An Introduction to PID Control for Drones

Sebastian Schlecht, CDTM Deep Learning Drones

10.08.2016

First things first: Rigid body dynamics

Newton's laws:

1. An object either remains at rest or continues to move at a constant velocity, unless acted upon by a net force.

2. The vector sum of the forces **F** on an object is equal to the mass **m** of that object multiplied by the acceleration vector **a** of the object.

3. When one body exerts a force on a second body, the second body simultaneously exerts a force equal in magnitude and opposite in direction on the first body.

Source: https://en.wikipedia.org/wiki/Newton%27s_laws_of_motion

First things first: Rigid body dynamics

Some basic formulae:

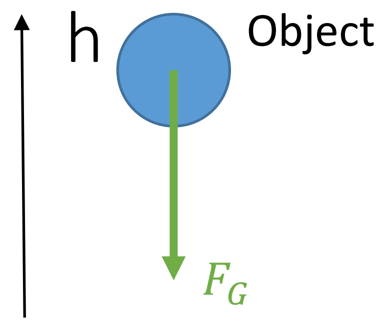
$$F(t) = \mathrm{m} * \mathrm{a(t)}$$

$$x(t) = x_0 + \int v(t)dt \quad \boldsymbol{or} \quad x_0 + vt = x_0 + \frac{1}{2}at^2 \quad for\ v, a\ const.$$

$$v(t) = v_0 + \int a(t)dt = v_0 + at \quad for\ a\ const.$$

Describing an object in space

Example: Free fall (without friction)



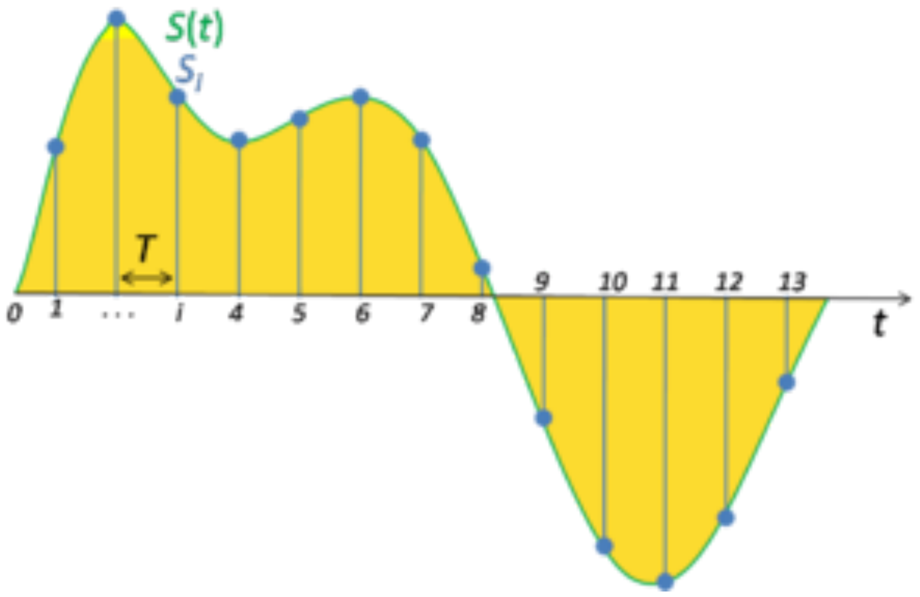
Initial condition: object at height **h** = 10 m, velocity **v** = 0 m/s, mass **m** = 1 kg

Thus:

$$h(t) = h_0 + \int v(t)dt = 10m + vt = 10m + \frac{1}{2}at^2 = 10m - \frac{1}{2} * 9{,}81\frac{m}{s^2} * t^2$$

Putting it into software needs discretization

Digital systems have to discretize our continuous world in order to be able to work with it.

