Modelling of Uncertainties with Modelica

Daniel Bouskela Audrey Jardin
EDF R&D
6 quai Watier, 78401 Chatou Cedex, France
daniel.bouskela@edf.fr audrey.jardin@edf.fr

Zakia Benjelloun-Touimi
IFP Energies nouvelles
1 & 4, avenue de Bois-Préau, F-92852 Rueil-Malmaison Cedex, France
zakia.benjelloun-touimi@ifpenergiesnouvelles.fr

Peter Aronsson
MathCore Engineering
Teknikringen 1F, SE-58330 Linköping, Sweden
peter.aronsson@mathcore.com

Peter Fritzson
Linköping University, Department of Computer and Information Science
SE-58183 Linköping, Sweden
peter.fritzson@liu.com

Abstract

Many industrial applications, e.g. in power systems, need to use uncertain information (e.g. coming from sensors). The influence of uncertain measurements on the behavior of the system must be assessed, for safety reasons for instance. Also, by combining information given by physical models and sensor measurements, the accuracy of the knowledge of the state of the system can be improved, leading to better plant monitoring and maintenance.

Three well established techniques for handling uncertainties using physical models are presented: data reconciliation, propagation of uncertainties and interpolation techniques. Then, the requirements for handling these techniques in Modelica environments are given. They apply to the Modelica language itself: how to specify the uncertainty problem to be solved directly in the Modelica model. They also apply to model processing: what are the pieces of information that must be automatically extracted from the model and provided to the standard algorithms that compute the uncertainties.

Modelica language extensions in terms of two new pre-defined attributes, `uncertain` and `distribution`, are introduced for Real and Integer variables. This is needed to differentiate between *certain* (the usual kind) variables and *uncertain* variables which have associated probability distributions. An algorithm for extracting from the Modelica model the auxiliary conditions needed by the data reconciliation algorithm is given. These new features have been partially implemented in the MathModelica tool (and soon OpenModelica).

Keywords: data reconciliation; propagation of uncertainties; distribution probability laws; Jacobian matrix; Modelica language extensions; model processing

1 Introduction

The major power plant projects at EDF mainly concern improvements of existing plants (e.g., lifetime extension up to 60 years, power upgrading, reliability improvements, etc.) as well as the construction of new plants (e.g., nuclear, renewable energy, etc.). In that context, EDF has acquired a strong background in the modeling and simulation of electrical and power plant applications for improved investigation and operation.

The physical state of a plant is given by sensor measurements which are subject to uncertainties. Hence, good uncertainty assessment is necessary for the proper monitoring of the plant operation set point, and to comply with the safety margins. By combining information given by physical models and sensor measurements the accuracy of the knowledge of the state of the system can be improved, leading to better plant monitoring and maintenance [6].

Today, engine simulation is important at IFP to solve the problems of air pollution and energy depletion. Many aspects of engine operation are poorly understood because of the problems encountered when attempting to measure the variables that describe the engine's operation. This issue complicates both the engine's control and design. Uncertainty management with parameter sensitivity studies are promising techniques to improve the measurement quality, and subsequently the control and design of internal combustion engines in the years to come.

In order understand the strongly non-linear behaviour of the physical systems under study, IFP has developed a new methodology based on an evolutionary experimental design, kriging and statistical modeling concepts, that are more adequate and accurate than the traditional linear regression techniques [4].

EDF and IFP are currently developing advanced libraries for modeling power plants and engines in Modelica, and are therefore interested in reusing these models for uncertainty computations. The main benefit will lie in the delivery of integrated environments for system modeling and uncertainty studies.

The objective of this article is to make a proposal to extend the Modelica language for the handling of uncertainties. First, several important use cases are presented: data reconciliation, propagation of uncertainties, kriging and response surface methodology. Then the basic requirements for the handling of such techniques with Modelica are established. Finally, a technical proposal for an extension of the Modelica language is given.

2 Techniques for handling uncertainties

2.1 Data reconciliation

The objective of the data reconciliation technique is to improve the knowledge of the physical state of a system using redundant physical measurements of the system and the physical laws that govern the behavior of the system. The increase of knowledge is obtained by reducing the uncertainty intervals of the variables of interest (i.e., that define the state of the system), or in other words by finding the reconciled values of the physical state which are closest to the true values of the physical state, which by definition cannot be exactly known.

By definition, the vector of improvements ν is defined such as:

$$\hat{x} = x + \nu$$

where x and $\hat{x}$ denote resp. the vector of measured values and the vector of improved values, also called reconciled values.

The data reconciliation technique can be formally expressed as an optimization problem, where the goal is to find the vector of improvements such as the objective function ξ_0 attains its minimum value.

$$\xi_0 = \nu^T \cdot S_x^{-1} \cdot \nu \Rightarrow \min_{\nu}(\xi_0)$$

S_x is the covariance matrix, which is symmetric by definition. Its diagonal elements are:

$$s_{x_i}^2 = \left(\frac{w_{x_i}}{\lambda_{95\%}} \right)^2$$

with w_{x_i} being the half-width confidence interval of the measured value x_i, and $\lambda_{95\%} \approx 1.96$ corresponding to a level of confidence of 95%.

Its off-diagonal elements are:

$$s_{x_{i,k}} = r_{x,ik} \cdot s_{x_i} \cdot s_{x_k}$$

where $r_{x,ik}$ is an empirical (estimated) correlation coefficient. Because of the difficulty of estimating these coefficients, they are often set to zero by making the assumption that the measured variables are uncorrelated. Then:

$$\xi_0 = \sum_i \left(\frac{\hat{x}_i - x_i}{s_{x_i}} \right)^2 = \sum_i \left(\frac{\nu_i}{s_{x_i}} \right)^2$$

The reconciled values are constrained by the physical laws such as mass, energy and momentum balances, state functions, correlations, etc, which are expressed in the mathematical model of the system. The subset of the model equations that constrain the reconciled values are called the auxiliary conditions, and denoted $f(\cdot)$. Hence $f(\hat{x}) = 0$, whereas $f(x) \neq 0$ in general. This is why $f(x)$ is called the vector of contradictions.

