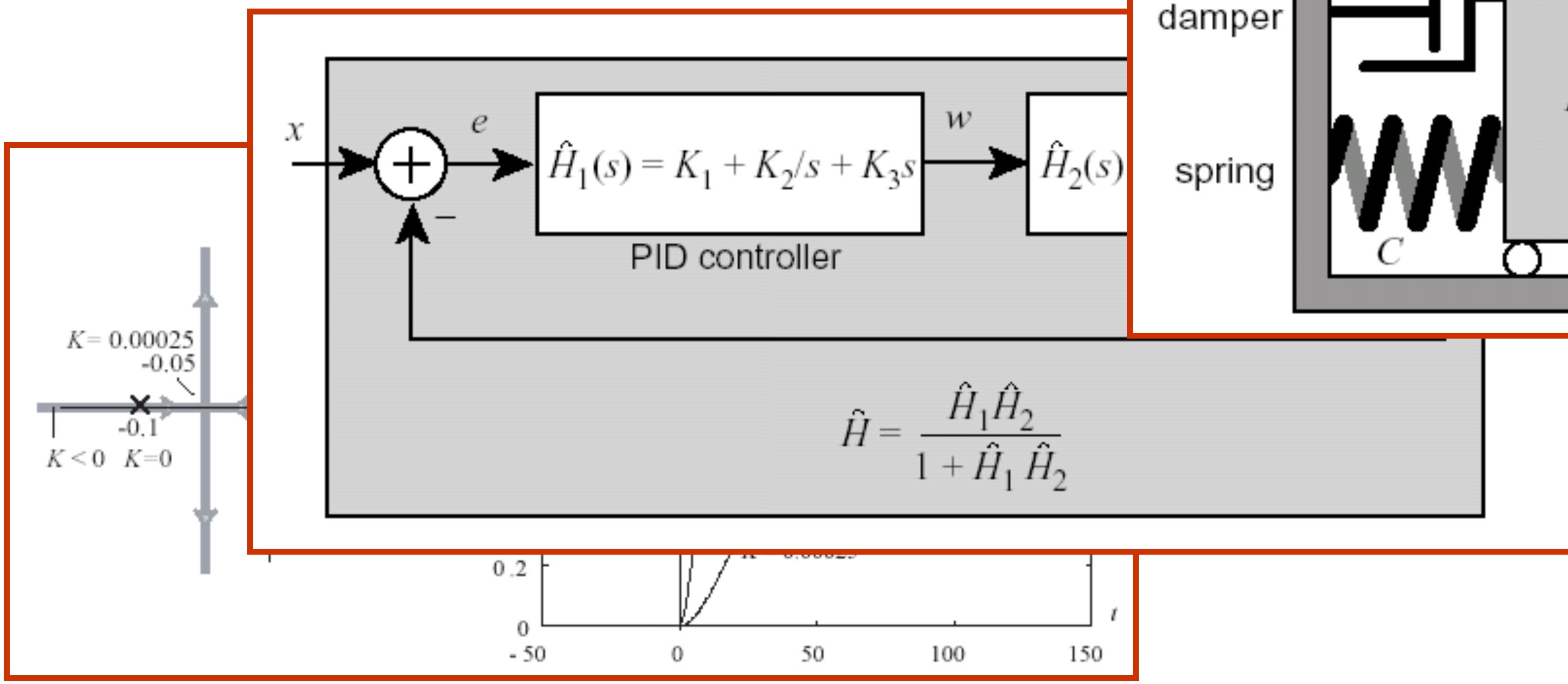
Modeling Techniques

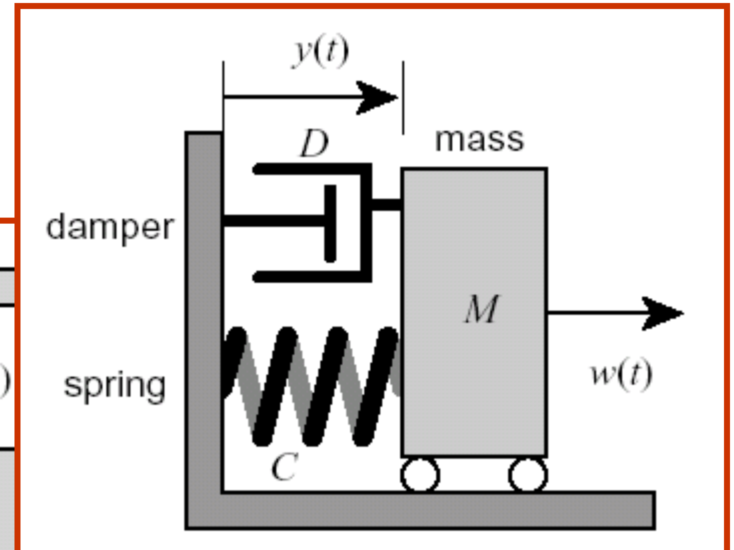
Models that are abstractions of **system dynamics**
(how system behavior changes over time)

