Enterprise Architect

User Guide Series

Parametric Simulation

Sparx Systems Enterprise Architect provides integration with OpenModelica to support rapid and robust simulation of how a SysML Parametric model will behave under different circumstances; the simulation properties are defined in a Simulation Artifact.

Author: Sparx Systems
Date: 2020-09-07
Version: 15.2

Table of Contents

Parametric Simulation 4
Creating a Parametric Model 7
Model Analysis using Datasets 24

Parametric Simulation

Enterprise Architect provides integration with both OpenModelica and MATLAB Simulink to support rapid and robust evaluation of how a SysML model will behave in different circumstances.

The OpenModelica Libraries are comprehensive resources that provide many useful types, functions and models. When creating SysML models in Enterprise Architect, you can reference resources available in these Libraries.

Enterprise Architect's MATLAB integration connects via the MATLAB API, allowing your Enterprise Architect simulations and other scripts to act based on the value of any available MATLAB functions and expressions. You can call MATLAB through a Solver Class, or export your model to MATLAB Simulink, Simscape and/or Stateflow.

This section describes the process of defining a Parametric model, annotating the model with additional information to drive a simulation, and running a simulation to generate a graph of the results.

Introduction to SysML Parametric Models

SysML Parametric models support the engineering analysis of critical system parameters, including the evaluation of key metrics such as performance, reliability and other physical characteristics. These models combine

Requirements models with System Design models, by capturing executable constraints based on complex mathematical relationships. Parametric diagrams are specialized Internal Block diagrams that help you, the modeler, to combine behavior and structure models with engineering analysis models such as performance, reliability, and mass property models.

For further information on the concepts of SysML Parametric models, refer to the official OMG SysML website and its linked sources.

SysMLSimConfiguration Artifact

Enterprise Architect helps you to extend the usefulness of your SysML Parametric models by annotating them with extra information that allows the model to be simulated. The resulting model is then generated as a model that can be solved (simulated) using either MATLAB Simulink or OpenModelica.

The simulation properties for your model are stored against a Simulation Artifact. This preserves your original model and supports multiple simulations being configured against a single SysML model. The Simulation Artifact can be found on the 'Artifacts' Toolbox page.

SysPhS Standard Support

The *SysPhS Standard* is a *SysML Extension for Physical Interaction and Signal Flow Simulation*. It defines a standard way to translate between a SysML model and either a Modelica model or a Simulink/Simscape model, providing a simpler model-based method for sharing simulations. See the *SysPhS Standard Support* Help topic.

User Interface

The user interface for the SysML simulation is described in the *Configure SysML Simulation Window* topic.

Examples

To aid your understanding of how to create and simulate a SysML Parametric model, three examples have been provided to illustrate three different domains. All three examples happen to use the OpenModelica libraries. These examples and what you are able to learn from them are described in the *SysML Simulation Example*s topic.

Creating a Parametric Model

In this topic we discuss how you might develop SysML model elements for simulation (assuming existing knowledge of SysML modeling), configure these elements in the Configure SysML Simulation window, and observe the results of a simulation under some of the different definitions and modeling approaches. The points are illustrated by snapshots of diagrams and screens from the SysML Simulation examples provided in this chapter.

When creating a Parametric Model, you can apply one of three approaches to defining Constraint Equations:

- Defining inline Constraint Equations on a Block element
- Creating re-usable Constraint Blocks, and
- Using connected Constraint properties

You would also take into consideration:

- Flows in physical interactions
- Default Values and Initial Values
- Simulation Functions
- Value Allocation, and
- Packages and Imports

Access

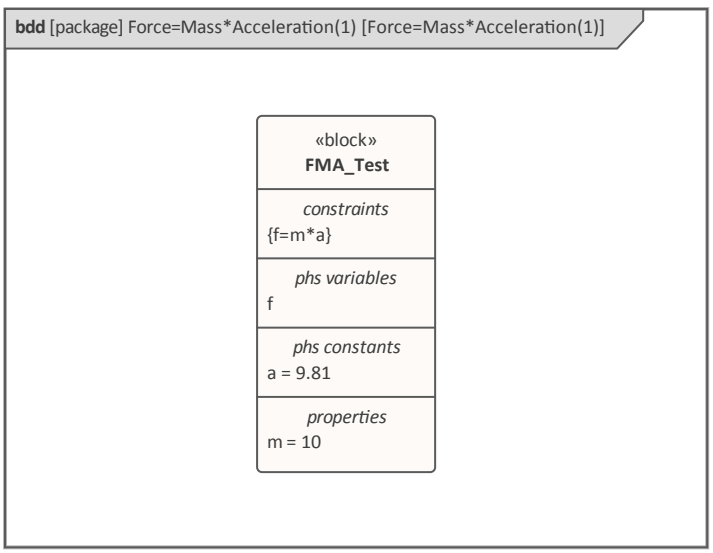
Ribbon	Simulate > System Behavior >

	Modelica/Simulink > SysMLSim Configuration Manager

Defining inline Constraint Equations on a Block

Defining constraints directly in a Block is straightforward and is the easiest way to define constraint equations.

