Data-driven and Learning-based Control

1st hands-on session

Erica Salvato

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Table of Contents

Dynamical Systems

- Definition

- Overview of the different kinds of dynamical systems:
  - continuous time (CT)
  - discrete-time (DT)
  - from CT to DT
  - linear
  - non-linear
  - equilibrium point
  - linearization around an equilibrium
  - stability analysis

What we know so far?

Given:

- $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^p$ the p-dimensional output
- $t \in \mathbb{R}_0^+$ the time
- $f_{CT} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function
- $g_{CT} : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ the output function

Given:

- $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ the n-dimensional state
- $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ the m-dimensional input
- $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^p$ the p-dimensional output
- $t \in \mathbb{R}_0^+$ the time
- $f_{CT} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the state transition function
- $g_{CT} : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ the output function

A **non-linear CT time-invariant dynamical system** is defined by a set of differential equations of the form:

$$\dot{x}(t) = f_{CT}\left(x(t), u(t)\right)$$

$$y(t) = g_{CT}\left(x(t), u(t)\right)$$

What we know so far?

A state $\bar{x} \in \mathcal{X}$ is an **equilibrium state** of the CT system if setting $u(t) = \bar{u} \in \mathcal{U}$ to a constant value

$$f_{CT}\left(\bar{x}, \bar{u}\right) = 0.$$

The pair $(\bar{x},\ \bar{u})$ is an **equilibrium point** of the CT system and can be:

- stable
- asymptotically stable
- globally asymptotically stable
- instable

What we know so far?

A state $\bar{x} \in \mathcal{X}$ is an **equilibrium state** of the CT system if setting $u(t) = \bar{u} \in \mathcal{U}$ to a constant value

$$f_{CT}\left(\bar{x}, \bar{u}\right) = 0.$$

The pair $(\bar{x},\ \bar{u})$ is an **equilibrium point** of the CT system and can be:

- stable
- asymptotically stable
- globally asymptotically stable
- instable

Given the equilibrium point $(\bar{x},\ \bar{u})$ we can linearize non-linear CT time-invariant dynamical system, thus obtaining a **linear CT time-invariant dynamical system**

What we know so far?

Denote by $\delta x^{(k)} = x^{(k)} - \bar{x}$ and $\delta u^{(k)} = u^{(k)} - \bar{u}$ the variations of $x^{(k)}$ and $u^{(k)}$ from their equilibrium values, respectively, the linearization of

$$\dot{x}(t) = f_{DT}\left(x(t), u(t)\right)$$

around $(\bar{x},\ \bar{u})$ is given by

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

Where $A = \left[\frac{\partial f_{DT}}{\partial x}\right]_{x=\bar{x},u=\bar{u}}$ and $B = \left[\frac{\partial f_{DT}}{\partial u}\right]_{x=\bar{x},u=\bar{u}}$

What we know so far?

The stability of $(\bar{x}, \bar{u})$ can also be analyzed for the linearized system by observing the eigenvalues of A:

- **asymptotically stable** if all the solutions of $\det(A - \lambda I) = 0$ have negative real part
- **unstable** if at least one solutions of $\det(A - \lambda I) = 0$ has positive real part
- if at least one solution of $\det(A - \lambda I) = 0$ has null real part, A is not enough to define the stability

What we know so far?

Given a CT time-invariant dynamical system we can define T_s the sampling time, and denoting by $x^{(k)}$ the state of the CT system at $t = kT_s$:

$$x^{(k)} = x(kT_s),\ \ k \in \mathbb{Z}_0^+.$$

we can discretize the CT time-invariant dynamical system and obtain a **DT time-invariant dynamical system**:

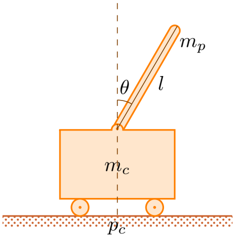
$$x^{(k+1)} = x(kT_s) + \int_{kT_s}^{(k+1)T_s} f_{CT}\left(x(\tau), u(\tau)\right) d\tau = f_{DT}\left(x^{(k)}, u^{(k)}\right).$$

