MPCPy User Guide Documentation

Release 1

David H. Blum

February 27, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

MPCPy facilitates the testing and implementation of Model Predictive Control (MPC) for building systems. The software package focuses on the use of data-driven simplified physical or statistical models to predict building performance and optimize control. Four main modules contain object classes to import data, interact with a real or emulated system, estimate and validate data-driven models, and optimize control inputs. Three other modules contain classes to help track units and provide additional, mainly internal, functionality to MPCPy.

1. **ExoData** classes collect external data and process it for use within MPCPy.
2. **System** classes represent real or emulated systems to be controlled, collecting measurements from and providing control inputs to the systems.
3. **Models** classes represent system models for MPC, managing model simulation, estimation, and validation.
4. **Optimization** classes formulate and solve the MPC optimization problems using Models objects.
5. **Variable** and **Unit** classes together maintain the association of static or timeseries data with units.
6. **Utility** classes provide functionality needed across modules and for interactions with external components.

CHAPTER

TWO

GETTING STARTED

Dependencies

MPCPy takes advantage of many third-party software packages, listed below. It has been tested on Ubuntu 16.04.

Python Packages

- matplotlib 1.5.1
- numpy 1.11.0
- pandas 0.17.1
- python-dateutil 2.4.2
- pytz 2014.10
- scikit-learn 0.18.1
- tzwhere 2.3
- sphinx 1.3.6
- estimationpy

Modelica Compiler and Optimizer, FMU Simulator

- JModelica 1.17

Modelica Packages

- Modelica Standard Library 3.2.2
- Modelica Buildings Library 3.0.0

Installation

1. Install all dependencies listed above according to their respective processes.
2. Create the following environmental variables, where ".../" is replaced by the full directory:
   - JMODELICA_HOME = ".../Jmodelica-1.17"
   - IPOPT_HOME = ".../Ipop-3.12.5"
   - SUNDIALS_HOME = ".../Jmodelica-1.17/ThirdParty/Sundials"
   - CPPAD_HOME = ".../Jmodelica-1.17/ThirdParty/CppAD/"
   - SEPARATE_PROCESS_JVM = ".../jvm/java-8-openjdk-amd64/"

- JAVA_HOME = ".../jvm/java-8-openjdk-amd64/"

3. Add the following to the PYTHONPATH environmental variable, where ".../" is replaced by the full directory:
   - ".../Jmodelica-1.17/Python"
   - ".../Jmodelica-1.17/Python/pymodelica"
   - ".../MPCPy"
4. Add the Modelica Standard Library and Modelica Buildings Library to the MODELICAPATH environmental variable.
5. Test the installation and explore MPCPy use-cases by running the unit tests.

Run Unit Tests

The script bin/runUnitTests.py runs the unit tests of MPCPy. By default, all of the unit tests are run. An optional argument -s [module.class] will run only the specified unit tests module or class.

To run all unit tests from command-line, use the command (shown from the parent directory):

```
> python bin/runUnitTests
```

To run only unit tests in the module test_models from command-line, use the command (shown from the parent directory):

```
> python bin/runUnitTests -s test_models
```

To run only unit tests in the class Estimate_Jmo from the module test_models from the command-line, use the command (shown from the parent directory):

```
> python bin/runUnitTests -s test_models.Estimate_Jmo
```

CHAPTER

THREE

UNITS

CHAPTER

FOUR

VARIABLES

CHAPTER

FIVE

UTILITY

CHAPTER

SIX

EXODATA

CHAPTER

SEVEN

SYSTEMS

CHAPTER

EIGHT

MODELS

CHAPTER

NINE

OPTIMIZATION

CHAPTER

TEN

OCCUPANT