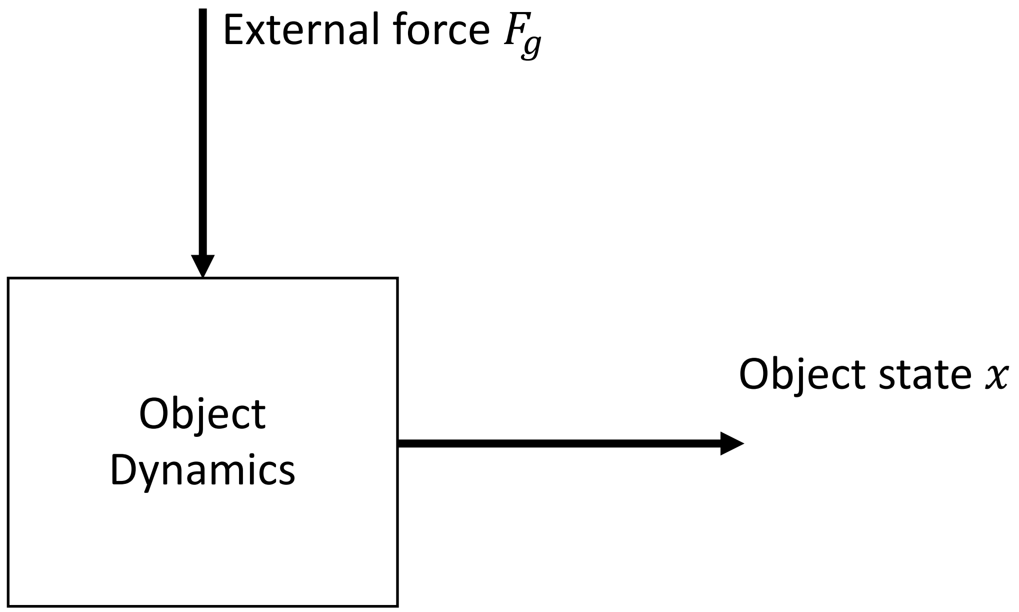


Source: https://en.wikipedia.org/wiki/Sampling_(signal_processing)

