[dcc]

MODEL BASED DESIGN

Pedro Brandão, Sérgio Crisóstomo

Sistemas Embutidos 2020/21

U. PORTO

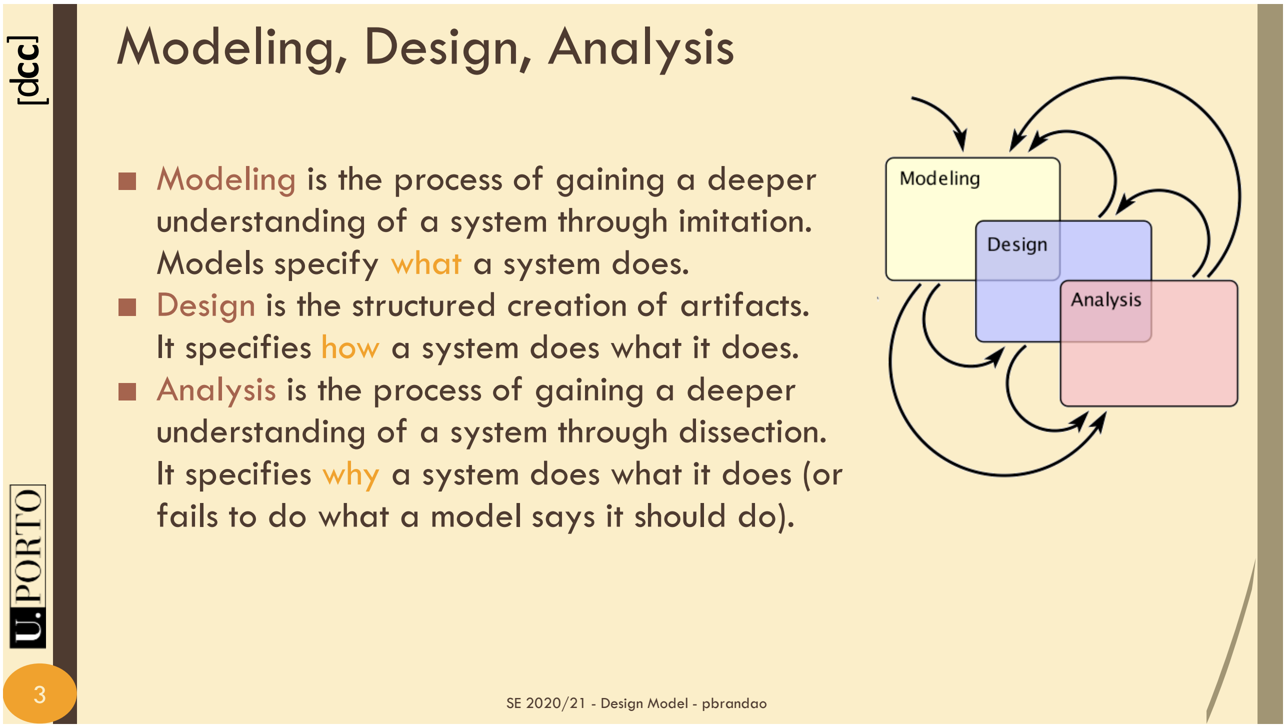
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

1

References

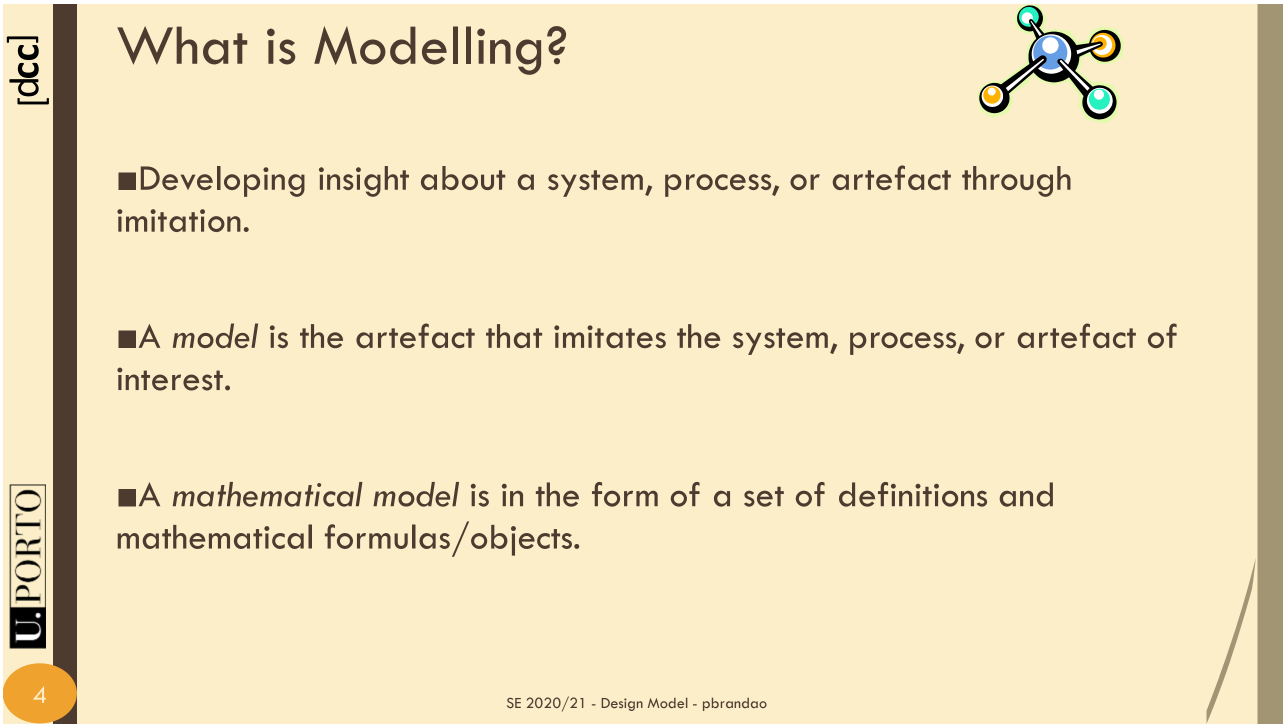
- Slides are from Edward A. Lee & Sanjit Seshia, UC Berkeley, EECS 149 Fall 2013
  - *Copyright © 2008-2016, Edward A. Lee & Sanjit A. Seshia, All rights reserved*
- Some slides from Philip Asare w/ Joanne Dugan and Ron Williams from UVA

2



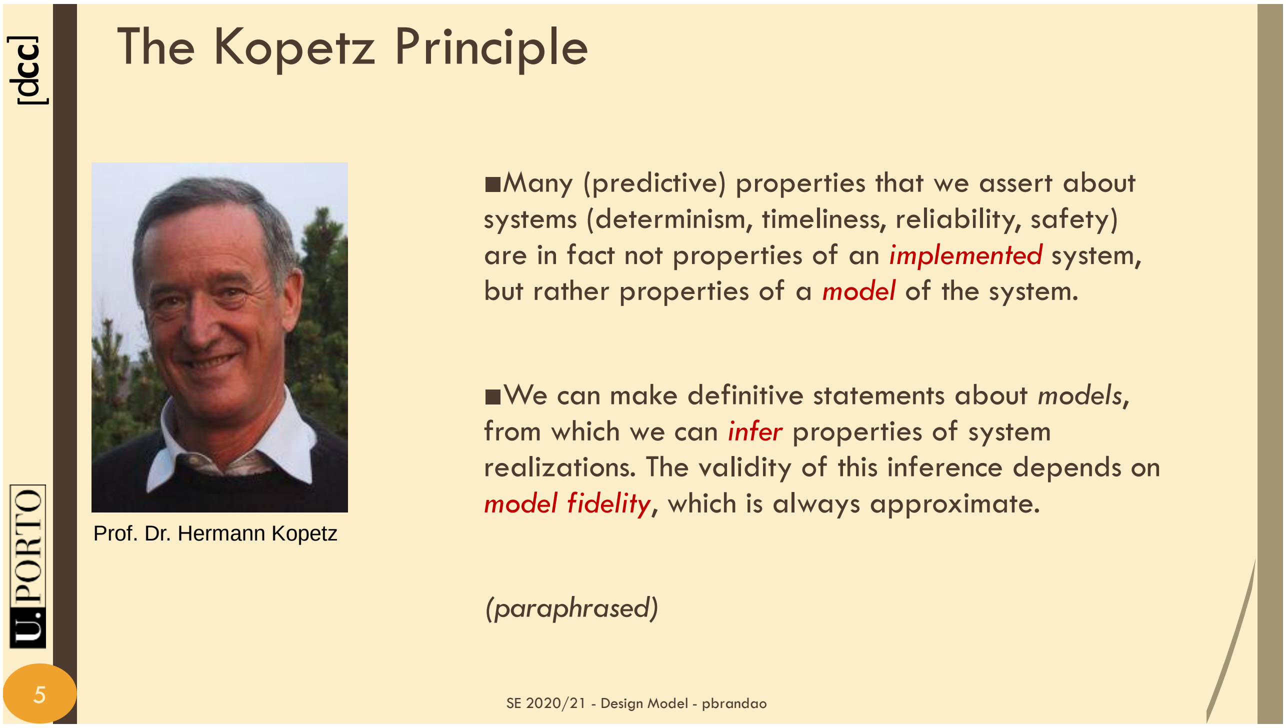
Modeling, Design, Analysis

- Modeling is the process of gaining a deeper understanding of a system through imitation. Models specify what a system does.
- Design is the structured creation of artifacts. It specifies how a system does what it does.
- Analysis is the process of gaining a deeper understanding of a system through dissection. It specifies why a system does what it does (or fails to do what a model says it should do).



What is Modelling?

