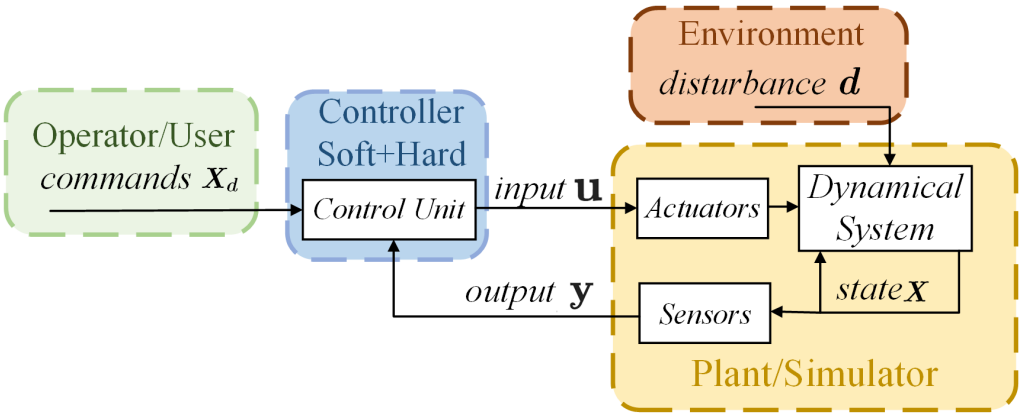
Modern Control Paradigms:

Lecture 1: Notion of Dynamical system, State space models, Control System and Implementation Issues

Modeling of Dynamical Systems

During our course we will discuss **control methods over dynamical systems**, our task can be summarized as follows: to **design a control algorithms** that will cause the **controlled object to perform the desired behavior** even in the **presence of possible disturbances**

But before we will dig in to details let us recall what is the main components of control system are:



From the scheme above we may distinguish following:

- **Operator/User** - is providing desired behavior to the control system by means of **desired commands**
- **Plant** - is the device/software we are trying to control and it's consist of:
  - **Actuators** - transform the software signals from controller to dynamical system
  - **Dynamical system** - take the inputs and change its behavior according to some mechanism of our robot which is completely described by some variables (state)
  - **Sensors** - measure some combination of states and produce the output
  - The Plant can be replaced by **simulator** software that mimic all of the components above.
- **Controller** - is the software + hardware that take **outputs** from the plant/simulator and produce appropriate **input** based on **user commands**, the controller itself may consist of different blocks, i.e. planner, regulator, observer, estimator etc.
- **Environment** - the system and world is in the constant bilateral interacting process that may be either beneficial or reduce the system performance by inducing **disturbances**

In this course we will mainly focus on **design of advanced controllers**, however to facilitate this we mast ensure that we have a clear picture of the object we are going to control.

So today we will try to:

- briefly recall what a dynamical system is
- how to build **mathematical model**
- **appropriately simulate** behavior of plant

While doing so we will have **two assumptions** (at least for today) namely plant actuators are ideal and supply the **control inputs to plant without any distortions**. The **full state of plant is measurable** without any noise (sensors are ideal as well)

Dynamical System

A dynamical system is a system whose behaviour, indicated by its output signal, **evolves over time**, possibly under the influence of **external inputs**.

Examples of dynamical systems include:

- Electrical circuits
- Mechanical systems
- Biological systems
- Stock market, and many others...

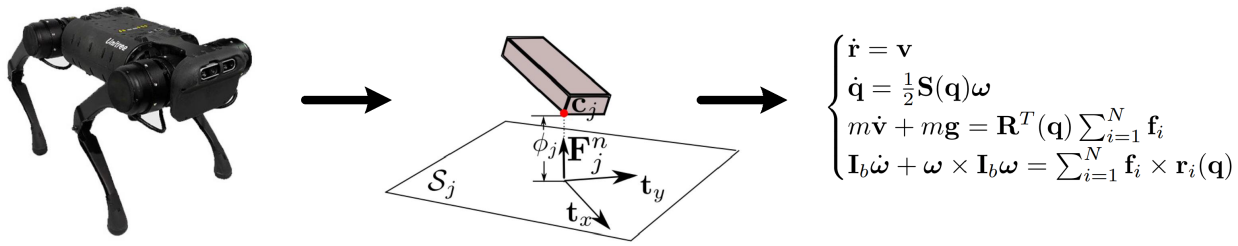
Whatever objects with some quantities changing in time can be viewed as dynamical system