- Modeling physical phenomena – differential equations
- Feedback control systems – time-domain modeling
- Modeling modal behavior – FSMs, hybrid automata, …
- Modeling sensors and actuators –calibration, noise, …
- Hardware and software – concurrency, timing, power, …
- Networks – latencies, error rates, packet losses, …

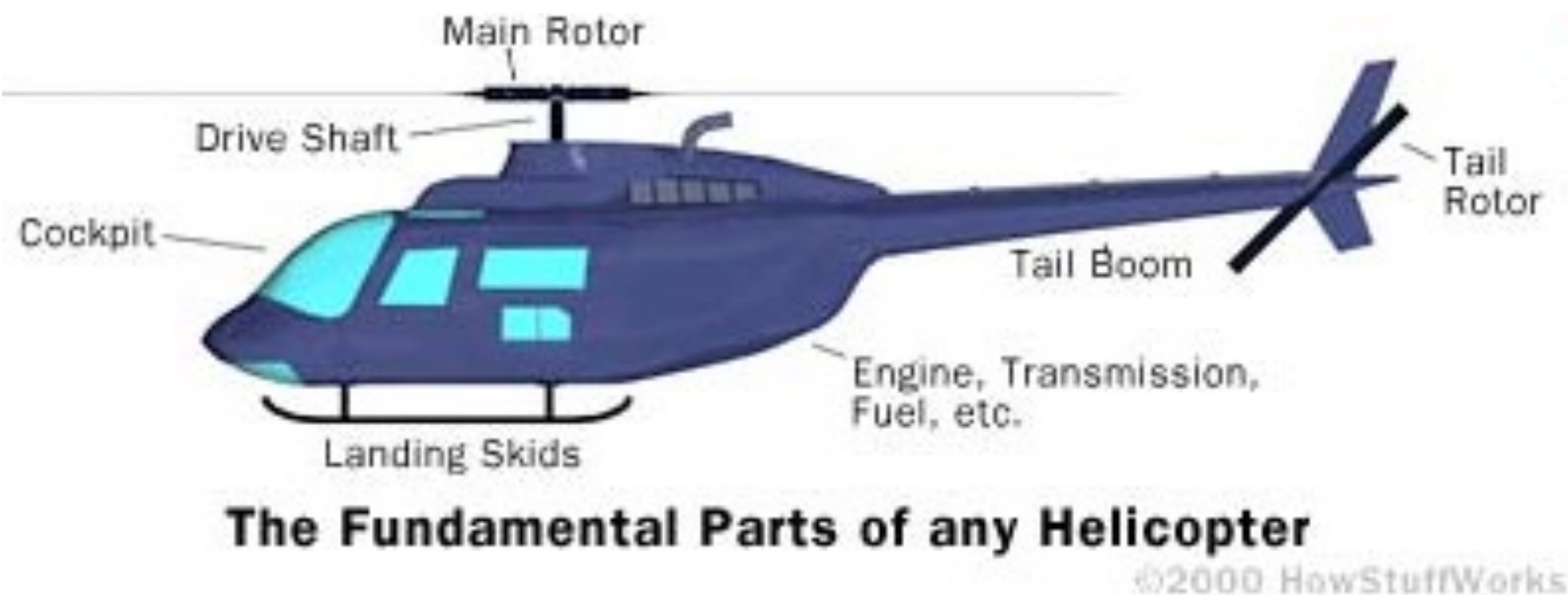
Modeling of Continuous Dynamics

- Ordinary differential equations, Laplace transforms, feedback control models, …





