[Basic Equations](http://book.xogeny.com/behavior/equations/)

[Getting Physical](http://book.xogeny.com/behavior/equations/physical/)



[http://book.xogeny.com/_static/images/TitleHeading.png](http://book.xogeny.com/)

Release: v0.3.6-1-gc28d7f1-Early Access

[Basic Equations](http://book.xogeny.com/behavior/equations/)

**Simple First Order System**

Table Of Contents

* [Basic Equations](http://book.xogeny.com/behavior/equations/)
  + [Simple First Order System](http://book.xogeny.com/behavior/equations/first_order/)
    - [Adding Some Documentation](http://book.xogeny.com/behavior/equations/first_order/#adding-some-documentation)
    - [Initialization](http://book.xogeny.com/behavior/equations/first_order/#initialization)
    - [Experimental Conditions](http://book.xogeny.com/behavior/equations/first_order/#experimental-conditions)

Simple First Order System

Let us consider an extremely simple differential equation:

x˙=(1−x)

Looking at this equation, we see there is only one variable, x. This equation can be represented in Modelica as follows:

**model** **FirstOrder**

Real x;

**equation**

der(x) = 1-x;

**end** **FirstOrder**;

This code starts with the keyword model which is used to indicate the start of the model definition. The model keyword is followed by the model name,FirstOrder. This, in turn, is followed by a declaration of all the variables we are interested in.

Since the variable x in our equation is clearly meant to be a continuous real valued variable, its declaration in Modelica takes the form Real x;. The Realtype is just one of the types we can use (a more complete description of the various possibilities will be discussed in the upcoming section on *[Variables](http://book.xogeny.com/behavior/equations/variables/" \l "variables)*).

Once all the variables have been declared, we can begin including the equations that describe the behavior of our model. In this case, we can use the deroperator to represent the time derivative of x. Thus,

der(x) = (1-x)

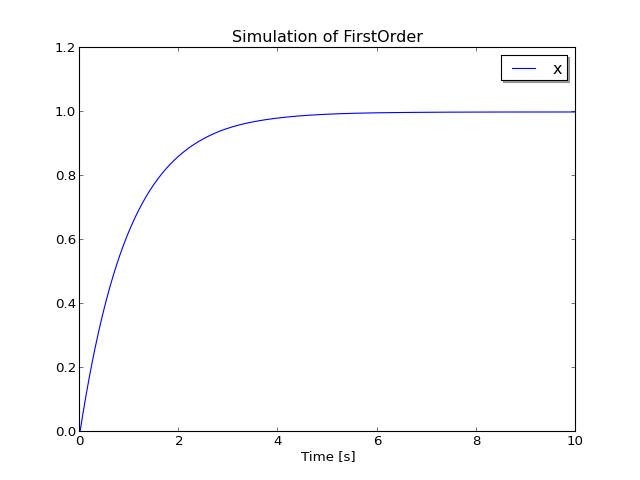
is equivalent to:

x˙=(1−x)

Unlike most programming languages, we don’t approach code like this as a “program” that can be interpreted as a set of instructions to be executed one after the other. Instead, we use a Modelica compiler to transform this model into something that we can simulate. This simulation step essentially amounts to solving (usually numerically) the equation and providing a solution trajectory like this:

([Source code](http://book.xogeny.com/plots/FO.py))

 Simulate FirstOrder in your browser



This gives you the first initial hint at one of the compelling aspects about using a modeling language to describe mathematical behavior. We didn’t need to describe how to solve this differential equation. The focus is entirely on behavior. As we work our way through more complex examples, we will see that**much of the tedious work involving the solution process is handled automatically by the Modelica compiler**.

Adding Some Documentation

Now that we’ve solved this simple mathematical equation, let’s turn our attention briefly to how we can make the model a bit more readable. Consider the following model:

**model** **FirstOrderDocumented** "A simple first order differential equation"

Real x "State variable";

**equation**

der(x) = 1-x "Drives value of x toward 1.0";

**end** **FirstOrderDocumented**;

Note the quoted text in this model.