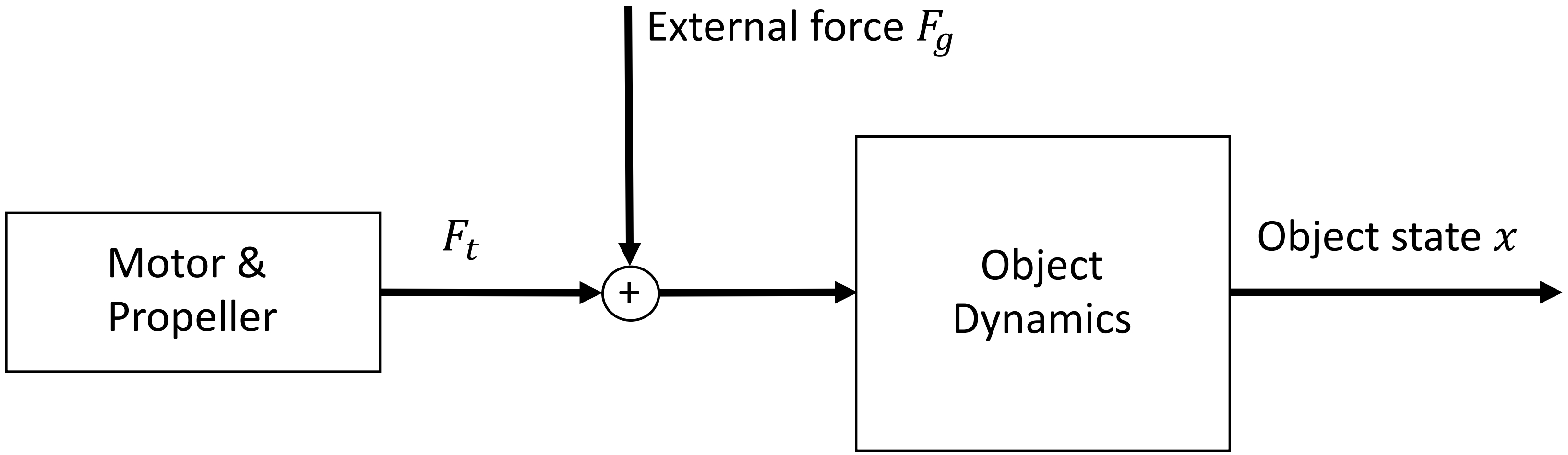
MATLAB Module #1: Free fall

MATLAB

A simple model for a falling object

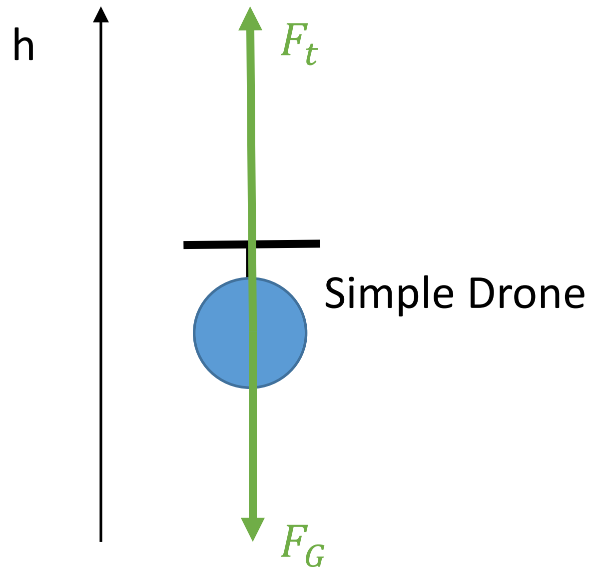


Towards Drones: Adding a propeller



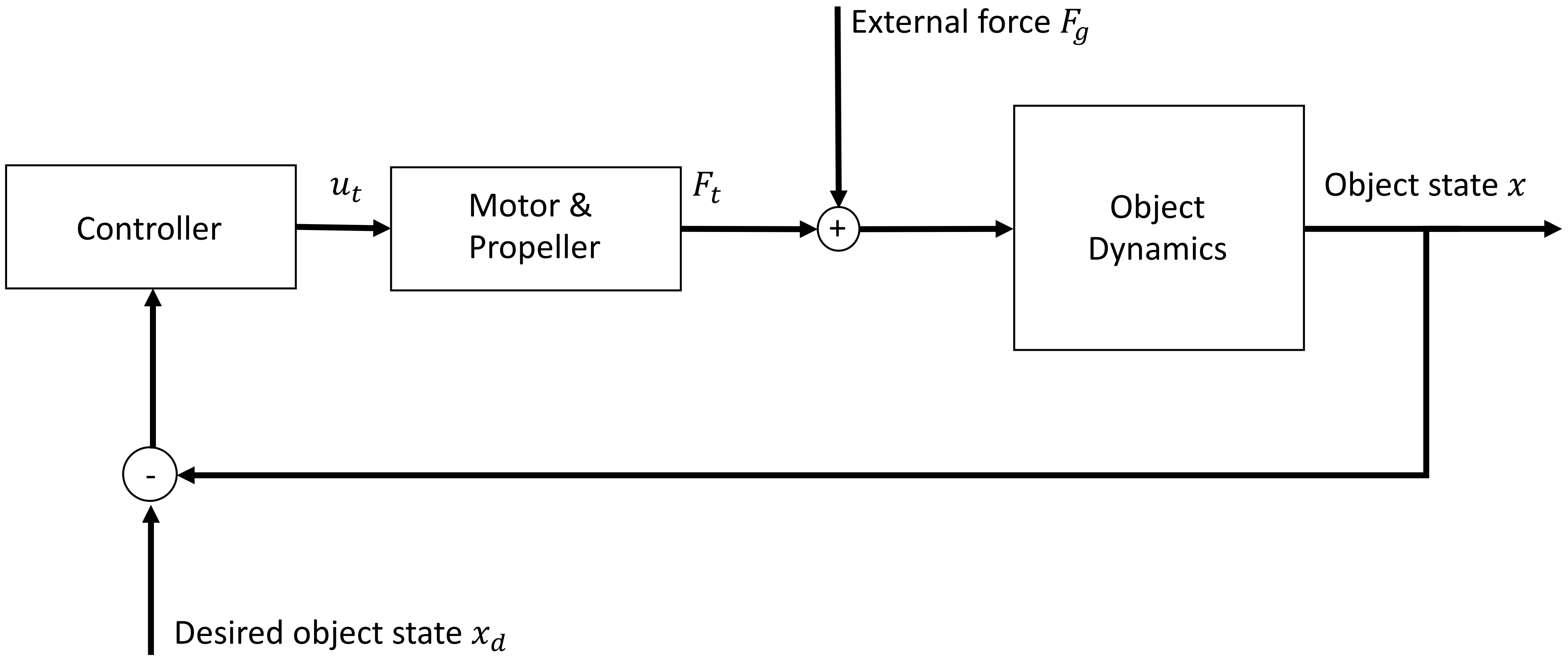
Actuators

Example: Simple Drone



MATLAB Module #2: Simple Drone Model

Towards Drones: Adding a controller



Towards Drones: Adding a P controller

Proportional control (P-Control) is the most basic way to control a system.

Control Law: $\boldsymbol{u_t = K_P(x_d \; - x)}$

MATLAB Module #3: P-Control

Towards Drones: Adding a PD controller

