Embedded Systems Design and Modeling

Chapter 2
Modeling
with an Emphasis on
Continuous Dynamics
Part 1

Outline (Part 1)

- Modeling Definition
- Types of Models
- Requirements
- Models of Computation
- Elements of MoC
- Continuous MoC Example

What Is Modeling?

- Modeling is the process of:
  1. specifying what a system's behavior should be given a particular set of inputs
  2. gaining a deeper understanding of a system by imitating its behavior given those inputs
- In the early stages of the design process, models are used mainly for item 1 above
- In the later stages of the design process, models are used mainly for item 2 above

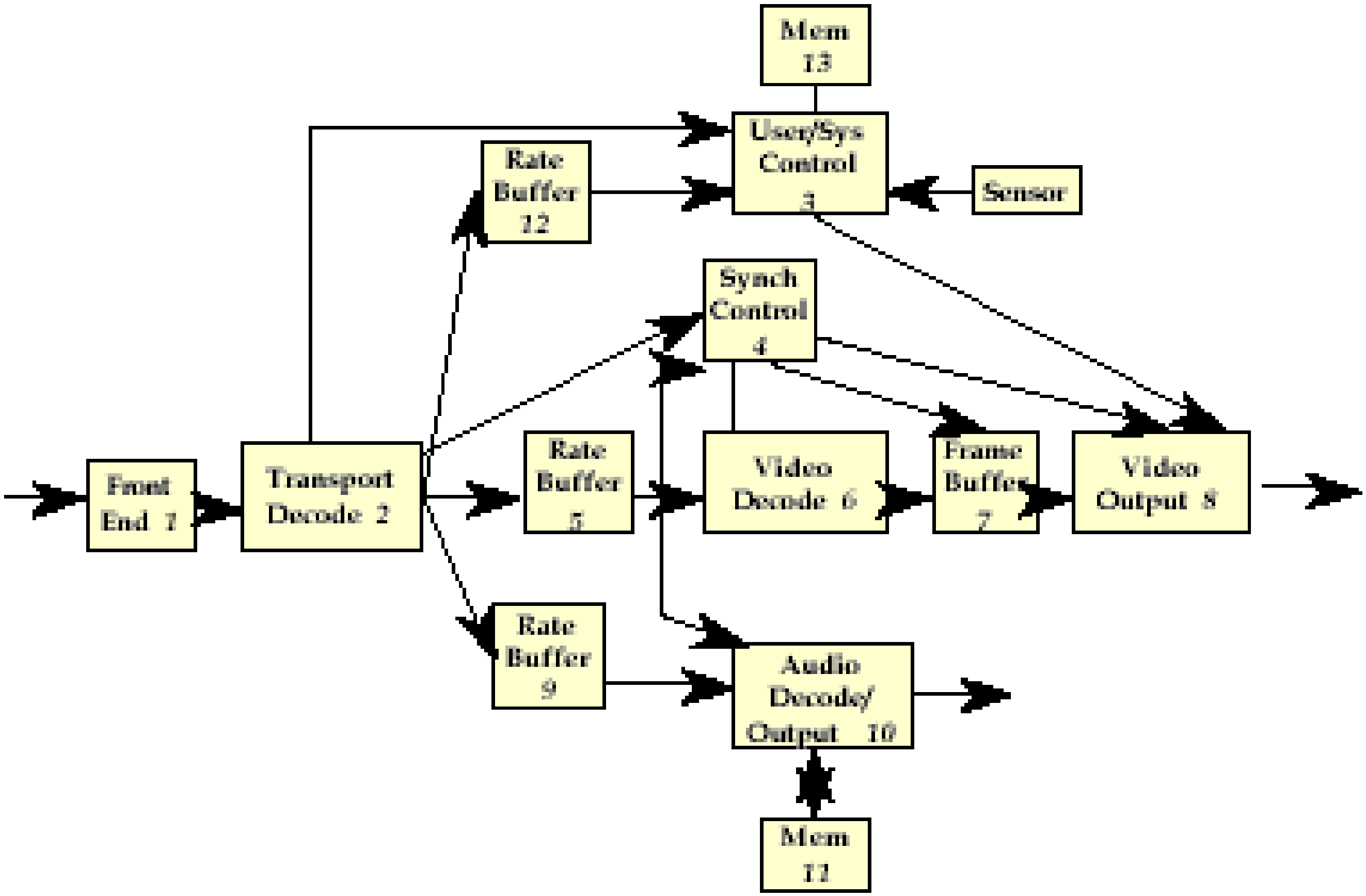
Modeling Principals

- A good model should give us insight about a system, process, or artifact through imitation
- Modeling should lead to systems with predictable performance
- A good model should allow us to evaluate the correctness, performance, and other key characters of a system BEFORE AND AFTER the design.
- Different models:
  - Mathematical: a set of definitions and mathematical relations
  - Structural, behavioral, etc.