- Developing insight about a system, process, or artefact through imitation.
- A *model* is the artefact that imitates the system, process, or artefact of interest.
- A *mathematical model* is in the form of a set of definitions and mathematical formulas/objects.



The Kopetz Principle

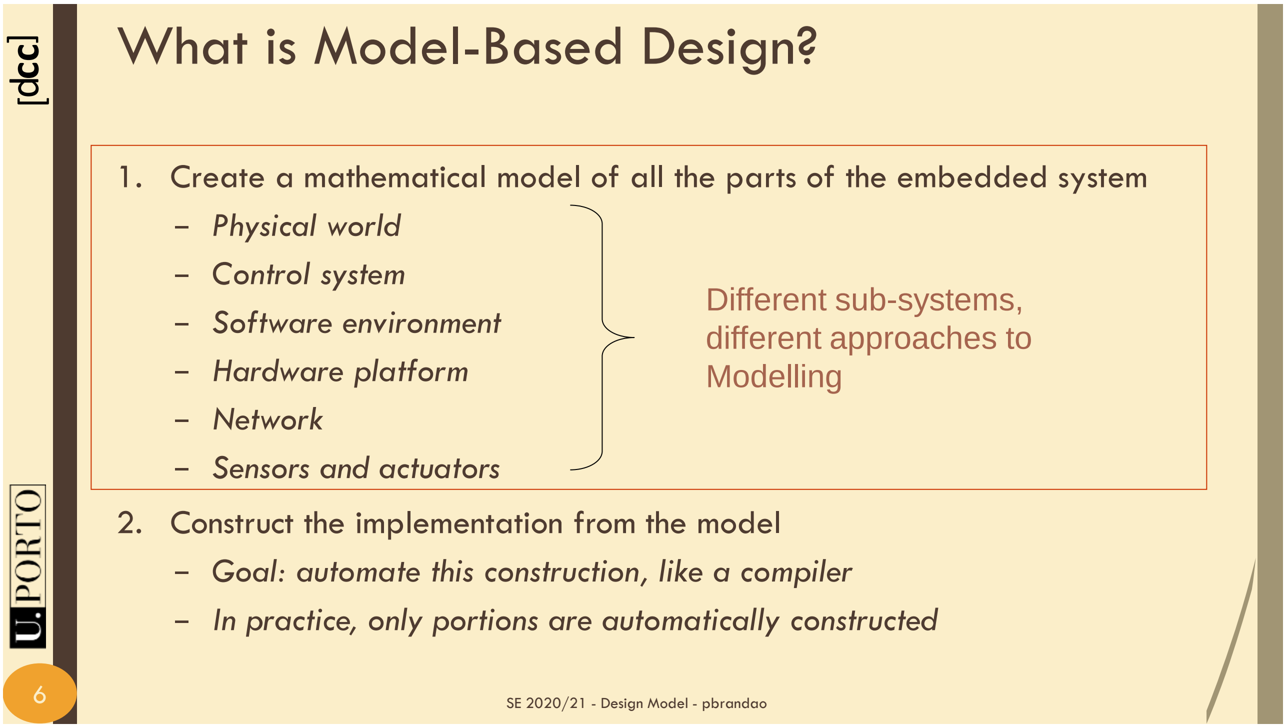
Prof. Dr. Hermann Kopetz

■Many (predictive) properties that we assert about systems (determinism, timeliness, reliability, safety) are in fact not properties of an *implemented* system, but rather properties of a *model* of the system.

■We can make definitive statements about *models*, from which we can *infer* properties of system realizations. The validity of this inference depends on *model fidelity*, which is always approximate.

(paraphrased)

5



What is Model-Based Design?

1. Create a mathematical model of all the parts of the embedded system
   - *Physical world*
   - *Control system*
   - *Software environment*
   - *Hardware platform*
   - *Network*
   - *Sensors and actuators*

Different sub-systems, different approaches to Modelling

2. Construct the implementation from the model
   - *Goal: automate this construction, like a compiler*
   - *In practice, only portions are automatically constructed*

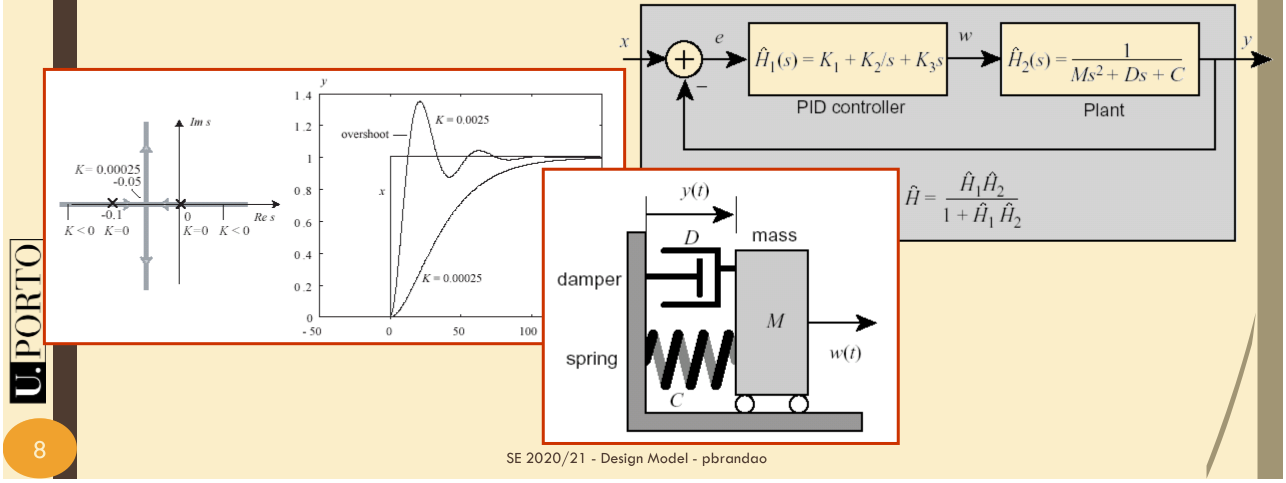
6

Modelling Techniques

- Models that are abstractions of system dynamics
  - *(how things change over time)*
- Examples:
  - *Modelling physical phenomena – Ordinary Differential Equations (ODEs)*
  - *Feedback control systems – time-domain Modelling*
  - *Modelling modal behaviour – FSMs, hybrid automata*
  - *Modelling sensors and actuators – calibration, noise*
  - *Modelling software – concurrency, real-time models*
  - *Modelling networks – latencies, error rates, packet loss*

Modelling of Continuous Dynamics

- Ordinary differential equations, Laplace transforms, feedback control systems, stability analysis, robustness analysis, …



An Example: Modelling Helicopter Dynamics

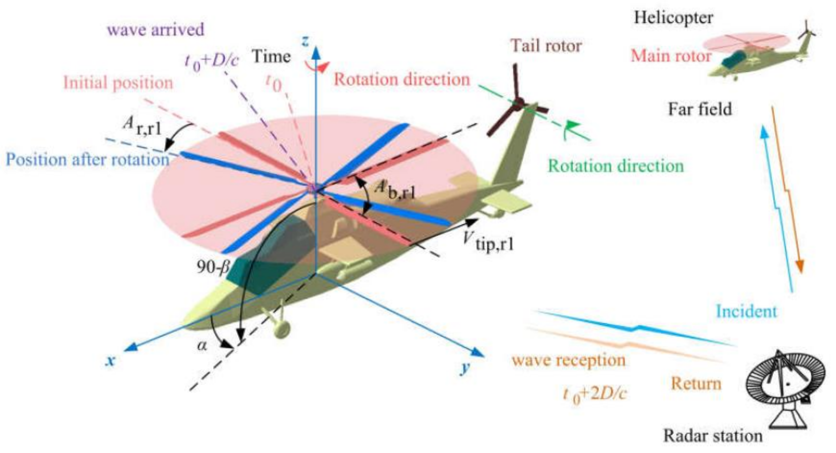
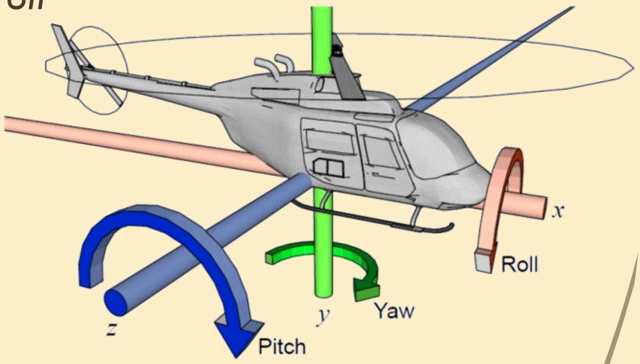
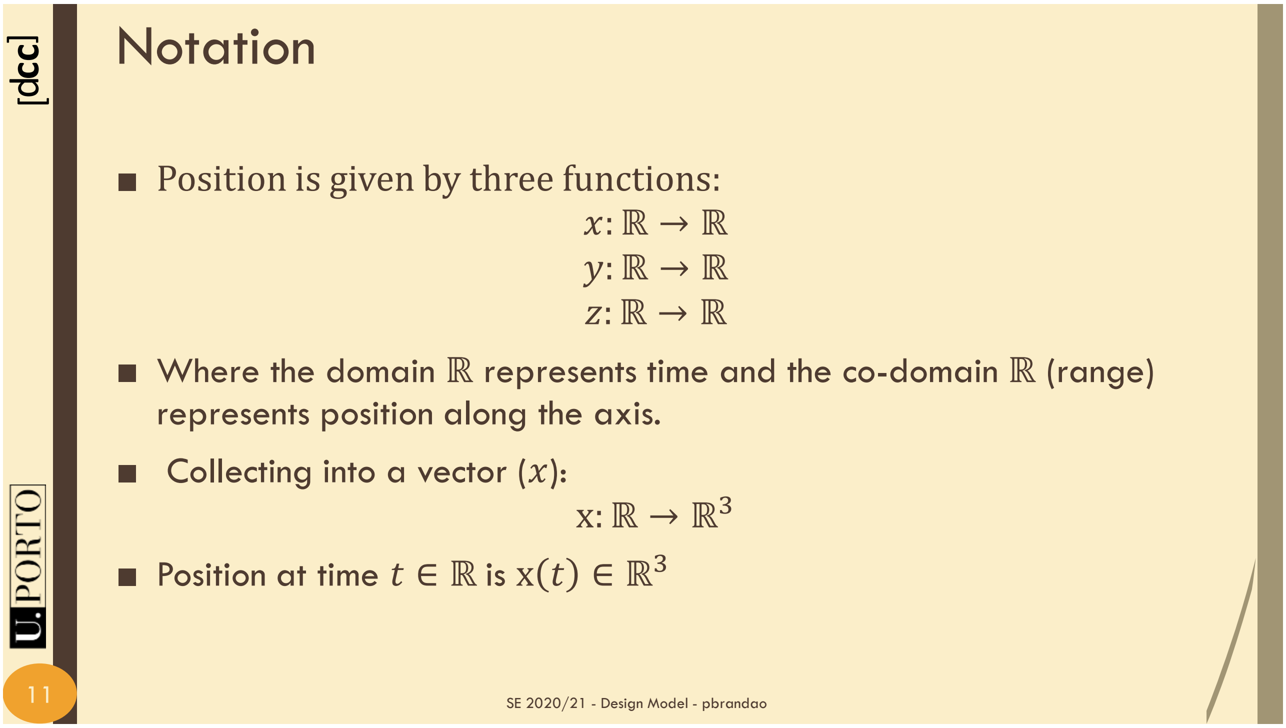


