# ModelClass User Manual

Fernando Nóbel Santos Navarro (fersann1@upv.es)

September 9, 2020

## Contents

**1 Manual version** **1**

**2 Introduction to ModelClass** **1**

**3 Extends** **1**
3.1 Usage . . . . . 2
3.2 Example . . . . . 2

## 1 Manual version

This manual was written for the version **0.4.1** of ModelClass. Please check your local version of ModelClass using the following command:

```
ModelClass.version()
```

, then if your local version does not match, please upgrade to the latest version in the GitHub repository.

## 2 Introduction to ModelClass

ModelClass is a MATLAB class which simplifies working with ODE models. The main objetive is to simplify the process of coding and simulating an ODE model, and therefore reducing the time spent in this task. With ModelClass one can program ODE models from the symbolic equations and then simulate directly. This class provides also some functionality like OpenModelica (i.e. extendable classes, simulation of DAE models, etc).

Apart from that, ModelClass provides us more classes for different tasks. For example, there is a class for mathematical analysis that can calculate equilibrium points, linearize the model, and calculate eigenvalues from the model defined in ModelClass. On other hand, there is class for a contractivity test to check whether a model is contractive or no.

Lastly it is even possible to define ModelClass models from chemical reactions directly and then perform QSSA analysis and simulate.

For more information please contact *fersann1@upv.es*.

## 3 Extends

The keyword `extends` allows us to create new ModelClass models that extends the functionality of a base class.

When we use the keyword `extends` with the filename of a base model, the parser of the ModelClass will read all the data in the base model and it will add it to the model which we are working on. So it can be understood as pasting all the data from the base model to the new one.

This way we can reuse model for building new models that expand the original functionality, and it a change is made in the original model, it will be also done in the extended models. This helps the maintenance of the models and reduces the time spent programming.

## 3.1 Usage

When defining a ModelClass `.mc`, we can use the following structure for extending a base model:

```
extends /path/to/filename.mc;
```

, where `/path/to/filename.mc` it the path with the name of the model we want to extend (it is possible to use a relative path or a absolute path). For example:

This command will extend the file `model.mc` present in the working directory.

```
extends ./model.mc;
```

This command will extend the file `model.mc` present in the parent directory of the working directory.

```
extends ../model.mc;
```

This command will extend the file `model.mc` present in the path `/home/user/MATLAB/`.

```
extends /home/username/MATLAB/model.m;
```

NOTE: It is best to declare the `extends` at the beginning of the ModelClass model.

## 3.2 Example

In `./examples/ex1_extends` there is a example code for using the `extends` keyword.

The example starts with a `baseModel.mc` and we would like to expand its functionality by adding a equation to show the reference dynamically.

The `baseModel.mc` defines the following variables and equations:

```
% Variables

Variable x1;
Variable x2;
Variable x3;

% Parameters

Parameter k1;
Parameter k2;
Parameter k3;
Parameter d1;
Parameter d2;
```

```
Parameter d3;
Parameter gamma12;

% Equations

Equation der_x1 == k1    - gamma12*x1*x2 - d1*x1;
Equation der_x2 == k2*x3 - gamma12*x1*x2 - d2*x2;
Equation der_x3 == k3*x1 - d3*x3;
```

And the `extendeModel.mc` is defines the variable and equation needed for the reference:

```
% I would like to represent the reference of the baseModel.mc dynamically.

% First, extend the functionality defined in baseModel.mc.
extends ./baseModel.mc;

% Then, add a variable for the reference.
Variable ref;

% And add the equation to calculate the reference value.
Equation ref == k3/d3;
```

Notice how the equation for the reference `ref == k3/d3` makes use of parameters previously defined in the `baseModel.mc`. The extended model has access to all the information defined in the base model.

Then if we execute the following command:

```
m = loadModelClass('extendedModel');
```

, the file `extendedModel.m` is autogenerated, and it will contain all the model information defined by `baseModel.mc` and `extendedModel.mc`. Here is shown the contents of `extendedModel.m`:

```
classdef extendedModel < ModelClass
    methods
        function [obj] = extendedModel()
            v = VariableClass('x1');
            obj.addVariable(v);

            v = VariableClass('x2');
            obj.addVariable(v);

            v = VariableClass('x3');
            obj.addVariable(v);

            p = ParameterClass('k1');
            obj.addParameter(p);

            p = ParameterClass('k2');
            obj.addParameter(p);

            p = ParameterClass('k3');
            obj.addParameter(p);
```

```
            p = ParameterClass('d1');
            obj.addParameter(p);

            p = ParameterClass('d2');
            obj.addParameter(p);

            p = ParameterClass('d3');
            obj.addParameter(p);

            p = ParameterClass('gamma12');
            obj.addParameter(p);

            e = EquationClass('');
            e.eqn = 'der_x1 == k1    - gamma12*x1*x2 - d1*x1';
            obj.addEquation(e);

            e = EquationClass('');
            e.eqn = 'der_x2 == k2*x3 - gamma12*x1*x2 - d2*x2';
            obj.addEquation(e);

            e = EquationClass('');
            e.eqn = 'der_x3 == k3*x1 - d3*x3';
            obj.addEquation(e);


            v = VariableClass('ref');
            obj.addVariable(v);

            e = EquationClass('');
            e.eqn = 'ref == k3/d3';
            obj.addEquation(e);

        obj.checkValidModel();
        end
    end
end
```

, where we can confirm that the base model has been successfully extended.

Finally, at this point we can work as usually with the model `extendedModel.mc`, and we could even extended it to create a `extendedExtendedModel.md`!