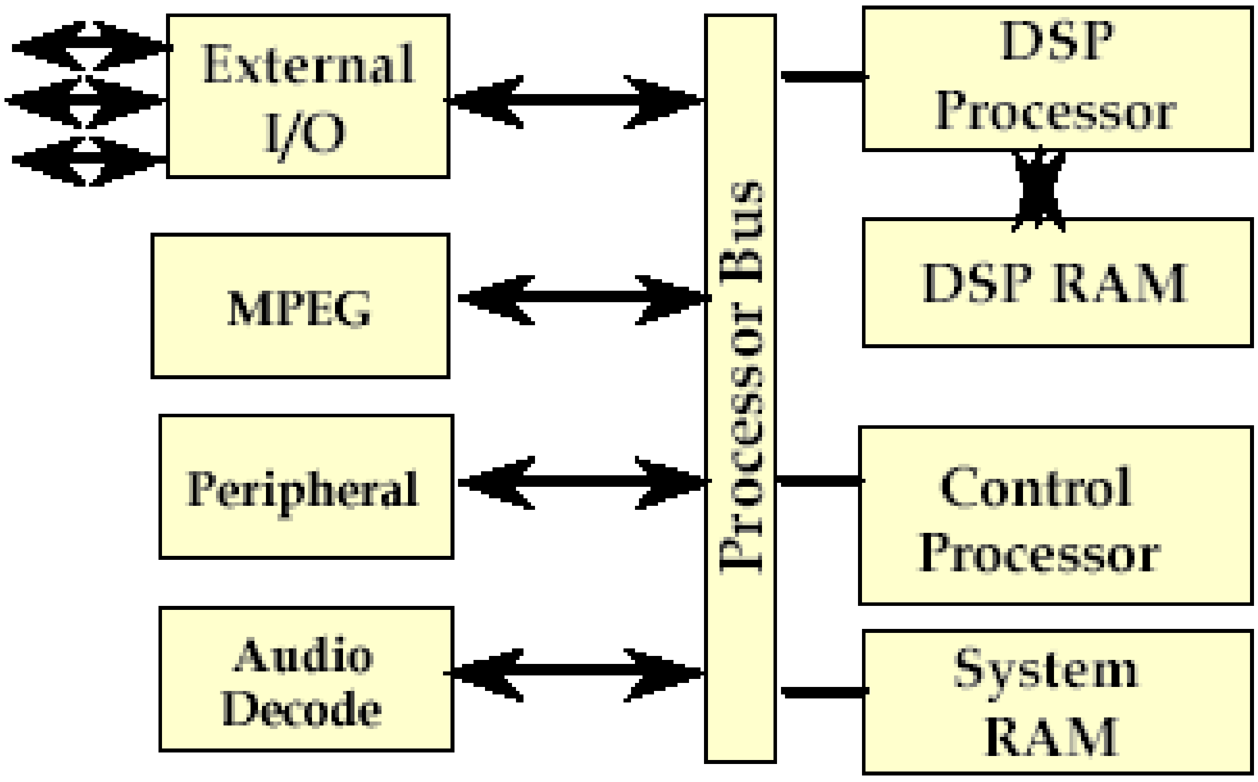
Abstractions and Modeling

- Modeling has been the foundation of scientific research and engineering practice
- Two general types
  1. System behavior
     - Functional specification of components
     - No distinction between HW or SW
     - No specific type of communication among components
  2. System structure
     - Implementation details <u>partially</u> considered
     - Coordinates communications among computational components

System Behavioral Model Example



System Structural Model Example



Modeling Challenges

- Finding most suitable models (or MoC's) that can describe all aspects of a cyberphysical system accurately
- Modeling both hardware and software elements of a system simultaneously (HW/SW cosimulation)
- Modeling discrete time and continuous time aspects of a system accurately
- How to combine various MoC's

Good Model's Requirements

1. Concrete representation of knowledge and ideas about the system being developed
2. Deliberately omits details (abstraction) but concretely represents certain properties to be analyzed, understood and verified
3. Precise and unambiguous semantics:
   - Implicit or explicit relations: inputs, outputs and (possibly) state variables
   - Cost functions
   - Constraints

Modeling of Embedded Systems

- In embedded systems, modeling functional behavior (what the system does) is not sufficient
- Also need to know properties such as:
  - Dynamic vs. static processes
  - Flat vs. nested
  - Asynchronous vs. synchronous
  - Blocking vs. non-blocking
  - Shared memory vs. message passing
  - Relation with the physical world: time
  - Physical: size, weight, power, etc.