A dynamical system may be

- **SISO or MIMO**: A SISO system is single-input single-output system and a MISO system is multiple-input multiple-output system
- **Continuous-time or discrete-time**: A continuous-time system accepts and generates signals at all continuous times continuously.
  A discrete system accepts and generates signals at discrete times.
- **Causal or non-causal**: A system is causal if its output at some time t depends on inputs up to time t but not after t. Otherwise it is non-causal.
- **Time invariant or time-varying**: A system is time-invariant if its input-output relationship is independent of time. Otherwise it is time-varying.
- **Linear or nonlinear**: The process that we want to model may be extremely complicated with different non-linearities, it is rare that real life systems are well described by linear models along the whole operational range.

Plant Models

Mathematical model is the **abstraction** of real world. While one building model of process there are a lot of simplifications may be made, and even seemingly accurate models are never perfectly describe the underlying process. **A model should be as simple as possible, and no simpler**.



Anything in the physical or biological world, whether natural or involving technology, is subject to analysis by mathematical models if it can be described in terms of mathematical expressions.

For the purpose of control design, a suitable model of the system should be used.
Different kinds of models used in control design are:

- **Impulse response** models (time domain)
- **Transfer function** models (frequency domain)
- **State-space** models (time domain)

At present, mostly state-space models are used due to their generality, simplicity of implementation and mature mathematical apparatus that simplifies their analysis. However, impulse responce and transfer function may be usefull as well. For instance if one is interested in input output relationships, or som terminal properties of a system, impulse
response models or transfer function descriptions can be used.

However in **this course we will stick to the state space models**, since they are widely used in modern control systems.

State-space models

These models are based on the concept of the **state** of the system.
The state $\mathbf{x}$ of a system is the smallest set of variables (called state variables) such that the knowledge of these variables $\mathbf{x}_0$ at some time t_0 together with the knowledge of the input $\mathbf{u}(\tau)$ for all τ from t_0 to t completely determines the behavior of the system for any time $t > t_0$.

A state-space model of a system is expressed with a set of first-order differential equations, one for each state variable.

Linear Systems

Linear control theory has been predominantly concerned with the study of **linear timeinvariant** (LTI) control systems, of the form:

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}$$

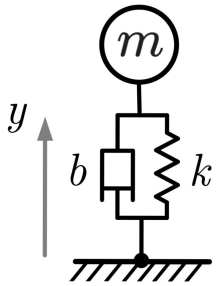
with $\mathbf{x} \in \mathbb{R}^n$ being a vector of **states**, $\mathbf{A} \in \mathbb{R}^{n \times n}$ the **system matrix**, $\mathbf{u} \in \mathbb{R}^m$ **input** (control) vector and $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the **input matrix**.

LTI systems have quite simple properties, such as:

- a linear system has a **unique equilibrium point** if $\mathbf{A}$ is full rank
- the equilibrium point is **stable** if all eigenvalues of $\mathbf{A}$ have negative real parts, regardless of initial conditions
- the transient response of a linear system is composed of the natural modes of the system, and the general solution can be found **analytically**
- **response** satisfies the principle of superposition.
- a sinusoidal input leads to a sinusoidal output of the **same frequency**.

Example: Mass-Spring-Damper

$$m\ddot{y} + b\dot{y} + ky = 0$$



And one can formulate this system in state space as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

Nonlinear Systems

Physical systems are inherently nonlinear. Thus, all control systems are nonlinear to a certain extent.
Nonlinear control systems can be described by nonlinear differential equations.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}, t)$$

where $\mathbf{f} \in \mathbb{R}^n$ is some nonlinear smooth function.