The algorithm for computing the reconciled values is given in the VDI 2048 standard [5]. This standard considers that the probability function of each measured value follows a Gaussian distribution law that the measurements are performed while the system is in steady-state, and that the measured values are reliable enough so that the vector of improvements is small.

As a consequence of the two last hypotheses, $f(\cdot)$ needs only to be static, and may be linearized around the measured values:

$$f(x) = -\frac{\partial f(x)}{\partial x} \cdot v = -F \cdot v$$

where F is the Jacobian matrix of the auxiliary conditions.

It can then be shown that the reconciled values are given by:

$$\hat{x} = x - S_x \cdot F^T \cdot \left(F \cdot S_x \cdot F^T\right)^{-1} \cdot f(x)$$

The covariance matrix of the reconciled values is given by:

$$S_{\hat{x}} = S_x - S_x \cdot F^T \cdot \left(F \cdot S_x \cdot F^T\right)^{-1} \cdot F \cdot S_x$$

The uncertainties associated with the reconciled values can then be computed from the knowledge of this matrix:

$$s_{\hat{x}_i}^2 = \left(\frac{w_{\hat{x}_i}}{\lambda_{95\%}}\right)^2$$

The Gaussian distribution hypothesis is verified a posteriori by applying a χ^2 test on the reconciled values. This check is done globally by checking the following condition:

$$\xi_0 \leq \chi^2_{r,95\%}$$

where $\chi^2_{r,95\%}$ is given by a χ^2-table and r is the number of auxiliary conditions. Local checks on individual variables may be performed as well.

In practice, the actual implementation of the method differs slightly from its theoretical formulation in order to avoid the cumbersome matrix inversions. To that end, they are replaced by the resolution of linear equations. For instance, the reconciled values are computed using the following procedure:

$$\hat{x} = x - S_x \cdot F^T \cdot f^*$$

with f^* such as:

$$\left(F \cdot S_x \cdot F^T\right) \cdot f^* = f(x)$$

f^* may be solved numerically using the Newton algorithm.

2.2 Propagation of uncertainties

From the knowledge of estimated sources of uncertainties, it is possible to derive the uncertainties of variables of interest by propagating the sources of uncertainties through a model of the system.

To that end, EDF R&D has developed a complete methodology. This methodology is divided into four steps, as illustrated in Figure 1.

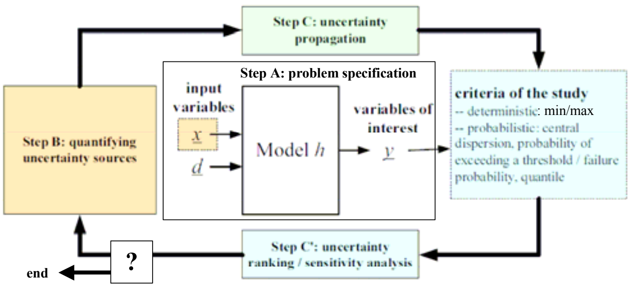

Step C: uncertainty propagation
Step A: problem specification
input variables
$\underline{x}$
$\underline{d}$
Model h
variables of interest
$\underline{y}$
Step B: quantifying uncertainty sources
criteria of the study
-- deterministic: min/max
-- probabilistic: central dispersion, probability of exceeding a threshold / failure probability, quantile
Step C': uncertainty ranking / sensitivity analysis
?
end

Figure 1: Steps of the uncertainty propagation methodology developed at EDF R&D [8].

Step A aims at identifying the sources of uncertainties $\underline{x}$, the variables of interest $\underline{y}$, and the model of the system $h(\cdot)$. The uncertainty study is then formally expressed as:

$$\underline{y} = h(\underline{x}, \underline{d})$$

where $\underline{d}$ denotes the variables treated as certain (i.e. whose uncertainties can be neglected).

A decision criterion must also be defined to indicate how the uncertainties of the variables of interest have to be quantified and to determine the final objective of the uncertainty study. The criterion can be deterministic, by assessing a minimum and a maximum value for each variable of interest, or probabilistic like the probability of exceeding a given threshold, a quantile, or a central dispersion.

Step B aims at quantifying the sources of uncertainties. When the deterministic criterion is chosen, a min and a max value must be associated with each variable. When the probabilistic criterion is chosen, the sources of uncertainties are treated as the components of a random vector $\underline{X}$. For each individual component X^i, the probability distribution must be assessed. Also, the statistical dependency between two components X^i and X^j should be evaluated in the form of correlation coefficients.

Step C is the uncertainty propagation through the model. When a deterministic criterion is chosen, finding the minimum and the maximum values of $\underline{y}$ is quite easy if the model is monotonous wrt. $\underline{x}$. Otherwise this search may become a potentially complex optimization problem. To alleviate this difficulty, some optimization algorithms or simplified approaches based on design of experiments to estimate extreme values of $\underline{y}$ may be used. When a probabilistic criterion is chosen, the difficulty is to characterize the probability distributions of the random vector $\underline{Y} = h(\underline{X}, \underline{d})$. For the assessment of expectation/variance or the probability of exceeding a threshold, both approximation methods (e.g. quadratic combination, FORM-SORM methods) and sampling methods (e.g. Monte Carlo simulations, Latin Hypercube simulations) can be used. For the assessment of a quantile, only sampling methods can be applied (e.g. Wilk's formula). Contrary to approximation methods, sampling methods make no assumption on the model $h(\cdot)$, but may be very CPU-time consuming.