What Is Model-Based Design?

1. To create a mathematical model of all parts of an embedded system such as:
   - Physical world
   - Control system
   - SW environment
   - HW platform
   - Network
   - Sensors and actuators
2. Construct an implementation from the model
   - Ultimate goal is to automate this process, similar to a compiler
   - In reality, only portions are automated

Model of Computation

- An essential part of model-based design is to have a good model of computation (MoC)
- Definition:
  - A mathematical description that has a syntax (rules of notations) and a semantics (meanings of notations)
  - The semantics specifies the system behavior (computations and concurrency)
  - The syntax has to be unambiguous and compositional

MoC Examples

- Continuous physical phenomena: ordinary differential equations (ODE)
  - Almost always there is a smooth transition such as motion, heat transfer, etc.
  - The concepts of linearity, time invariance, continuity, stability, etc. can be defined
- Abrupt (non-smooth) transitions: modal models such as FSM's, automaton
  - Usually discrete and instantaneous events
- Cyber-physical behavior: hybrid models

Elements of MoCs

- Describes a system consisting of components
  - What is each component?
  - What knowledge do components share?
  - How do they communicate?
  - What do they communicate?
- MoCs allow
  - Global optimization of computation and communication
  - Scheduling and communication that is correct by construction

Elements of MoCs (Cont’d)

- Have associated language(s) and use different tools
  - Examples: SystemC, Matlab, LabView
- Synthesize all or part of the design at that level of abstraction
- Verify correctness of the functional specification w.r.t. to the given properties at each level of abstraction
- Handle BOTH data and control flows

Data vs. Control Flow

- Fuzzy distinction, yet useful for:
  - specification (language, model, ...)
  - synthesis (scheduling, optimization, ...)
  - validation (simulation, formal verification, ...)
- Control flow:
  - deals with data arrival time instead of value
  - may not know when exactly data arrives (at higher levels of abstraction, or in real world app's)
- Data flow:
  - data value matters more than timing and format (burst, continuous stream, tokens, etc.)

Methods Emphasize on …

- For control:
  - event/reaction relation
  - response time (Real-Time scheduling for deadline satisfaction)
  - priority among events and processes
- For data:
  - functional dependency between input and output
  - memory/time efficiency (e.g. Dataflow scheduling for efficient pipelining)

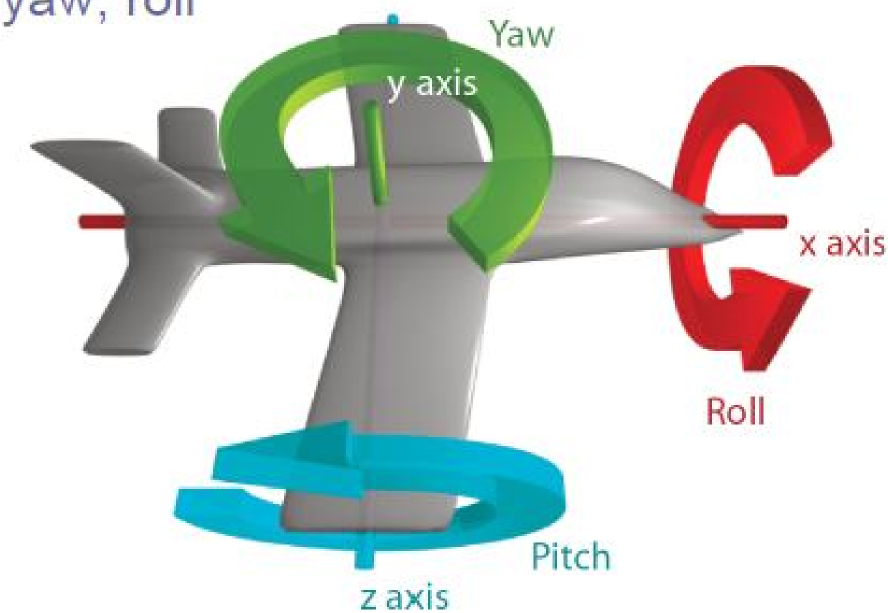
Modeling Continuous Dynamics

- A simple mechanical example:

Modeling Physical Motion

Six degrees of freedom:

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



Continuous Dynamics Example

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 $\mathbf{x}(t) \in \mathbb{R}^3$.