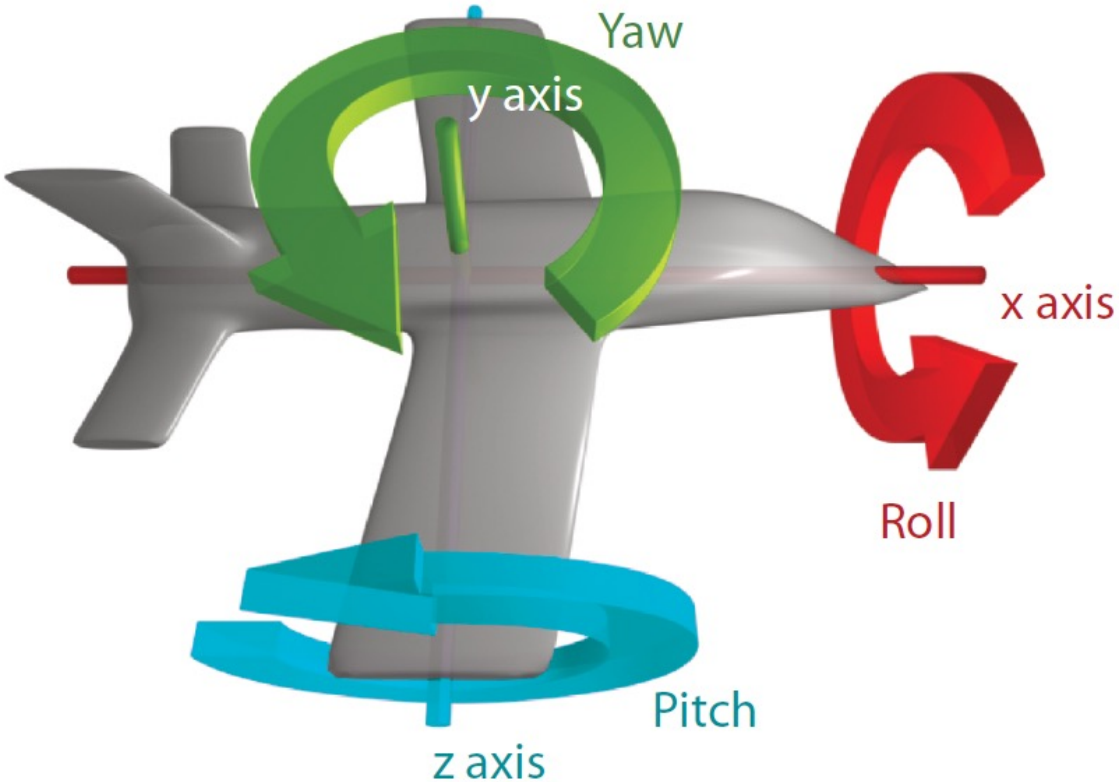
An Example: Helicopter Dynamics



Modeling Physical Motion

Six degrees of freedom:

- Position: x, y, z
- Orientation: pitch, yaw, roll



Notation

Position is given by three functions:

$$x : \mathbb{R} \to \mathbb{R}$$
$$y : \mathbb{R} \to \mathbb{R}$$
$$z : \mathbb{R} \to \mathbb{R}$$

where the domain $\mathbb{R}$ represents time and the co-domain (range) $\mathbb{R}$ represents position along the axis. Collecting into a vector:

$$\mathbf{x} : \mathbb{R} \to \mathbb{R}^3$$

Position at time $t \in \mathbb{R}$ is $\mathbf{x}(t) \in \mathbb{R}^3$.

Notation

Velocity

$$\dot{\mathbf{x}} \colon \mathbb{R} \to \mathbb{R}^3$$

is the derivative, $\forall\, t \in \mathbb{R}$,

$$\dot{\mathbf{x}}(t) = \frac{d}{dt}\mathbf{x}(t)$$

Acceleration $\ddot{\mathbf{x}} \colon \mathbb{R} \to \mathbb{R}^3$ is the second derivative,

$$\ddot{\mathbf{x}} = \frac{d^2}{dt^2}\mathbf{x}$$

Force on an object is $\mathbf{F} \colon \mathbb{R} \to \mathbb{R}^3$.

