21/6/2024

OctoPyUs Simulation Package (status update)

Matteo Montani, Alessandro Longo
matteo.montani@uniurb.it alessandro.longo@uniurb.it

What's new

- A preliminary version of the OctoPyUs simulation package is ready to be downloaded and tested (Valerio, Lucia, Paolo already contacted)

What is available:
- Components: Mass, Beam, Blades, Spring
- Connections: Series, Parallel, Derivation, Ground
- Dynamical load computation
- Automatic topology solving

- A preliminary documentation has been uploaded on the TDS (VIR-0584A-24)

Next steps
- Controls
- Integrate in official gitlab repository
- Tutorial video

How to test the simulation package

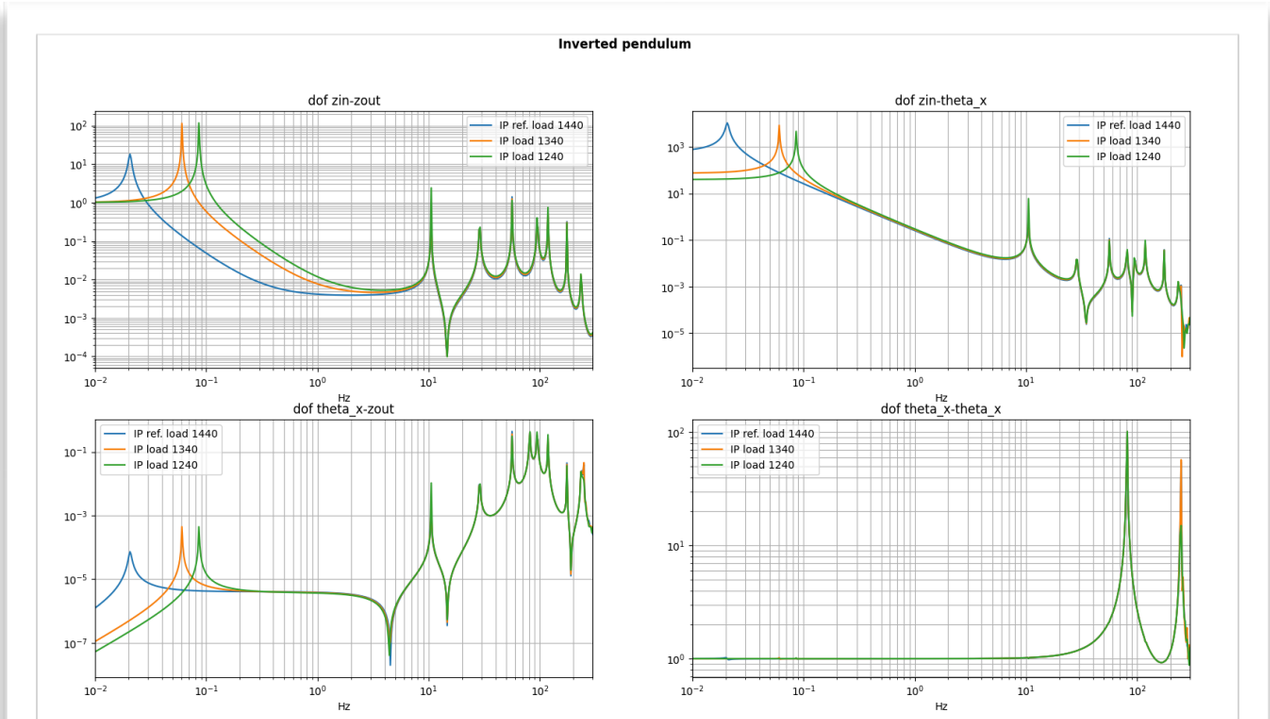
- Download the .zip file **octopyus_pack_v0.0.1.zip** at the following **link**
- Unzip in a dedicated folder (e.g. OCTOPYUS). There are three folders (**/data, /dist, /notebook**)
- From inside OCTOPYUS folder where the .zip file has been unzipped

pip3 install ./dist/octopyus-0.0.1.tar.gz

- To test the software: **cd notebook** (contains example scripts), then

python example.py or **python example2.py**

- The second example will simulate an inverted pendulum with different loads in the z θ degrees of freedoms (see Figure)
- A file **ip.png** should be now present in **data/figure** while in **data/simulation** there should be a file **IP1.pkl** with simulation data
- The folder **data/json** contains the .json files used for the simulation i.e., the description of the simulated elements



example2.py: IP under different loads

example2.py

```
import logging
import octopyus.engine.simulation as SIM
import octopyus.engine.dataview as DW

#set the log level during the simulation process: FATAL,
CRITICAL, ERROR, WARNING, DEBUG
logging.basicConfig(level=logging.WARNING)

#set the path of directories
jsonfilepath = "../data/json/"
simfilepath = "../data/simulation/"

#simulation
sim1 = SIM.simulate(jsonfilepath + "IP.json")

sim2 = SIM.simulate(jsonfilepath + "IP.json",
params=[{"elName": "susp_load", "param_list": {"mass": 1340}}])

sim3 = SIM.simulate(jsonfilepath + "IP.json",
params=[{"elName": "susp_load", "param_list": {"mass": 1240}}])

#save simulation
SIM.savesim(sim1, simfilepath + "IP1")
```