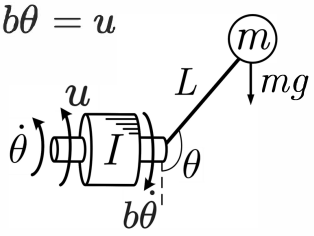
Nonlinear systems in contrast to linear:

- in general one can't obtain analytical solution
- a nonlinear system has a **multiple equilibrium points**
- the **stability** of these equilibriums is much harder to analyze
- **response** does not satisfies the principle of superposition.

The form above is fairly general, however there are known special cases, like control-affine and drift-less systems which we will study a bit later.

Example: Nonlinear Pendulum

$$(mL^2 + I)\ddot{\theta} + mgL\sin\theta + b\dot{\theta} = u$$

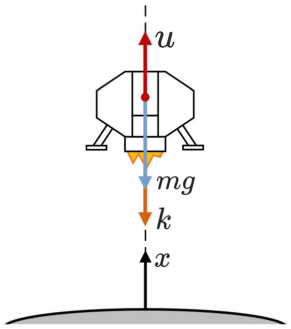


Given state $\mathbf{x} = [\theta, \dot{\theta}]^T$ we may formulate equation above as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{mL^2+I}(u - mgL\sin x_1 - bx_2) \end{bmatrix}$$

Example: Variable Mass Lander

$$m\ddot{x} + mg = -k\dot{m} = -ku$$



Noting that m is vary we need to include it to state, thus $\mathbf{x} = [x, \dot{x}, m]^T$ and equation above is equalient to:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} v \\ -g - \frac{k}{m}u \\ u \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ -g - \frac{k}{x_3}u \\ u \end{bmatrix}$$

Generilized Mechanical System

Equation of motion for most mechanical systems may be written in following form:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{d}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{Q} = \mathbf{B}(\mathbf{q})\mathbf{u}$$

where:

- $\mathbf{Q} \in \mathbb{R}^n$ - generalized forces corresponding to generilized coordinates
- $\mathbf{B} \in \mathbb{R}^{n \times m}$ - input mapping matrix that mapped actual forces to the
- $\mathbf{d} \in \mathbb{R}^n$ - disturbances (for instance friction or external forces)
- $\mathbf{q} \in \mathbb{R}^n$ - vector of generilized coordinates
- $\mathbf{M} \in \mathbb{R}^{n \times n}$ - positive definite symmetric inertia matrix
- $\mathbf{h} \in \mathbb{R}^n$ nonlinear term that describe the internal forces (coriolis and centrifugal terms)

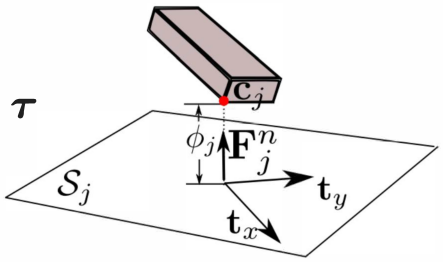
One can easily transform the mechanical system to the state space form by defining the state $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{M}^{-1}(\mathbf{x}_1)\big(\mathbf{B}(\mathbf{x}_1)\mathbf{u} - \mathbf{d}(\mathbf{x}_1, \mathbf{x}_2, t) - \mathbf{h}(\mathbf{x}_1, \mathbf{x}_2)\big) \end{bmatrix}$$

Example: Floating Rigid Body

The model of floating rigid body described by its postion $\mathbf{p}$, linear $\mathbf{v} = \dot{\mathbf{p}}$ and angular velocity $\boldsymbol{\omega}$ subject to external force $\mathbf{f}$ and torque $\boldsymbol{\tau}$:

$$\begin{cases} m\dot{\mathbf{v}} + m\mathbf{g} = \mathbf{f} \\ \mathcal{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathcal{I}\boldsymbol{\omega} = \boldsymbol{\tau} \end{cases}$$