<u>Step C'</u> is the ranking of the sources of uncertainties and the sensitivity analysis. Chosen indicators are used to rank the uncertainty sources with respect to their impact on the uncertainties of the system's characteristics of interest. Depending on the result of this ranking, the modeling of the sources of uncertainty can be adjusted (or some sources of uncertainties can be neglected) to perform another propagation.

An uncertainty study rarely finishes after a first round of steps A, B, C and C'. Step C' actually plays a crucial role since ranking the results highlights the variables that truly determine the relevance of the final results. If the uncertainty laws of some input variables have been chosen too roughly during step B, it is necessary to collect additional information on the influential sources of uncertainty and re-apply the whole methodology to refine the analysis, and so-on until satisfaction.

Probability distributions may be expressed in the form of parametric distribution laws with the help of a limited number of parameters $\underline{\theta}$.

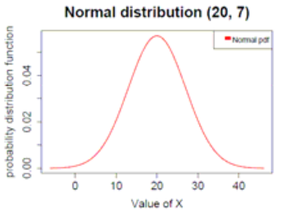
Two examples of such parametric distribution laws are given below [8].

<u>Normal (or Gaussian) distribution</u>

$\underline{\theta} = (\mu, \sigma)$

μ is the mean value

σ is the standard deviation

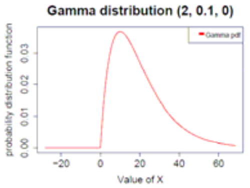


$$f_X(x,\underline{\theta}) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{x-\mu}{\sigma} \right)^2 \right)$$

<u>Gamma distribution</u>

$\underline{\theta} = (\lambda, k, \gamma)$

$\lambda > 0$, $k > 0$



$$f_X(x,\underline{\theta}) = \frac{\lambda}{\Gamma(k)} \cdot (\lambda \cdot (x-\gamma))^{k-1} \cdot \exp(-\lambda \cdot (x-\gamma)) \cdot 1_\gamma(x)$$

where $\Gamma(\cdot)$ is the gamma function

$$\Gamma(k) = \int_0^{+\infty} t^{k-1} \cdot \exp(-t) \cdot dt$$

The choice of the correct distribution law depends on the application. For instance, the normal distribution is relevant in metrology.

The dependency between variables may be expressed using the copula theory or the Pearson correlation coefficient [15, 16]. The latter is defined as:

$$\rho^P_{X^i,X^j} = \frac{Cov(X^i, X^j)}{\sigma_{X^i} \cdot \sigma_{X^j}}$$

where $Cov(X^i, X^j)$ is the covariance between X^i and X^j, and σ_{X^i} and σ_{X^j} are respectively the standard deviations of X^i and X^j.

Several techniques may be used for propagating the uncertainties.

The quadratic combination method is a probabilistic approach based on the Taylor decomposition of $\underline{Y}$ wrt. $\underline{X}$ around the mean point $\underline{\mu}_X$.

The Monte-Carlo method is a numerical integration method using sampling, which can be used, for example, to determine the expectation μ_{Y^i} and the standard deviation σ_{Y^i} of each variable of interest Y^i.

Several ranking techniques may be used, such as those based on the quadratic combination's importance factor or the Pearson correlation. They aim at finding the influence of the inputs X^i on the outputs Y^j.

2.3 Interpolation techniques

Mathematical models of physical systems are important tools in many fields of scientific research. But better knowledge of systems behavior and increasingly desired accuracy lead to higher complexity of models, which, in this context, sometimes are not sufficient to meet the expectations of the experimenters.

Uncertainty studies must in particular be adapted to handle such complex models. The following section describes some examples of advanced methods that are especially used at IFP: the kriging method, the experimental design theory and the Response Surface Methodology (RSM).

2.3.1 Kriging with non-linear trend

One of the studies at IFP using the kriging method is the analysis of a catalytic system for pollution control, which consists in post-treating smoke produced by diesel engines through NOx trap (i.e. Nitrogen-Oxides trap [3]).

As a surrogate of the real system, a kinetic model was developed to represent the physico-chemical phenomenon, depending on parameters (e.g. pre-exponential factors, activation energies, adsorption constants) that cannot be obtained from theoretical considerations. Therefore, experiments are required to calibrate the model. A criterion is suggested for experimental designs adapted to kinetic parameters identification, when the model is highly non-linear and the kinetic model does not fit well experimental data. These differences observed between kinetic model and experimental data, can be represented by a Gaussian process realization. Gaussian process often accounts for correlated errors due to lack of fit.

More explicitly, the model is represented by:

$$y = f(x, \beta)$$

where y is the response vector (*e.g.* the NOx concentration), x the experimental conditions and β the kinetic parameters of the model represented by the non-linear function f.

Then, the first model is corrected and replaced by:

$$y = f(x, \beta) + Z_{\sigma^2,\theta}(x)$$

where $Z_{\sigma^2,\theta}(x)$ is a centered Gaussian process with Gaussian covariance kernel specified by a variance σ^2 and a vector θ of scale parameters.

As an example, the covariance kernel can be given by a kriging approach which is commonly used in the field of computer experiments. However, in traditional use, the trend is linear. Its estimation is obtained through an analytical formula as well as its uncertainty.

The first difficulty is to estimate the trend parameters considering its non-linear behavior. Similarly to non-linear regression, the traditional analytical formula for β is then replaced by a minimization procedure. In this case, the theory of kriging with non-linear trend can be applied as summarized below.