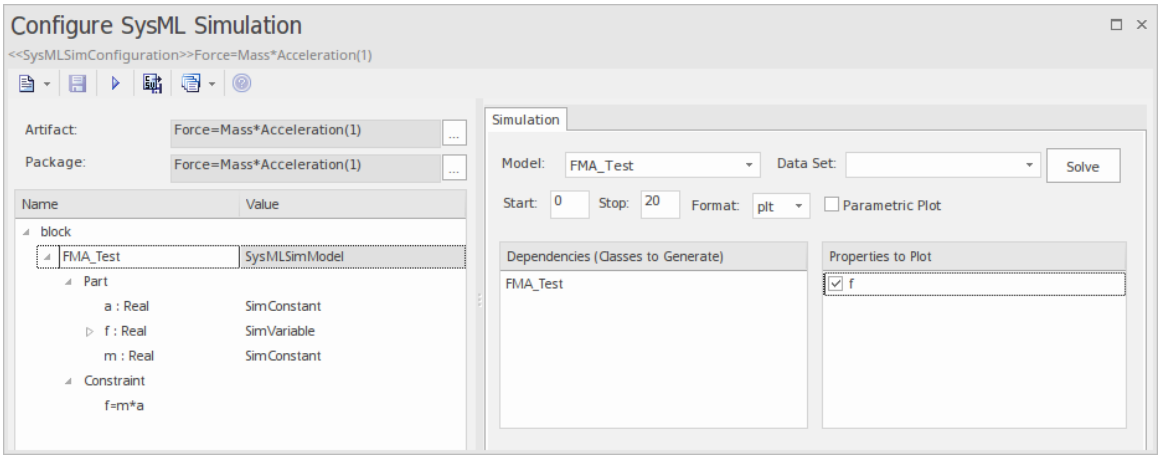
In this figure, constraint 'f = m * a' is defined in a Block element.



Tip: You can define multiple constraints in one Block.

1. Create a SysMLSim Configuration Artifact 'Force=Mass*Acceleration(1)' and point it to the Package 'FMA_Test'.
2. For 'FMA_Test', in the 'Value' column set 'SysMLSimModel'.

3. For Parts 'a', 'm' and 'f', in the 'Value' column: set 'a' and 'm' to 'PhSConstant' and (optionally) set 'f' to 'PhSVariable'.
4. On the 'Simulation' tab, in the 'Properties to Plot' panel, select the checkbox against 'f'.
5. Click on the Solve button to run the simulation.

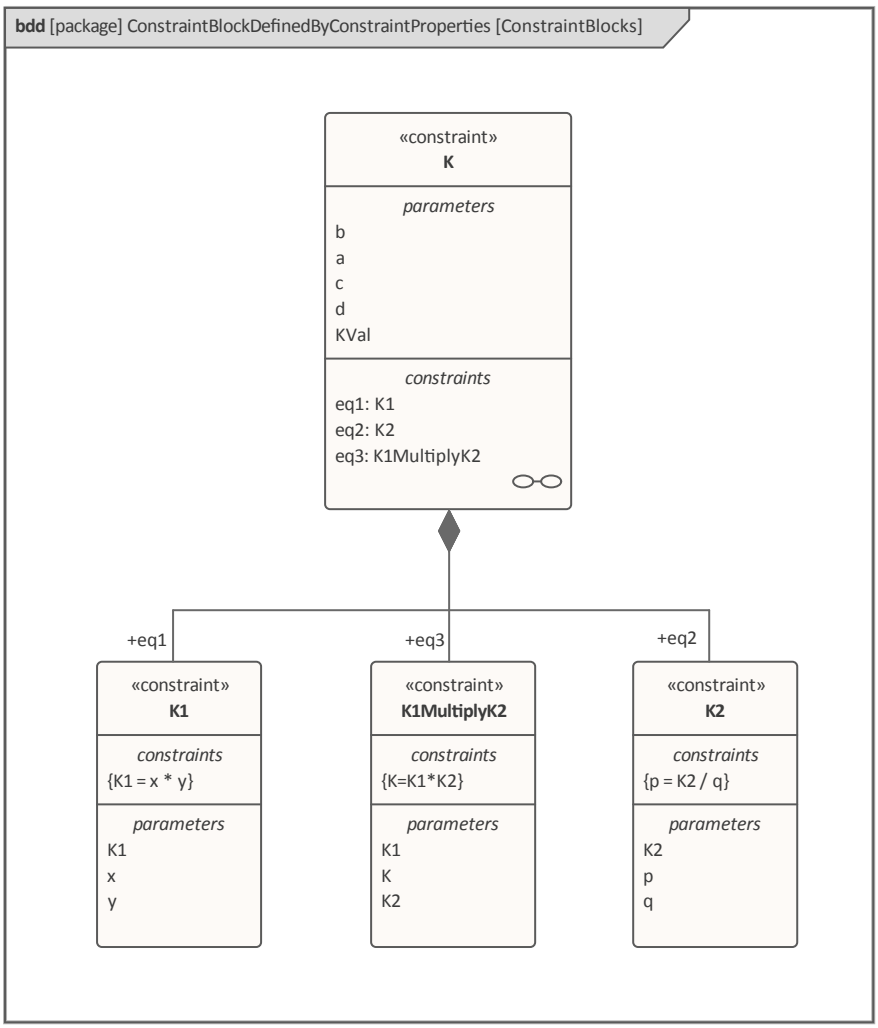


A chart should be plotted with f = 98.1 (which comes from 10 * 9.81).