Image from Zhou Z, Huang J. Influence of Rotor Dynamic Scattering on Helicopter Radar Cross-Section. *Sensors.* 2020; 20(7):2097.

Modelling Physical Motion

- Six degrees of freedom:
  - *Position: x, y, z*
  - *Orientation: pitch, yaw, roll*





Notation

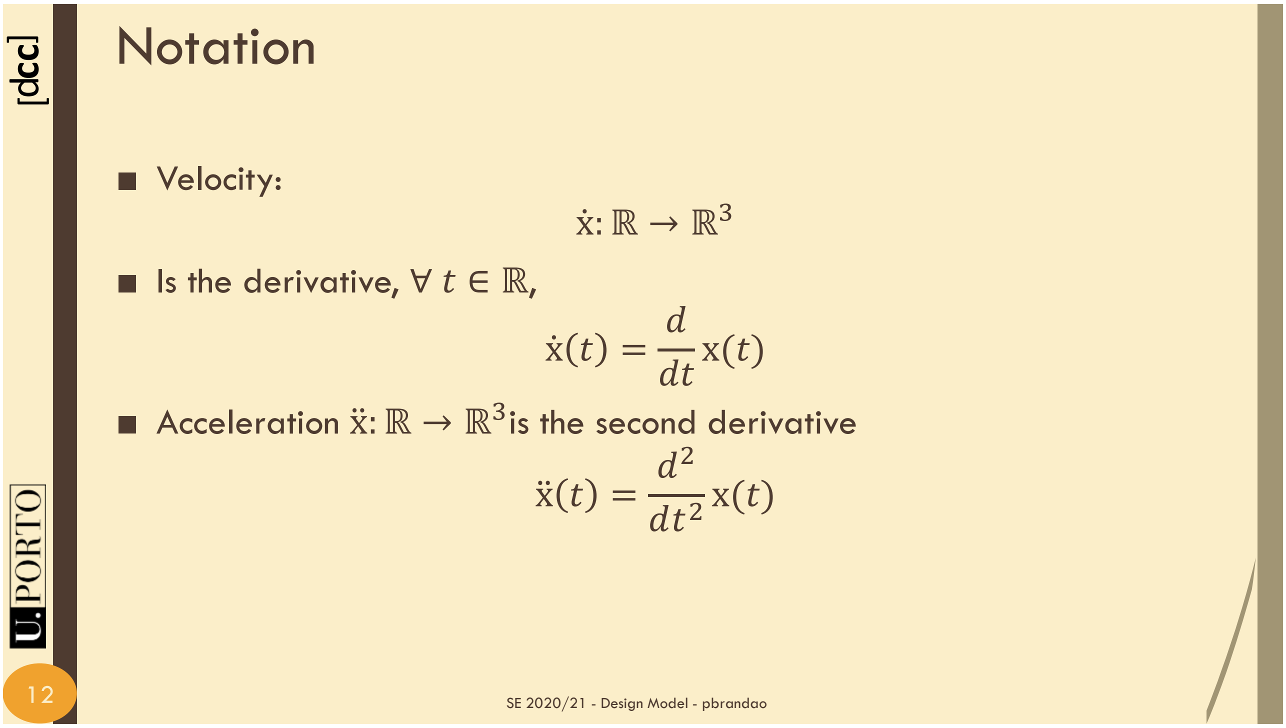
- Position is given by three functions:

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

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

$$\mathrm{x}: \mathbb{R} \rightarrow \mathbb{R}^3$$

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



Notation

- Velocity:

$$\dot{\mathrm{x}}: \mathbb{R} \rightarrow \mathbb{R}^3$$

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

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

- Acceleration $\ddot{\mathrm{x}}: \mathbb{R} \rightarrow \mathbb{R}^3$is the second derivative

$$\ddot{\mathrm{x}}(t) = \frac{d^2}{dt^2}\mathrm{x}(t)$$

Newton’s Second Law

- Force on an object is $F\colon \mathbb{R} \to \mathbb{R}^3$
- Newton’s second law states $\forall t \in \mathbb{R}$,

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

- where M is the mass.

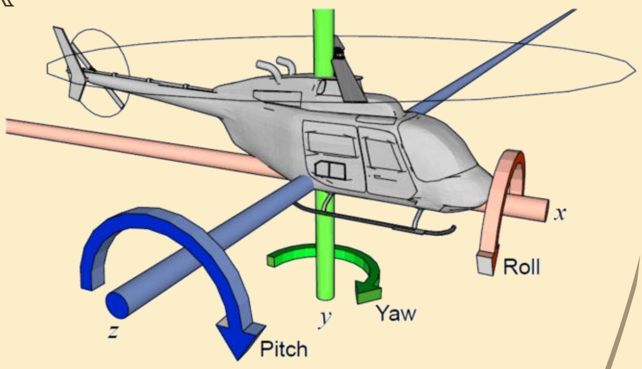
- Converting to integral equation:

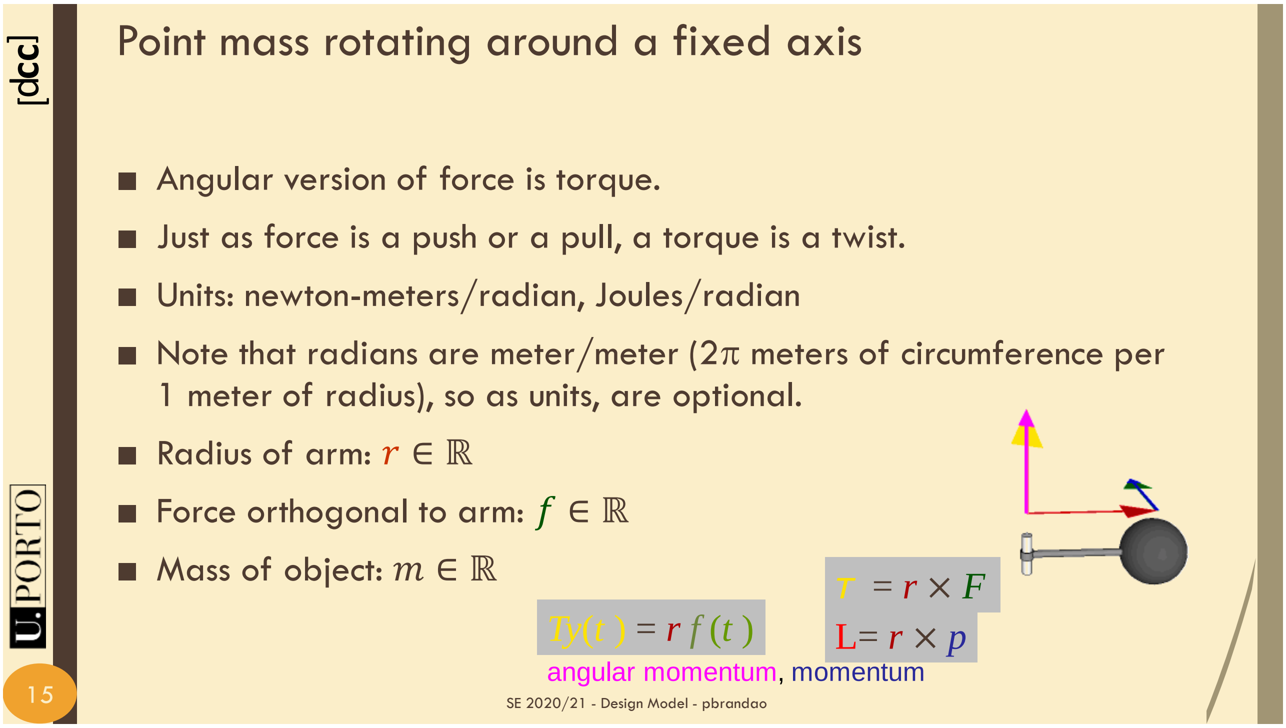
$$\begin{aligned} \mathrm{x}(\mathrm{t}) &= \mathrm{x}(0) + \int_0^t \dot{x}(\tau)d\tau \\ &= \mathrm{x}(0) + \int_0^t [\dot{x}(0) + \int_0^\tau \ddot{x}(\alpha)d\alpha]d\tau \\ &= \mathrm{x}(0) + t\dot{x}(0) + \frac{1}{M}\int_0^t \int_0^\tau F(\alpha)d\alpha d\tau \end{aligned}$$