Newton's Second Law

Newton's second law states $\forall\, t \in \mathbb{R}$,

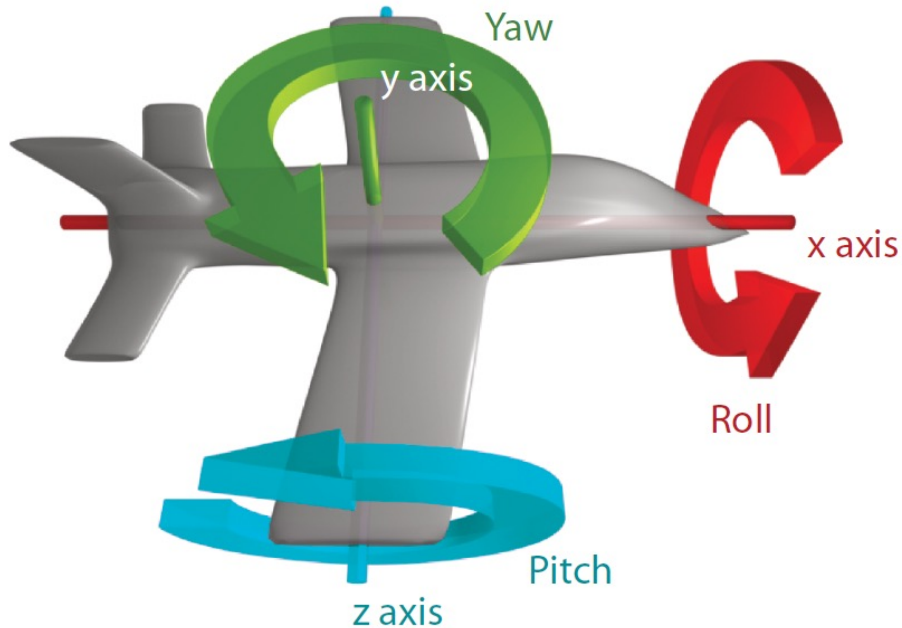
$$\mathbf{F}(t) = M\ddot{\mathbf{x}}(t)$$

where M is the mass. To account for initial position and velocity, convert this to an integral equation

$$\begin{aligned}
\mathbf{x}(t) &= \mathbf{x}(0) + \int_0^t \dot{\mathbf{x}}(\tau)d\tau \\
&= \mathbf{x}(0) + t\dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \int_0^\tau \mathbf{F}(\alpha)d\alpha d\tau,
\end{aligned}$$

Orientation

- Orientation: $\theta : \mathbb{R} \to \mathbb{R}^3$
- Angular velocity: $\dot{\theta} : \mathbb{R} \to \mathbb{R}^3$
- Angular acceleration: $\ddot{\theta} : \mathbb{R} \to \mathbb{R}^3$
- Torque: $\mathbf{T} : \mathbb{R} \to \mathbb{R}^3$

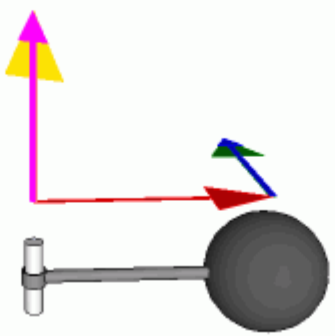


$$\theta(t) = \begin{bmatrix} \dot{\theta}_x(t) \\ \dot{\theta}_y(t) \\ \dot{\theta}_z(t) \end{bmatrix} = \begin{bmatrix} \text{roll} \\ \text{yaw} \\ \text{pitch} \end{bmatrix}$$

Angular version of force is torque

For a point mass rotating around a fixed axis:

- radius of the arm: $r \in \mathbb{R}$
- force orthogonal to arm: $f \in \mathbb{R}$
- mass of the object: $m \in \mathbb{R}$



$$T_y(t) = r f(t)$$

angular momentum, momentum

- Just as force is a push or a pull, a torque is a twist.
- Units: newton-meters/radian, Joules/radian

- Note that radians are metres/metre (2π metres of circumference per 1 metre of radius), so as units, are optional.