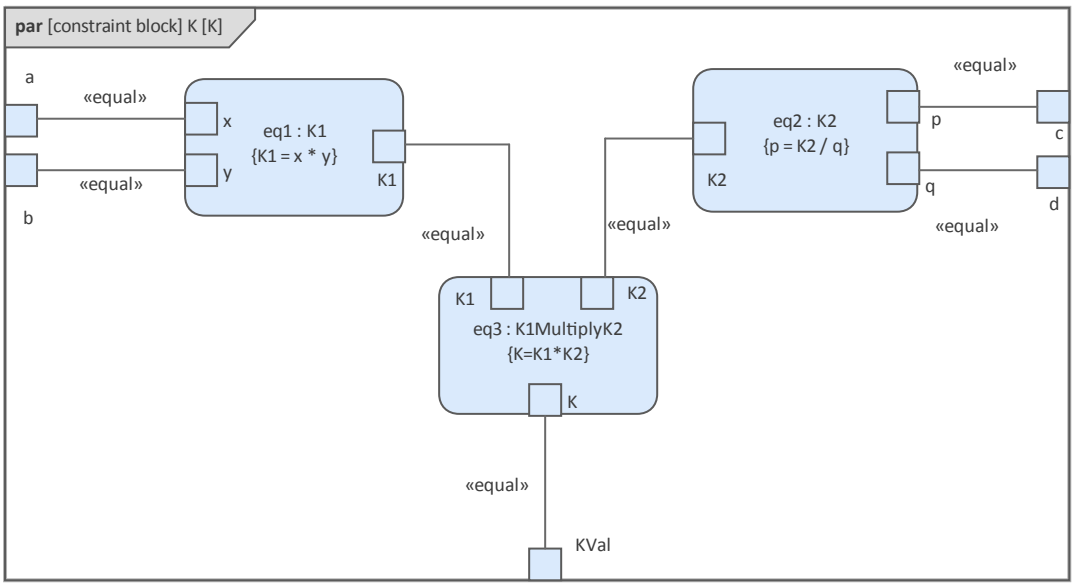
Connected Constraint Properties

In SysML, constraint properties existing in Constraint Blocks can be used to provide greater flexibility in defining constraints.

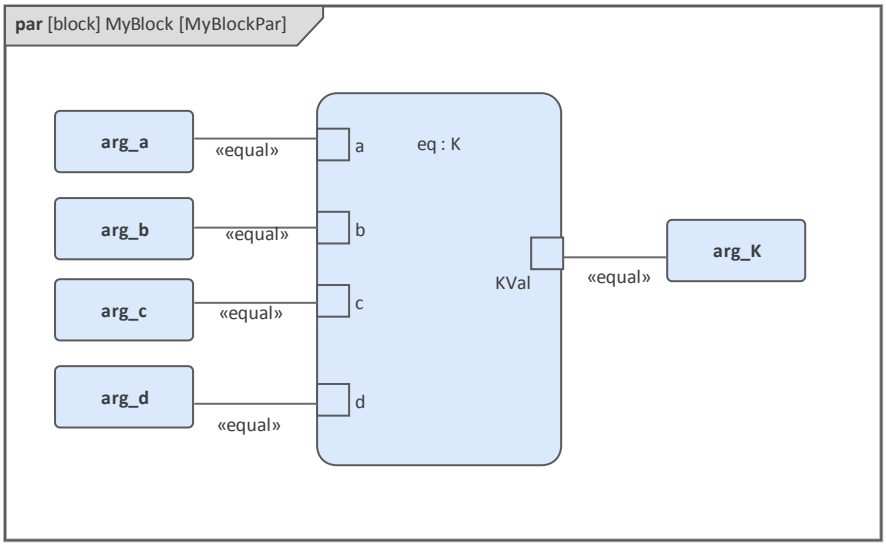
In this figure, Constraint Block 'K' defines parameters 'a', 'b', 'c', 'd' and 'KVal', and three constraint properties 'eq1', 'eq2' and 'eq3', typed to 'K1', 'K2' and 'K1MultiplyK2' respectively.



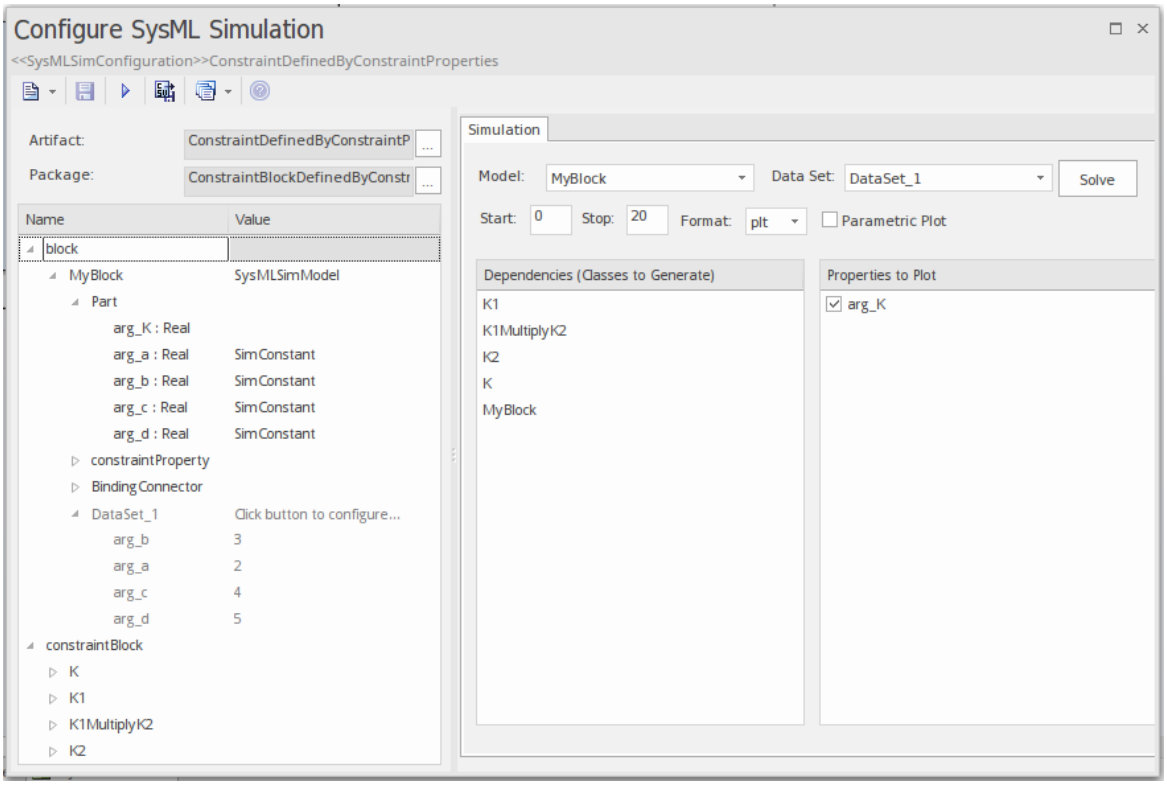
Create a Parametric diagram in Constraint Block 'K' and connect the parameters to the constraint properties with Binding connectors, as shown:



- Create a model MyBlock with five Properties (Parts)
- Create a constraint property 'eq' for MyBlock and show the parameters
- Bind the properties to the parameters



- Provide values (arg_a = 2, arg_b = 3, arg_c = 4, arg_d = 5) in a data set
- In the 'Configure SysML Simulation' dialog, set 'Model' to 'MyBlock' and 'Data Set' to 'DataSet_1'
- In the 'Properties to Plot' panel, select the checkbox against 'arg_K'
- Click on the Solve button to run the simulation



The result 120 (calculated as 2 * 3 * 4 * 5) will be computed and plotted. This is the same as when we do an expansion with pen and paper: K = K1 * K2 = (x*y) * (p*q), then bind with the values (2 * 3) * (4 * 5); we get 120.

What is interesting here is that we intentionally define K2's equation to be 'p = K2 / q' and this example still works.

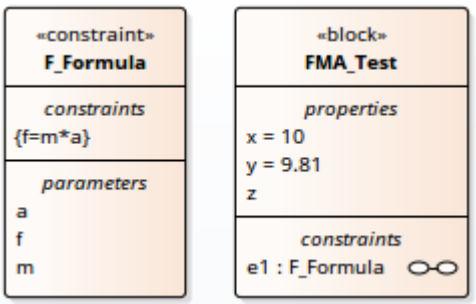
We can easily solve K2 to be p * q in this example, but in some complex examples it is extremely hard to solve a variable from an equation; however, the Enterprise Architect SysMLSim can still get it right.

In summary, the example shows you how to define a Constraint Block with greater flexibility by constructing the constraint properties. Although we demonstrated only one layer down into the Constraint Block, this mechanism could work on complex models for an arbitrary level of use.

Creating Reuseable Constraint Blocks

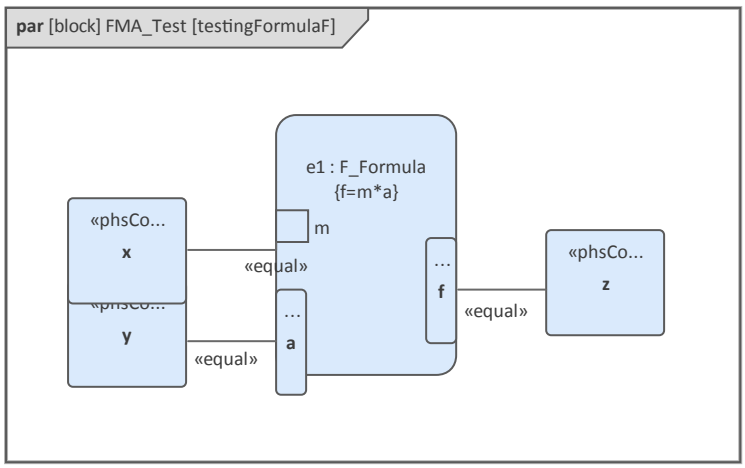
If one equation is commonly used in many Blocks, a Constraint Block can be created for use as a constraint property in each Block. These are the changes we make, based on the previous example:

- Create a Constraint Block element 'F_Formula' with three parameters 'a', 'm' and 'f', and a constraint 'f = m * a'

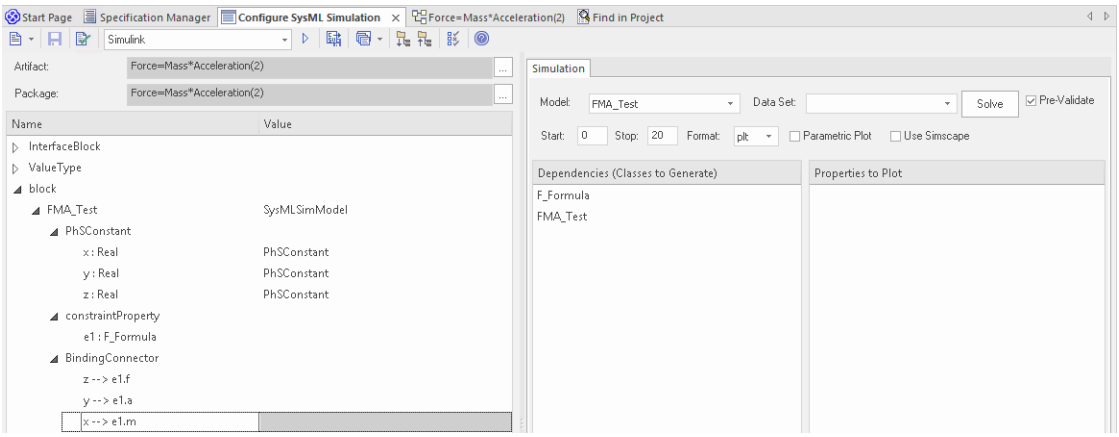


Tip: Primitive type 'Real' will be applied if property types are empty

- Create a Block 'FMA_Test' with three properties 'x', 'y' and 'z', and give 'x' and 'y' the default values '10' and '9.81' respectively
- Create a Parametric diagram in 'FMA_Test', showing the properties 'x', 'y' and 'z'
- Create a ConstraintProperty 'e1' typed to 'F_Formula' and show the parameters
- Draw Binding connectors between 'x—m', 'y—a', and 'f—z' as shown:



- Create a SysMLSimConfiguration Artifact element and configure it as shown in the dialog illustration:
  - In the 'Value' column, set 'FMA_Test' to 'SysMLSimModel'
  - In the 'Value' column, set 'x' and 'y' to 'PhSConstant'
  - In the 'Properties to Plot' panel select the checkbox against 'Z'
  - Click on the Solve button to run the simulation



A chart should be plotted with f = 98.1 (which comes from 10 * 9.81).

Flows in Physical Interactions

When modeling for physical interaction, exchanges of conserved physical substances such as electrical current, force, torque and flow rate should be modeled as flows, and the flow variables should be set to the attribute 'isConserved'.

Two different types of coupling are established by connections, depending on whether the flow properties are potential (default) or flow (conserved):

- Equality coupling, for potential (also called effort) properties
- Sum-to-zero coupling, for flow (conserved) properties; for example, according to Kirchoff's Current Law in the electrical domain, conservation of charge makes all charge flows into a point sum to zero

In the generated OpenModelica code of the 'ElectricalCircuit' example:

```
connector ChargePort
    flow Current i;          //flow keyword will be generated if 'isConserved' = true
    Voltage v;
end ChargePort;

model Circuit
    Source source;
```

```
        Resistor resistor;
        Ground ground;
equation
        connect(source.p, resistor.n);
        connect(ground.p, source.n);
        connect(resistor.p, source.n);
end Circuit;
```

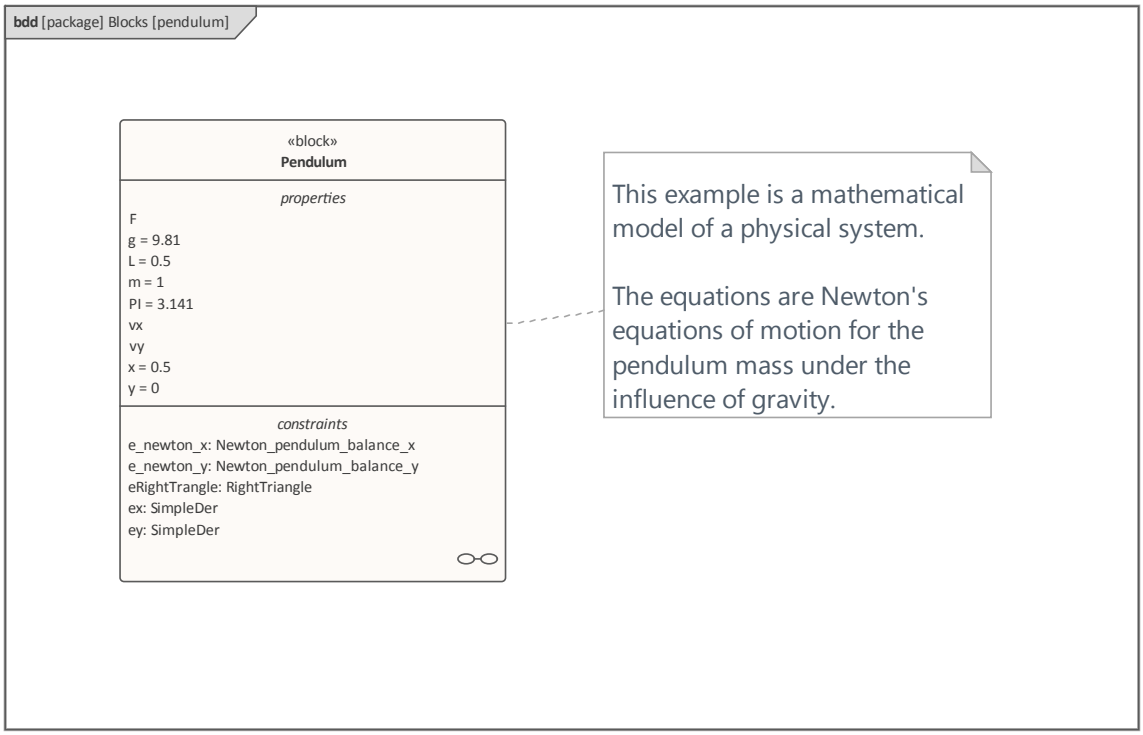
Each connect equation is actually expanded to two equations (there are two properties defined in ChargePort), one for equality coupling, the other for sum-to-zero coupling:

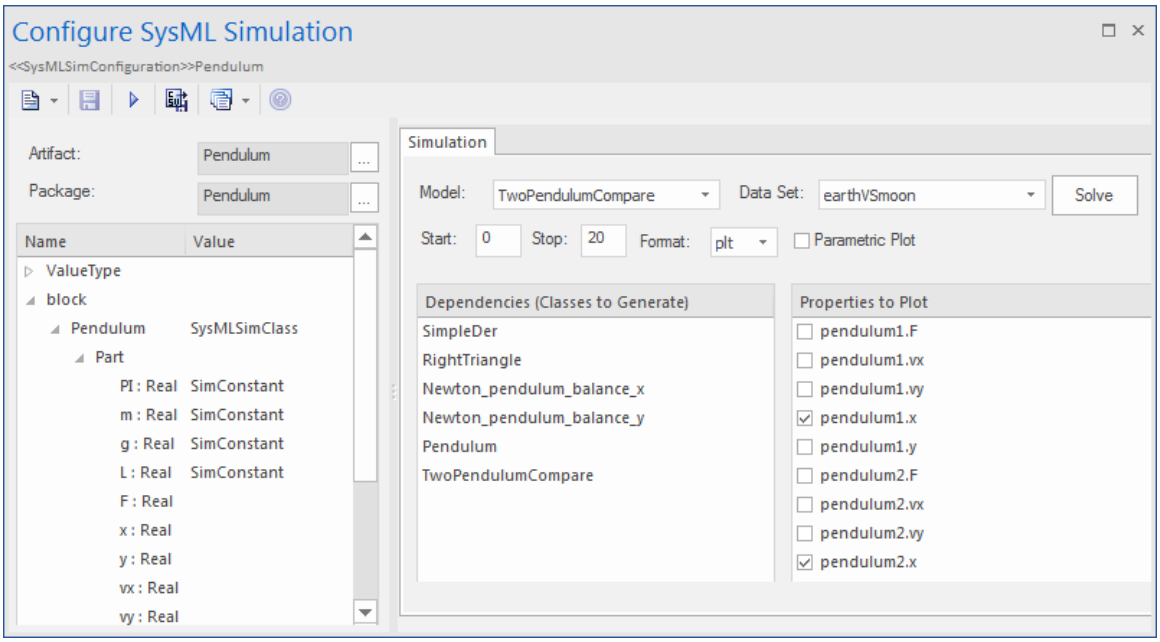
```
source.p.v = resistor.n.v;
source.p.i + resistor.n.i = 0;
```

Default Value and Initial Values

If initial values are defined in SysML property elements ('Properties' dialog > 'Property' page > 'Initial' field), they can be loaded as the default value for a PhSConstant or the initial value for a PhSVariable.

In this Pendulum example, we have provided initial values for properties 'g', 'L', 'm', 'PI', 'x' and 'y', as seen on the left hand side of the figure. Since 'PI' (the mathematical constant), 'm' (mass of the Pendulum), 'g' (Gravity factor) and 'L' (Length of Pendulum) do not change during simulation, set them as 'PhSConstant'.





The generated Modelica code resembles this:

```
class Pendulum
    parameter Real PI = 3.141;
```

```
        parameter Real m = 1;
        parameter Real g = 9.81;
        parameter Real L = 0.5;
        Real F;
        Real x (start=0.5);
        Real y (start=0);
        Real vx;
        Real vy;
        ......
    equation
    ......
    end Pendulum;
```

- Properties 'PI', 'm', 'g' and 'L' are constant, and are generated as a declaration equation
- Properties 'x' and 'y' are variable; their starting values are 0.5 and 0 respectively, and the initial values are generated as modifications

Simulation Functions

A Simulation function is a powerful tool for writing complex logic, and is easy to use for constraints. This section describes a function from the TankPI example.

In the Constraint Block 'Q_OutFlow', a function

'LimitValue' is defined and used in the constraint.

- On a Block or Constraint Block, create an operation ('LimitValue' in this example) and open the 'Operations' tab of the Features window
- Give the operation the stereotype 'SimFunction'
- Define the parameters and set the direction to 'in/out'

Tips: Multiple parameters could be defined as 'out', and the caller retrieves the value in format of:

(out1, out2, out3) = function_name(in1, in2, in3, in4, ...); //Equation form

(out1, out2, out3) := function_name(in1, in2, in3, in4, ...); //Statement form

- Define the function body in the text field of the 'Code' tab of the Properties window, as shown:

```
pLim :=
        if p > pMax then
            pMax
        else if p < pMin then
            pMin
        else
            p;
```

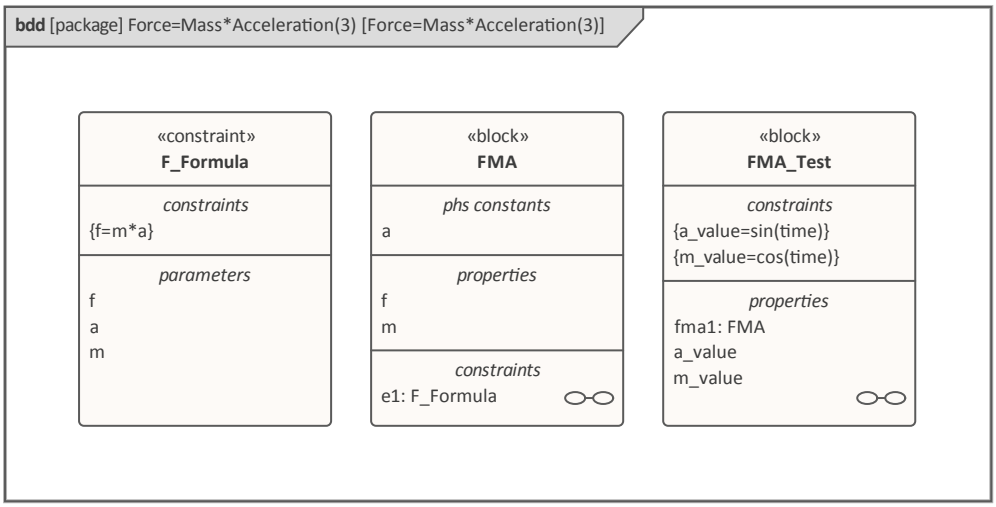
When generating code, Enterprise Architect will collect all the operations stereotyped as 'SimFunction' defined in Constraint Blocks and Blocks, then generate code resembling this:

```
function LimitValue
    input Real pMin;
    input Real pMax;
    input Real p;
    output Real pLim;
        algorithm
            pLim :=
                if p > pMax then
                    pMax
                else if p < pMin then
                    pMin
                else
```

```
                    p;
end LimitValue;
```