Orientation

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

$$\theta(t) = \begin{bmatrix} \theta_x(t) \\ \theta_y(t) \\ \theta_z(t) \end{bmatrix} = \begin{bmatrix} roll \\ yaw \\ pitch \end{bmatrix}$$





Point mass rotating around a fixed axis

- Angular version of force is torque.
- Just as force is a push or a pull, a torque is a twist.
- Units: newton-meters/radian, Joules/radian
- Note that radians are meter/meter (2π meters of circumference per 1 meter of radius), so as units, are optional.
- Radius of arm: $r \in \mathbb{R}$
- Force orthogonal to arm: $f \in \mathbb{R}$
- Mass of object: $m \in \mathbb{R}$

$Ty(t) = r\,f(t)$

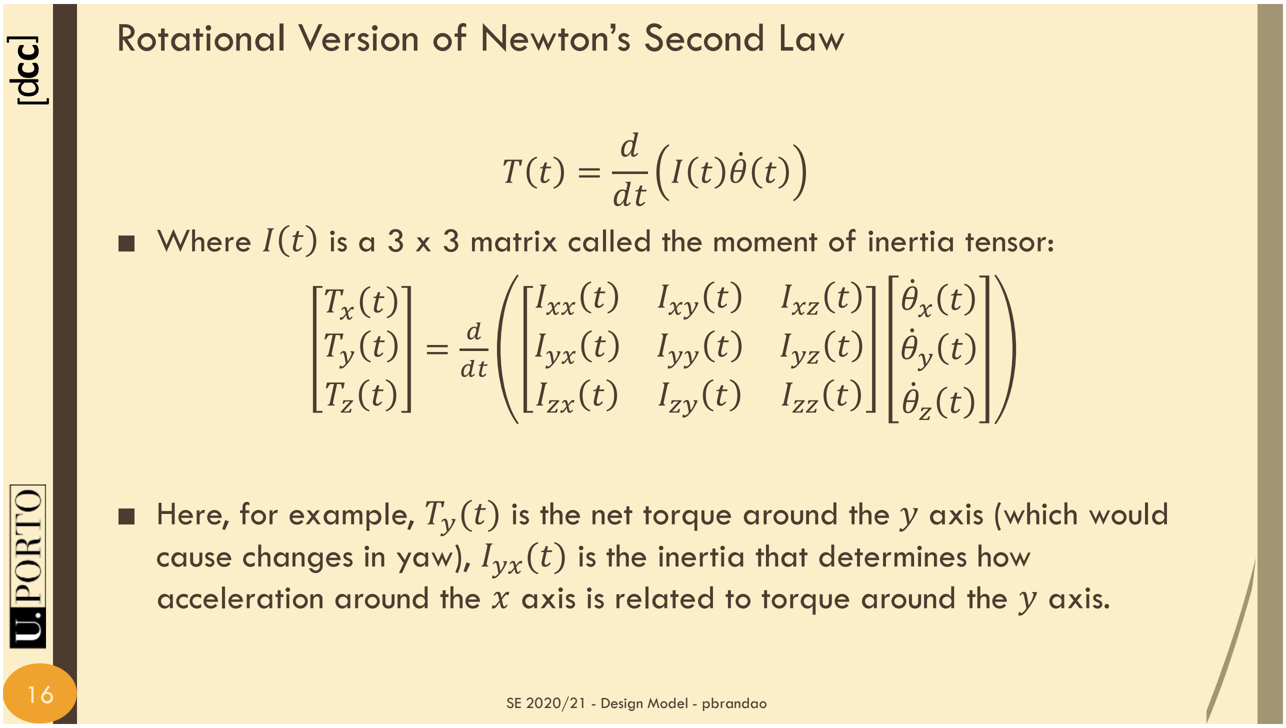
$T = r \times F$

$L = r \times p$

angular momentum, momentum

SE 2020/21 - Design Model - pbrandao

15



Rotational Version of Newton's Second Law

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

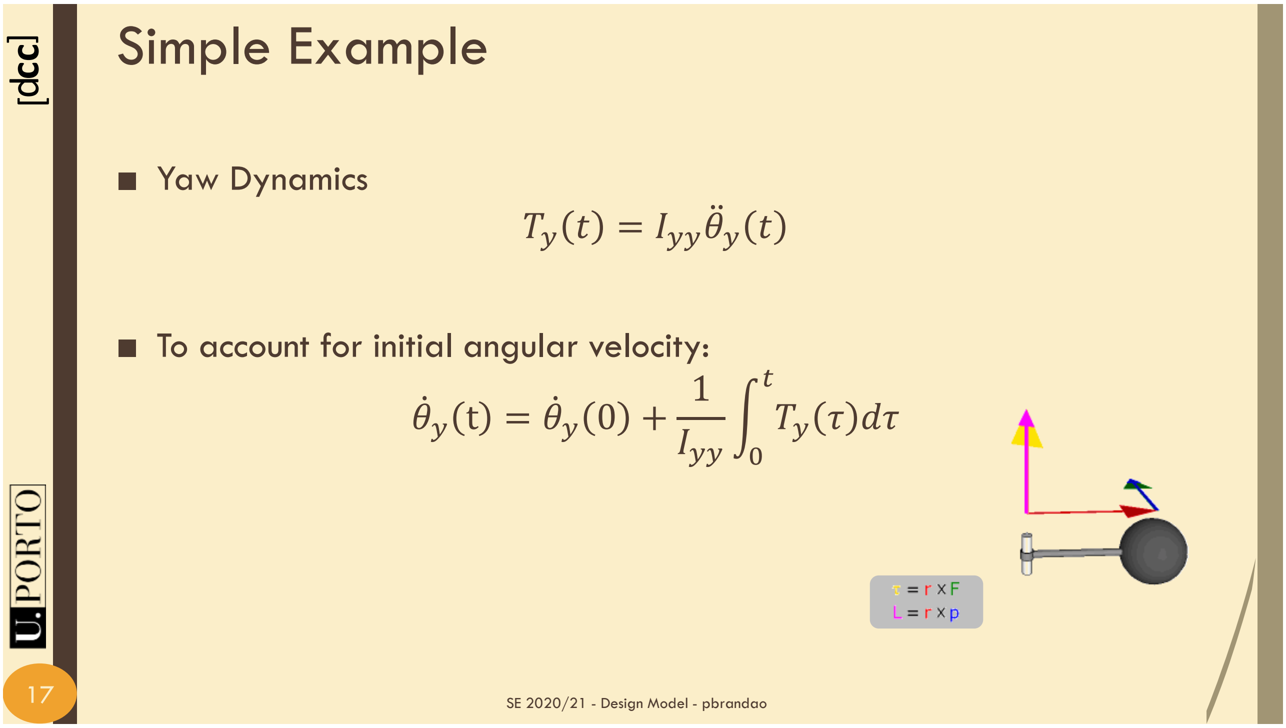
- Where $I(t)$ is a 3 x 3 matrix called the moment of inertia tensor:

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

- 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.

SE 2020/21 - Design Model - pbrandao

16



Simple Example

- Yaw Dynamics

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

- To account for initial angular velocity:

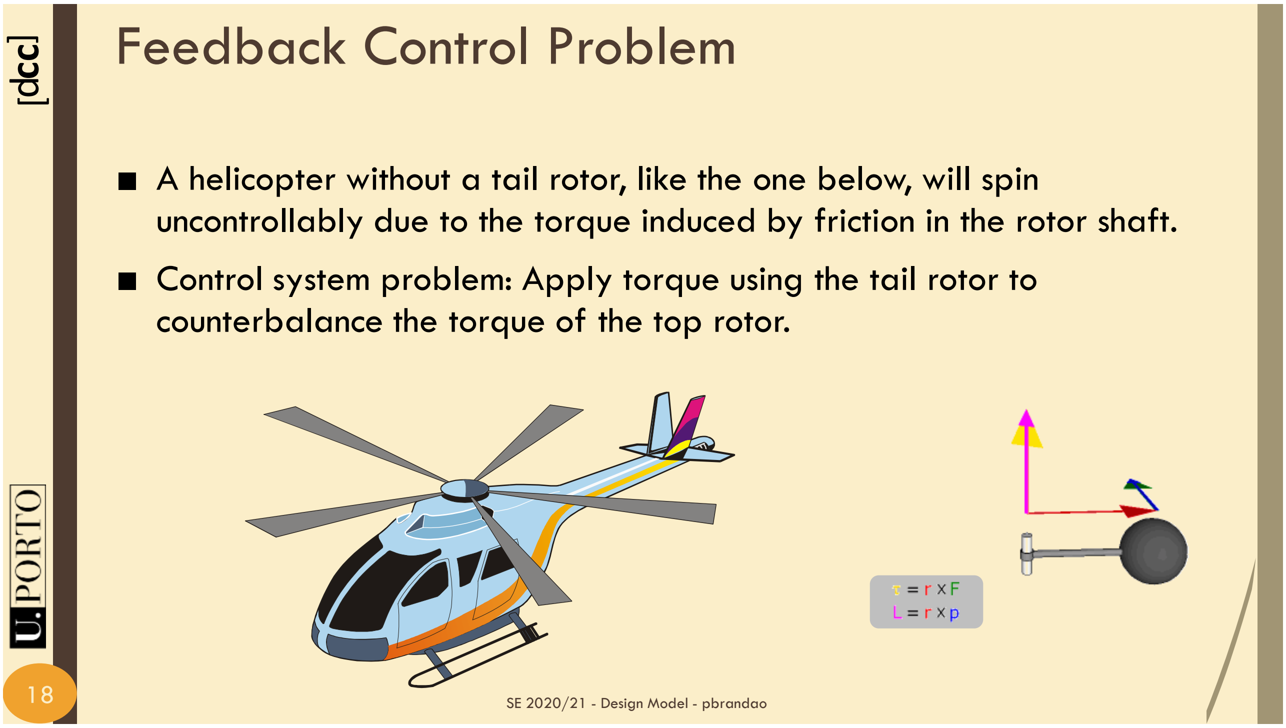
$$\dot{\theta}_y(\mathrm{t}) = \dot{\theta}_y(0) + \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau$$