The covariance kernel of the centered Gaussian process is defined by:

$$Cov\left(Z_{\sigma^2,\theta}(x), Z_{\sigma^2,\theta}(x+h)\right) = \sigma^2 \cdot R_\theta(h)$$

where the Gaussian spatial correlation is used and expressed by:

$$\forall h \in \Re^k, R_\theta(h) = \exp\left(-\sum_{i=1}^{k} \theta_i h_i^2\right)$$

Let m be the number of design points and $y = \begin{pmatrix} y_1 & \dots & y_m \end{pmatrix}^T$ the outputs observed at location $s = \begin{pmatrix} s_1 & \dots & s_m \end{pmatrix}^T$, $s_i \in \Re$. Using maximum likelihood estimation, expression of kriging predictor $\hat{y}$ and variance prediction φ at a new location x_0 are given by:

$$\hat{y}(x_0) = r \cdot R^{-1} \cdot y - \left(F^T \cdot R^{-1} \cdot r - f\right)^T \cdot \left(F^T \cdot R^{-1} \cdot F\right)^{-1} \cdot F^T \cdot R^{-1} \cdot y$$

$$\varphi(x_0) = \sigma^2 \cdot \left(1 + \left\|F^T \cdot R^{-1} \cdot r - f\right\|_{F^T \cdot R^{-1} \cdot F} + \|r\|_R\right)$$

where:

$$\|u\|_A = u^T \cdot A^{-1} \cdot u$$

$$r = \begin{pmatrix} R_{\hat{\theta}}(x_0 - s_1) & \dots & R_{\hat{\vartheta}}(x_0 - s_m) \end{pmatrix}^T$$

$$\forall i \in [\![1;m|], \forall j \in [\![1;m|], R_{ij} = R_{\hat{\theta}}(s_i - s_j)$$

$$F = f(s, \hat{\beta})$$

$$f = f(x_0, \hat{\beta})$$

and the parameters are obtained by solving recursively and simultaneously the following simultaneous equations:

$$\hat{\beta} = \min_\beta\left((y - F)^T \cdot R^{-1} \cdot (y - F)\right)$$

$$\hat{\sigma} = m^{-1} \cdot (y - F)^T \cdot R^{-1} \cdot (y - F)$$

$$\hat{\theta} = \arg\min\left(\hat{\sigma}^2 \cdot \left|R^{-1}\right|^{1/m}\right)$$

The minimization algorithm determines $\hat{\theta}$ through the modified Hooke and Jeeves method, described in Kowalik and Osborne [11]. For more details, see Lophaven *et al.* [12]

Notice that the predictor $\hat{y}$ and the variance prediction φ depend on the estimator $\hat{\beta}$ through F and f. As a consequence, in universal kriging, kriging predictor and prediction variance expressions cannot be interpreted as conditional expectation and variance (see Helbert *et al.* [13]). This is due to the fact that only uncertainties induced by the estimation of trend parameters are taken into account, and not those created by approximating variance and correlation parameters. Hence, it can underestimate the uncertainty on the response and lead to very important difficulties for non-linear models.

2.3.2 Experimental Design theory and Response Surface Methodology

The Response Surface Methodology (RSM) has been described in detail by Dejean *et al.* [14]. The purpose is to approximate a complex process with respect to uncertain parameters belonging to a region of interest. Engineers define a performance measure of the process called response y (*e.g.* cumulative oil production, field net present value, …) and some input variables x_1, x_2, x_3, … called factors, that are assumed to influence the response (*e.g.* petrophysics, field structural map, well locations, economic factors, …). The input variables correspond to the prior uncertainties on the process. RSM provides tools for identifying the factors that are influential factors, such as illustrated below:

$$y = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3 + \ldots$$
$$+ a_1 \cdot a_2 \cdot x_1 \cdot x_2 + \ldots$$

where a_0, a_1, … are constant coefficients obtained by fitting a set of numerical simulations.

The main interest of this RSM model is its negligible cost to get new values of the response compared to CPU-time consuming simulations. This regression model can then be used to make predictions of the process over the uncertain domain and to generate probabilistic distribution of the response using Monte Carlo sampling technique.

A sufficient number of response values corresponding to different factor values is necessary in order to fit this model. These values should be representative enough of the behavior of the response in the domain of variation of the factors. Thus the experimental design theory is applied, since, for a given objective and a given uncertain domain, it delivers the right set of model simulations to be performed in order to properly model the response behavior in the uncertain domain. Many experimental designs are available depending on both the objective (sensitivity study or risk analysis study) and the acceptable CPU-time. The selected model simulations must:

(a) be numerous enough so that all the coefficients a_i of the model can be estimated and

(b) ensure good quality of the model, both in terms of accuracy and prediction.

For sensitivity studies, the objective is to identify the uncertain parameters that influence the response. In that case, the RSM model does not need to be very accurate and classical sensitivity designs are the two-level fractionals.

For risk analysis purposes, the RSM model should be of good quality in order to deliver accurate predictions. The composite designs are in that case the most appropriate. All those designs are well known and tabulated. They have the best properties with respect to the objective of the study, but they can still be too expensive or on the contrary too coarse in a specific context. In that case, some other designs such as small composite designs or optimal designs can be used.

Once the RSM model has been fitted, it can be used to compute probabilistic distributions of the production forecasts as a function of the main model uncertainties. This is commonly called a technical risk analysis.

3 Requirements on the Modelica language and tools

In this section, the requirements are given independently from the existing Modelica specifications and tools. There are two kinds of requirements: those who apply to the language itself, and those who apply to the model processor. The model processor is defined as the tool that has the analysis and symbolic manipulation capabilities to produce the desired result from the Modelica model.

3.1 Identifying the uncertain variables

In the following paragraphs, only continuous variables will be considered.

There are two broad kinds of variables: *certain* or *uncertain*. *Certain* variables are single valued variables, whose values are known (explicitly given as inputs of the model), or unknown (implicitly given as outputs of the model). *Uncertain* variables are random variables which represent probability distributions.

Formally, a certain variable could be seen as an uncertain variable with a normal distribution of zero standard deviation (i.e. a Dirac function). It is however preferable to continue handling these two types of variable separately, because of the infinities associated with Dirac functions.

In the case of uncertainty propagation, the sources of uncertainty are the inputs and the variables of interest are the outputs of the computation. Input means that the distribution law of the variable is given, and output means that the distribution law is computed. Output uncertain variables may also be called observation variables.

