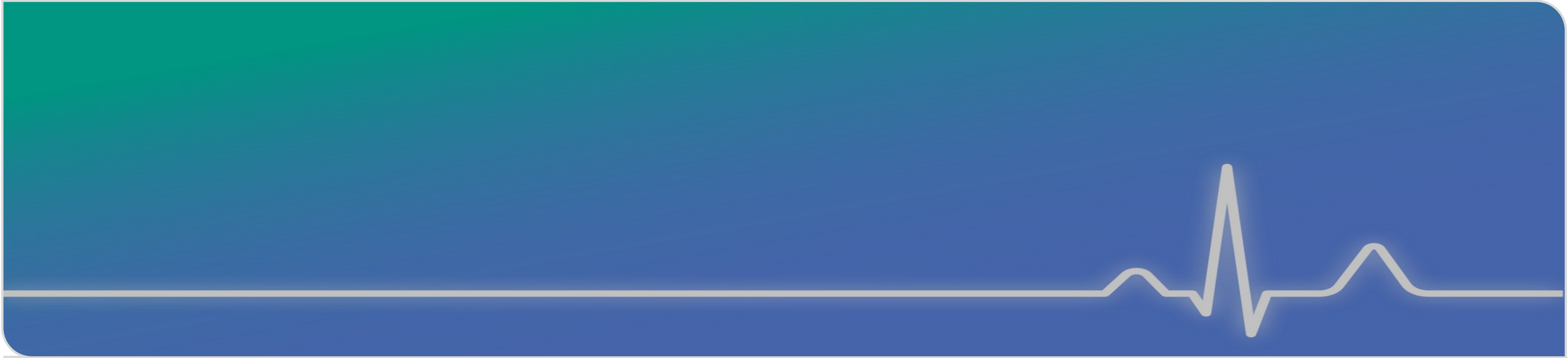
Web Interface for Cardiac Simulations

Tomas Stary, advised by **Axel Loewe**



MICROCARD Objectives

From Proposal for WP1:

6. Integrate the components delivered by the other work packages into a production-ready HPC simulation platform for cardiac electrophysiology.

Results of a Survey

- 28 people (20 users) participated in the survey
- The majority runs bench and openCARP directly
- People have difficulties with installing dependencies and compiling openCARP (9 out of 15 that install openCARP from sources)
- Only a few (3) use docker

Results of a Survey

Desired features:

- Allow to retrieve results of the simulation
- Easy run on HPC resources
- Connection with published experiments
- Visualization in the web-based system
- Templates for carputils experiments
- Submission of carputils experiments from the interface
- Interactive control of the parameters of carputils script
- Access to carputils tutorials
- Simplified sharing of experiments with the community