$\tau = r \times F$
$L = r \times p$

17

SE 2020/21 - Design Model - pbrandao

17



Feedback Control Problem

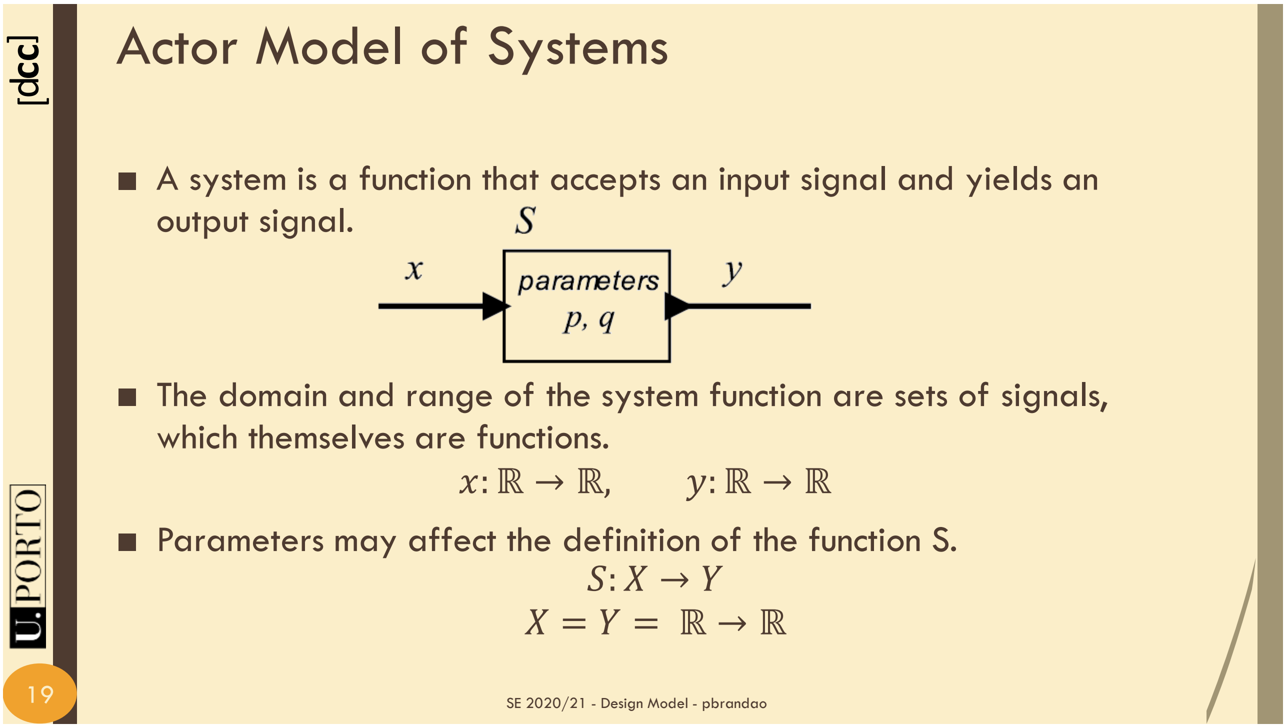
- A helicopter without a tail rotor, like the one below, will spin uncontrollably due to the torque induced by friction in the rotor shaft.
- Control system problem: Apply torque using the tail rotor to counterbalance the torque of the top rotor.

$\tau = r \times F$
$L = r \times p$

18

SE 2020/21 - Design Model - pbrandao

18



Actor Model of Systems

- A system is a function that accepts an input signal and yields an output signal.

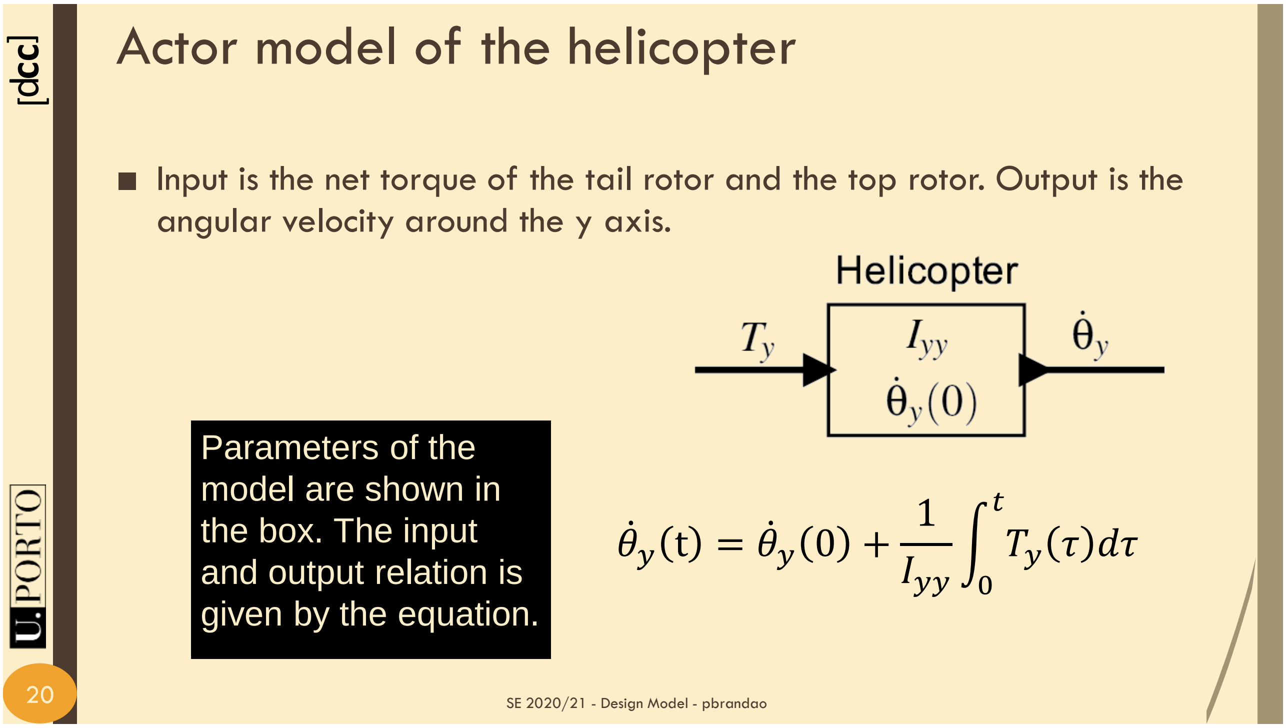
- The domain and range of the system function are sets of signals, which themselves are functions.

$$x: \mathbb{R} \to \mathbb{R}, \qquad y: \mathbb{R} \to \mathbb{R}$$

- Parameters may affect the definition of the function S.

$$S: X \to Y$$
$$X = Y = \mathbb{R} \to \mathbb{R}$$

19



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.

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

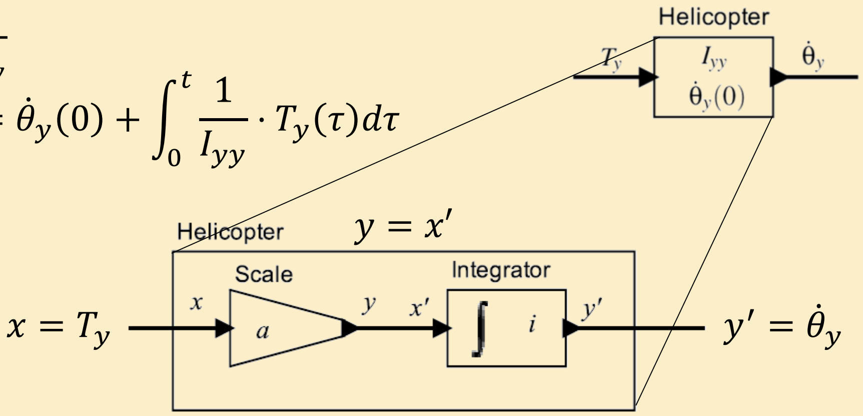
20

Composition of actor models

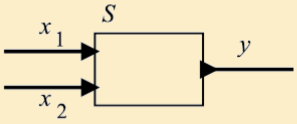
$i = \dot{\theta}_y(0)$ (initial value of integration)

$$a = \frac{1}{I_{yy}}$$

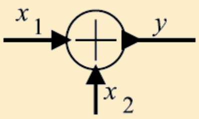
$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \int_0^t \frac{1}{I_{yy}} \cdot T_y(\tau)d\tau$$