IP.json

- .json file containing names of elements and nodes (and parameters of the system) to be simulated. See documentation

Classes

- SIM, simulation method
- DW, plot results

- Based on nodes names (in - out) in the json file, the right connections are automatically implemented

- Parameters can be modified (e.g. mass) with **params**

- **With one file json different configurations can be simulated**

example2.py

```
#PLOT creation
data = DW.dataview(sim1)
fig, ax = DW.dataview.new_figure(title="Inverted pendulum", subplots=[2, 2], figsize=(10, 5))

label = label1

data.plotTF(ax[0,0], dof_in=2, dof_out=2, label=label, type_of_tf='XoXi', title="dof zin-zout")
data.plotTF(ax[0,1], dof_in=2, dof_out=3, label=label, type_of_tf='XoXi', title="dof zin-theta_x")
data.plotTF(ax[1,0], dof_in=3, dof_out=2, label=label, type_of_tf='XoXi', title="dof theta_x-zout")
data.plotTF(ax[1,1], dof_in=3, dof_out=3, label=label, type_of_tf='XoXi', title="dof theta_x-theta_x")

if "sim2" in locals():
    label = label2
    data2 = DW.dataview(sim2)
    data2.plotTF(ax[0,0], dof_in=2, dof_out=2, label=label, type_of_tf='XoXi', title="dof zin-zout")
    data2.plotTF(ax[0,1], dof_in=2, dof_out=3, label=label, type_of_tf='XoXi', title="dof zin-theta_x")
    data2.plotTF(ax[1,0], dof_in=3, dof_out=2, label=label, type_of_tf='XoXi', title="dof theta_x-zout")
    data2.plotTF(ax[1,1], dof_in=3, dof_out=3, label=label, type_of_tf='XoXi', title="dof theta_x-theta_x")

if "sim3" in locals():
    label = label3
    data3 = DW.dataview(sim3)
    data3.plotTF(ax[0,0], dof_in=2, dof_out=2, label=label, type_of_tf='XoXi', title="dof zin-zout")
    data3.plotTF(ax[0,1], dof_in=2, dof_out=3, label=label, type_of_tf='XoXi', title="dof zin-theta_x")
    data3.plotTF(ax[1,0], dof_in=3, dof_out=2, label=label, type_of_tf='XoXi', title="dof theta_x-zout")
    data3.plotTF(ax[1,1], dof_in=3, dof_out=3, label=label, type_of_tf='XoXi', title="dof theta_x-theta_x")

#save the picture in a png file
DW.dataview.save_figure(fig, "../data/figure/ip.png")

#show the result (this function is blocking)
DW.dataview.show_figure(fig)
```

Dataview

- Script example2.py also generate a plot of the TF
- plotTF: Matplotlib to generate, show and save plots

Preliminary Documentation

PRELIMINARY

OctoPyUs simulation results: Inverted pendulum and standard filter

Matteo Montani[a,b], Alessandro Longo[a,b], Paolo Ruggi[c]

[a] *Università degli Studi di Urbino "Carlo Bo", I-61029 Urbino, Italy*
[b] *INFN, Sezione di Firenze, I-50019 Sesto Fiorentino, Firenze, Italy*
[c] *European Gravitational Observatory (EGO), I-56021 Cascina, Pisa, Italy*

matteo.montani@uniurb.it, alessandro.longo@uniurb.it, paolo.ruggi@ego-gw.it

1. Introduction
2. Impedance Matrix Method
3. Impedance of elementary components
4. Simulated objects (IP, standard filter)
5. Overview of the software
6. Simulation results
7. Appendix

5.4) How to start using the preliminary version of the software

Contents

1 Introduction 4

2 Impedance Matrix Method Overview 6
2.1 Connection types . . . 6
2.2 Transfer function computation . . . 7

3 Impedance of elementary components 8
3.1 Mass element . . . 8
3.2 Spring element . . . 9
3.3 Impedance of a beam in three dimensions . . . 9
3.3.1 Beam impedance in the horizontal directions x-z: BEAM function . . . 10
3.3.2 Beam thin approximation in the horizontal directions x-z . . . 10
3.3.3 Impedance of a wire/beam in the vertical direction . . . 12
3.3.4 Beam impedance in three dimensions . . . 13
3.4 Displacement outside the center of mass . . . 14

4 Simulated objects 16
4.1 Inverted Pendulum . . . 16
4.2 Physical parameters of the inverted pendulum . . . 17
4.3 Seismic Filter and Standard Filter Approximation . . . 18
4.4 Physical parameters of the standard filter . . . 19

5 Overview of OctoPyUs simulation package 21
5.1 Software structure overview . . . 21
5.2 The input JSON file . . . 22
5.2.1 netlist . . . 23
5.2.2 WIRE element . . . 23
5.2.3 BLADE element . . . 24
5.2.4 LIBRARY element . . . 25
5.2.5 Connections . . . 32
5.2.6 globalProprieties . . . 32
5.3 JSON files for IP and standard filter simulation . . . 33
5.4 How to install the simulation package of OctoPyUs . . . 40

6 Simulation Results 41

A Appendix 45
A.1 Formula for the Parallel connection . . . 45
A.2 Parameters for the wire and beam impedance (Wset function) 47
A.3 Blade Impedance . . . 49