Table of Contents

2 A case study

The pole balancing problem

2 A case study



$$\ddot{\theta} = \frac{g\sin(\theta) \; + \; \cos(\theta)\left[\frac{-F - m_p\, l\, \dot{\theta}^2 \sin(\theta)}{m_c + m_p}\right] - \frac{\mu_p \dot{\theta}}{m_p\, l}}{l\left[\frac{4}{3} - \frac{m_p\, cos^2(\theta)}{m_c + m_p}\right]}$$

$$\ddot{p}_c = \frac{F \; + \; m_p\, l\left[\dot{\theta}^2 \; \sin(\theta) \; - \; \ddot{\theta} \; \cos(\theta)\right]}{m_c \; + \; m_p}$$

Why?

- **Simplicity and complexity balance:** it is simple enough to grasp quickly how it works and, at the same time, it introduces non-linear dynamics and the need for sophisticated control strategies.
- **Physical intuition:** the physical nature of the problem (balancing a pole on a moving cart) resonates and helps in developing an intuitive understanding of control challenges and solutions.
- **Real-time control and stability:** implementing control strategies for the Cart and Pole system requires addressing real-time control challenges and stability issues.

Table of Contents

How it works?

You can decide to complete this hands-on with Matlab live script or Python notebook (I will provide both templates).

How it works?

You can decide to complete this hands-on with Matlab live script or Python notebook (I will provide both templates).

In both cases, it is necessary to fill in the forms with all the answers to the respective questions.

- 5 queries, each of which can be graded within $0 - 6$
- 3 hours in class. If not enough you can complete the experience at home
- The completed form must be submitted to the lecturer 1 week before the exam

How it works?

You can decide to complete this hands-on with Matlab live script or Python notebook (I will provide both templates).

In both cases, it is necessary to fill in the forms with all the answers to the respective questions.

- 5 queries, each of which can be graded within $0-6$
- 3 hours in class. If not enough you can complete the experience at home
- The completed form must be submitted to the lecturer 1 week before the exam

A mark < 18 is not valid for admission to the exam.

Hands-on outline

Given the cart and pole system previously defined:

1. Clearly define the state vector x and the control input vector u. Define also the domains to which they belong, $\mathcal{X}$ and $\mathcal{U}$ respectively.

Expectation: clear and formal definition of the variable composing x and u, the respective size and a definition of the set of admissible value $\mathcal{X}$ and $\mathcal{U}$

Given the cart and pole system previously defined:

1. Clearly define the state vector x and the control input vector u. Define also the domains to which they belong, $\mathcal{X}$ and $\mathcal{U}$ respectively.

Expectation: clear and formal definition of the variable composing x and u, the respective size and a definition of the set of admissible value $\mathcal{X}$ and $\mathcal{U}$

2. Compute the equilibrium state $\bar{x}$ when the control input $\bar{u}$ is set equal to 0.

Expectation: Formally define the equilibrium point and how to compute it for the cart and pole dynamical system. Create a code that can return the equilibrium point of the dynamic system when $\bar{u} = 0$ and define the stability of the equilibrium point.

Hands-on outline

3 1st hands-on session

3. Given a sequence of control inputs $\left[u^{(0)},\, u^{(1)}, \cdots, u^{(N)}\right]$ simulate the behavior of the non-linear cart and pole system.

Expectation: Create a code able to simulate the dynamics of the dynamical system previously defined when subjected to a sequence of N control inputs. Try different settings of the systems: initial conditions, and time-step size. Create a code that plots significant behavior.

4. Linearise the non-linear cart and pole system around the computed equilibrium point and simulate the behavior of the linear systems given the same sequence of control inputs.

Expectation: Create a code able to linearise the non-linear system around the equilibrium point and simulate the dynamics of the linearised dynamic system when subjected to a sequence of N control inputs. Try different system settings: initial conditions, size of time steps. Create a code that tracks meaningful behavior.

4. Linearise the non-linear cart and pole system around the computed equilibrium point and simulate the behavior of the linear systems given the same sequence of control inputs.

