

WORKSHOP 3.1

Writing Modelica code – “Hello World”

Contents

Introduction.....	1
A cake model	1
What is the mathematics involved?	1
Modeling.....	2
Aiming at a better style	4
Adding cake color	5
Making the cake look like its result.....	5
Finalizing the cake model.....	6
A bonus?	6

Introduction

In this workshop, you will write your first contained model from scratch in an incremental approach. “Contained” refers to the fact that the model does not have any interface and thus can be ran alone. This enables focusing on the basics – such as time variability, types, equations, etc. – and the organization of the parameters in the Modelon Impact GUI.

Note: At any time, if you need the perfect Modelica syntax, you can open the help center of Modelon Impact, search for “Modelica Specification” and click on the corresponding link.

A cake model

Yes, you read it well... In this workshop, you will develop a cake model! To be honest, this model will **not** be culinary correct, but it should be a fun example to get your hands in Modelica language and hopefully learn many nice customization capabilities.

What is the mathematics involved?

As said above, there is no culinary correctness in this model. However, we want to observe a given behavior for which we will make equations for. Our criteria are the height and color of the cake.

Let’s assume the following list of ingredient and associated effects:

Recipe parameters	Effect on cake height	Effect on cake color
Number of eggs	Linearly increasing	None
Mass of flour	Small influence (10%) Parabola (increase/decrease)	None
Mass of sugar	None	None
Baking time	High influence Parabola (increase/decrease)	Incrementally dependent on both variables

Oven temperature	Linearly decreasing	
------------------	---------------------	--

Each of these parameters are bounded between minimum and maximum values. This way, we avoid diverging behavior such as adding more and more effects to increase the cake height.

Here is a suggestion of such a model:

- Create variables which are saturation and normalization of each of these parameters
- Add a formula for the cake height h that adds to the minimum height of the cake h_{min} , a fraction f of the delta to the maximum height h_{max} :

$$h = h_{min} + (h_{max} - h_{min}) * f$$

This fraction is defined as the product of the effect of the parameters as height:

$$f = (1 - T_{hoven_sat}) * nEggs_sat * (4 * t_{baking_sat} * (1 - t_{baking_sat})) * (0.9 + (4 * mFlour_sat * (1 - mFlour_sat)) / 10)$$

where the subscript “sat” refers to the saturated and normalized variables of each ingredient.

The color of the cake is defined as follow:

Color	RGB	Qualifier	Condition
	{240,195,0}	Raw	$r < 0.75$
	{167,103,38}	Fondant	$0.75 \leq r < 0.90$
	{136,66,29}	Perfect	$0.90 \leq r < 1.05$
	{91,60,17}	Dry	$1.05 \leq r < 1.20$
	{0,0,0}	Burned	$1.20 \leq r$

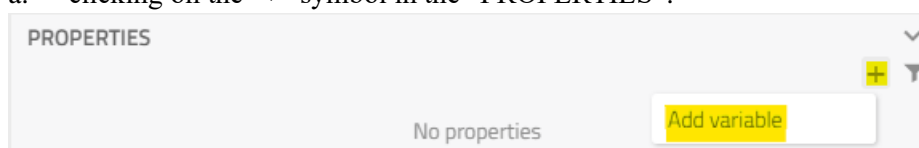
Where r is the ratio:

$$r = \frac{t_{baking} * T_{oven}}{t_{ideal} * T_{ideal}}$$

The “ideal” subscript corresponds to the secret recipe.

Modeling

1. Create a new package **W1** and create a new model class in that package, called **Cake**
2. Open this new **Cake** model and start adding all ingredients as variables. This can be done in two ways:
 1. Graphically by:
 - a. clicking on the “+” symbol in the “PROPERTIES”.



- b. A dialog opens to assist the creation of the parameter.

- c. For each variable, select the time variability (“parameter” is what we need), the type (select “Real” or Integer for now), add a name, description and you can experiment with Expression (default variable) and “tab” and “group” annotations.

2. Coding in Modelica language in the source code editor:

- a. Change view by switching from “Diagram” to “Code” in the toolbar.

```

1 model Cake
2   annotation(...);
3 end Cake;
4

```

- b. The code editor opens, and you are ready to type the Modelica code yourself.

Note: You can combine both solutions so if you do not remember the syntax, add a variable graphically and look at what has been added in the code editor.

3. At this step, you should have a parameter defined for each ingredient. Except from the number of eggs – that is a Integer –, all other parameters are Real.
4. Add a variable for the cake height. We will let aside the color for now.
5. Add the normalized and saturated variable variant for each of the ingredients using the `min()` and `max()` operators. It should look something like: $varSatNorm = \frac{\min(\max(var, minValue), maxValue) - minValue}{maxValue - minValue}$; Note that `minValue` and `maxValue` can be substituted by their value for now.

6. Add the equation for the cake height. One option is:

$$height = hMin + (hMax - hMin) * (1 - T_hoven_sat) * nEggs_sat * (4 * t_baking_sat * (1 - t_baking_sat)) * (0.9 + (4 * mFlour_sat * (1 - mFlour_sat))/10);$$

You should be able to simulate this model. If it does simulate then congratulations, you have created your first model from scratch! If not, look at your code or reach for help, you should not be too far from the goal. Search for variables that are not defined, or parameter values not assigned.

Aiming at a better style

Using Real type is convenient but is not really specific. SIunits define the quantities and units involved in the type to properly define the variable. Also, as for Real, these units have min and max attributes. It is a good time to add dedicated parameters to these attributes so we can easily modify them to change the boundary conditions of our recipe.