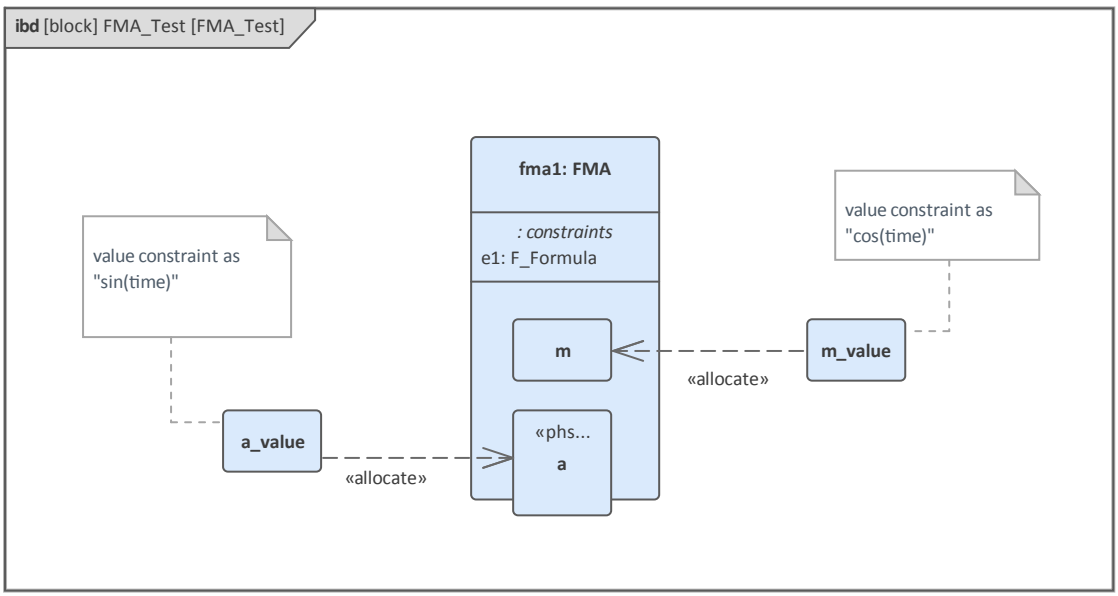
Value Allocation

This figure shows a simple model called 'Force=Mass*Acceleration'.

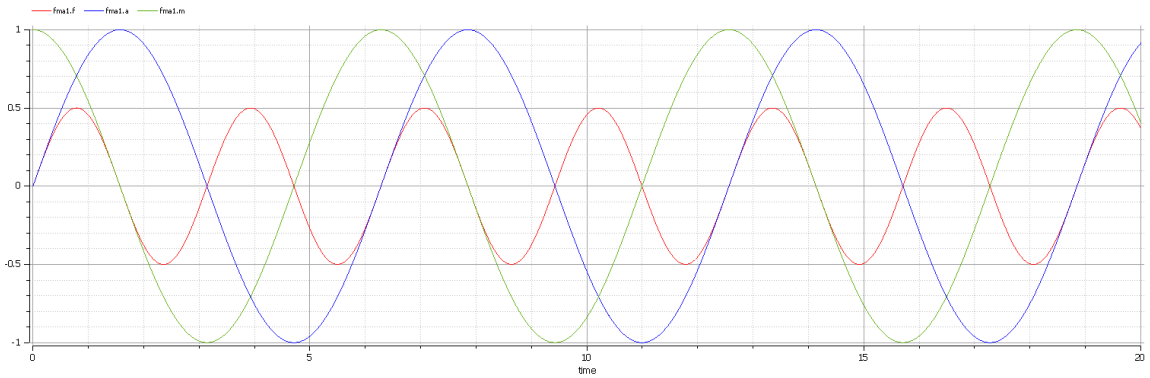


- A Block 'FMA' is modeled with properties 'a', 'f', and 'm' and a constraintProperty 'e1', typed to Constraint Block 'F_Formula'
- The Block 'FMA' does not have any initial value set on its properties, and the properties 'a', 'f' and 'm' are all variable, so their value change depends on the environment in which they are simulated
- Create a Block 'FMA_Test' as a SysMLSimModel and add the property 'fma1' to test the behavior of Block 'FMA'

- Constraint 'a_value' to be 'sin(time)'
- Constraint 'm_value' to be 'cos(time)'
- Draw Allocation connectors to allocate values from environment to the model 'FMA'



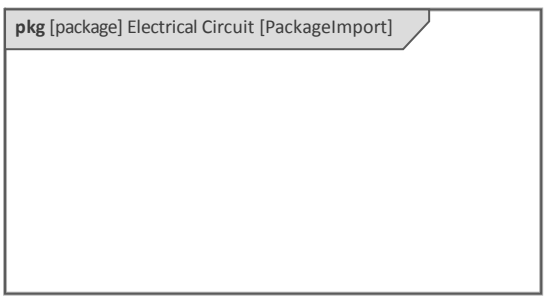
- Select the 'Properties to Plot' checkboxes against 'fma1.a', 'fma1.m' and 'fma1.f'
- Click on the Solve button to simulate the model



Packages and Imports

The SysMLSimConfiguration Artifact collects the elements (such as Blocks, Constraint Blocks and Value Types) of a Package. If the simulation depends on elements not owned by this Package, such as Reusable libraries, Enterprise Architect provides an Import connector between Package elements to meet this requirement.

