python

Jump to bottom

Jorn Bruggeman edited this page on Jul 5 · 61 revisions

FABM comes with a "pyfabm" package for the Python programming language. This package enables you to access FABM's biogeochemical models directly from Python. For instance, you can enumerate a model's parameters and variables, or obtain temporal derivatives and diagnostics for any given environment and model state. In combination with a Python-based time integration scheme (e.g., scipy.integrate.solve_ivp), this also allows you to perform model simulations.

You can try pyfabm online without installing anything:

launch binder

Build and install

First, obtain the FABM code from GitHub. You can either download the latest official release of the source code as a zip or tar.gz file, or check out the current developer's version of the code from its git repository (recommended!)

You will need:

- CMake, version 3.0 or later
- Python 2.7 or Python 3.4 or later
- pip (installed by default with Python 2 ≥ 2.7.9 or Python 3 ≥ 3.4)
- wheel (install with `python -m pip install --user wheel`)

On Linux-like systems, pyfabm is built and installed by creating a new directory for building the code, executing cmake with the path to the source code, and then running "make install". For instance:

```
mkdir -p ~/build/python-fabm
cd ~/build/python-fabm
cmake <FABMDIR>/src/drivers/python
make install
```

The above `make install` creates the pyfabm package within the build directory, packs it in wheel format, and then calls pip to install it (by default pyfabm is installed in the per user site-packages directory). To skip the installation step, use `make all` instead of `make install` . You can then still import pyfabm if you first add the build directory to your Python path (`sys.path.insert(0, '<BUILDDIR>')` .

CMake will try to locate your default Python interpreter to create the pyfabm wheel and call pip. If CMake does not select the correct Python interpreter, you will need to set the `PYTHON_EXECUTABLE` variable to the full path of your Python interpreter. To do so when using cmake on the command line, add `-DPYTHON_EXECUTABLE=<PYTHONPATH>` ; when using cmake-gui or ccmake, edit this variable in the user interface, then configure and generate.

After the pyfabm package has been built and installed, you can access it from python by typing `import pyfabm` . Please try this before continuing.

Examples

In addition to examples below, pyfabm comes with several Jupyter Notebooks. These can be found at `<FABMDIR>/testcases/python`. You can also try them online: launch binder

Enumerating variables

```
import pyfabm
model = pyfabm.Model("fabm.yaml")
for variable in model.state_variables:
   print(f"  {variable.name} = {variable.long_name} ({variable.units})")
```

Run a simulation

```
import scipy.integrate
from matplotlib import pyplot
import pyfabm

# Create model (loads fabm.yaml)
model = pyfabm.Model()

# Configure the environment
# Note: the set of environmental dependencies depends on the loaded biogeochemical model.
model.cell_thickness = 10.0
model.dependencies["surface_downwelling_photosynthetic_radiative_flux"].value = 50.0
model.dependencies["downwelling_photosynthetic_radiative_flux"].value = 25.0

# Verify the model is ready to be used
assert model.start(), "One or more model dependencies have not been fulfilled."

# Time derivative
def dy(t, y):
    model.state[:] = y
    return model.getRates()

# Time-integrate over one year (note: FABM's internal time unit is seconds!)
result = scipy.integrate.solve_ivp(dy, [0.0, 365.0 * 86400], model.state)

# Plot results
fig, ax = pyplot.subplots()
ax.plot(result.t / 86400, result.y.T)
ax.legend([variable.path for variable in model.state_variables])
pyplot.show()
```

Included utilities

pyfabm comes with a number of utilities:

- **fabm_complete_yaml.py** takes a FABM configuration (fabm.yaml) and adds descriptions of all parameters and variables (e.g., long names, units) in comments at the end of each line.
- **fabm_describe_model.py** takes a FABM configuration (fabm.yaml) and writes out a list of all variables contained in the resulting model.
- **fabm_evaluate.py** evaluates a FABM configuration (fabm.yaml) for a particular set of inputs (state variables, environmental dependencies). The values of these inputs can be provided in a YAML file with key: value pairs, in a NetCDF file or on the command line with `-v` / `--values`. This script can be used to diagnose model behaviour under circumstances that are known to cause problems, e.g., they crash 3D simulations.
- **fabm_stress_test.py** takes a FABM configuration (fabm.yaml) and verifies that the resulting biogeochemical model returns valid derivatives under a wide variety of inputs (state variables, environmental dependencies). Different tests can be run, using either random values or extremes for inputs, and running randomized or exhaustive tests.

`make install` puts all above utilities in your script directory. This directory may be already in your PATH: try running `fabm_complete_yaml.py -h` (Linux, Mac) or `fabm_complete_yaml -h` (Windows) on the command line. If this fails, you will need to add the script directory to your path. Typically, it is `~/.local/bin` on Linux and Mac, and `%APPDATA%\Python\<PYTHON_VERSION>\Scripts` on Windows.

Each of the utilities can be run on the command line with `-h` as argument; this will list information about the script and its command line arguments.

Should you wan to use these scripts as example of how to use pyfabm: they can be found at `<FABMDIR>/src/drivers/python/pyfabm/utils`.

Build and test passing DOI 10.5281/zenodo.7737951

Clone this wiki locally

```
https://github.com/fabm-model/fabm.wiki.git
```