Expectation: Create a code able to linearise the non-linear system around the equilibrium point and simulate the dynamics of the linearised dynamic system when subjected to a sequence of N control inputs. Try different system settings: initial conditions, size of time steps. Create a code that tracks meaningful behavior.

5. Comment on the results obtained with the non-linear and the linearised system.

Expectation: Provide evidence (plots) of the observed behaviors and justify them with the theory studied.

Remarks

- The Matlab live script or the Python notebook must be executable on any PC. Include all necessary packages by line of code in the Matlab live script or Python notebook before submitting them. A code that does not run equals a grade $= 0$
- Comments on the code are not negligible. An uncommented code will not be rated full marks

Useful links

- Matlab live script:

  Matlab live script tutorial

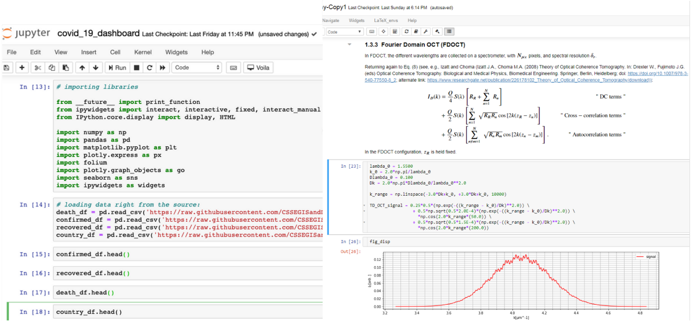
- Python notebook

  Python notebook general tutorial
  Python notebook ode solving tutorial

Table of Contents

Jupyter Notebooks are both:

- human-readable documents containing text elements that allow to provide comments on the results and analysis description and the results
- executable documents that can run computer code



How do they work?

You can create edit and run Jupyter Notebooks both on **Google Colab**:

- No configuration needed
- Free GPU access
- Easy sharing

Or on your PC by properly installing the **Jupyter Notebook App**

Functionalities

4 Jupyter Notebooks

- Install packages

```
!pip install numpy
```

- Comment the code

```
# A well-commented code is easy to use even after a while
```

- Import libraries

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
```

- Define functions

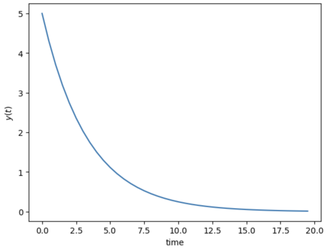
```
# function that returns dy/dt
def model(y,t):
    k = 0.3
    dydt = -k * y
    return dydt
```

Functionalities

4 Jupyter Notebooks

- Plot results

```
# plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('$y(t)$')
plt.show()
```



Functionalities

4 Jupyter Notebooks

- Solve ode

```
# initial condition
y0 = 5

# time points
t = np.arange(0, 20, 0.5)
#t = np.linspace(0,20)

# solve ODE
y = odeint(model,y0,t)
```

- Include text and equations

```
# Esempio Notebook per simulare un'equazione differenziale e plottare i
risultati

$\dot{y} = -ky$
```

Esempio Notebook per simulare un'equazione differenziale e plottare i risultati

$\dot{y} = -ky$

Esempio Notebook per simulare un'equazione differenziale e plottare i risultati

$\dot{y} = -ky$

```
[3] #!pip install numpy
```

```
[4] import numpy as np
    from scipy.integrate import odeint
    import matplotlib.pyplot as plt
```

```
# function that returns dy/dt
def model(y,t):
    k = 0.3
    dydt = -k * y
    return dydt
```