Considered Solutions

Target groups

- Beginner users – example tutorials to facilitate learning

- Experienced users – implementation of own experiments

Technology

- Carputils GUI
- Streamlit or Trame Python frameworks

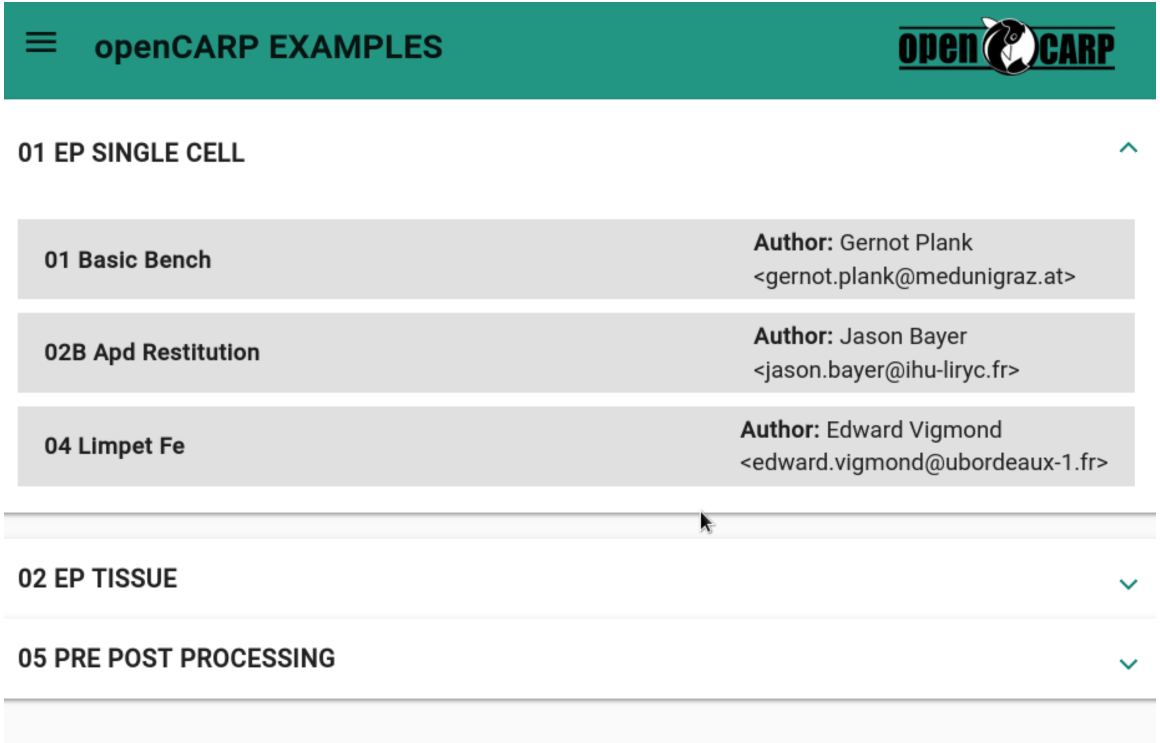
- JupyterLab

Carputils GUI

- Python backend using Flask API
- JavaScript frontend using Angular framework
- ParaViewWeb for visualizations
- Postgress database
- Nginx web server
- Deployed as docker-compose services

Carputils GUI

- Automatic conversion from selected tutorials
- Both single cell and tissue simulations
- Self-hosted or provided as service



openCARP EXAMPLES

01 EP SINGLE CELL

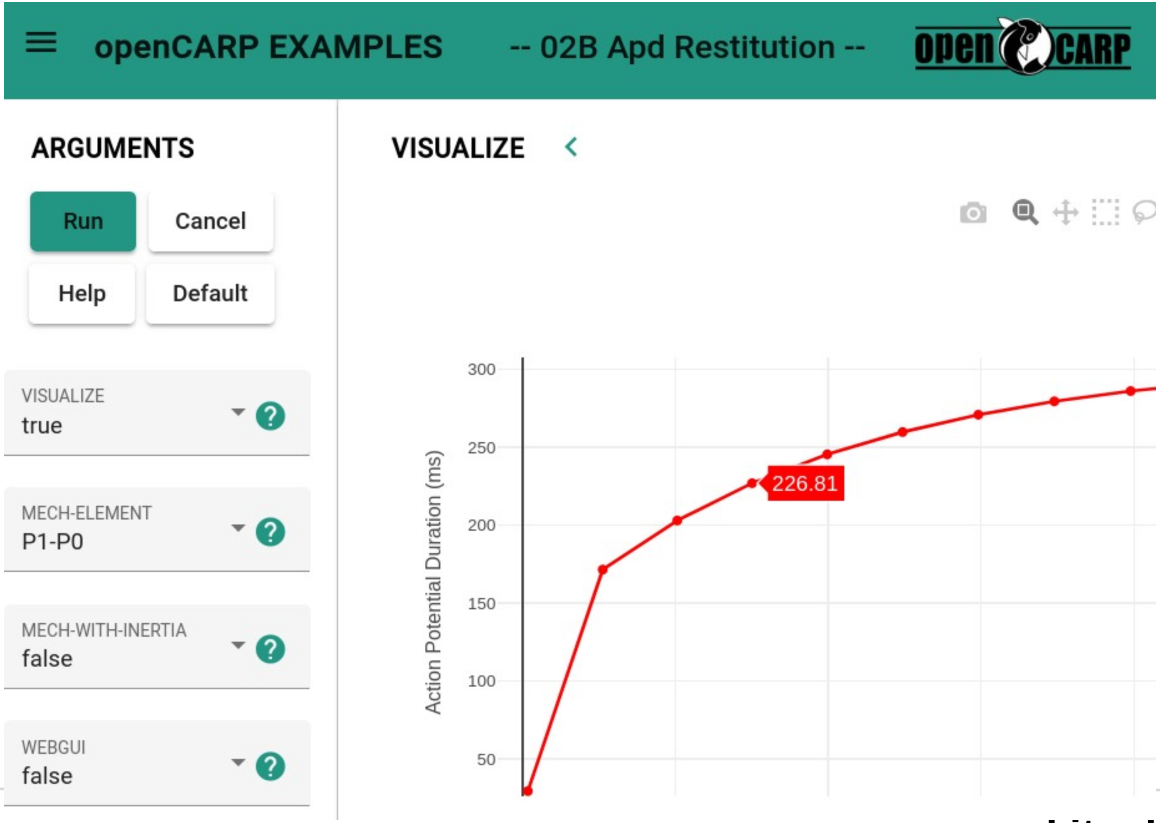
01 Basic Bench	**Author:** Gernot Plank <gernot.plank@medunigraz.at>
02B Apd Restitution	**Author:** Jason Bayer <jason.bayer@ihu-liryc.fr>
04 Limpet Fe	**Author:** Edward Vigmond <edward.vigmond@ubordeaux-1.fr>

