### pypet: Python Parameter Exploration Toolkit

Robert Meyer, Klaus Obermayer  
Neural Information Processing Group Technische Universität Berlin &  
Bernstein Center for Computational Neuroscience Berlin  
Robert.meyer@mi.tu-berlin.de

**pypet**

pypet is a new multi-platform python toolkit for management of simulations and storage of numerical data.

No longer waste your time writing I/O functionality to serialize the results and parameter settings of your numerical experiments. Put your data into the novel Trajectory container and pypet handles storage into HDF5 [1] files for you. For instance, let pypet help you explore and analyze different parameter configurations of a neural network model.

### Features

- **Novel tree container** Trajectory for managing parameters and results
- **Sort** your parameters and results into groups and categories
- **Access data via natural naming.** e.g. traj.parameters.network.neurons.Vm
- **Automatic storage** of simulation data into HDF5 [1] files via PyTables [2]
- **Support for many data formats**
  - python native data types, lists, dictionaries, etc.
  - NumPy arrays and SciPy sparse matrices
  - pandas DataFrames [3]
  - BRIAN Neural Network Simulator quantities and monitors [4]
  - and more
- **Easily extendable** to other data formats
- **Easy exploration** of the parameter space and parameter ranges
- **Support for multiprocessing**, pypet can run your simulations in parallel
- **Dynamic loading**, load only the parts of your data you need
- **Annotate your data**
- **Git integration**, let pypet make automatic commits of your codebase
- and many more

1. HDF5: [http://www.hdfgroup.org/HDF5](http://www.hdfgroup.org/HDF5)  
3. pandas: [http://pandas.pydata.org](http://pandas.pydata.org)  
4. BRIAN: [http://briansimulator.org](http://briansimulator.org)

### Example Code Snippet

```python
from pypet.environment import Environment
from pypetutils.explore import cartesian_product

def multiply(traj):
    """Example of a sophisticated numerical experiment that involves multiplying two integer values."
    :param traj:
        Trajectory containing the parameters in a particular combination, it also serves as a container for results.
    ...

    z = traj.x * traj.y
    traj.add_result('z', z, comment = 'Result of x * y')

    # Create an environment that handles running the experiment
    env = Environment(trajectory = 'Multiplication',
                       filename = 'multiply.hdf5',
                       comment = 'A sophisticated simulation of multiplication')

    # The environment provides a trajectory container for us
    traj = env.trajectory

    # Add two parameters, both with default value 0
    traj.add_parameter('x', 0, comment = 'First dimension')
    traj.add_parameter('y', 0, comment = 'Second dimension')

    # Explore the Cartesian product of x in {1,2,3,4} and y in {6,7,8}
    traj.explore(cartesian_product( {'x': [1,2,3,4], 'y': [6,7,8] } ))

    # Run simulation function 'multiply' with all parameter combinations
    env.run(multiply)
```

### Control Flow

Distributes to individual run of user's simulation  

Environment  
Runs experiments, Multiprocessing, Logging,  
Saves and loads

### StorageService  
Write & loads HDF5 files

### Where to get it?

Documentation: [http://pypet.readthedocs.org](http://pypet.readthedocs.org)  
Releases: [http://pypi.python.org/pypi/pypet](http://pypi.python.org/pypi/pypet)  
Sourcecode: [http://github.com/SmokinCaterpillar/pypet](http://github.com/SmokinCaterpillar/pypet)

### Acknowledgements

pypet was created at the Neural Information Processing Group TU Berlin and supported by the Research Training Group GRK 1589/1 BCCN Berlin.