To stabilize the body at the desired location, we can insert a D-Control unit which “damps” movement.

Control Law: $\boldsymbol{u_t} = \boldsymbol{K_P}(\boldsymbol{x_d} - \boldsymbol{x}) + \boldsymbol{K_D}(\dot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{\boldsymbol{x}})$ with $\dot{x}_d$ often set to 0.

MATLAB Module #4: PD-Control

Towards Drones: Adding a PID controller

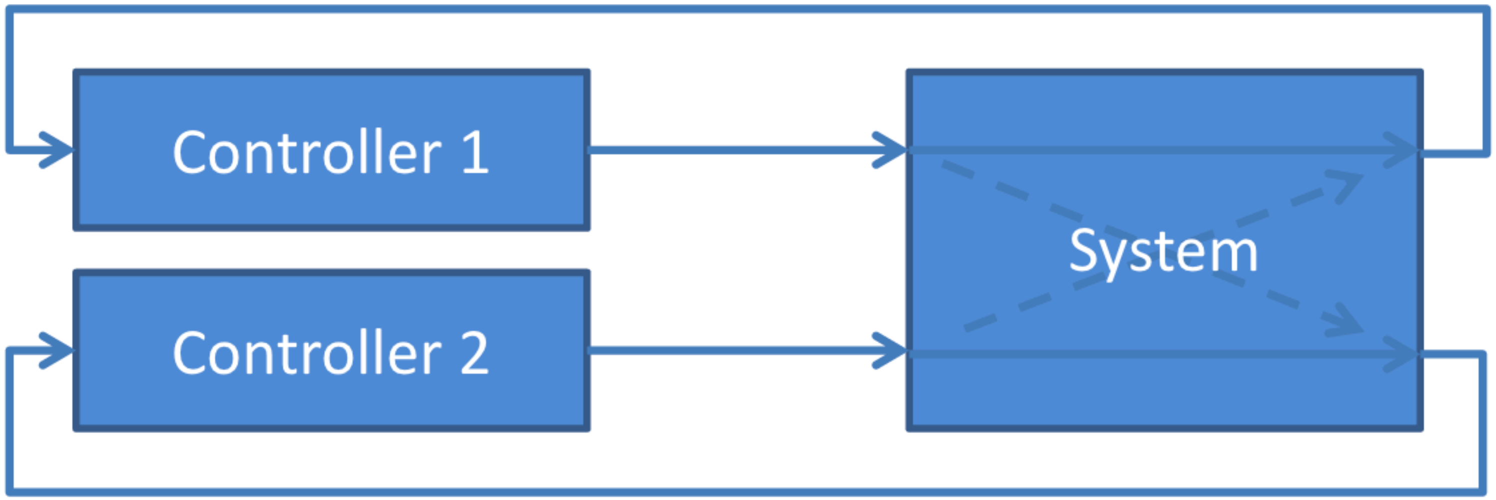
To counter constant error or offset, we can insert an integrative part.

Control Law: $\boldsymbol{u_t = K_P(x_d - x) + K_D(\dot{x_d} - \dot{x}) + K_I \int x_d - x \, dt}$

MATLAB Module #5: PID-Control

How to handle multiple inputs/outputs?