```
[6] # initial condition
    y0 = 5

    # time points
    t = np.arange(0, 20, 0.5)
    #t = np.linspace(0,20)

    # solve ODE
    y = odeint(model,y0,t)
```

Example

4 Jupyter Notebooks

```
# plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('$y(t)$')
plt.show()
```

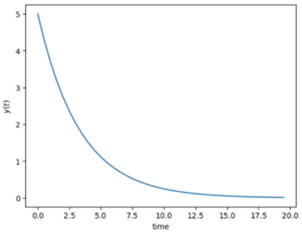


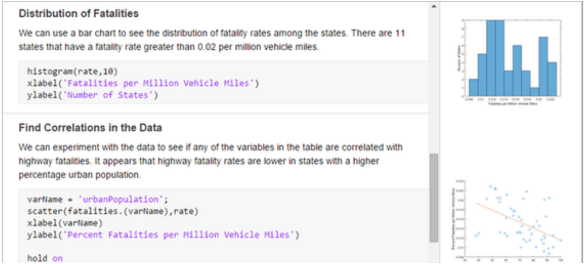
Table of Contents

How do they work?
5 Matlab live scripts

Matlab live scripts are interactive documents that combine:

- MATLAB code
- Formatted text
- Equations
- Images

in a single environment called the Live Editor



How do they work?

MATLAB supports live scripts and live functions in **versions R2019b and above**.

UniTS students have a free MATLAB license.

Some MATLAB features are **unsupported** in the Live Editor:

- Classes: create classes as plain code files (.m) instead
- MATLAB preferences: e.g., custom keyboard shortcuts

How do they work?

MATLAB supports live scripts and live functions in **versions R2019b and above**.

UniTS students have a free MATLAB license.

Some MATLAB features are **unsupported** in the Live Editor:

- Classes: create classes as plain code files (.m) instead
- MATLAB preferences: e.g., custom keyboard shortcuts

When you need a specific toolbox or add-on for your code execution, specify which one and the versioning in comments.

Functionalities

5 Matlab live scripts

- Comment the code

```
% A well-commented code is easy to use even after a while
```

- Define functions

```
function dydt = odefcn(t,y,A,B)
    % function that returns dy/dt
    dydt = zeros(2,1);
    dydt(1) = y(2);
    dydt(2) = (A/B)*t.*y(1);
end
```

- Solve ode

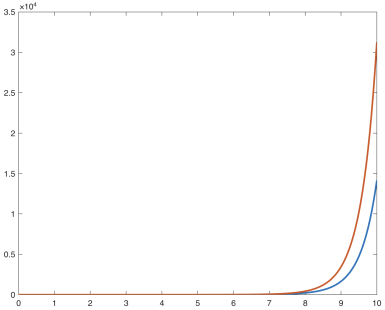
```
%timepoints
tspan=0:0.05:10;
%solve ODE
[t,y] = ode45(@(t,y)odefcn(t,y,A,B), tspan, y0);
```

Functionalities

5 Matlab live scripts

- Plot results

```
% plot
plot(t,y(:,1),'-',t,y(:,2),'-','LineWidth',2)
```



Functionalities

5 Matlab live scripts

- Include text and equations

Esempio Notebook per simulare un'equazione differenziale e plottare i risultati

$$\ddot{y} = \frac{A}{B} t y$$

Example

Live Editor – /Users/salvato/Desktop/example_livescript.mlx

example_livescript.mlx

Esempio Notebook per simulare un'equazione differenziale e plottare i risultati

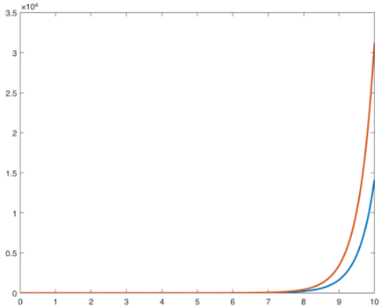
$$\ddot{y} = \frac{A}{B} t y$$

```
% A well-commented code is easy to use even after a while

% constants and initial condition
A = 1;
B = 2;
y0=[0 0.01];

%timepoints
tspan=0:0.05:10;
%solve ODE
[t,y] = ode45(@(t,y)odefcn(t,y,A,B), tspan, y0);
```

```
function dydt = odefcn(t,y,A,B)
    % function that returns dy/dt
    dydt = zeros(2,1);
    dydt(1) = y(2);
    dydt(2) = (A/B)*t.*y(1);
end
```

Example


```
% plot
plot(t,y(:,1),'-',t,y(:,2),'-','LineWidth',2)
```



Questions' time!

UNIVERSITÀ DEGLI STUDI DI TRIESTE