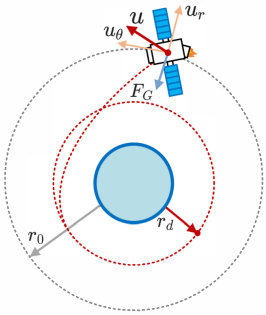
To rewrite the following in the general form one may define following:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} mI & 0 \\ 0 & \mathcal{I} \end{bmatrix}, \quad \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} mg \\ \boldsymbol{\omega} \times \mathcal{I}\boldsymbol{\omega} \end{bmatrix}, \quad \mathbf{B} = \mathbf{I}$$

Example: Artificial Satellite

The artificial satellite orbiting planet may be described using Newton gravity theory as:

$$\begin{cases} m\ddot{r} = mr\dot{\theta}^2 - G\dfrac{mM}{r^2} + u_r \\ mr\ddot{\theta} = -2m\dot{r}\dot{\theta} + u_\theta \end{cases}$$



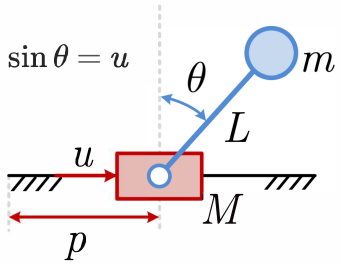
Given the generilized coordinates as $\mathbf{q} = [r, \theta]$ and control $\mathbf{u} = [u_r, u_\theta]^T$ one can express above in the general form:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m & 0 \\ 0 & mr \end{bmatrix}, \quad \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -mr\dot{\theta}^2 + G\dfrac{mM}{r^2} \\ 2m\dot{r}\dot{\theta} \end{bmatrix}, \quad \mathbf{B} = \mathbf{I}$$

Example: Cart Pole

Let us consider the cart pole described by:

$$\begin{cases} (M+m)\,\ddot{r} - mL\ddot{\theta}\cos\theta + mL\dot{\theta}^2\sin\theta = u \\ mL^2\ddot{\theta} - mLg\sin\theta = mL\ddot{r}\cos\theta \end{cases}$$



Defyning the generilized coordinates as $\mathbf{q} = [r, \theta]$ and matching the terms yields:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} M+m & -mL\cos\theta \\ -mL\cos\theta & mL^2 \end{bmatrix}, \quad \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} mL\dot{\theta}^2\sin\theta \\ -mLg\sin\theta \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Note that the choice of a set of state variables for a system is **not unique**. So a state-space model for a system is also not unique. There can be **infinitely many possibilities**.

When the model is obtained from a known differential equation, you can always select meaningful state variables. However, state variables may not directly relate to physical variables if the model is obtained from **identification methods**.

Other Models

It should be noted that there are special cases when the equations above maybe simplified or some interesting properties may be exploited, examples are:

- Control affine systems: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u}$
- Drift-less systems: $\dot{\mathbf{x}} = \mathbf{G}(\mathbf{x})\mathbf{u}$
- Differential-Algebraic Equations: $\mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}) = \mathbf{0}$
- Mechanical Systems in Regressor Form: $\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\mathbf{p} = \mathbf{B}(\mathbf{q})\mathbf{u}$
- Differential-Inclusions: $\dot{\mathbf{x}} \in \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}, t)$

We will consider some of these later on.

Discrete-Time Systems

Some dynamical systems are described in discrete-times (which can be counted) rather than in continuous time. e.g., system representing a bank account whose balance is reported once everyday.

Discrete systems are described by **difference equations**:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k$$
$$\mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k, k)$$

Some systems are inherently discrete, whereas many continuous-time systems are modeled in discrete-time for easier analysis and design with digital computers.

Discretization and Simulation

In the field of robotics the most models are actually derived from continues differential equations while the controller is implemented in digital form (software). Thus the overall control system may be described as follows:

- Output signals from the system $\mathbf{x}(t)$ are sampled regularly at times $t_k = kT$, $k = 0, 1, 2, \ldots$, where T is the sampling period resulting in the **state samples** $\mathbf{x}_k$.
- These samples $\mathbf{x}_k$ are fed to a **digital controller** which calculates **control input** $\mathbf{u}_k$.
- A **constant input** $\mathbf{u}_k$ is applied to real system over the sampling period using zero-order hold.

So for the purpose of control and analysis, it always useful to obtain a model of the system in **discrete times**. Obtaining a discrete-time model is referred to as **discretization**.

