MCsim Python toolbox

Mathias Marley, Department of Marine Technology, NTNU

February 17, 2022

1 Introduction

This note is intended for users and developers of the Marine Cybernetics simulation toolbox in Python (MCsim Python).

In accordance with Python terminology, we refer to a **script** for a Python file that runs on its own, and a **module** for a Python file containing a library of functions. The toolbox consists of

- **lib**: Library of generic functions for marine vessels and control systems.
- **models**: Aggregated models for simulating vessel and system responses.
- **demos**: Packages of parameterized vessel or system models for demonstrating simulation of complete systems.

2 Models

Models are placed in subfolders of the **models** folder. Each model subfolder contains

- Model function library: a Python module of functions simulating the model dynamics. This may be a single function, that uses the generic functions in the **lib** folder as subfunctions.
- Model data: a file containing model parameters, stored using pickle. For general models, an example set of model data shall be provided.
- Example initialization script.
- Documentation.

2.1 Model functions

The dynamics of a model are given by a differential equation

$$\dot{x}(t) = f(x(t), u(t), w(t), p) \tag{1}$$

with state x, control input u, disturbance input w and fixed parameters p. Eq. (1) is implemented as a function

dx = dot_model(x,u,w,par1,...,parN),

which takes the system states, inputs and parameters as arguments, and return the derivative of the state vector.

The dynamics of a model may also be described by the integral equation

$$x(t_{i+1}) = \int_{t_i}^{t_{i+1}} f(x(\tau), u(t_i), w(t_i), p) d\tau, \tag{2}$$

where t_i is the current time step, and t_{i+1} is the next time step. Eq (2) is implemented as a function

x_next = int_model(x,u,w,par1,...,parN),

that includes single time step numerical integration of the system dynamics.

3 Guidelines for developers

To ensure consistency in the format of the toolbox, please use existing models and modules as template when developing new models and modules.

If developing generic models (i.e. not specific for a vessel or similar), a set of example model data should be included.

3.1 Programming tips

As far as possible, abide by the PEP 8 style rules [2]. If you are coming from MATLAB, useful tips are given in [1]. Highlighted tips are listed below:

- Do not use numpy.matrix objects or associated functions.
- Avoid nested numpy arrays. Ensure that functions do not unintentionally return nested numpy arrays.
- For vectors: be aware of the difference between 1D arrays (vectors without orientation), and 2D vertical or horizontal vectors, i.e. 2D arrays with shape (n,1) or (1,n). Extracting a column from a 2D array returns a 1D array. For this reason, aim at using 1D arrays to represent vectors.

3.2 Documentation

3.2.1 Models

All models shall be documented, although the documentation need not be exhaustive. It is often sufficient with an overview of the theory, and source of the model data. As an example, for a vessel model, the documentation should state which hydrodynamic load theory is used, and describe how the vessel data is obtained.

3.2.2 Function library

Documentation is not needed for functions that are considered standard within marine cybernetics. More advanced modules should be documented, e.g. with reference to a relevant paper.

3.3 Testing

All code must be tested, to verify that they produce the expected output. A statement of the degree of testing should be included in the header of each module/script, and in some cases also in the header of each function within a module. More advanced code may warrant a separate test script, placed in the **testing** subfolder of the **lib** folder.

As an example of the latter, consider a function generating an irregular sea state from a wave spectrum. If you generate a sufficiently long realization of the sea state, taking the inverse Fourier transform of the time series should recreate the wave spectrum, thus ensuring correctness of code. If you write a script to perform such a test, please include it in the **testing** folder.

References

[1] *numpy for MATLAB users.* `https://numpy.org/doc/stable/user/numpy-for-matlab-users.html`.

[2] *PEP 8 Python style guide.* `https://www.python.org/dev/peps/pep-0008/`.