In the Electrical Circuit example, the Artifact is configured to the Package 'ElectricalCircuit', which contains almost all of the elements needed for simulation. However, some properties are typed to value types such as 'Voltage', 'Current' and 'Resistance', which are commonly used in multiple SysML models and are therefore placed in a Package called 'CommonlyUsedTypes' outside the individual SysML models. If you import this Package using an Import connector, all the elements in the imported Package will appear in the SysMLSim Configuration Manager.



Model Analysis using Datasets

Every SysML Block used in a Parametric model can, within the Simulation configuration, have multiple datasets defined against it. This allows for repeatable simulation variations using the same SysML model.

A Block can be typed as a SysMLSimModel (a top-level node that cannot be generalized or form part of a composition) or as a SysMLSimClass (a lower-level element that can be generalized or form part of a composition). When running a simulation on a SysMLSimModel element, if you have defined multiple datasets, you can specify which dataset to use. However, if a SysMLSimClass within the simulation has multiple datasets, you cannot select which one to use during the simulation and must therefore identify one dataset as the default for that Class.

Access

Ribbon	Simulate > System Behavior > Modelica/Simulink > SysMLSim Configuration Manager > in "block" group > Name column > Context menu on block element > Create Simulation DataSet

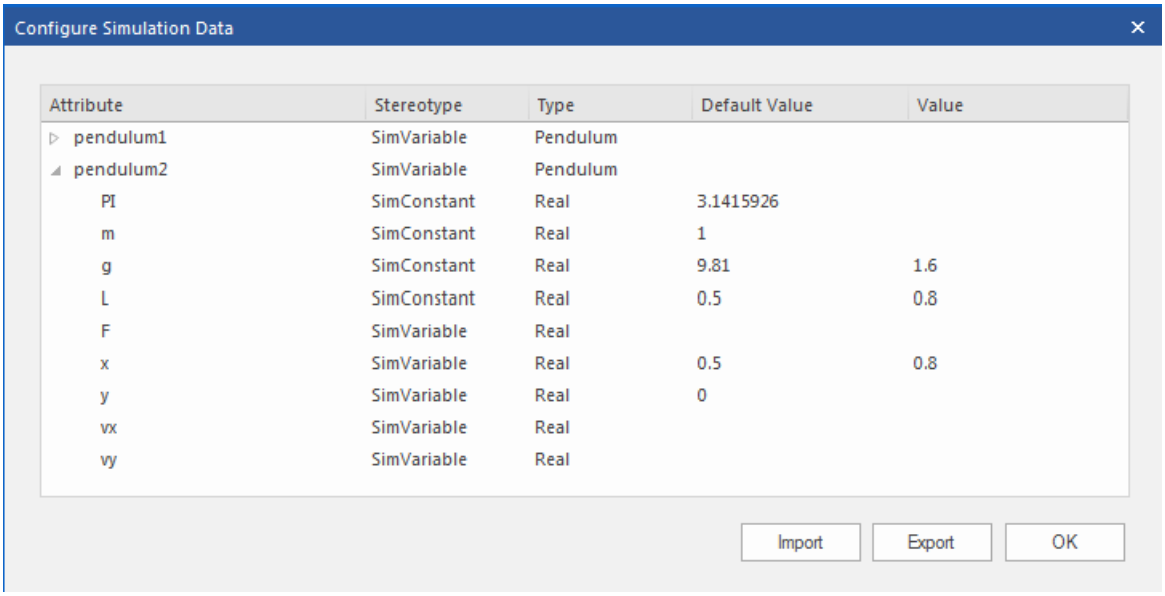
Dataset Management

Task	Action
Create	To create a new dataset, right-click on a Block name and select the 'Create Simulation Dataset' option. The dataset is added to the end of the list of components underneath the Block name. Click on the  button to set up the dataset on the 'Configure Simulation Data' dialog (see the *Configure Simulation Data* table).
Duplicate	To duplicate an existing dataset as a base for creating a new dataset, right-click on the dataset name and select the 'Duplicate' option. The duplicate dataset is added to the end of the list of components underneath the Block name. Click on the  button to edit the dataset on the 'Configure Simulation Data' dialog (see the *Configure Simulation Data* table).
Delete	To remove a dataset that is no longer

	required, right-click on the dataset and select the 'Delete Dataset' option.
Set Default	To set the default dataset used by a SysMLSimClass when used as a property type or inherited (and when there is more than one dataset), right-click on the dataset and select the 'Set as Default' option. The name of the default dataset is highlighted in bold. The properties used by a model will use this default configuration unless the model overrides them explicitly.

Configure Simulation Data

This dialog is principally for information. The only column in which you can directly add or change data is the 'Value' column.



Column	Description
Attribute	The 'Attribute' column provides a tree view of all the properties in the Block being edited.
Stereotype	The 'Stereotype' column identifies, for each property, if it has been configured to be a constant for the duration of the simulation or variable, so that the value is expected to change over time.
Type	The 'Type' column describes the type used for simulation of this property. It can be either a primitive type (such as 'Real') or a reference to a Block contained in the model. Properties referencing Blocks will show the child properties specified by the referenced Block below

	them.
Default Value	The 'Default Value' column shows the value that will be used in the simulation if no override is provided. This can come from the 'Initial Value' field in the SysML model or from the default dataset of the parent type.
Value	The 'Value' column allows you to override the default value for each primitive value.
Export / Import	Click on these buttons to modify the values in the current dataset using an external application such as a spreadsheet, and then re-import them to the list.