For a LTI system, we can obtain a exact discrete-time model if the input remains constant over each sampling period.

Approximated discretization

Exact discretization of time-varying/nonlinear systems are difficult or may not be analytically possible.

Approximate discrete-time models are widely used in practice. For a small sampling time, T, we can write, for the nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}, t)$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T\mathbf{f}\big(\mathbf{x}(kT), \mathbf{u}(kT), \mathbf{d}(kT), kT\big)$$

And for a linear system, this becomes

$$\mathbf{x}_{k+1} = (\mathbf{I} + T\mathbf{A}(kT))\mathbf{x}_k + T\mathbf{B}(kT)\mathbf{u}_k$$

This is called discretization by Euler method.

Example:

Consider the nonlinear pendulum described by following state space representation (assuming all parameters are 1):

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ u - \sin x_1 - x_2 \end{bmatrix}$$

The discrete model is then given by following set of difference equations:

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \end{bmatrix} = \begin{bmatrix} x_{1_k} \\ x_{2_k} \end{bmatrix} + T \begin{bmatrix} x_{2_k} \\ u - \sin x_{1_k} - x_{2_k} \end{bmatrix} = \begin{bmatrix} x_{1_k} + T x_{2_k} \\ x_{2_k} + T u_k - T \sin x_{1_k} - T x_{2_k} \end{bmatrix}$$

Simulation of ODE

While studying ODE $\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}, t)$, one is often interested in its solution $\mathbf{x}(t)$ (integral curve):

$$\mathbf{x}(t) = \int_{t_0}^{t} \boldsymbol{f}(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau, \quad \text{s.t: } \mathbf{x}(t_0) = \mathbf{x}_0$$

The simulation is nothing but tacking the integral above.

However, in most practical situations the above cannot be solved analytically and one should consider numerical integration instead, thus ending up with descrete system:

$$\mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k, k), \quad \text{s.t: } \mathbf{x}_0 = \mathbf{x}(t_0)$$

Thus simulation is just **iteration over the descrete dynamics** starting from initial point $\mathbf{x}_0 = \mathbf{x}(t_0)$