It is important to understand that the quoted text blocks, called “strings” in computer science, are **not** comments. They are “descriptive strings” and, unlike comments, they cannot be added in arbitrary places. Instead, they can only be inserted in specific places to provide additional documentation about the elements of the model they are associated with.

For example, the first string “A simple first order differential equation” is used to describe what this is a model of. Note how it follows the name of the model. If you wish to include such documentation about the model, it must appear immediately after the model name as shown.

As we will see later, this kind of documentation can be used by tools in many ways. For example, when searching for models, a tool may use these descriptive strings when identifying matches. This text may also be associated with a graphical representation of the models. And, of course, this kind of documentation is very useful for anybody reading the model.

As this example demonstrates, there are other places that the descriptive text can appear. For example, it may be included after the declaration of a variable or equation to document them.

Initialization

As we have seen already, Modelica allows us to describe model behavior in terms of differential equations. But the initial conditions we choose are just as important as the equations.

For this reason, Modelica also provides constructs for describing the initialization of our system of equations. For example, if we wanted the initial value ofx in our model to be *2*, we could add an initial equation section to our model as follows:

**model** **FirstOrderInitial** "First order equation with initial value"

Real x "State variable";

**initial equation**

x = 2 "Used before simulation to compute initial values";

**equation**

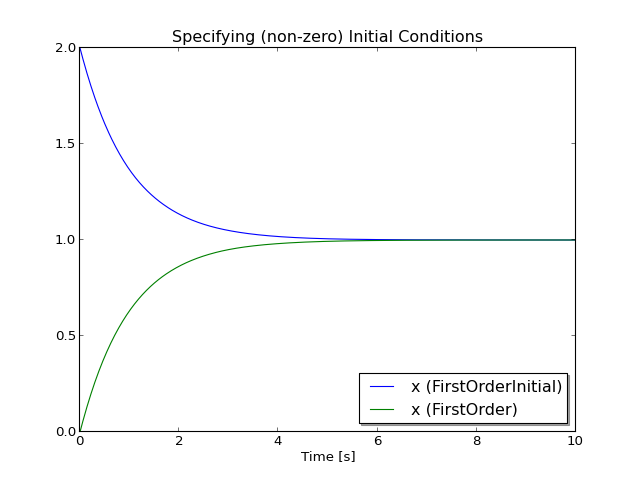
der(x) = 1-x "Drives value of x toward 1.0";

**end** **FirstOrderInitial**;

Note that the only difference between this model and the previous one, presented in the section on [*Adding Some Documentation*](http://book.xogeny.com/behavior/equations/first_order/#first-order-doc), is the addition of theinitial equation section which contains the equation x = 2. In the previous example, the initial value of x at the start of the simulation was unspecified. Generally speaking, this means that the initial value for x will be the value of its start attribute (which is zero by default). However, because each tool uses their own specific algorithms to formulate the final system of equations, it is always best to state initial conditions explicitly, as we have done here. By adding this equation to the initial equation section, we are explicitly specifying the initial condition for x.

As a result, the solution trajectory is quite different as we can see in the following figure:

([Source code](http://book.xogeny.com/plots/FOI.py))



The model FirstOrderInitial shows a typical way of initializing a system by providing explicit initial values for the states of the system. In fact, a system of differential equations is incomplete without a specification for how the initial conditions are determined. Our FirstOrderInitial model would be represented mathematically as:

x˙=(1−x);x(0)=2

However, there are many cases where you want a more sophisticated type of initialization. An initial equation section can contain more than just explicit equations for the initial values of the state variables.