02 EP TISSUE

05 PRE POST PROCESSING

Carputils GUI

- Single cell experiment visualized using plotly graphics library



openCARP EXAMPLES -- 02B Apd Restitution --

ARGUMENTS

VISUALIZE
true

MECH-ELEMENT
P1-P0

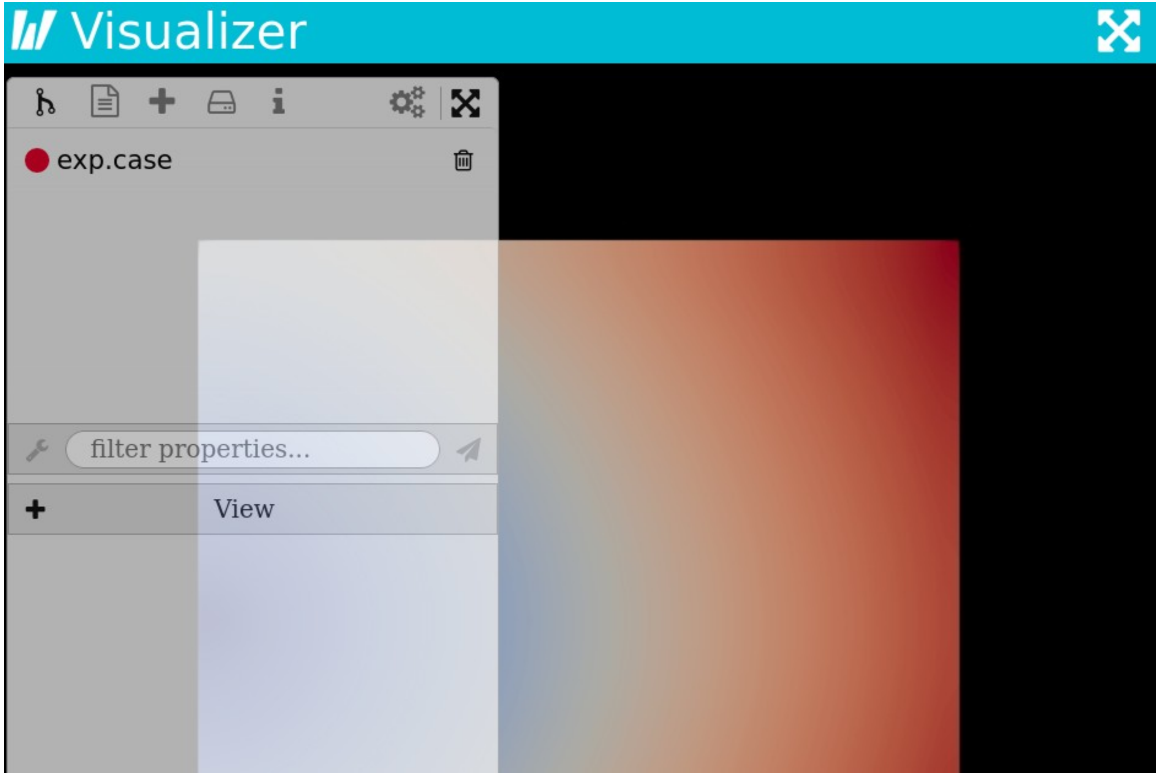
MECH-WITH-INERTIA
false

WEBGUI
false

VISUALIZE

Carputils GUI

- Tissue level simulations visualized through ParaViewWeb interface



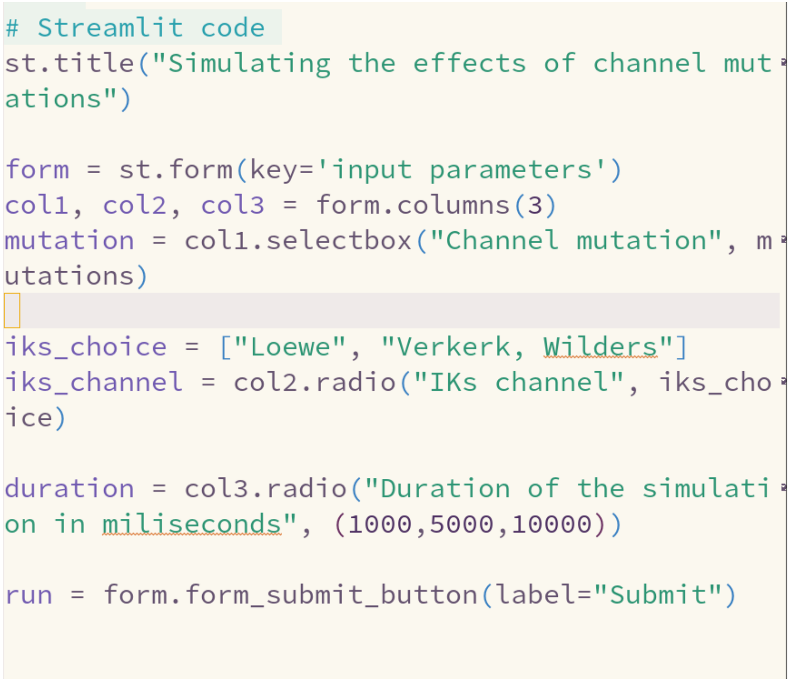
Streamlit and Trame

- Python frameworks for data science
- Simple implementation
- Python libraries for processing and visualization

