Uncertainties in OpenModelica

Peter Aronsson ([paronsson@wolfram.com](mailto:paronsson@wolfram.com))  
Leonardo Laguna Ruiz (leonardo@wolfram.com)  
Adrian Pop (Adrian.Pop@liu.se)

This document describes the language extensions for uncertainties in OpenModelica and the interface for data reconciliation and connection to OpenTURNS ([www.openturns.org](http://www.openturns.org)). The document is divided into three sections. The first section describes the Modelica language extensions required for representing uncertainties in Modelica and the annotations required for the two interfaces. The second section describes the interface provided to be able to perform data reconcilitation by extracting equations into a Mathematica syntax. The third section describes the connectivity to OpenTURNS.

# Modelica Language extensions

This section describes the language extensions for uncertainties.

## Uncertain variables

Uncertainties of variables are described by a new attribute ‘uncertain’ for the built in classes Real and Integer. This attribute is defined as an enumeration:

**type** Uncertainty = **enumeration**(given,sought,refine);

To declare a variable in Modelica as uncertain you modify the uncertain attribute, like for instance:

Real u(uncertain = Uncertainty.given);

## Distributions

To association a stochastic distribution for a variable, the attribute ‘distribution’ is used. It is also available for the built in classes Real and Integer. The distribution attribute is of type Distrbution, a record defined as:

**record** Distribution   
 String name;  
 Real params[:];  
 String paramNames[:];  
**end** Distribution;

To assign a distribution to a Modelica variable you modify the distribution attribute, like for instance:

Real u(uncertain = Uncertainty.given,  
 distribution = Distribution(“Normal”,{0,0.1},{“my”,”sigma”});

# Data reconciliation

TBD

# OpenTURNS

The openTURNS connection in OpenModelica makes it possible to run e.g. uncertainty propagation algorithms in OpenTURNS based on a Modelica model. OpenModelica will automatically build a simulator for the model that can be plugged into OpenTURNS for further analysis. It will also build a pythonscript, based on a template pyton script file provided by the user (as described below), and an XML file to indicate the inputs and outputs of the simulator wr.t. the OpenTURNS interface. The inputs are the uncertain variables declared as Uncertain.given, and the outputs are the variables declared as Uncertain.sought. Uncertain.refine has no meaning for OpenTURNS connection, and should be considered as an error (it is only used for data reconciliation where a variable is both considered as an input and an output).

## Correlation matrix

The correlation matrix of correlations between uncertain variables should be declared at the system level (top model scope) as a vector of Correlation records. Currently, due to limitations in OpenModelica, the vector MUST be declared as this:

Correlation correlation[1];  
algorithm  
correlation:={Correlation(L,I,-0.2)};

Note that the size of the vector must be given a constant value in the declaration, and that the correlation must be given its value in a single algorithm section.

## Template script file

The template script file provided by the user must have the following strings that will be replaced by OpenModelica with information from the Modelica model.

|  |  |  |
| --- | --- | --- |
| Template string | Description | Example (CantileverBeam) |
| <%distributions%> | Declares all distributions defined in the Modelica Model | distributionI=Beta(2.5,4.0,310.0,450.0) distributionL = Uniform(250.0,260.0) distributionE = Beta(0.93,3.2,28000000.0,48000000.0) distributionF = LogNormal(30000.0,9000.0,15000.0) |
| <%correlationMatrix%> | Declares the correlation matrix RS (hardcoded), its size, and the correlation elements outside diagonal that are non-zero | RS = CorrelationMatrix(4)  RS[0,1] = -0.2 |
| <%collectionDistributions%> | Creates a collection of all the distrutions using the variable collectionMarginals (currentlyhardcoded) | collectionMarginals=DistributionCollection()  collectionMarginals.add(Distribution(distributionI,"I"))  collectionMarginals.add(Distribution(distributionL,"L"))  collectionMarginals.add(Distribution(distributionE,"E"))  collectionMarginals.add(Distribution(distributionF,"F")) |
| <%inputDescriptions%> | Currently not used, description set in <%collectionDistributions%> | - |

## Script command

Building an OpenTURNS compliant dll is done with the script command:

buildOpenTURNSInterface(modelname,templateFileName)

where modelname is the simulation model and templateFileName is the python script template containing string templates as described in the section above.

This command will build

* An OpenTURNS compliant dll
* An XML file that is used by OpenTURNS to indicate inputs and outputs
* A python script for the analysis based on the template file.

runOpenTURNSPythonScript(filename)

Calls OpenTURNS from the default installation given a python script file name.