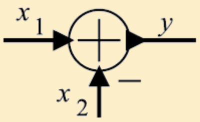


Actor models with multiple inputs

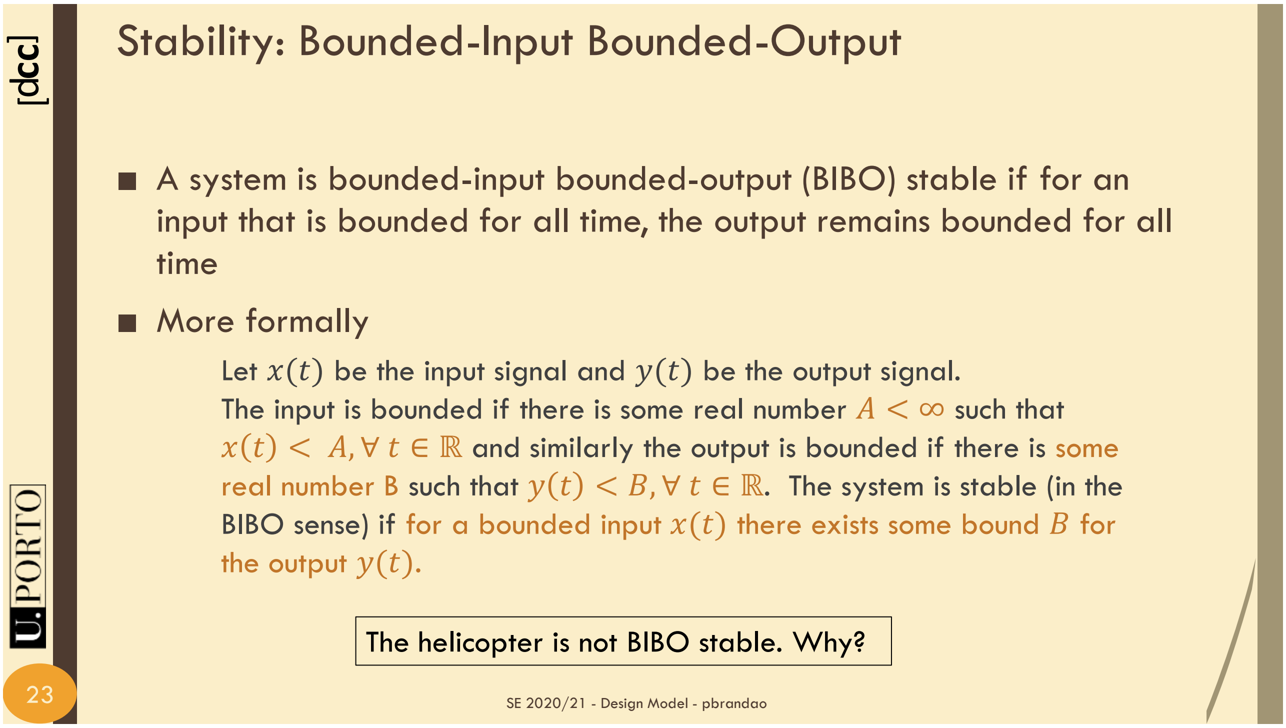


$S: (\mathbb{R} \to \mathbb{R})^2 \to (\mathbb{R} \to \mathbb{R})$





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



[dcc]
U.PORTO
23

Stability: Bounded-Input Bounded-Output

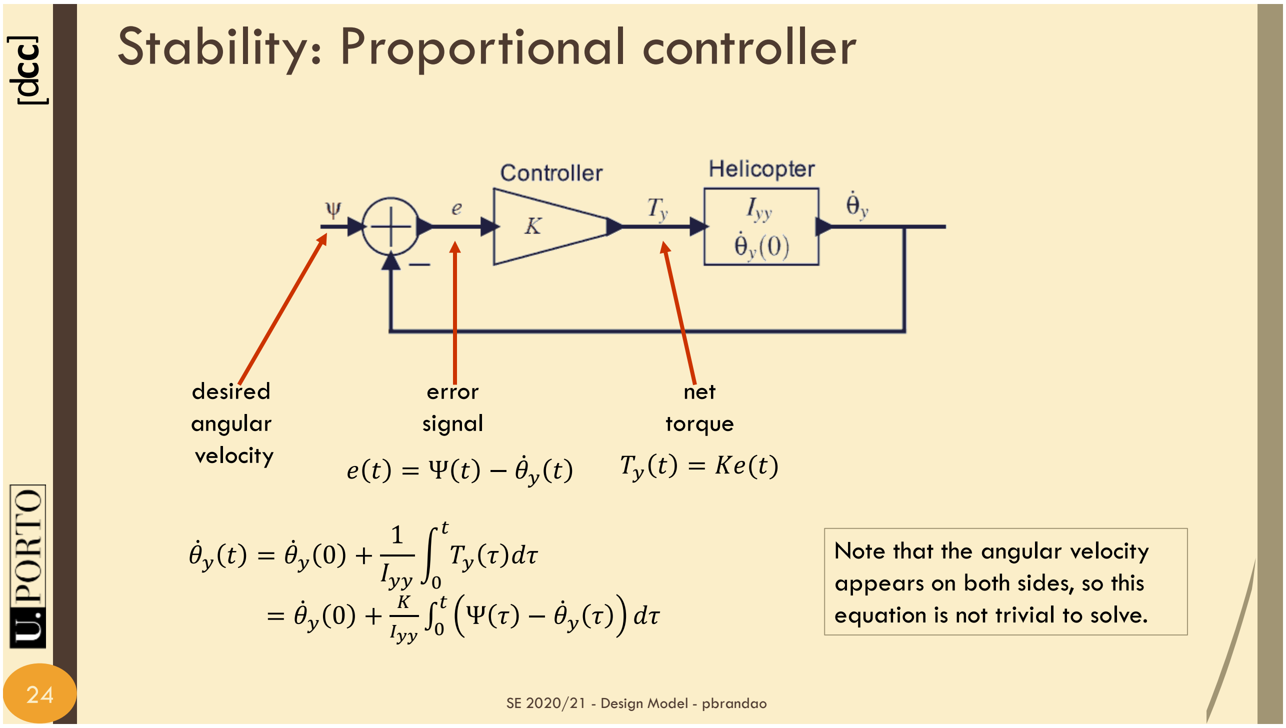
- A system is bounded-input bounded-output (BIBO) stable if for an input that is bounded for all time, the output remains bounded for all time
- More formally

Let $x(t)$ be the input signal and $y(t)$ be the output signal. The input is bounded if there is some real number $A < \infty$ such that $x(t) < \ A, \forall\ t \in \mathbb{R}$ and similarly the output is bounded if there is some real number B such that $y(t) < B, \forall\ t \in \mathbb{R}$. The system is stable (in the BIBO sense) if for a bounded input $x(t)$ there exists some bound B for the output $y(t)$.

The helicopter is not BIBO stable. Why?

SE 2020/21 - Design Model - pbrandao

23



Stability: Proportional controller

desired angular velocity

error signal

net torque

$e(t) = \Psi(t) - \dot{\theta}_y(t)$ $\quad T_y(t) = Ke(t)$

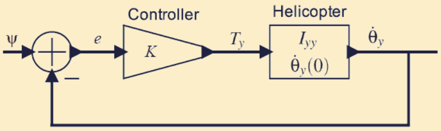
$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau$$
$$= \dot{\theta}_y(0) + \frac{K}{I_{yy}}\int_0^t \Big(\Psi(\tau) - \dot{\theta}_y(\tau)\Big)\, d\tau$$

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

SE 2020/21 - Design Model - pbrandao

24

Behaviour of the controller



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

Desired angular velocity: $\Psi(\tau) = 0$

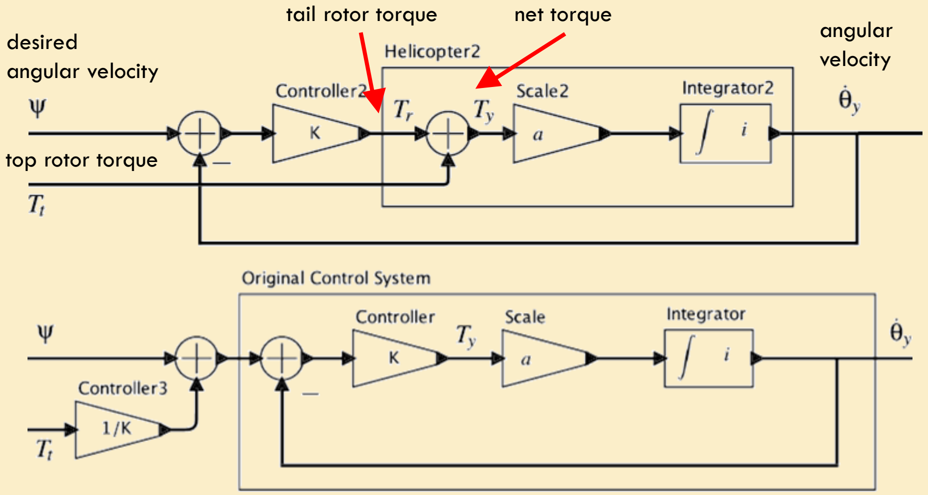
Simplifies differential equation to:

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

Which can be solved as follows (see textbook):

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

More Realistic Scenario

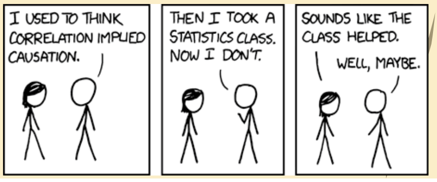


System Properties – Causality

- The **current output** can only depend on **the current and past inputs**
- More formally:

Let $x|_{t\leq\tau}$ be an input that has values only at times $t \leq \tau$
(this is called a restriction in time)

IF $x_1|_{t\leq\tau} = x_2|_{t\leq\tau}$
THEN a system S for which x_1 and x_2 are valid inputs
is causal if and only if $\mathrm{S}(x_1)|_{t\leq\tau} = \mathrm{S}(x_2)|_{t\leq\tau}$


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Source: XKCD -Correlation

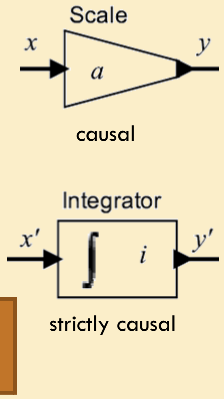
System Properties – Strict Causality

Strict causality $\Longrightarrow$ causality

- The **current output** can only depend on **the past inputs**
- More formally:

Let $x|_{t<\tau}$ be an input that has values only at times $t < \tau$

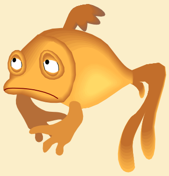
IF $x_1|_{t<\tau} = x_2|_{t<\tau}$
THEN a system S for which x_1 and x_2 are valid inputs
is strictly causal if and only if $\mathrm{S}(x_1)|_{t\leq\tau} = \mathrm{S}(x_2)|_{t\leq\tau}$
Note that the output of $\boldsymbol{S}$ is for times up to **and including** t



For the integrator, the value of x at time τ is irrelevant for $y(\tau)$. So, x_1 can differ from x_2 at time τ.