1. Substitute each Real keyword in the Modelica code by the corresponding type. It should be enough to deal with Modelica.SIunits.Mass, Modelica.SIunits.Temperature, Modelica.SIunits.Time and Modelica.SIunits.Height.
2. For each (at least for some) ingredients, add min and max attributes to the associated parameters as new parameters. This can look like:

```
5 parameter Integer nEggs(
6   min=nEggs_min,
7   max=nEggs_max) = 1 "Number of eggs" annotation (...);
8 parameter Integer nEggs_min(
9   min=0,
10  max=nEggs_max) = 0 "Number of eggs" annotation (...);
11 parameter Integer nEggs_max(
12  min=nEggs_min,
13  max=6) = 6 "Number of eggs" annotation (...);
```

3. Optionally, you can set the ingredient in a group called “Ingredients” and the min and max parameters in a tab called “Problem statement”.
4. For some units, it is convenient to add the attribute “displayUnit” to a given unit – e.g. “degC” for degree Celsius or “min” for minutes or “cm” for centimeter. Here is an example:

```
parameter Modelica.SIunits.Time t_baking(
  displayUnit="min",
  min=t_baking_min,
  max=t_baking_max) = 1800 "Baking time" annotation (Dialog(group="Baking process"));
```

When writing code, it is inconvenient to duplicate code. Each expression that can be reused ends up being a function or a model. A good candidate for that is the normalization and saturation of the variables.

5. Let’s define a function inside the model that does exactly what we expect:

```
92 function SatNorm "Saturation and normalization of variables"
93   input Real var "variable to saturate and normalize";
94   input Real minValue "minimum bound";
95   input Real maxValue "maximum bound";
96   output Real varSatNorm "variable saturated and normalized";
97   algorithm
98     varSatNorm := (min(max(var,minValue),maxValue)-minValue)/(maxValue-minValue);
99   end SatNorm;
```

6. Now, for each saturated and normalized variables, you can define it by using the new function.

```
101 // Saturated and normalized parameters or variables for convenience
102 parameter Modelica.SIunits.DimensionlessRatio T_hoven_sat=SatNorm(T_hoven, T_hoven_min, T_hoven_max);
```

This should look like a decent code already. Congratulations!

Adding cake color

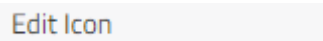

Modelica language enables you to create your own types. While many are available in the Modelica Standard Libraries, it would be impossible to gather them all – for all application. Let’s see how we can create one easily.

A color is often defined as a combination of Red Green Blue (RGB). The value of each primitive is given a value between 0 and 255 (which makes 2^8 options). 0 corresponds to an absence of the color while 255 corresponds to a full presence of it. Therefore, $RGB=\{255,0,0\}$ corresponds to red, $\{0,255,0\}$ to green, $\{0,0,255\}$ to blue, $\{255,255,255\}$ to white (all primitives superimposed) and $\{0,0,0\}$ to black (absence of all primitives). Let’s create a new type for the RGB coloring.

1. Create a new type with the “type” keyword. For RGB, this new type will correspond to an array of integers of size 3.
`type RGB = Integer[3];`
2. We know the bounds of the integer values so let’s add them. As the bounds apply to each of the scalar of the array, the keyword “each” should be used.
`type RGB = Integer[3](each final min=0, each final max=255);`
3. Now we can conveniently use the new RGB type as any other (like Real). Create colors following the table describing the colors using the qualifiers as name. A first is shown as hint:
`RGB raw={240,195,0} "Color for rawDough";`
4. You can now add a variable and associated equation to define the cake color. Of which type should the cakeColor variable be? Use if-conditions to match the cake color based on the conditions expressed in the table describing the colors above.

Making the cake look like its result

Why did we spend so much time adding colors if we cannot see the result of the cake? Of course we can... it just requires some coding.

1. Right click on the cake model in the Workspace and edit the icon.

2. Use the shapes to draw something that looks like a cake. You can spend much time on it so that it would look very nice. A less artistic icon could look like that:

3. Save and close the icon editor.
4. Open the source code editor.
5. Expand the last annotation (at the very bottom of the model) and observe the corresponding code of the icon. You should be able to identify the shapes. The icon above is composed of a Polygon (rectangle with pseudo-rounded bottom – indeed, these as small consecutive lines) and an Ellipse. You should also be able to identify “fillColor” and “lineColor” attributes

Note: Modelica has a specific operator called DynamicSelect that enables changing the values of these attributes with the simulation time. It is constructed as follow: DynamicSelect(default, variable) or simply DynamicSelect(variable) if the variable has a default. The icon will display the default until the simulation is ran, then it will display the variable.

6. Change the fillColor of your cake to be “raw” by default” and “cakeColor” during simulation.

Bonus: Change the height of your cake to be a function of the height variable. This one is a bit more cumbersome as you must scale the height to match the coordinates of the canvas. It is perfectly feasible though. Taking the challenge?

Finalizing the cake model

A model is never complete without a proper documentation and testing.

1. Right click on the cake model in the Workspace and show documentation.

Show documentation

2. Open edit mode by clicking on the pencil on the top right of the documentation window:



3. The pencil turns orange. Write a small documentation. Try also writing some formatted text (bold, italics) in Word and copy paste it in the documentation, the formatting is preserved.
4. Click on the floppy disc icon to save your changes and on the pencil to quit the editing mode.



5. Create a new model and name it “BakingCakes”.
6. Instantiate several cake models in it and play around with the parameters, observe your different cakes.



A bonus?

If you were fast in developing the model, you can continue playing with the Modelica language. Wouldn't that be great to have a gourmet review of your cake? Something simple, printed on the simulation log? Interested?

Modelica.Utilities.Streams.print() function enables printing messages in the log.

The function **getInstanceName()** returns a String with the name of the instance of the class in which it belongs.

Try adding in the equation section, the call of the print function that prints a message including the name of the cake instance and a message that is function of the color of the cake.

Congratulations in making such a great cake!