In the case of data reconciliation, uncertain variables are considered as both inputs and outputs, the difference between the outputs and the inputs being the vector of improvements.

Note that the words “input” and “output” should not be confused here with the semantics of the Modelica keywords “input” and “output”. As they indeed have different meanings, other keywords will be used when implementing these notions in Modelica, as described in Section 4.

[R1] The Modelica language should give the possibility to declare uncertain variables as inputs only or as outputs only (for the propagation of uncertainties), or as inputs/outputs (for data reconciliation).

3.2 Assigning parametric distribution laws to uncertain variables

An uncertain variable x_i may be characterized by the parameters of its distribution law $\underline{\theta}_{x_i} = (\alpha_{x_i}, \beta_{x_i}, ...)$.

In the frequent case of a normal distribution, these parameters are the mean value μ_{x_i} and the standard deviation σ_{x_i} . Then, x_i may be written as:

$$x_i = \mu_{x_i} \pm w_{x_i}$$

where the half-interval w_{x_i} is a multiple of σ_{x_i} :

$$w_{x_i} = \lambda \cdot \sigma_{x_i}$$

λ being a function of the level of confidence. For instance, for a level of confidence of 95%, $\lambda_{95\%} = 1.96$. This expression means that there are 95% chances that the value of x_i is in the interval $\left[\mu_{x_i} - \lambda_{95\%} \cdot \sigma_{x_i} ; \mu_{x_i} + \lambda_{95\%} \cdot \sigma_{x_i}\right]$.

[R2] The Modelica language should give the possibility to assign one parametric distribution law to each uncertain variable by specifying its name (e.g. normal distribution) and the values of its parameters (e.g. mean value and standard deviation, or alternatively mean value, confidence level and half-interval in the case of a normal distribution).

3.3 Specifying dependencies between uncertain variables

[R3] The Modelica language should give the possibility to specify the dependencies between uncertain variables in the form of correlation matrices (covariance matrices, matrices of Pearson correlation coefficients, etc.).

3.4 Handling redundant information for data reconciliation

When performing data reconciliation, redundant information is fed into the model as shown in the following example (see Figure 2).

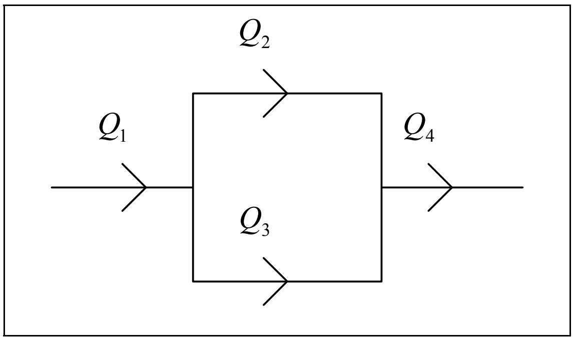


Figure 2: Example of data reconciliation for flow measurements

In Figure 2, the hydraulic circuit is instrumented with four flow meters, which give respectively the following values and uncertainties (all quantities are given in kg/s):

$$\begin{cases} Q_1 = 5.0 \pm 1.0 \\ Q_2 = 2.5 \pm 0.5 \\ Q_3 = 2.6 \pm 0.1 \\ Q_4 = 5.5 \pm 0.5 \end{cases}$$

Writing the mass balance equations for the system yields:

$$\begin{cases} Q_1 = Q_2 + Q_3 \\ Q_4 = Q_2 + Q_3 \end{cases}$$

Considering the values at the center of the uncertainty intervals, it is obvious that the equations system is not satisfied. However, the equations may be satisfied by carefully choosing the proper values within the uncertainty ranges. This is what is actually done when applying the data reconciliation technique, which yields:

$$\begin{cases} \hat{Q}_1 = 5.3 \pm 0.3 \\ \hat{Q}_2 = 2.7 \pm 0.3 \\ \hat{Q}_3 = 2.6 \pm 0.1 \\ \hat{Q}_4 = 5.3 \pm 0.3 \end{cases}$$

Note that the uncertainty intervals have been reduced.

At the present time, the Modelica language only handles square systems of physical equations (having as many unknowns as equations). The question is how to consider the four variables for the model: are they inputs or outputs? If they are all considered as inputs, then the system is over-constrained. If they are all considered as outputs, then the system is under-constrained. Two of them could be considered as inputs, and the other two as outputs (6 possibilities) to obtain a square system. Note that the last alternative is applicable to the reconciled values, all possibilities being equivalent as the reconciled values satisfy the auxiliary conditions.

The most natural way is to consider the four variables as both inputs and outputs of the data reconciliation algorithm, and use requirement [R1] to that end, so no additional requirement on the Modelica language is necessary. However, as the term "input/output" for data reconciliation is somewhat ambiguous, because it may be confused with the terms "input" and "output" for DAE simulation, the vector of inputs/outputs for data reconciliation will be called the vector of control variables (or control vector) in the rest of the paper, because it controls (defines) the state of the system.

Once the data reconciliation algorithm is completed and the variables are assigned their reconciled values, they can be considered as standard inputs or outputs to have a square system, as required for DAE simulation. Doing so, the same model could be used for data reconciliation and simulation. Data reconciliation would be used to compute an improved state from redundant measurements, and the result would be readily used as the initial state of subsequent simulations.

In the above example, all mass flow rates would be declared as inputs/outputs for data reconciliation, but only two of them (no matter which) would be declared as standard input variables for simulation.

3.5 Extracting the auxiliary conditions for data reconciliation

The auxiliary conditions $f(\cdot)$ constitute the subset of the model equations that constrain the control variables. $f(\cdot)$ must be extracted from the model because, as shown in Section 2.1, $f(x) \neq 0$ before data reconciliation and $f(x)$ must be evaluated by the data reconciliation algorithm, where x is the control vector.