Helicopter Example (Cont'd)

Notation

Velocity

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

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

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

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

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

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

Helicopter Example (Cont'd)

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

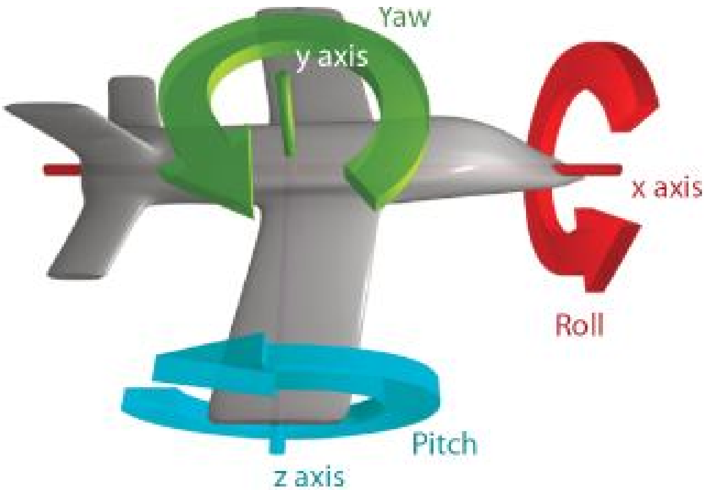
$$\begin{aligned}
\mathbf{x}(t) &= \mathbf{x}(0) + \int\limits_0^t \dot{\mathbf{x}}(\tau)d\tau \\
&= \mathbf{x}(0) + t\dot{\mathbf{x}}(0) + \frac{1}{M}\int\limits_0^t \int\limits_0^\tau \mathbf{F}(\alpha)d\alpha d\tau,
\end{aligned}$$

Helicopter Example (Cont'd)

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: $\mathbf{T} \colon \mathbb{R} \to \mathbb{R}^3$

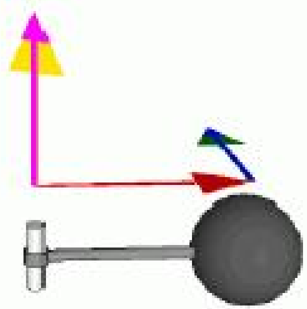
$$\theta(t) = \begin{bmatrix} \theta_x(t) \\ \theta_y(t) \\ \theta_z(t) \end{bmatrix} = \begin{bmatrix} \text{roll} \\ \text{yaw} \\ \text{pitch} \end{bmatrix}$$



Helicopter Example (Cont'd)

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

Helicopter Example (Cont'd)

Rotational Version of Newton's Second Law

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

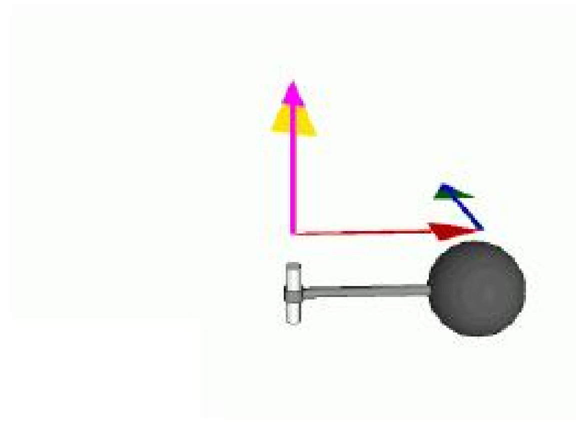
where $I(t)$ is a 3×3 matrix called the moment of inertia tensor.

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

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.