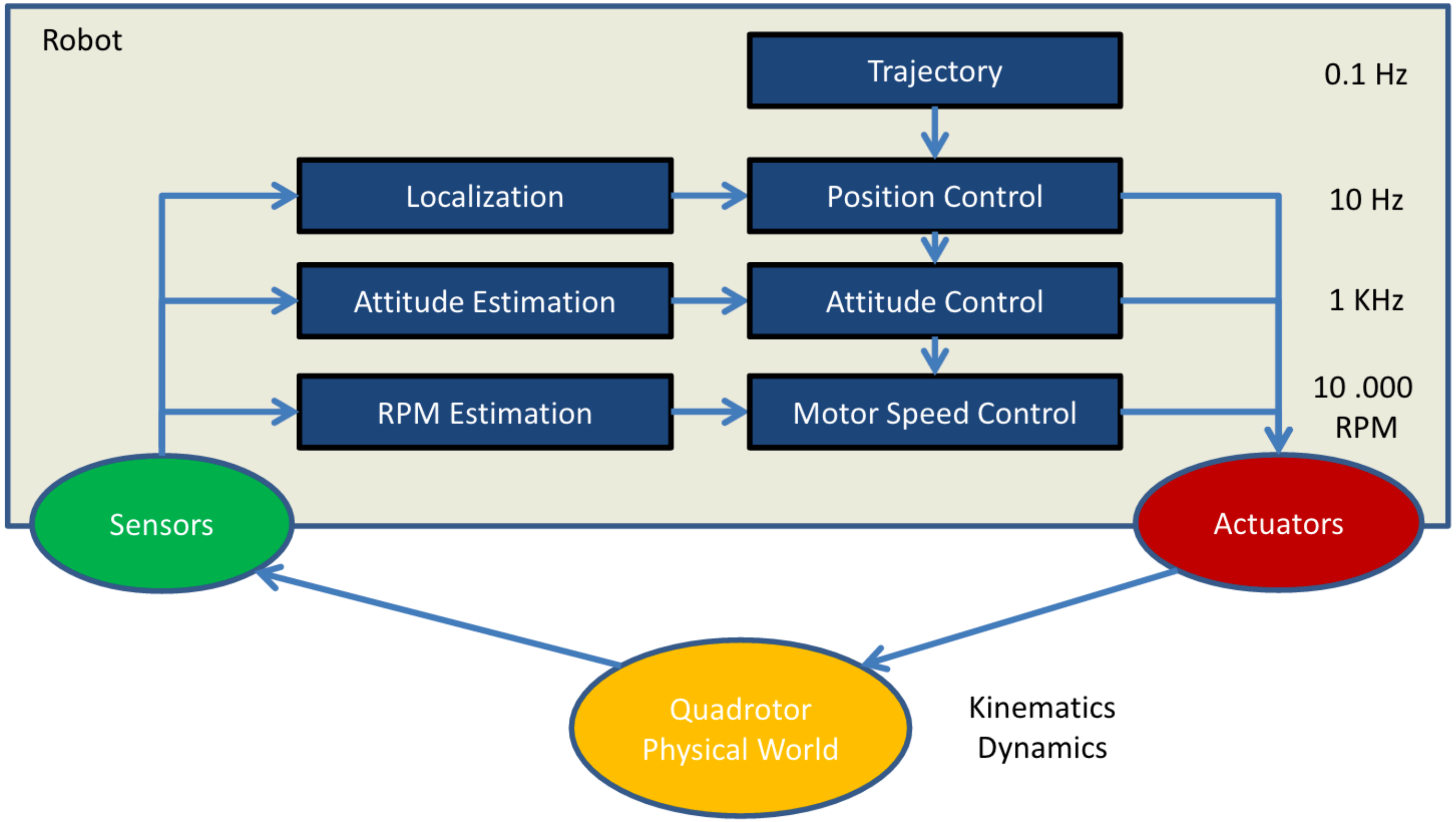
For MIMO Systems (Multiple Input, multiple output), control is often decoupled



Source: Jürgen Sturm, PID Control, http://jsturm.de/publications/data/lecture_4_part_4.pdf