For example, we might want our initial conditions to be such that the derivative of x was zero at the start of the simulation. In this case, only a bit of trivial algebra is required to realize that this could be accomplished by specifying an initial condition of x(0)=1. However, for more complex systems, it is far from trivial to determine the initial state values that would satisfy such a requirement. In those cases, it is possible to express the constraint that x˙(0)=0directly in Modelica as follows:

**model** **FirstOrderSteady**

"First order equation with steady state initial condition"

Real x "State variable";

**initial equation**

der(x) = 0 "Initialize the system in steady state";

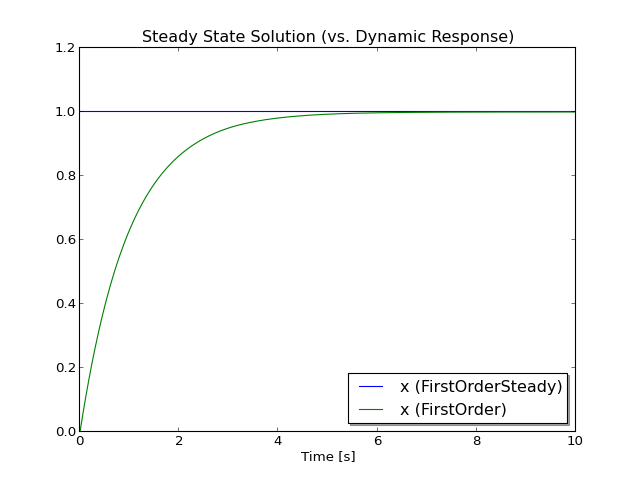
**equation**

der(x) = 1-x "Drives value of x toward 1.0";

**end** **FirstOrderSteady**;

Simulating this system gives the following solution:

([Source code](http://book.xogeny.com/plots/FOS.py))



As we see from these results, the initial derivative of x is zero at the start of the simulation and remains zero because there are no external influences acting on this system that would disrupt this equilibrium.

This provides a glimpse of the initialization capablities in Modelica. More complete coverage of the initialization topic can be found in the *[Initialization](http://book.xogeny.com/behavior/equations/initialization/" \l "initialization)*section later in this chapter.

Experimental Conditions

When building a model, the model developer might wish to associate specific experimental conditions with the model. This can be done using something called an annotation. An annotation includes information that is not directly related to the behavior of the model.

For example, experimental conditions describe information like the start time of the simulation, the stop time, solution tolerance and so on. This is not information about the behavior of the model itself, but rather information about how to approach simulating that behavior. Experimental conditions are stored in a model using a specific annotation called the experiment annotation.

There are four pieces of information that can be specified in an experiment annotation. All of them are optional. The following is a model of our first order system that includes an experiment annotation:

**model** **FirstOrderExperiment** "Defining experimental conditions"

Real x "State variable";

**initial equation**

x = 2 "Used before simulation to compute initial values";

**equation**

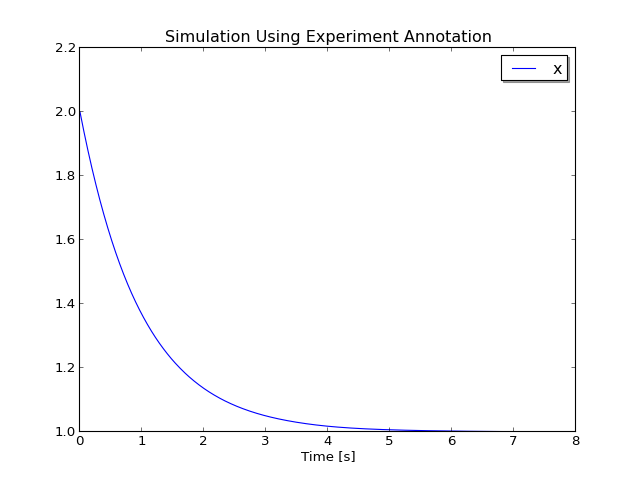
der(x) = 1-x "Drives value of x toward 1.0";

**annotation**(experiment(StartTime=0,StopTime=8));

**end** **FirstOrderExperiment**;

The following trajectory was simulated using these experimental conditions:

([Source code](http://book.xogeny.com/plots/FOE.py))



The trajectory terminates at 8 seconds because the simulator used the experiment annotation to determine how long to run the simulation.

**Annotation Support**

The experiment annotation is widely supported. But it is important to keep in mind that, in general, a tool is free to ignore any or all annotations.