Karlsruhe Institute of Technology

Streamlit Framework

Karlsruhe Institute of Technology



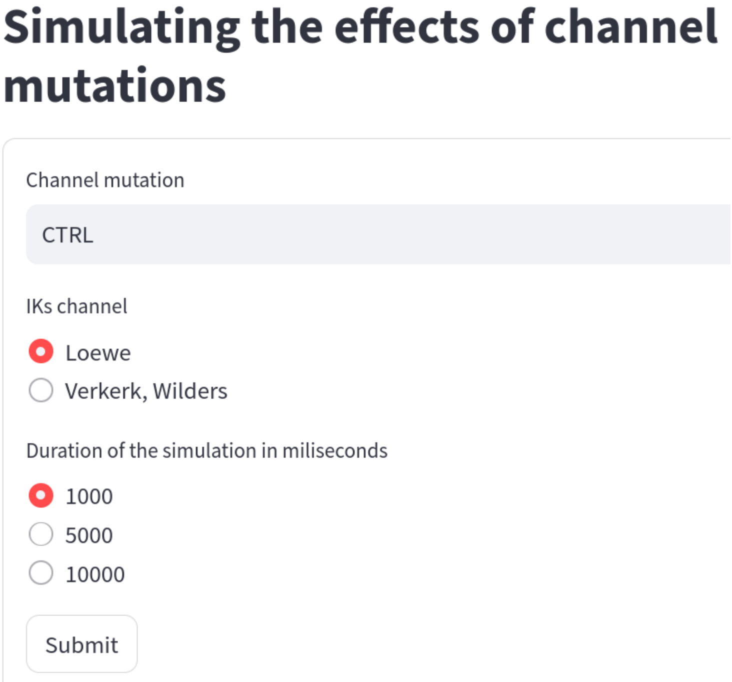
```
# Streamlit code
st.title("Simulating the effects of channel mutations")

form = st.form(key='input parameters')
col1, col2, col3 = form.columns(3)
mutation = col1.selectbox("Channel mutation", mutations)

iks_choice = ["Loewe", "Verkerk, Wilders"]
iks_channel = col2.radio("IKs channel", iks_choice)

duration = col3.radio("Duration of the simulation in miliseconds", (1000,5000,10000))

run = form.form_submit_button(label="Submit")
```



