PyADAPT Tutorial

ADAPT was originally writen in MATLAB and now we at CBio group would like to write it in python a.k.a, pyADAPT. There are mainly three classes in the python package, `DataSet`, `Model`, and `ADAPT`. They are organized in basically the same way as the MATLAB version.

Getting started

Python

To use pyADAPT, one should have some basic knowledge about python programming. Python is a dynamic programming language and is very similar to MATLAB (in which ADAPT was written originally). These are some good resources to learn python. I would recommend learning list comprehension, lambda expression and object oriented programming (class definition) **besides** basic syntax.

- lambda expression:
  - used a lot in ODE solvers and optimizers
  - https://docs.python.org/3/reference/expressions.html#lambda
- object oriented programming:
  - how to define a model for example the toy model
  - https://python.swaroopch.com/oop.html
- list comprehension
  - used a lot in this project for simplicity
  - https://www.python-course.eu/python3_list_comprehension.php

Scipy suite

Python is free and open source. Unlike MATLAB, which is built to do scientific computing, python is a general purpose programming language. One can use it to write HTTP servers, image processing apps or instant messagers. But researchers are more interested in using python to do what MATLAB could do. Luckily, there's scipy suite for this purpose. It provides an alternative for almost everything in MATLAB. To make this tutorial more specific to pyADAPT, only those tools used will be covered.

- matplotlib: plotting in python
- scipy.integrate.solve_ivp: `ode15s` in python, solve a initial value problem
- scipy.optimize.least_squares: `lsqnonlin` in python, optimze the paramters using Levenberg–Marquardt algorithm. I am not using it directly but through a wrapper called `lmfit.minimize`.

It is useful to check out these third-party libs briefly before writting you own model.

pyADAPT

Either you get the source code of `pyADAPT` by downloading an archive or by cloning it from my github repo, what you get on should be a folder containing a file named "setup.py". Change to this directory in the terminal and type in:

```
pip install -e .
```

`pip` will read the rules defined in "setup.py" and handle all the dependencies for you. This will install pyADAPT as a package, along with all the dependencies such as numpy and scipy, to your python environment in editable mode (what `-e` stands for). Make sure you have pip executable from the proper environment included in the path of the operating system. Python from official website or other scientific distribution should both work fine.

Then you can try the toy model by running the script "run_toy.py". Hopefully there's no error.

DataSet

DataSet class contains the experimental data we obtained from different sources. It is organized into a list of all the states and fluxes in the dataset. And It is able to produce a spline interpolation evaluated at different time intervals.

To instantiate DataSet, a raw data and a corresponding data description file should be provided.

```
data = DataSet(raw_data_path='data/toyModel/toyData.mat',
            data_specs_path='data/toyModel/toyData.yaml')
```

Since there's no way to standardize the way raw data is stored and defined, it is also possible for users to first read the raw data and the data specs as python dictionaries and the use them to instantiate the dataset.

Currently, this DataSet class meets all the needs of the toy model. For more complicated models which contains more than one measurement, such as the clamp model with four measurments, class `DataSets` will be a future solution.

Model

`Model` is a abstract class that provide a convenient interface to define an ADAPT style model. The goal is to keep the definition of models in both ADAPT and pythonic way. In order to define a new model `NewModel`, one should first inherit it from `NewModel(Model)`, add parameters, constants, states in `__init__` method and extend three important methods: `odefunc`, `reactions` and `inputs`. Check `pyADAPT.models.toy_model.ToyModel` for example.

odefunc

`odefunc` will be passed as an argument of `scipy.integrate.solve_ivp`. There's not much restriction on how `odefunc` should be implemented as long as the it takes time, initial values and paramters as the arguments. The `Model` class works by remembering the order of the states and parameters. Thus, `odefunc` should always return the derivatives in the same order as the initial values (`x0`).

reactions

Reactions are actually any arbitrary expression that uses the components of the model to obtain a value. In the MATLAB implementation, it can be either real reaction fluxes or a interesting intermediate value in the simulation that we would like to keep track of. In some models, the model outcome is fitted to the experimentally measured fluxes. I will implement this function when dealing with clamp model.

inputs

Inputs is not useful in the toy model but is important in the clamp model. It's function is overlapping with the constants in the model, so further inspectation is needed. I will work on it after I learn more about the clamp model.

ADAPT

ADAPT class contains all the procedure to perform an ADAPT simulation. It takes a model instance and a dataset instance as the argumnets. And it will calculate the parameter trajectory. It supports parallel computing with python multiprocessing module. In an ADAPT simulation, there are usually many iterations, class `ADAPT` is able to run multiple iterations at a time by assigning them to different proceses.