Rotational Version of Newton's Second Law

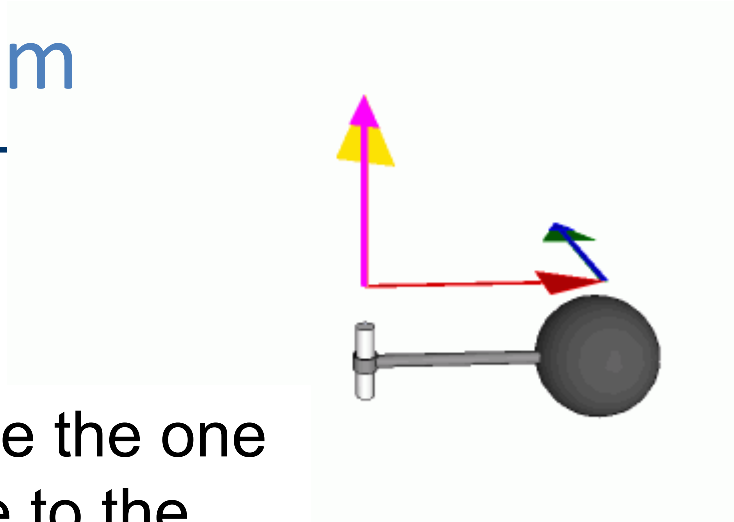
$$\mathbf{T}(t) = \frac{d}{dt}\left(I(t)\dot{\theta}(t)\right),$$

where $I(t)$ is a 3×3 matrix called the moment of inertia tensor.

$$\begin{bmatrix} T_x(t) \\ T_y(t) \\ T_z(t) \end{bmatrix} = \frac{d}{dt}\left(\begin{bmatrix} I_{xx}(t) & I_{xy}(t) & I_{xz}(t) \\ I_{yx}(t) & I_{yy}(t) & I_{yz}(t) \\ I_{zx}(t) & I_{zy}(t) & I_{zz}(t) \end{bmatrix}\begin{bmatrix} \dot{\theta}_x(t) \\ \dot{\theta}_y(t) \\ \dot{\theta}_z(t) \end{bmatrix}\right)$$

Here, for example, $T_y(t)$ is the net torque around the y axis (which would cause changes in yaw), $I_{yx}(t)$ is the inertia that determines how acceleration around the x axis is related to torque around the y axis.

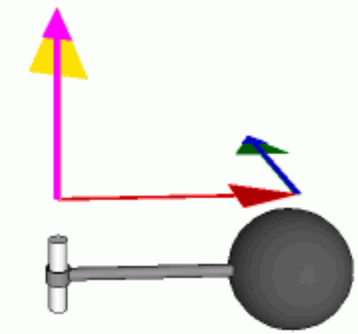
Feedback Control Problem



A helicopter without a tail rotor, like the one below, will spin uncontrollably due to the torque induced by friction in the rotor shaft.

<u>Control system problem</u>: Apply torque using the tail rotor to counterbalance the torque of the top rotor.

Simplified Model



Yaw dynamics:

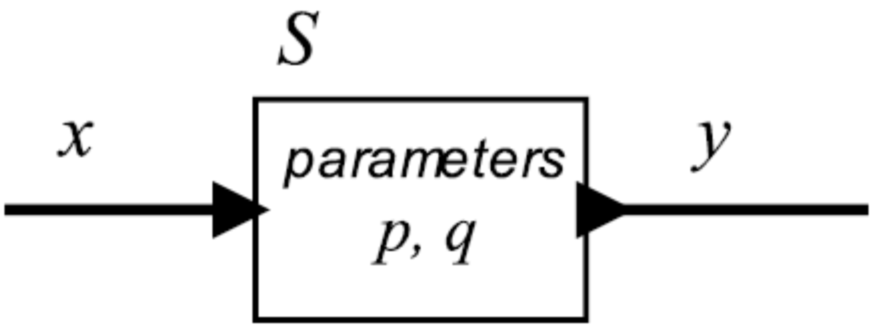
$$T_y(t) = I_{yy}\ddot{\theta}_y(t)$$

To account for initial angular velocity, write as

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau.$$

Actor Model of Systems

- A *system* is a function that accepts an input signal and yields an output signal.
- The domain and range of the system function are sets of signals, which themselves are functions.
- Parameters may affect the definition of the function *S*.