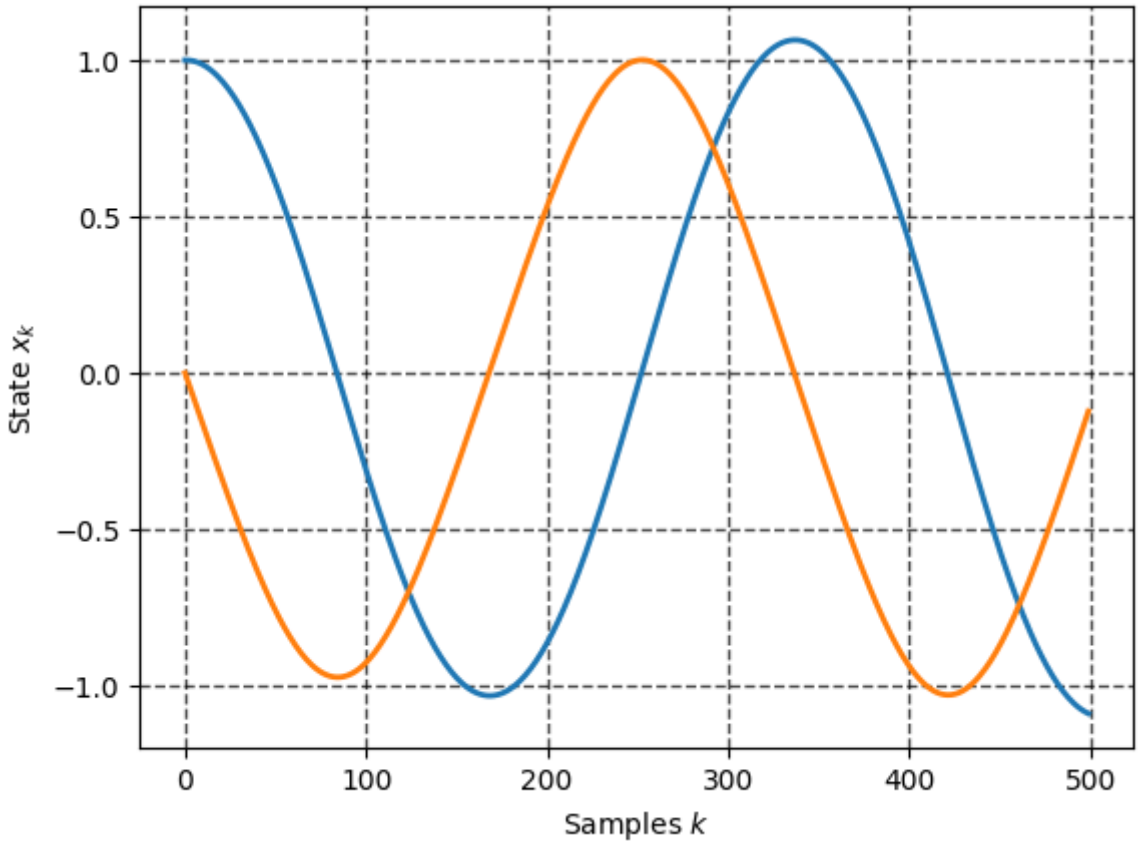
Let us implement the simulation of nonlinear pendulum via iterating the discrete dynamics:

```
import numpy as np

def f(state, t, control):
    u = control
    x1, x2 = state
    dx1 = x2
    dx2 = u - np.sin(x1) - 0*x2
    return np.array([dx1, dx2])
```

```
x_0 = np.array([1,0])
T = 2E-2
tf = 10
N = int(tf/T)
X = []


# ITERATE DESCRETE DYNAMICS
x_prev = x_0
for k in range(N):
    X.append(x_prev)
    u_k = 0
    x_new = x_prev + T*f(x_prev, k*T, u_k)
    x_prev = x_new

x_sol_simp = np.array(X)
```

Let us plot the result:

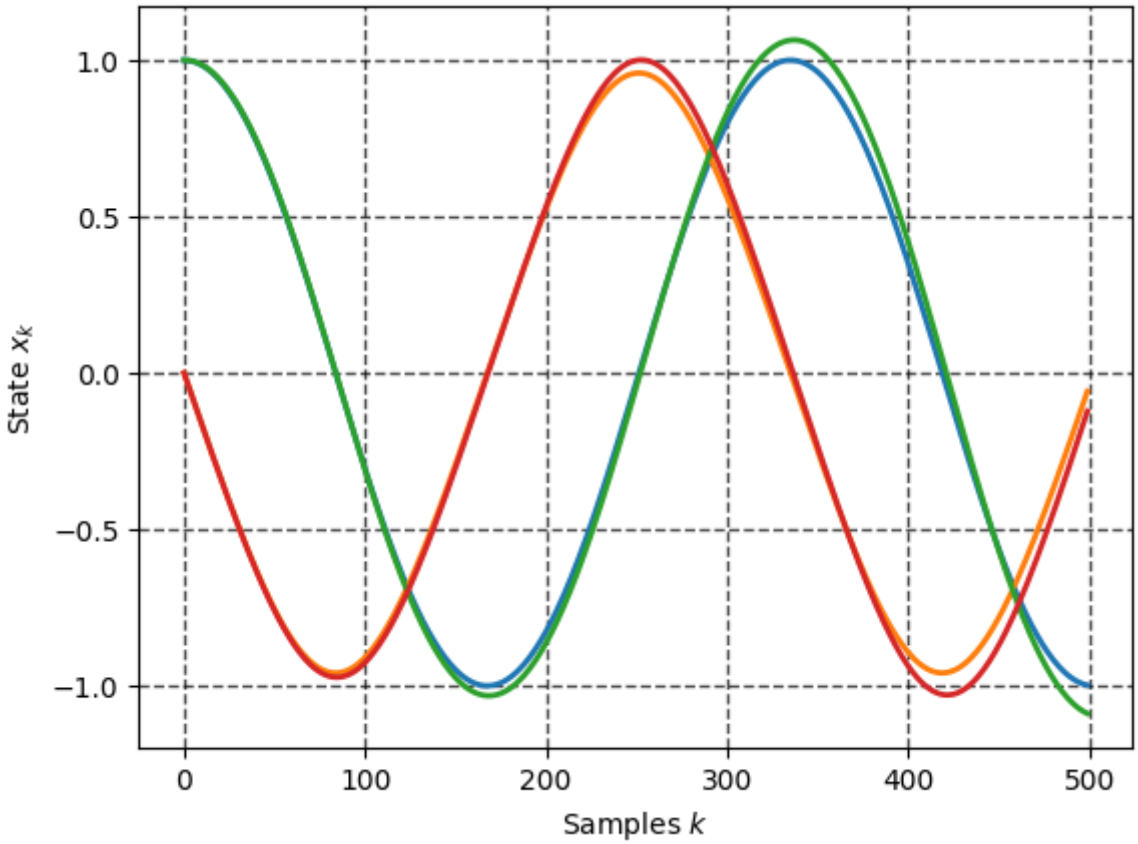
```
from matplotlib.pyplot import *

plot(x_sol_simp, linewidth=2.0)
grid(color='black', linestyle='--', linewidth=1.0, alpha = 0.7)
grid(True)
# xlim([t0, tf])
ylabel(r'State $x_k$')
xlabel(r'Samples $k$')
show()
```