The extraction of $f(\cdot)$ should be fully automatic, as the auxiliary conditions may be scattered throughout the whole model, which is usually a graph of connected model components.

[R5] The model processor should be able to extract the auxiliary conditions from the model.

3.6 Computing the Jacobian matrix of the auxiliary conditions for data reconciliation

The Jacobian matrix of the auxiliary conditions is defined as:

$$F = \frac{\partial f(x)}{\partial x}$$

where $f(\cdot)$ are the auxiliary conditions, extracted from the model equations, and x is the control vector. Note that F is not square as the number of control variables is greater than the number of auxiliary conditions due to the fact that data reconciliation is based on the use of redundant information.

[R6] The model processor should be able to compute the Jacobian matrix of the auxiliary conditions wrt. the control vector.

3.7 Performing the data reconciliation algorithm

Once the quantities x, S_x, $f(\cdot)$ and F are known, it is possible to run the data reconciliation algorithm. Two alternatives are possible. The first is to write a Modelica script of the algorithm that could be compiled with the model. The second is to write a program in another environment such as Python [7], that would have access to a Modelica functional interface that would provide the values of the above quantities upon request from the main program.

In the above example:

$$x = \begin{pmatrix} Q_1 & Q_2 & Q_3 & Q_4 \end{pmatrix}^T$$

$$S_x = diag\left(\left(\frac{w_{Q_1}}{\lambda_{95\%}}\right)^2, \left(\frac{w_{Q_2}}{\lambda_{95\%}}\right)^2, \left(\frac{w_{Q_3}}{\lambda_{95\%}}\right)^2, \left(\frac{w_{Q_4}}{\lambda_{95\%}}\right)^2\right)$$

$$f(x) = \begin{pmatrix} Q_1 - Q_2 - Q_3 \\ Q_2 + Q_3 - Q_4 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{pmatrix}$$

[R7] The model processor should be able to provide x, S_x, $f(\cdot)$ and F to the data reconciliation algorithm.

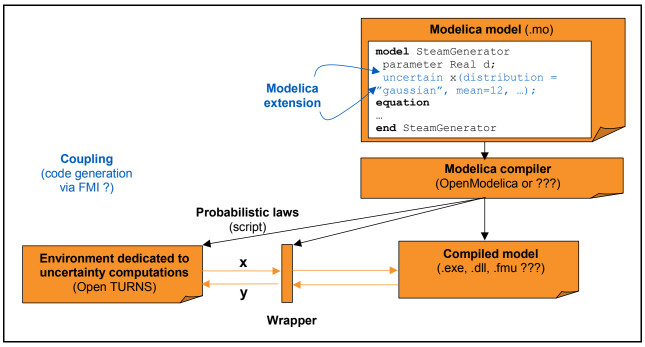
3.8 Performing uncertainty propagation algorithms

The uncertainty propagation methodology is implemented in OpenTURNS, which is an open source tool developed by EDF/R&D, Phimeca and EADS [8].

The knowledge of the sources x, the model $f(\cdot)$ and the variables of interest y is sufficient to run the OpenTURNS algorithms. $y = f(x)$ must be evaluated upon request from OpenTURNS, and its value y returned to OpenTURNS.

Note that it is much easier to use OpenTURNS than the data reconciliation in a Modelica environment, as in the case of OpenTURNS no processing is required on the model other than identifying which variables constitute the vectors x and y. $f(\cdot)$ is the whole original model, as compared to the extracted auxiliary conditions for data reconciliation.

The uncertainty propagation algorithms may thus be performed by coupling the Modelica environment with OpenTURNS, as shown in Figure 3. The coupling principle is to automatically generate the interface between OpenTURNS and the simulation code compiled from the Modelica model. This can especially be done by extending the Modelica compiler.



```
model SteamGenerator
 parameter Real d;
 uncertain x(distribution =
"gaussian", mean=12, …);
equation
…
end SteamGenerator
```

Figure 3: Principle for coupling OpenTURNS with a Modelica platform

[R8] The model processor should be able to interface the model with OpenTURNS.

4 Proposal for a Modelica language extension

This section gives a preliminary design proposal of Modelica language extensions for supporting uncertainties. This proposal will be considered for test implementation in MathModelica [10] and OpenModelica [9].

The proposal to support requirement [R1], described in Section 3.1, is to introduce a *new attribute* for the built-in classes Real and Integer. This attribute should have an enumeration type that allows specifying it as `given` (i.e., kind of "input"), `sought` (kind of "output") or `refine` (kind of "input/output" whose uncertainty is refined). In order to distinguish it from the already established semantics for input and output variables in Modelica we propose the following naming:

```
type Uncertainty = enumeration(
 given "a known uncertainty",
 sought "an unknown uncertainty",
 refine "a known uncertainty to be refined"
);
```

The attribute is named `uncertain`, here showed for the Real type (as described in the Modelica language specification):

```
type Real  // Note: defined with Modelica syntax although predefined
  …
  parameter Uncertainty uncertain;
equation
  …
end Real;
```

Let us illustrate how this is used by an example:

```
model Process
  parameter Real p1=0.1;
  Real v1;
  Real v2(uncertain=Uncertainty.given);
equation
…
end Process;
```

To support [R2], described in Section 3.2, we propose another *new attribute* called `distribution` for the built-in classes Real and Integer.

```
type Real  // Note: defined with Modelica syntax although predefined
  …
  parameter Distribution distribution;
equation
  …
end Real;
```

