pyScienceMode: an Open-Source Python Package to control electro-stimulator through the Hasomed's science mode protocol

Kevin Co[1], **Amedeo Ceglia**[2], **and Mickael Begon**[1,2]

1 School of Kinesiology and Human Kinetics, University of Montreal, Montreal, QC, Canada **2** Institute of Biomedical Engineering, Faculty of Medicine, University of Montreal, Canada

DOI: 10.21105/joss.08259

Software

- Review
- Repository
- Archive

Editor: Stefan Appelhoff

Reviewers:

- @efeanilaksoz
- @olilecompte

Submitted: 31 March 2025
Published: 10 July 2025

Summary

`pyScienceMode` is an open-source Python package that simplifies advanced and customizable functional electrical stimulation (FES) protocols and offers straightforward integration into research pipelines. The package supports the Rehastim2 and the P24 stimulator devices (Hasomed Inc., Magdeburg, Germany), enabling the creation of customized stimulation protocols by allowing the user to control key stimulation parameters such as frequency, pulse intensity, pulse width, and pulse train duration. Additionally, pyScienceMode supports the combined use of the MOTOmed rehabilitation bike (Reck-Technik GmbH & Co. KG, Betzenweiler, Germany) and the Rehastim2 stimulator, allowing pedal angle–based stimulation control (Figure 1), as well as direct adjustment of the bike's speed and resistance. Furthermore, pyScienceMode is designed to work with a variety of experimental tools including encoders, electromyography sensors, and force plates making it easy to build new stimulation strategies and control approach for research. By enabling an interface for multiple devices, this package helps researchers flexibly to implement and customize FES protocols for a range of clinical or experimental applications.

Statement of Need

To enhance FES rehabilitation, it is essential to provide optimized and personalized stimulation for both the patient and the performed task. Advanced control technology is needed to deliver new stimulation protocols (Ibitoye et al., 2016; Rouse et al., 2019) that can enhance motion (Molazadeh et al., 2021) or increase muscle force production (Doll et al., 2018). However, most commercially available electrical stimulators lack flexible device control, and customization is often restricted to manufacturer-provided interfaces, limiting real-time adaptation to experimental conditions. Hasomed's stimulators are widely used in research because their ScienceMode API facilitates advanced stimulation customization. Building on this API, `pyScienceMode` was developed to provide full and intuitive control of stimulator devices through the Python programming language. Furthermore, the package can be updated to integrate any new stimulation device that provides an API, ensuring ongoing adaptability and innovation in FES research. This capability is essential for testing new rehabilitation protocols and to gain a deeper understanding of the underlying mechanisms of FES.

Although similar packages exist, including a LabVIEW-based interface for the Rehastim2 device (RaviChandran et al., 2022) and a C library for the P24 stimulator (https://github.com/ScienceMode/ScienceMode4_c_library),the development of `pyScienceMode` in Python is particularly beneficial for fast prototyping, given Python's free and widely adopted programming language. pyScienceMode is the first open-source package enabling the customization of stimulation patterns for the Rehastim2 stimulator in Python. The P24 stimulator control was also included to the package (based on https://github.com/ScienceMode/ScienceMode4_python_wrapper) to provide a unified interface and a user-friendly coding environment, accessible from a simple installation procedure. It can also integrate other sensors and devices that have their own libraries via multiprocessing (example below). This will enable the scientific community to control the FES for different tasks and goals, for instance triggered onset/offset for the drop foot correction and cycling events. The package supports customization of stimulation parameters (e.g., doublets, triplets, ramp modifications) to address challenges such as muscle fatigue, pain reduction, and motion smoothness. By unifying these advanced controls in a single platform, `pyScienceMode` enables reproducible and adaptable FES interventions that help researchers to pursue their challenging research and to design innovative rehabilitation strategies.

Features

The main `pyScienceMode` features are:

- `Stimulator interfaces`: Allow for switching between stimulators easily
- `Acks`: Retrieve the device logs for debugging
- `Channel(s) configuration`: Enables the configuration of each stimulator channel and the customization of the stimulation's parameters (frequency, duration and intensity).
- `Motomed`: real-time communication between the computer and Motomed.