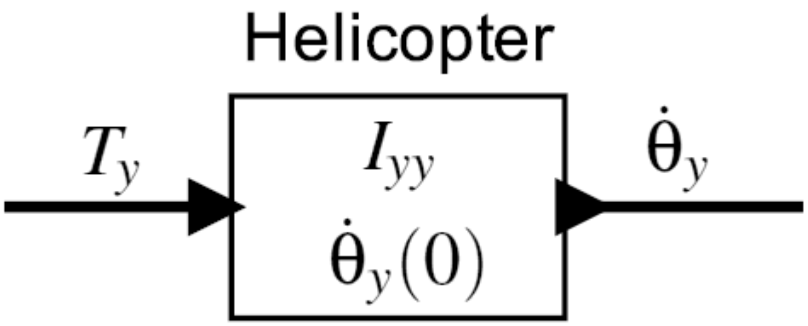
$$x\colon \mathbb{R} \to \mathbb{R}, \quad y\colon \mathbb{R} \to \mathbb{R}$$

$$S\colon X \to Y$$

$$X = Y = (\mathbb{R} \to \mathbb{R})$$

Actor Model of the Helicopter

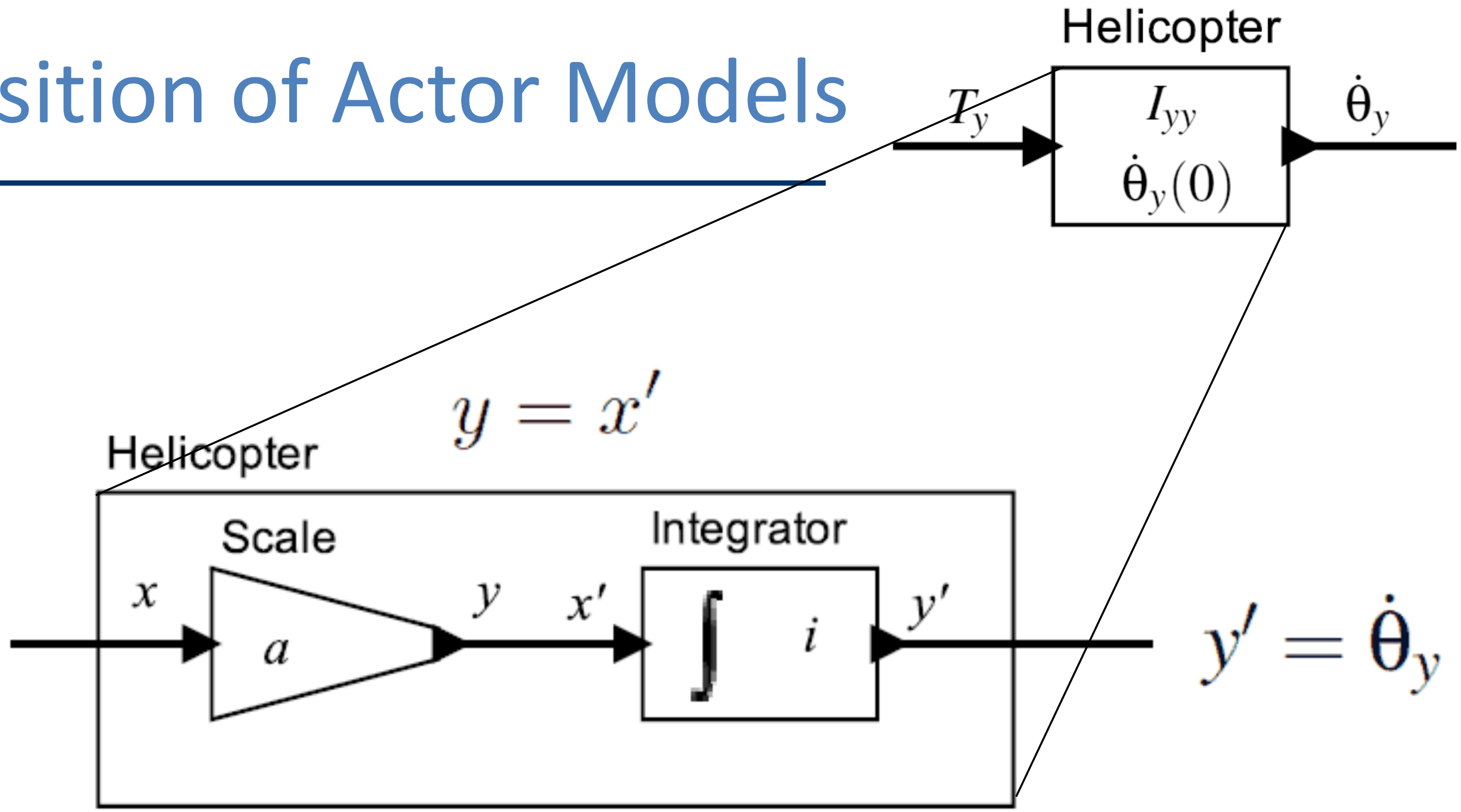
- Input is the net torque of the tail rotor and the top rotor. Output is the angular velocity around the y axis.