Helicopter Example (Cont'd)

Simple Example



Yaw dynamics:

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

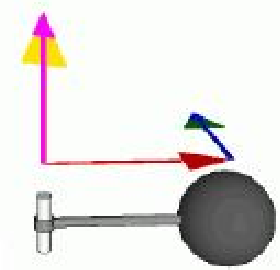
To account for initial angular velocity, write as

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

Helicopter Example: Problem Definition

- So far we have described the computational relations between components of the system's dynamics
- Next we need an objective function

Feedback Control Problem



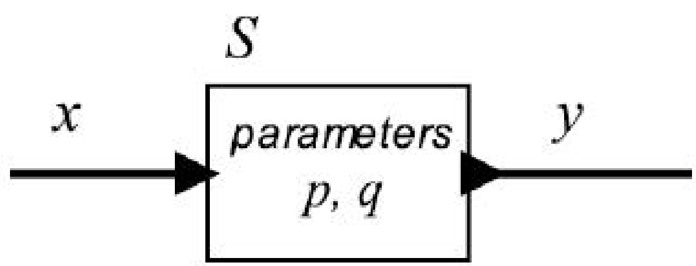
A helicopter without a tail rotor, like the one below, will spin uncontrollably due to the torque induced by friction in the rotor shaft.

Control system problem: Apply torque using the tail rotor to counterbalance the torque of the top rotor.

Actor Model

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.

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

$$S: X \to Y$$

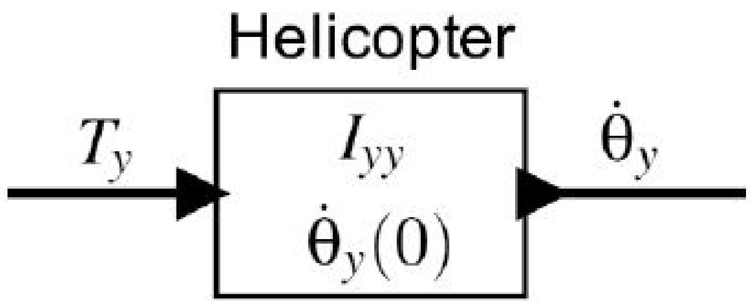
$$X = Y = (\mathbb{R} \to \mathbb{R})$$

Parameters may affect the definition of the function *S*.

Actor Model of Helicopter

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.

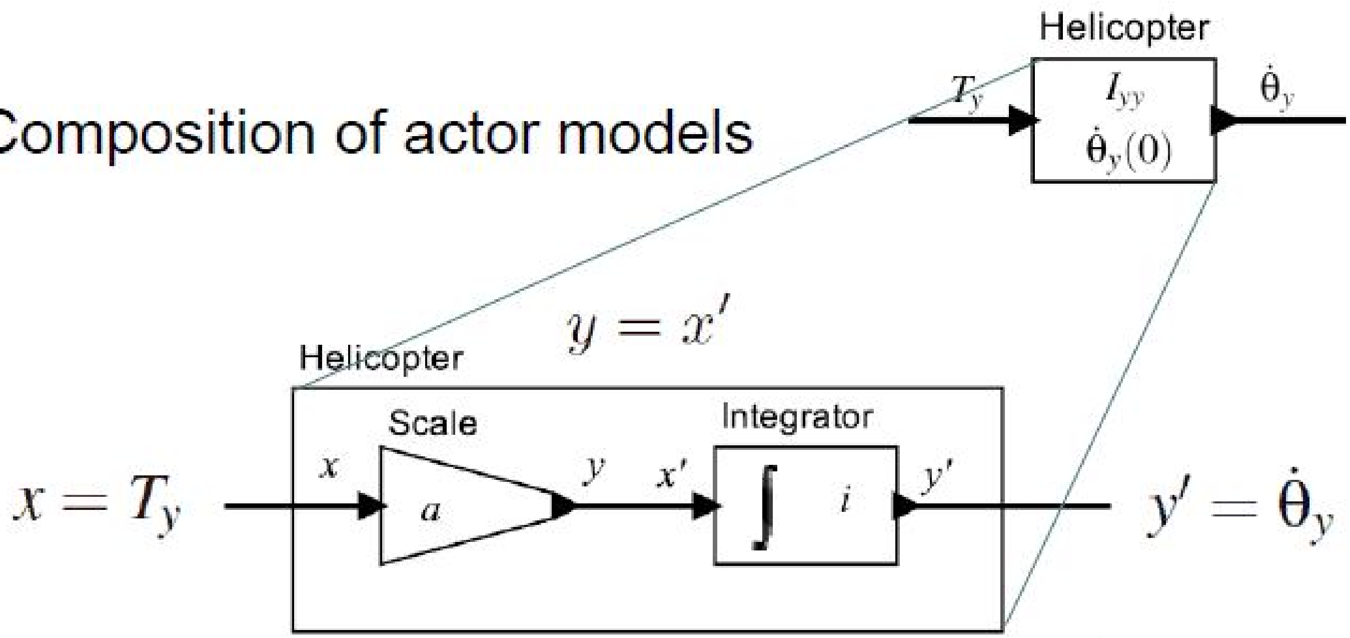


Parameters of the model are shown in the box. The input and output relation is given by the equation to the right.