[dcc]

System Properties – Memoryless



Note however that Fish's memories last for months, say scientists, the Guardian

- The **current output** only depends on the **current input**
- More formally:

Remember that if x is a signal, then x is a function, hence $x(t)$ is the value at time t of this function

Additionally, the system S is also a function and hence $(S(x))(t)$ is the output of the system at t for an input signal x

A system S is memoryless if there is some function $f: A \rightarrow \mathrm{B}$ such that $(S(x))(t) \ = \ f(x(t))$

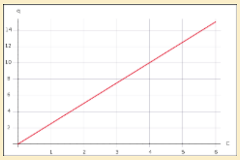
(i.e., the output at t depends on the input at t only)

U.PORTO

SE 2020/21 - Design Model - pbrandao

[dcc]

System Properties – Linearity



- Must satisfy superposition
  - *Additivity*

$$S(x_1 + x_2) = S(x_1) + S(x_2)$$

  - *Homogeneity*

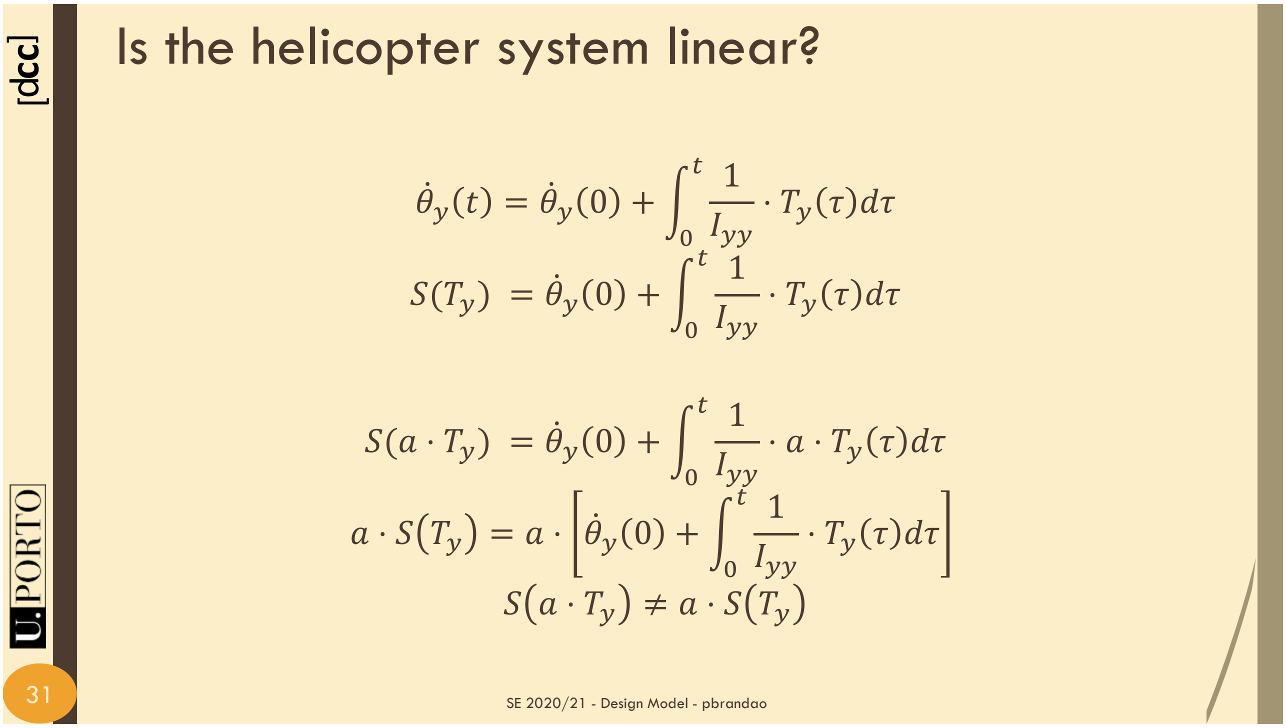
$$S(a \cdot x) = a \cdot S(x)$$

- Superposition

$$S(a \cdot x_1 + b \cdot x_2) = a \cdot S(x_1) + b \cdot S(x_2)$$

U.PORTO

SE 2020/21 - Design Model - pbrandao



Is the helicopter system linear?

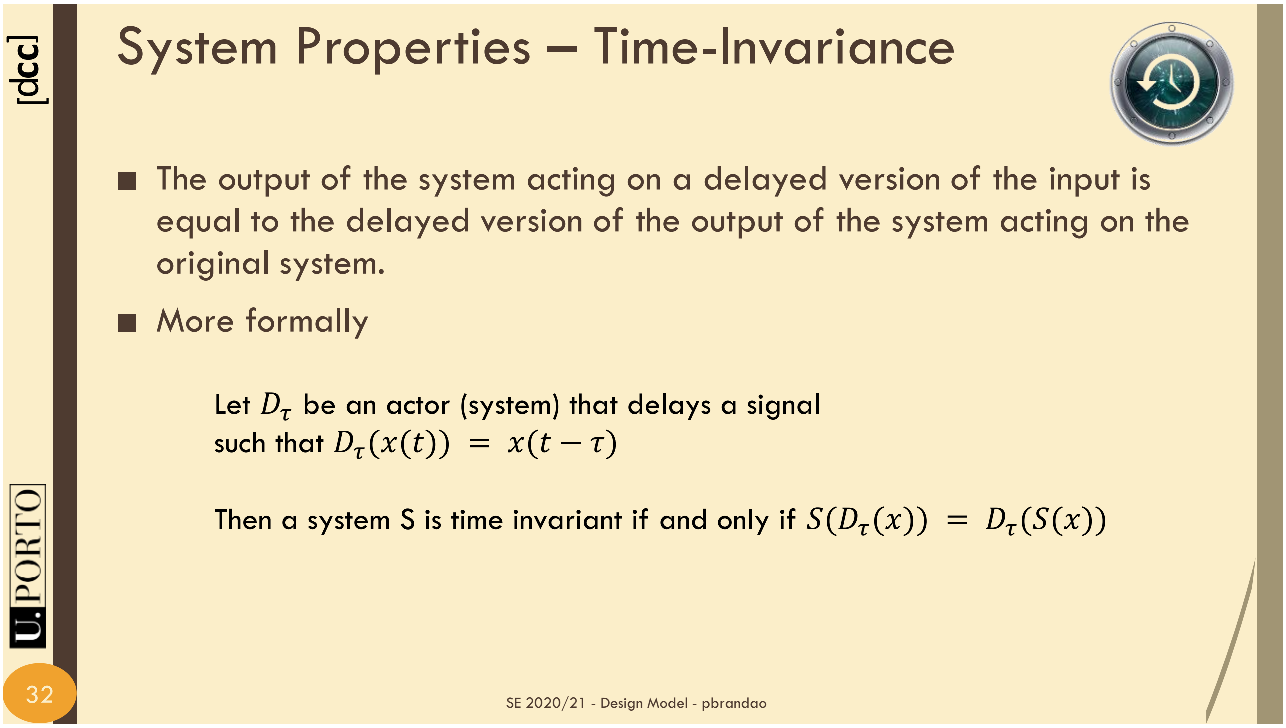
$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \int_0^t \frac{1}{I_{yy}} \cdot T_y(\tau)d\tau$$