The Euler method implemented above is highly dependent on the sampling period T, there are other suitable methods, the most widely used is the 4-th order Runge-Kutta and advanced variational integrators. However, we will not dig into the integration algorithm, instead for the purpose of simulation we will use `odeint` from the `scipy.integrate`:

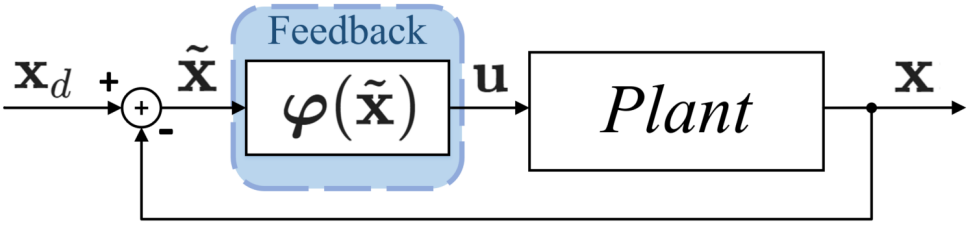
```
from scipy.integrate import odeint # import integrator routine
scale = 5

X = []
x_prev = x_0
for k in range(N):
    X.append(x_prev)
    t_k = np.linspace(k*T, (k+1)*T, scale)
    u_k = 0
    x_new = odeint(f, x_prev, t_k, args = (u_k,))
    x_prev  = x_new[-1,:]

x_sol = np.array(X)
```

```
plot(x_sol, linewidth=2.0)
plot(x_sol_simp, linewidth=2.0)
grid(color='black', linestyle='--', linewidth=1.0, alpha = 0.7)
grid(True)
ylabel(r'State $x_k$')
xlabel(r'Samples $k$')
show()
```



Controller and Implementation

How to make the given dynamical system display desired behavior? this is one of the questions of concern in the field of **control** theory. One of the most widely used approaches supporting the solution of the problems above is the so-called **feedback control**



Let us now assume that one have designed feedback law as follows:

$$\mathbf{u} = \boldsymbol{\varphi}(\mathbf{x})$$

One may substitute control law and obtain the equations of the **closed loop** system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\varphi}(\mathbf{x})) = \mathbf{f}_c(\mathbf{x})$$

now this system is unforced and may be analyzed as it is no control at all - basically we have changed the overall nature of plant - the governing dynamics.

In the **next lecture** we will learn how to **design the controller functions** and analyze **closed loop response** with different **numerical and analytical tools**, while for today we will move to the practice when we will focus on the **implementation side of the control system**.

