

## Lab Session 2 - Incorporating Data Connectivity into the Digital Twin

### 1 | Aim

The aim of this lab is to enhance the virtual model with real-time data from sensors.

### 2 | Preparation

For this lab session, you will need the following new files:

- `test_data.csv`;
- `plot_data.py`;
- `Make_recording.py`; and
- `Run_recording.py`.

You should already possess the last `Digital_twin.py` file and have a functional model implemented.

### 3 | Procedure

We begin by examining the data produced by our sensor system. The `test_data.csv` file contains a recording of the pendulum swinging. Execute the `plot_data.py` file to analyse the data:

- 3.1.** What type of noise is present? (You can also mention its origin, the spectrum).
- 3.2.** Which filtering technique would be most appropriate?
- 3.3.** What additional transformations are required to incorporate this data into your virtual model?


Reflect on our system dynamics lecture, where we used the phase portrait to analyze significant aspects of the dynamic system. To enhance our understanding from the data we collect, consider the following questions:

- 3.4.** What data is critical to collect?
- 3.5.** How can the collected data provide information on the energy dissipation within the system? This Pendulum Lab ([https://phet.colorado.edu/sims/html/pendulum-lab/latest/pendulum-lab\\_all.html](https://phet.colorado.edu/sims/html/pendulum-lab/latest/pendulum-lab_all.html)) may offer valuable insights.

- 3.6.** Once you have determined the data you wish to collect, proceed with running the `Make_recording.py` file. You can modify the actions in the actions array, with available actions ranging from 0 to 8, as defined in your digital twin file (`action_map`).

**Algorithm 1:** Define the actions.

```
1 actions = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2 digital_twin.connect_device()
3 digital_twin.start_recording("test")
```

 **Note:** If for some reason you don't get any data via the connected device, you might want to try to add the following in the digital twin file: `ser.flush()` after reading and writing the serial port. If it does not work, please continue the lab with the provided data or ask a classmate to share the collected data.

When starting a new recording, in order to avoid overwriting the previous collected data, be sure to rename the file in the `digital_twin.start_recording("test")` function. In this example, the file `test.csv` will be generated and stored in the same directory as the `Make_recording.py` script.

- 3.7.** Next, we will filter the data from the recording by executing the `Run_recording.py` file. This script implements three different filtering methods: the Kalman filter [1], a Median filter [2], and an Exponential Moving Average (EMA) filter [3].

## 4 | Tasks

- 4.1.** Adjust the different filters to achieve optimal signal filtration.
- 4.2.** Discuss the advantages and disadvantages of each filtering technique.
- 4.3.** Identify your preferred filtering method and explain your choice.
- 4.4.** Calibrate and scale the signal so that it can be accurately represented in the virtual model.
- 4.5.** Uncomment the section that enables the digital twin's visualization and display your chosen filter with the optimal parameters.
- 4.6.** Finally, integrate this filter and transformation into the `Digital_twin.py` file, applying a transformation to the `x_pivot` as well.

## 5 | References

- [1] M. Khodarahmi and V. Maihami, "A review on Kalman filter models," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 727–747, 2023. [Online]. Available: <https://link.springer.com/article/10.1007/s11831-022-09815-7>
- [2] FluCoMa Team, "Median Filters," 2024, accessed: 2024-01-23. [Online]. Available: <https://learn.flucoma.org/learn/median-filters/>

- [3] M. E. Ninja, "Exponential Moving Average (EMA) Filter," 2024, accessed: 2024-01-23. [Online]. Available: <https://blog.mbedded.ninja/programming/signal-processing/digital-filters/exponential-moving-average-ema-filter/>