Parameters of the model are shown in the box. The input and output relation is given by the equation to the right.

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau) d\tau$$

Composition of Actor Models



$x = T_y$

$y = x'$

$y' = \dot{\theta}_y$

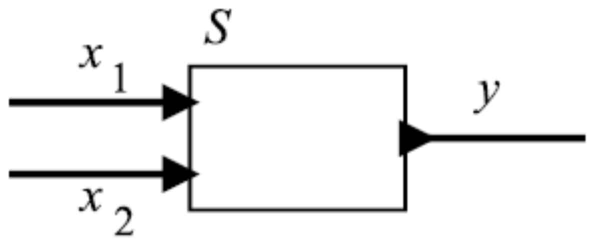
$$\forall\, t \in \mathbb{R}, \quad y(t) = ax(t) \qquad y'(t) = i + \int_0^t x'(\tau)\,d\tau$$

$$y = ax$$

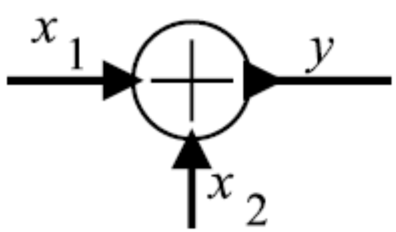
$$a = 1/I_{yy}$$

$$i = \dot{\theta}_y(0)$$

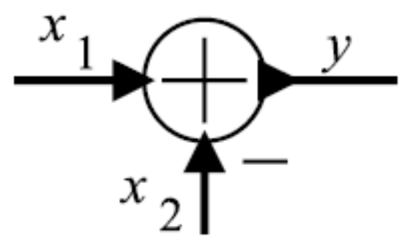
Actor Models with Multiple Inputs



$$S\colon (\mathbb{R} \to \mathbb{R})^2 \to (\mathbb{R} \to \mathbb{R})$$

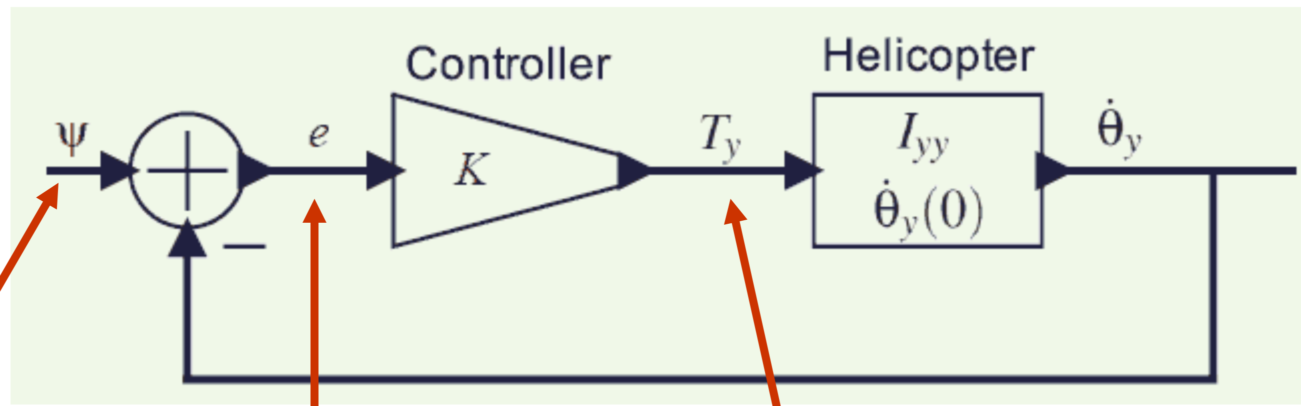


$$\forall\, t \in \mathbb{R}, \quad y(t) = x_1(t) + x_2(t)$$



$$(S(x_1, x_2))(t) = y(t) = x_1(t) - x_2(t)$$

Proportional controller



desired angular velocity

error signal

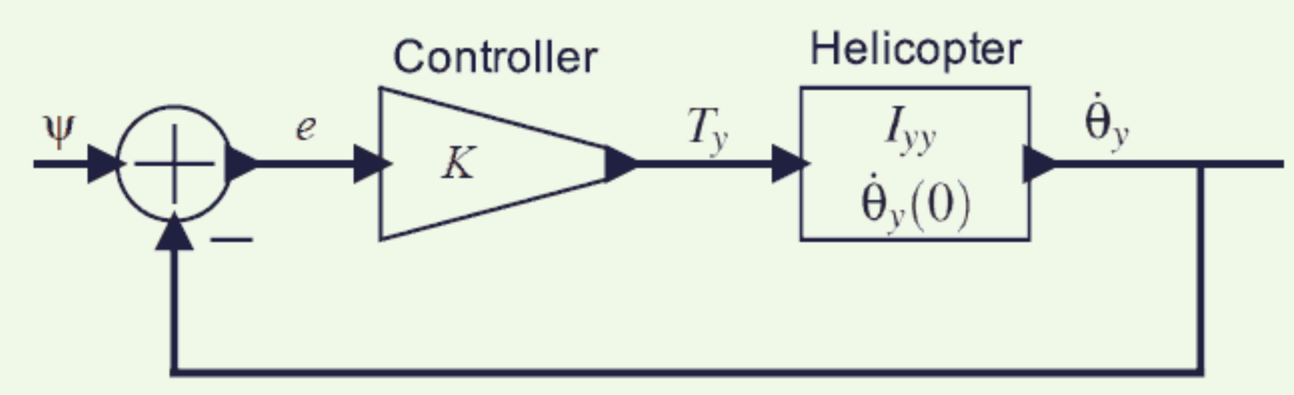
net torque

$$e(t) = \psi(t) - \dot{\theta}_y(t) \qquad T_y(t) = Ke(t)$$

$$\begin{aligned}\dot{\theta}_y(t) &= \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau)\,d\tau \\ &= \dot{\theta}_y(0) + \frac{K}{I_{yy}} \int_0^t (\psi(\tau) - \dot{\theta}_y(\tau))\,d\tau\end{aligned}$$