A Stimulation example: Electro stimulation pipeline

This example shows how to combine the `pyScienceMode` package with another library `Biosiglive` (Ceglia et al., 2023) to control the Rehastim 2 stimulator for a drop foot correction on an instrumented treadmill (left images in Figure1).

```python
import numpy as np
from biosiglive.interfaces.vicon_interface import ViconClient
from biosiglive.processing.data_processing import RealTimeProcessing
from biosiglive.gui.plot import LivePlot
from time import sleep, time
from pyScienceMode2 import Stimulator as St
from pyScienceMode2 import Channel as Ch
import multiprocessing as mp

def stream(foot_strike):
    vicon_interface = ViconClient(init_now=True)
    vicon_interface.add_device("Treadmill",
                               "generic_device",
                               rate=2000,
                               system_rate=100)
    vicon_interface.devices[-1].set_process_method(RealTimeProcessing().get_peaks)
    nb_min_frame = vicon_interface.devices[-1].rate * 10
    time_to_sleep = 1 / vicon_interface.devices[-1].system_rate
    count = 0
    force_z, force_z_process = [], []
    is_one = [False, False]

    while True:
        tic = time()
        vicon_interface.get_frame()
        data = vicon_interface.get_device_data(device_name="Treadmill")
        force_z_tmp = data[0][[2, 8], :]
        (cadence,
         force_z_process,
         force_z,
         is_one) = vicon_interface.devices[0].process_method(
                                                    new_sample=force_z_tmp,
                                                    signal=force_z,
                                                    signal_proc=force_z_process,
                                                    threshold=0.2,
                                                    nb_min_frame=nb_min_frame,
                                                    is_one=is_one,
                                                    min_peaks_interval=1300)
        if np.count_nonzero(force_z_process[:, -20:]):
            print("set")
            foot_strike.set()
            count += 1
        loop_time = time() - tic
        real_time_to_sleep = time_to_sleep - loop_time
        if real_time_to_sleep > 0:
            sleep(time_to_sleep - loop_time)

def stim(foot_strike, stimulation_delay, stimulation_duration):
    list_channels = []

    # Channel creation
    channel_1 = Ch.Channel("Single",
                           no_channel=1,
                           amplitude=50,
                           pulse_width=100,
                           stimulation_interval=33,
                           name="Soleus")
    list_channels.append(channel_1)
    stimulator = St.Stimulator(list_channels, "COM34")
    count = 0
    while True:
        foot_strike.wait()
        sleep(stimulation_delay * 0.001)
        stimulator.start_stimulation(stimulation_duration)
        print("stim_started")
        foot_strike.clear()

if __name__ == "__main__":
    stimulation_delay = 10  # ms
    stimulation_duration = 0.33  # s
    foot_strike = mp.Event()
    stim_proc = mp.Process(name="stim",
                           target=stim,
                           args=(foot_strike,
                                 stimulation_delay,
                                 stimulation_duration))
    stream_proc = mp.Process(name="stream", target=stream, args=(foot_strike,))
    stim_proc.start()
    stream_proc.start()
    stim_proc.join()
    stream_proc.join()
```