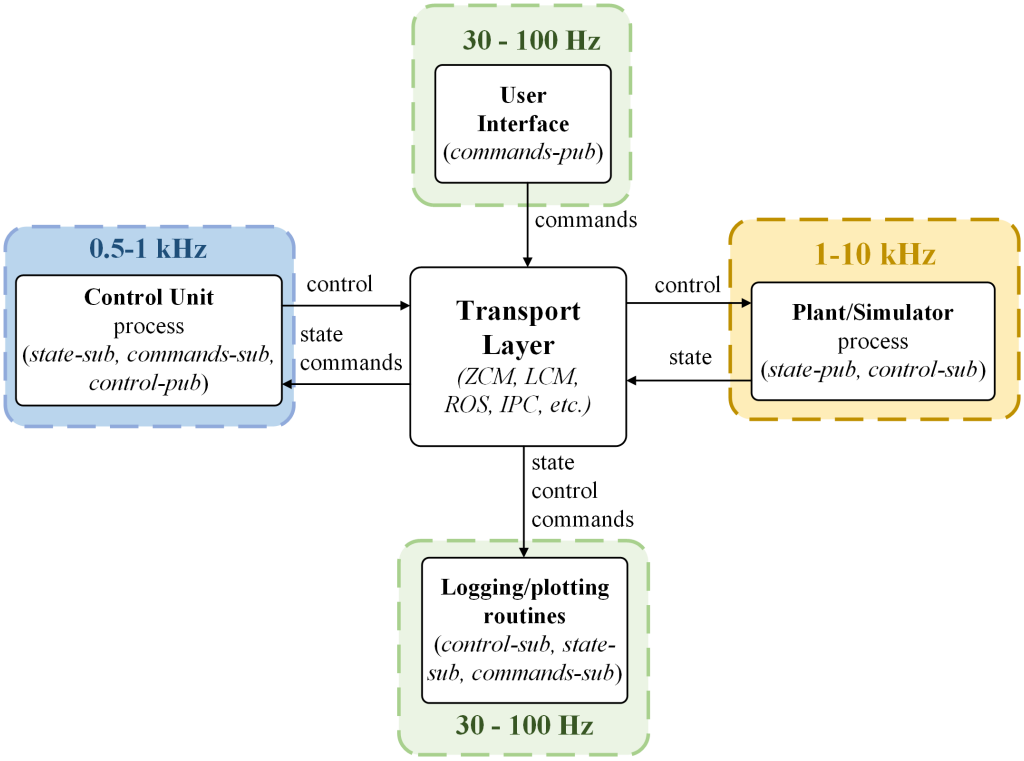
Practice 1: Architecture of The Control System, Controlled Dynamical System as Multiprocessing Software

From point of view of control and software engineers we would like to dig more into **implementation side**.

So far we have discussed that the purpose of **control** is to shape system in the way that it is **display desired behavior**. In paper everything is great, you just **design the controller function**, plug it to the **dynamics and analyze the resulting system**.

But in practice in most cases the system of choice is described in continues time while controller is in descrete domain. This can be achived if simulation run considerably **faster then controller** and ideally in **parallel** to control process. Thus it is natural to run controller in the separate process or thread.

Thus the overall software structure of the control system may be roughly described as follows:



Let us implement this architecture with help of `lcm` (lightweight communications and marshalling) library:

- Install the lcm library and it's python interface (I suggest to use virtualenv)
- Clone the course repository and go to `code/practice_01`
- Study the `.lcm` files in `transport/messages/` , these are the structures for the **controller**, **plant** and **user** messages.
- Checkout the `pub.py` `sub.py` , this are examples of publisher and subscriber
- Implement the plant simulator, controller and user interface (`simulator.py` , `controller.py` , `commands.py`)
  - use whatever dynamical system you want, for instance the nonlinear pendulum
  - simulation can be done either with simple euler method or odeint 4-th order runge-kutta (check the code from lecture for reference)
  - use whatever controller you familiar with (i.e PD, PID), or just apply the constant/random inputs for now
  - simulator should not be interrupted by the controller and be blocked
  - ensure that simulator is running faster then controller
- Implement simple logger and printing routine
- **BONUS** implement online plotter and checkout the trajectories