Simulating the effects of channel mutations

Channel mutation

CTRL

IKs channel

● Loewe
○ Verkerk, Wilders

Duration of the simulation in miliseconds

● 1000
○ 5000
○ 10000

Submit

Streamlit Framework

- Video demonstration

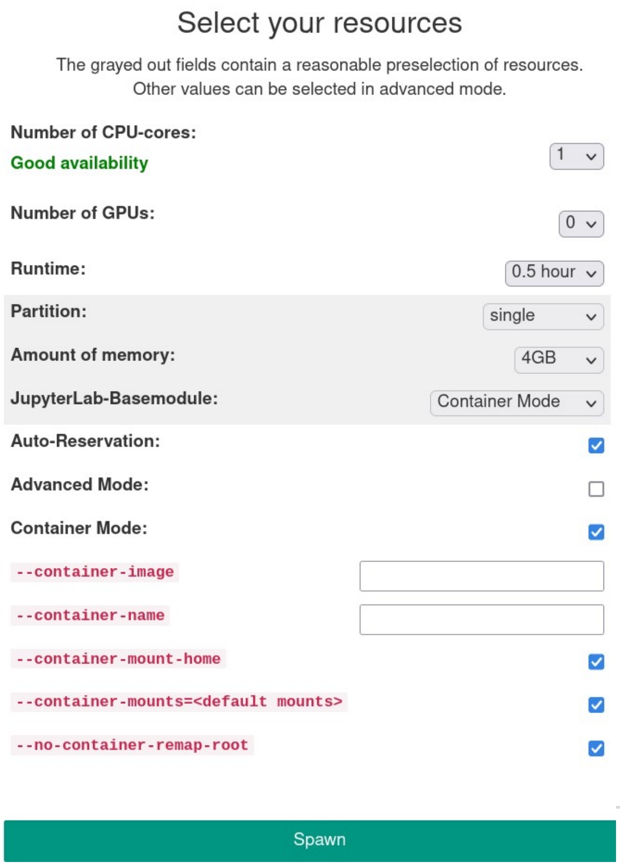
JupyterLab

- Documents combining **text**, formatted mathematical **formulas**, and **code** as well as **data, figures** and **tables** generated by that code
- Used to present linear data analysis pipelines
- Intuitive user interaction through selection widgets
- Large and supporting community of users
- Freely available and highly customizable

- Deployed on many HPC system such as:
  - BW HPC Center
  - Jülich Supercomputing Center
  - Cineca HPC Center

JupyterLab

- JupyterLab interface on BWUniCluster
- Container mode allows arbitrary image, e.g. docker.opencarp.org
- JupyterLab components automatically installed



Select your resources

The grayed out fields contain a reasonable preselection of resources.
Other values can be selected in advanced mode.