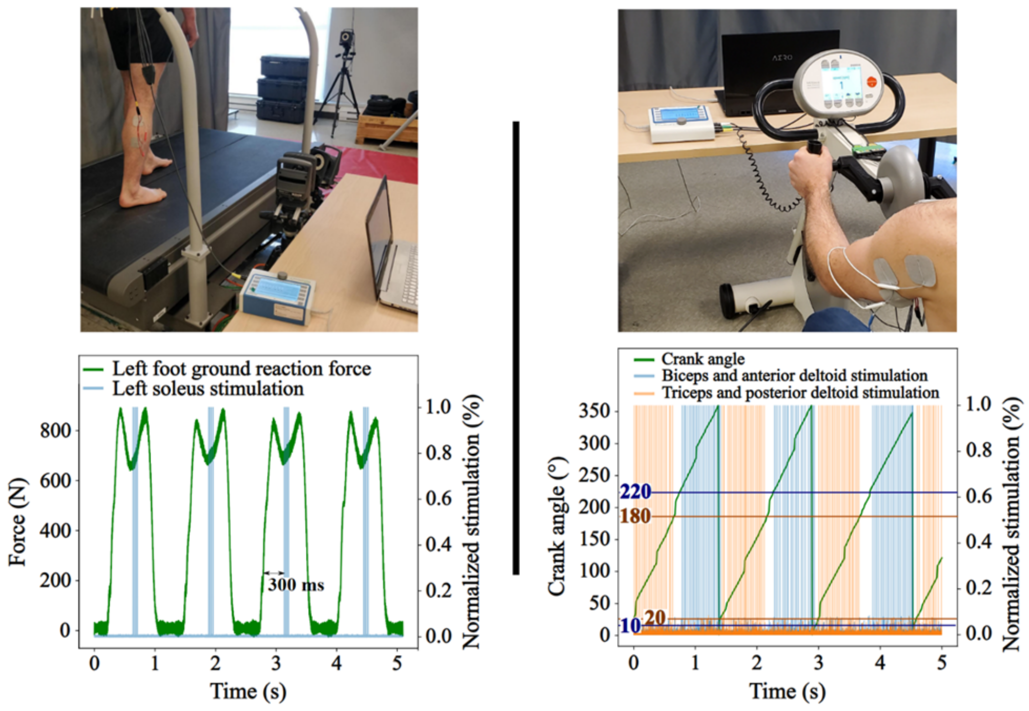

Figure 1: Figure 1

Figure 1: FES for foot drop correction on a treadmill (left) and arm cycling on a MOTOmed (right). The left panel shows the left foot ground reaction force (green) and the stimulation of the left soleus muscle (blue). The right panel displays the crank angle (green) along with the stimulation of the biceps and anterior deltoid (blue) and the triceps and posterior deltoid (orange).

Acknowledgements

The software development was supported by the "Ingénierie de technologies interactives en réadaptation" group (INTER) #160 OptiStim.

A special thanks to Arsene Baert and Benjamin Faresin for their contribution to the package development and documentation writing.

References

Ceglia, A., Verdugo, F., & Begon, M. (2023). Biosiglive: An open-source python package for real-time biosignal processing. *Journal of Open Source Software*, *8*(83), 5091. https://doi.org/10.21105/joss.05091

Doll, B. D., Kirsch, N. A., Bao, X., Dicianno, B. E., & Sharma, N. (2018). Dynamic optimization of stimulation frequency to reduce isometric muscle fatigue using a modified hill-huxley model. *Muscle & Nerve*, *57*(4), 634–641. https://doi.org/10.1002/mus.25777

Ibitoye, M. O., Hamzaid, N. A., Hasnan, N., Abdul Wahab, A. K., & Davis, G. M. (2016). Strategies for rapid muscle fatigue reduction during FES exercise in individuals with spinal cord injury: A systematic review. *PloS One*, *11*(2), e0149024. https://doi.org/10.1371/journal.pone.0149024

Molazadeh, V., Zhang, Q., Bao, X., Dicianno, B. E., & Sharma, N. (2021). Shared control of a powered exoskeleton and functional electrical stimulation using iterative learning. *Frontiers in Robotics and AI*, *8*, 711388. https://doi.org/10.3389/frobt.2021.711388

RaviChandran, N., Aw, K., & McDaid, A. (2022). *A LabVIEW interface for RehaStim 2*. https://doi.org/10.36227/techrxiv.21302865.v1

Rouse, C. A., Downey, R. J., Gregory, C. M., Cousin, C. A., Duenas, V. H., & Dixon, W. E. (2019). FES cycling in stroke: Novel closed-loop algorithm accommodates differences in functional impairments. *IEEE Transactions on Biomedical Engineering*, *67*(3), 738–749. https://doi.org/10.1109/TBME.2019.2920346