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

Actor Models Composition

Composition of actor models

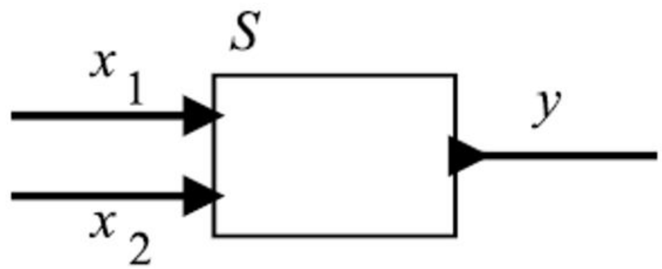


$$\forall t \in \mathbb{R}, \quad y(t) = ax(t) \qquad y'(t) = i + \int_0^t x'(\tau)d\tau$$

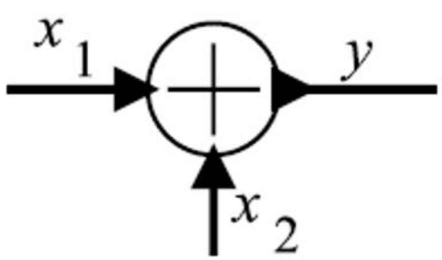
$$y = ax$$

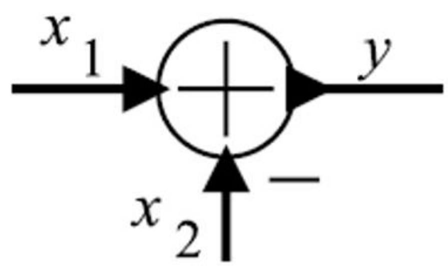
$$a = 1/I_{yy} \qquad\qquad i = \dot{\theta}_y(0)$$

Actor Models with Multiple Inputs



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

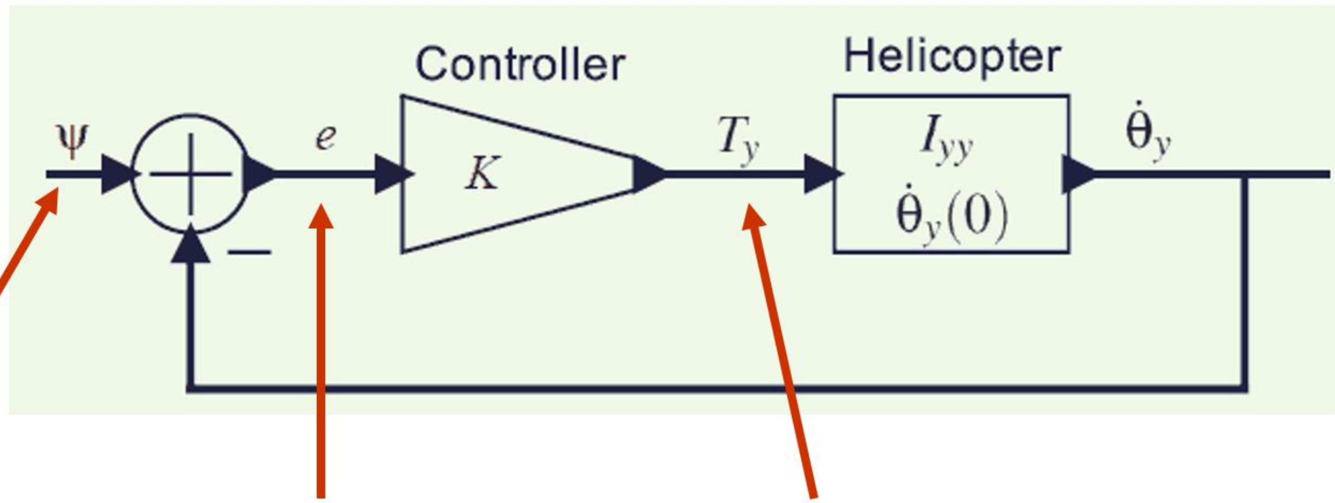




$$\forall\, t \in \mathbb{R}, \quad y(t) = x_1(t) + x_2(t)$$

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

Feedback Control Model



desired angular velocity

error signal

net torque

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

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