Number of CPU-cores: 1
Good availability

Number of GPUs: 0

Runtime: 0.5 hour

Partition: single

Amount of memory: 4GB

JupyterLab-Basemodule: Container Mode

Auto-Reservation: ☑

Advanced Mode: □

Container Mode: ☑

`--container-image`

`--container-name`

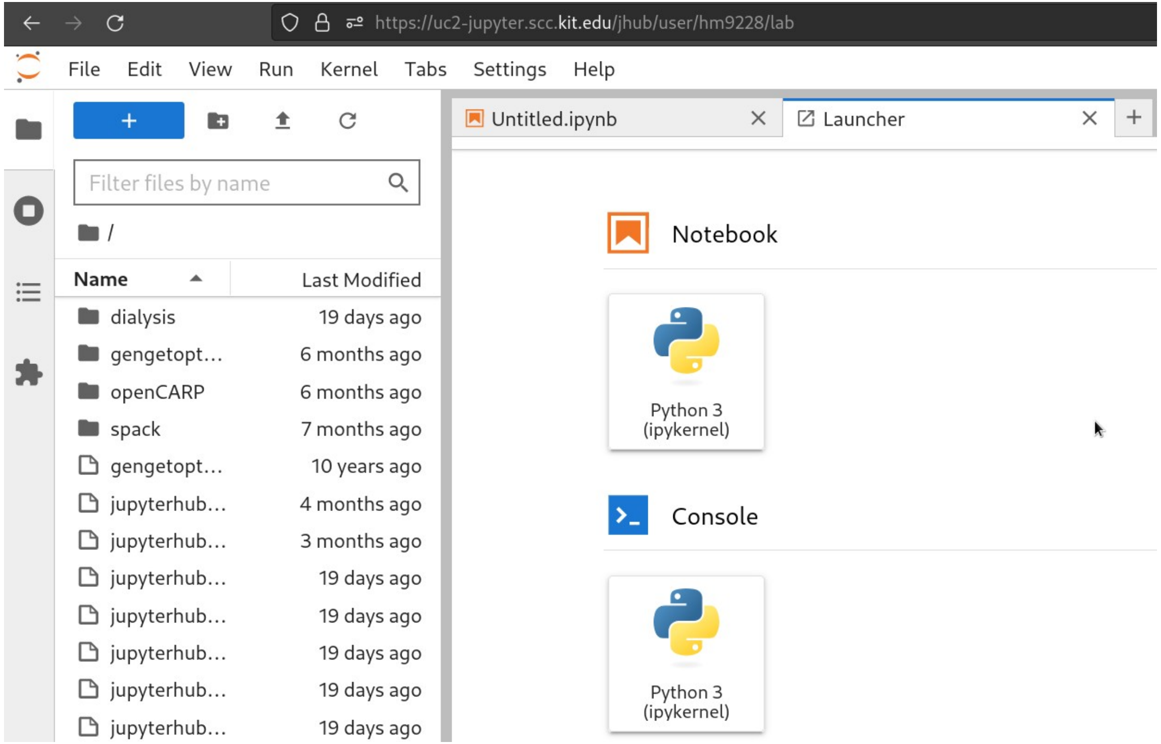
`--container-mount-home` ☑

`--container-mounts=<default mounts>` ☑

`--no-container-remap-root` ☑

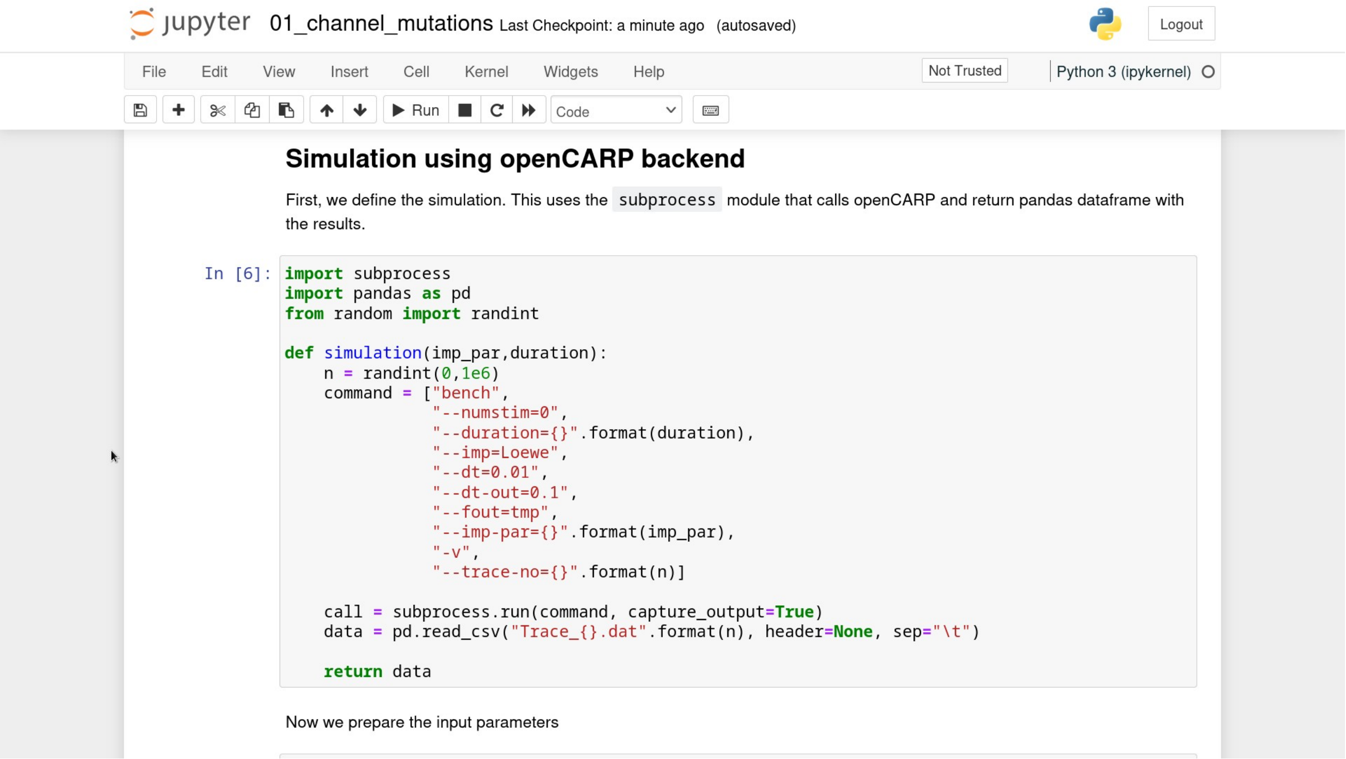
JupyterLab

- Mounted $HOME directory provides persistent access to notebook files



Name	Last Modified
dialysis	19 days ago
gengetopt…	6 months ago
openCARP	6 months ago
spack	7 months ago
gengetopt…	10 years ago
jupyterhub…	4 months ago
jupyterhub…	3 months ago
jupyterhub…	19 days ago
jupyterhub…	19 days ago
jupyterhub…	19 days ago
jupyterhub…	19 days ago
jupyterhub…	19 days ago

JupyterLab



Simulation using openCARP backend

First, we define the simulation. This uses the `subprocess` module that calls openCARP and return pandas dataframe with the results.

In [6]:

```
import subprocess
import pandas as pd
from random import randint

def simulation(imp_par,duration):
    n = randint(0,1e6)
    command = ["bench",
               "--numstim=0",
               "--duration={}".format(duration),
               "--imp=Loewe",
               "--dt=0.01",
               "--dt-out=0.1",
               "--fout=tmp",
               "--imp-par={}".format(imp_par),
               "-v",
               "--trace-no={}".format(n)]

    call = subprocess.run(command, capture_output=True)
    data = pd.read_csv("Trace_{}.dat".format(n), header=None, sep="\t")

    return data
```

Now we prepare the input parameters

JupyterLab

- Video demonstration

Future Work

- Document usage of JupyterLab with openCARP
- Convert some tutorial into JupyterLab format
  - Extract docstrings from `run.py` and convert from *.rst to markdown
  - Add Jupyter cells in markdown
  - `mystnb-to-jupyter` to convert markdown to *.ipynb format
- Explore possible usage of meshalyzer and ParaView for 3d visualization within JupyterLab

- Write a tutorial on Streamlit or Trame framework with openCARP

Karlsruhe Institute of Technology

Thank you for your attention!

MICROCARD

EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking EuroHPC (JU) under grant agreement No 955495. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and France, Italy, Germany, Austria, Norway, Switzerland.