The following distributions are proposed (tool vendors could be allowed to extend this list themselves):

```
partial record Distribution
  parameter String name;
end Distribution;

record NormalDistribution
   extends Distribution(
     final name="Normal");
   parameter Real mu,sigma;
end NormalDistribution;

record ExponentialDistribution
  extends Distribution(
     final name="Exponential");
  parameter Real lambda,gamma;
end ExponentialDistribution;
```

```
record GammaDistribution
  extends Distribution(final name="Gamma");
  parameter Real lambda,k,gamma;
end GammaDistribution;

record LogNormalDistribution
  extends Distribution(
     final name="LogNormal");
  parameter Real mu, sigma,gamma;
end LogNormalDistribution;

record TriangularDistribution
  extends Distribution(
     final name="Triangular");
  parameter Real a,b,m;
end TriangularDistribution;

record UniformDistribution
  extends Distribution(
     final name="Uniform");
  parameter Real a,b;
end UniformDistribution;

record BetaDistribution
  extends Distribution(
     final name="Beta");
  parameter Real r,t,a,b;
end BetaDistribution;

record PoissonDistribution
  extends Distribution(
     final name="Poisson");
  parameter Real mu;
   end PoissonDistribution;
```

The rationale for introducing the distributions and the uncertainty properties as attributes for the built-in classes Real and Integer is the flexibility that they bring. With this approach it becomes possible, for instance, to change several uncertainties at once by using parameterization:

```
model Process
  replaceable
   parameter Distribution d =
                  Distribution("");
   parameter Uncertainty u =
                  Uncertainty.given;
  Real x(distribution=d, uncertain=u);
  Real y(distribution=d, uncertain=u);
…
end Process;
```

The requirement [R3] to support the addition of dependencies between variables is performed at the top level of a model. It is only there that the modeler knows what the dependencies are, since they might appear due to e.g. connections of components. What is required is to be able to express a dependency between two uncertain variables and give a correlation coefficient. This coefficient can be of different kinds, as briefly mentioned in Section 3.3. The user has to select one kind of coefficients, e.g. covariance or Pearson. It is not possible to mix different kinds in a model. We propose to support R3, not by extending the Modelica language, but instead by allowing the user to introduce a set of equations at the top level that the tool can recognize, as follows:

```
model System
  Process p1,p3;
  Process2 p2;
  CovCorrelation[:] covCorrelation =
   {CovCorrelation(p1.q,p2.q,0.1),
    CovCorrelation(p2.q,p3.q,0.2)};
end System;
```

The rationale for this design to support R3 is that there is no need for a language extension (which avoids to make the language more complex). A similar record definition is required for Pearson correlation coefficients.

With the above language extensions, it becomes possible for a Modelica tool to automatically fulfill requirements [R4, R5 and R6]. How this is performed is explained in the next section.

5 New features for model processing

Apart from the necessary language extensions proposed in previous section, the analysis of models with uncertainties requires some new features from a Modelica tool perspective.

For data reconciliation it is required that the set of equations f that constrain the control variables can be extracted. Remember that these equations together with the uncertain variables typically result in an underdetermined system, i.e., there are more variables than equations. The method of extracting the equations is as follows. Given a model M with variables vector X:

1. Perform a causality analysis by running BLT sorting on the complete system. This yields a series of blocks B_i (i=1 to n).
2. Remove from M all blocks B_i that are square wrt. the subset of X solved in earlier blocks $(B_1 \ldots B_{i-1})$ referenced by B_i.

3. Remove from M all blocks B_i that do not influence X (i.e., downstream the causality analysis from X).
4. If there are one or more components from X that do not belong to any of the remaining blocks B_i, raise an error.
5. Otherwise, f is the set of the equations corresponding to the remaining blocks B_i.

The result from step 5 is the set of equations that constrain the control variables. This approach has one problem that needs some attention. The blocks identified can both contain uncertain control variables and normal variables. In such a case, it can be necessary to eliminate the normal variables to reduce the equations to only contain control variables. Such eliminations can only be performed if for instance all inverses of used functions are available.

The tool will also construct the Jacobian matrix of this function as required by the data reconciliation algorithm.

The connection with OpenTURNS to be able to perform uncertainty propagation is straightforward. OpenTURNS simply requires a computational block that can compute the outputs given a certain input.

6 Results

The proposed language extensions have been implemented in the OpenModelica compiler frontend, and the extraction algorithm for data reconciliation presented in the previous section has been implemented in MathModelica.

The example in Figure 4 is used as a test case. It is the model of a fluid pipe system with a pump feeding the system and a volume collecting the output flow from the system.

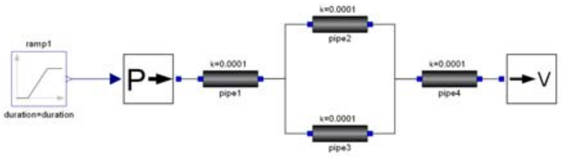


Figure 4 Example of pipe system with uncertainties

The pipe model is defined as:

```
model Pipe
  import SI = Modelica.SIunits;
  Port port_a;
  Port port_b;
  SI.VolumeFlowRate q
       "flow from port_a to port_b";
  SI.Pressure dp "pressure drop over pipe";
  parameter Real k=0.0001
       "Friction factor";
  parameter Real rho=1000.0
       "density (water by default)";
equation
  q=port_a.q;
  dp=port_a.p - port_b.p;
  port_a.q + port_b.q=0;
  dp=k/rho*q*abs(q);
end Pipe;
```

The uncertain variables are declared at the system level as follows (all pipes have q as uncertain variable to be refined):

```
model PipeSystem
  Pipe pipe1(q.uncertain = Uncer-
tainty.refine);
  Pipe pipe2(q.uncertain = Uncer-
tainty.refine);
…
end PipeSystem;
```

Figure 5 shows a simulation of the pipe system when a ramp signal is applied to the pump. Due to the same friction coefficients of the pipe segments the flows are symmetrically distributed through the system.