Note that the angular velocity appears on both sides, so this equation is not trivial to solve.

Behavior of the controller



$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{K}{I_{yy}} \int_0^t (\psi(\tau) - \dot{\theta}_y(\tau))d\tau$$

Desired angular velocity: $\psi(t) = 0$

Simplifies differential equation to:

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) - \frac{K}{I_{yy}} \int_0^t \dot{\theta}_y(\tau)d\tau$$

Which can be solved as follows (see textbook):

$$\dot{\theta}_y(t) = \dot{\theta}_y(0)e^{-Kt/I_{yy}}u(t)$$

Questions

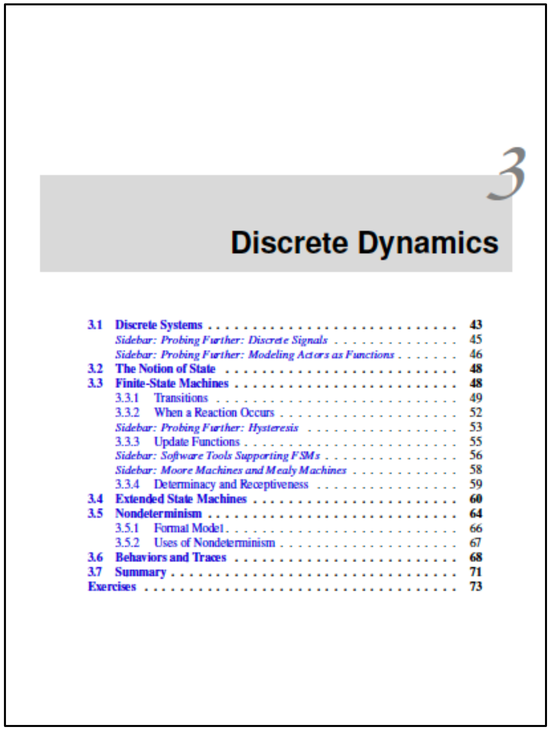
- Can the behaviour of this controller change when it is implemented in software?

- How do we measure the angular velocity in practice? How do we incorporate noise into this model?

- What happens when you have failures (sensors, actuators, software, computers, or networks)?

Things to do …

- Download the textbook and read Chapter 3
- Complete the assignment and return by Friday August 19 mid-night!
- Read over Workshop 3 and do the pre-workshop work



3

Discrete Dynamics