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

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

$$a \cdot S(T_y) = a \cdot \left[\dot{\theta}_y(0) + \int_0^t \frac{1}{I_{yy}} \cdot T_y(\tau)d\tau\right]$$

$$S(a \cdot T_y) \neq a \cdot S(T_y)$$

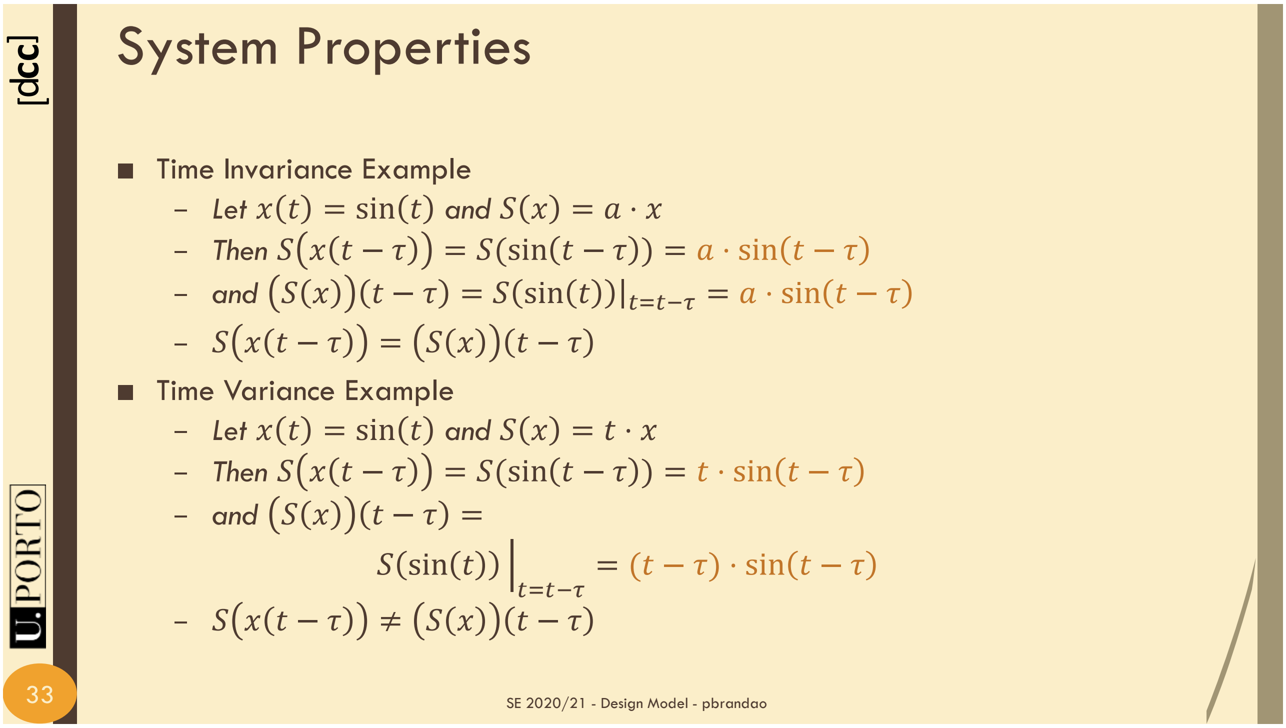


System Properties – Time-Invariance

- The output of the system acting on a delayed version of the input is equal to the delayed version of the output of the system acting on the original system.
- More formally

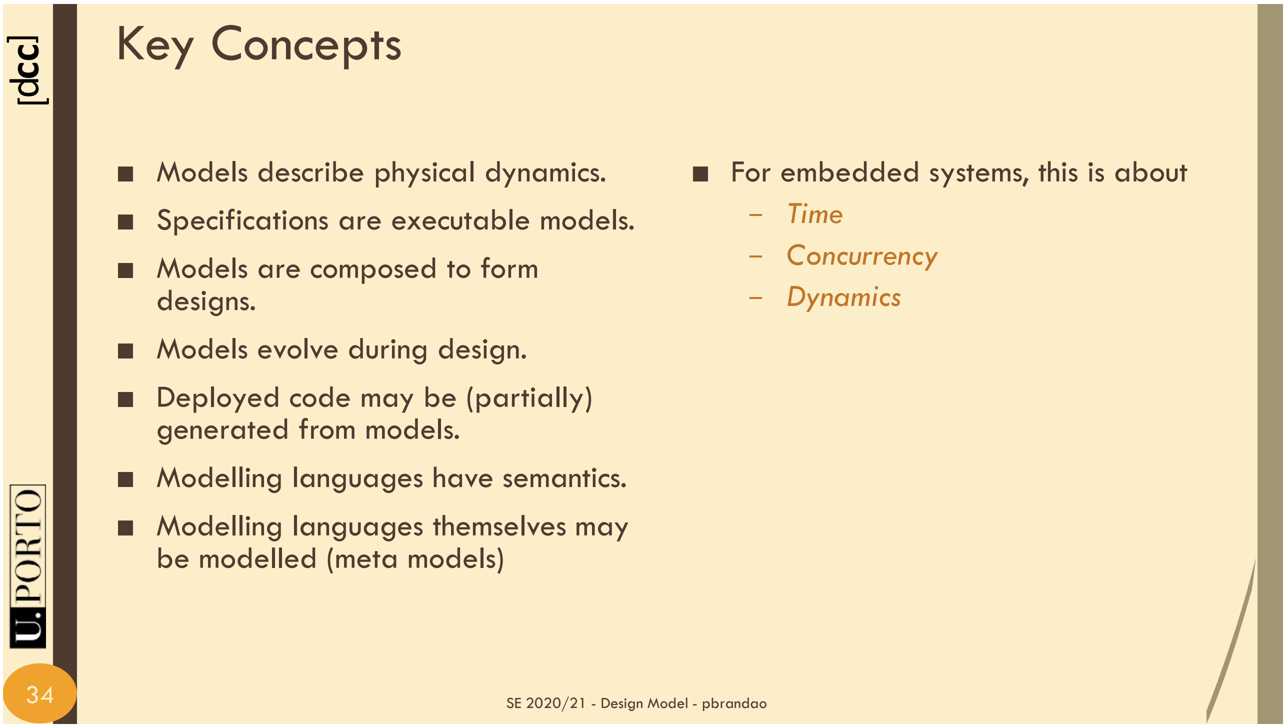
Let D_τ be an actor (system) that delays a signal
such that $D_\tau(x(t)) \ = \ x(t-\tau)$

Then a system S is time invariant if and only if $S(D_\tau(x)) \ = \ D_\tau(S(x))$



System Properties

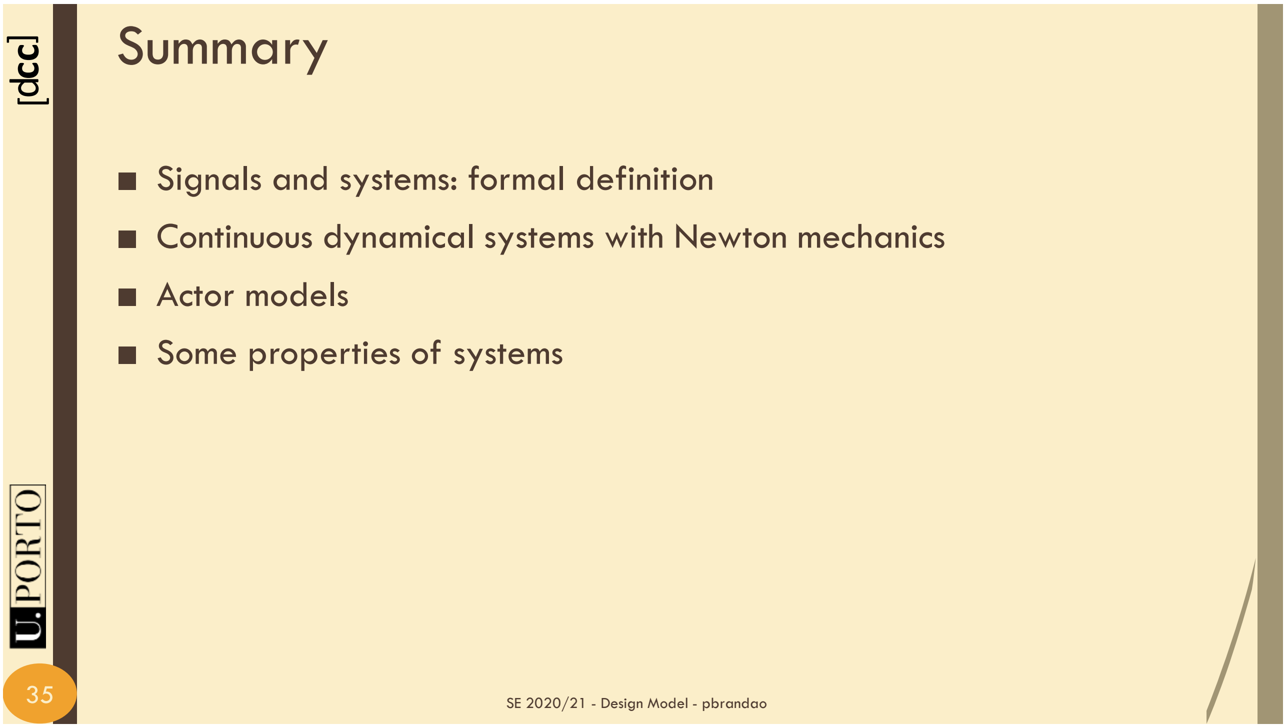
- Time Invariance Example
  - *Let* $x(t) = \sin(t)$ *and* $S(x) = a \cdot x$
  - *Then* $S\big(x(t-\tau)\big) = S(\sin(t-\tau)) = a \cdot \sin(t-\tau)$
  - *and* $\big(S(x)\big)(t-\tau) = S(\sin(t))|_{t=t-\tau} = a \cdot \sin(t-\tau)$
  - $S\big(x(t-\tau)\big) = \big(S(x)\big)(t-\tau)$
- Time Variance Example
  - *Let* $x(t) = \sin(t)$ *and* $S(x) = t \cdot x$
  - *Then* $S\big(x(t-\tau)\big) = S(\sin(t-\tau)) = t \cdot \sin(t-\tau)$
  - *and* $\big(S(x)\big)(t-\tau) =$
    $$S(\sin(t))\Big|_{t=t-\tau} = (t-\tau) \cdot \sin(t-\tau)$$
  - $S\big(x(t-\tau)\big) \neq \big(S(x)\big)(t-\tau)$



Key Concepts

- Models describe physical dynamics.
- Specifications are executable models.
- Models are composed to form designs.
- Models evolve during design.
- Deployed code may be (partially) generated from models.
- Modelling languages have semantics.
- Modelling languages themselves may be modelled (meta models)

- For embedded systems, this is about
  - *Time*
  - *Concurrency*
  - *Dynamics*



Summary

- Signals and systems: formal definition
- Continuous dynamical systems with Newton mechanics
- Actor models
- Some properties of systems