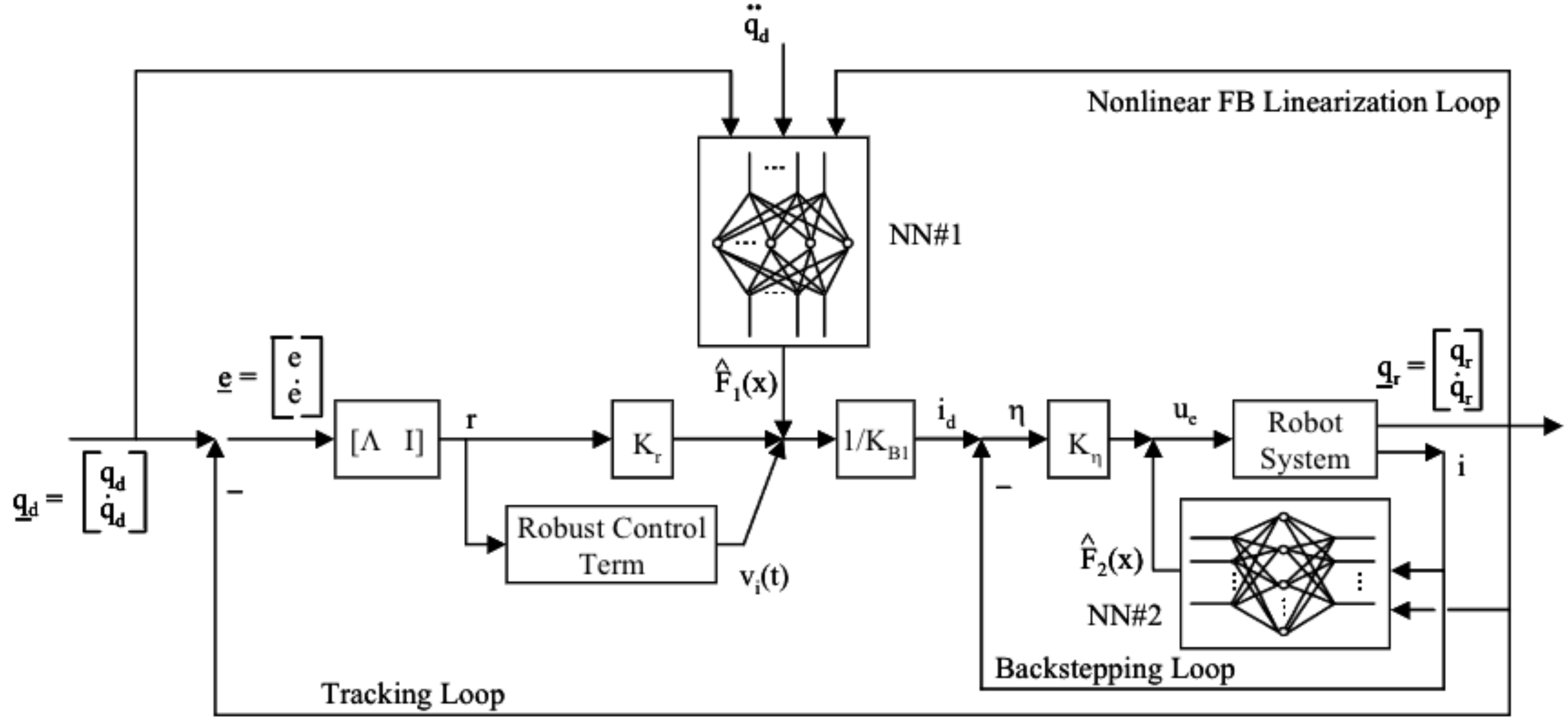
Cascaded Control

Enhanced control at various sampling frequencies allows for fine grained control at different levels.



Source: Jürgen Sturm, PID Control, http://jsturm.de/publications/data/lecture_4_part_4.pdf

Adding neural networks to control systems



Source: http://www.pdx.edu/sites/www.pdx.edu.sysc/files/media_assets/SySc576_FrankLewisNNsControl.pdf

Thank You