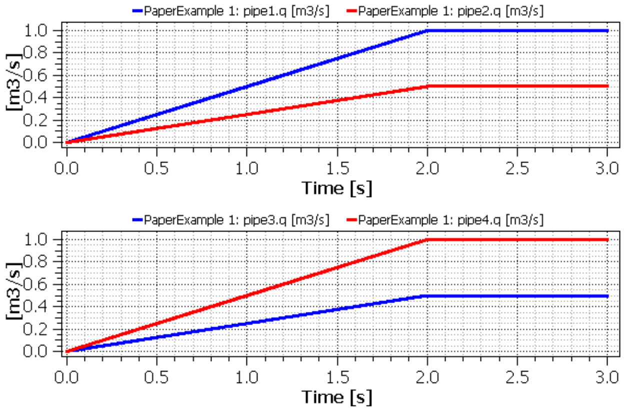


Figure 5 Simulation of the pipe system with MathModelica

Running the extraction algorithm and thereafter eliminating undesired variables result in the following equations, which constitute f(X) (note that X here is {q1,q2,q3,q4}):

$$q1-q2-q3 = 0; \tag{1}$$

$$0.0001*q2*abs(q2)-0.0001*q3*abs(q3) = 0; \tag{2}$$

$$q2+q3-q4 = 0; \tag{3}$$

In this example all undesired variables have been eliminated resulting in a system only containing the uncertain variables that are candidates for refinement. Equations 1 and 3 originate from the connect statements (flows are summed up to zero), and equation 2 comes from the pressure drop over the pipes. This example is by intent very similar to the example in Figure 2, so we can reuse the same measured values and uncertainty intervals. The only difference is that we have here three constraint equations instead of two.

The data reconciliation algorithm has been run with Mathematica, that used the auxiliary equations f extracted by MathModelica from the original model. Below are the calculations for new measurements and confidence matrix:

```
xr=x-Sx.Transpose[F].Inverse[F.Sx.Transpose[F]].fx
{{5.22801},{2.61526},{2.61275},{5.22801}}
Sxr=Sx-Sx.Transpose[F].Inverse[F.Sx.Transpose[F]].F.Sx
{{0.00865708,0.00441341,0.00424367,0.00865708},
{0.00441341,0.00224998,0.00216344,0.00441341},
{0.00424367,0.00216344,0.00208023,0.00424367},
{0.00865708,0.00441341,0.00424367,0.00865708}}
```

This results in new estimates as:

Q1 = 5.2 ± 0.2
Q2 = 2.6 ± 0.1
Q3 = 2.6 ± 0.1
Q4 = 5.2 ± 0.2

This result differs somewhat from the original example in Figure 2. The reason for this is the extra equation for the pressure drop, which adds another constraint to the system. A reasonable conclusion is that this reconciliation run performs a better job compared to the original example, simply because it has more knowledge of the system in the form of one additional constraint.

7 Conclusions

Two techniques for the handling of uncertainties with Modelica have been presented, and the requirements from a modeling language and tool perspective have been identified. Furthermore their support in the OpenModelica and MathModelica tools has been implemented in order to be able to specify the uncertainties directly in the Modelica models and to extract automatically the necessary pieces of information for the data reconciliation algorithms. The extraction algorithm has been tested and verified on a simple Modelica model. The results seem promising for future developments and real industrial validation, which will be done in the near future in the scope of the OPENPROD project. The connection with OpenTURNS has not been implemented yet, but the authors foresee no major issues in this work, which is also planned to be done soon.

Acknowledgements

This work was partially supported by the pan-European ITEA2 program and the French and Swedish governments through the OPENPROD project.

References

[1] Fang K.T., Li R., Sudjianto A., *Design and Modeling for computer experiments*, Chapmann & Hall/CRC, 2006

[2] Santner T., Williams B., Notz W., *The Design and Analysis of Computer Experiments*, Springer, Berlin, 2003.

[3] Canaud M., Wahl F., Helbert C., Carraro L., *Design of experiments for smoke depollution from the output of diesel engine*, submitted.

[4] Ferraille M., Busby D., *Uncertainty management on a reservoir workflow*, in International Petroleum Technology Conference, IPTC13768, 2009.

[5] Verein Deutscher Ingenieure, *Uncertainties of measurement during acceptance tests on energy-conversion and power plants – Fundamentals*, Standard VDI 2048, 2000.

[6] Zornoza J., Favennec J.-M., Szaleniec S., Piedfer O., *Feedwater Flow-rate And Thermal Power Monitoring And Adjustment By Data Reconciliation In NPPs*, in Proceedings of the 51st ISA POWID Symposium, June 8-13, Scottsdale, USA, 2008.

[7] Torabzadeh-Tari M., Fritzson P., Sjölund M., Pop A., *OpenModelica-Python Interoperability Applied to Monte Carlo Simulation*, in Proceedings of the 50th Scandinavian Conference on Simulation and Modeling (SIMS'2009), available at www.scan-sims.org. Fredericia, Denmark. October 7-8, 2009.

[8] EDF-EADS-PhiMeca, *OpenTURNS (version 0.13.1) – Reference Guide*, 2007, available at: http://trac.openturns.org/wiki/Documentation.

[9] Open Source Modelica Consortium. OpenModelica, www.openmodelica.org

[10] MathCore Engineering AB. MathModelica. **www.mathcore.com**

[11] Kowalik J., Osborne M.R., *Methods for unconstrained optimization problems,* Elsevier, New York, USA, 1968.

[12] Lophaven *et al.*, *Aspects of the Matlab toolbox DACE,* in Informatics and mathematical modeling, DTU, 44p., 2002.

[13] Helbert *et al.*, *Assessment of uncertainty in computer experiments: from universal kriging to Bayesian kriging*, in Applied Stochastic Models in Business and Industry, vol. 25, pp. 99-113, 2009.

[14] Dupraz P., *Modélisation et commande avancées d'un moteur diesel à injection directe,* 1998.

[15] Dixon W.J., Massey F.J., *Introduction to statistical analysis (4th edition)*, McGraw-Hill, 1983.

[16] Embrechts P., Lindskog F., McNeil A., *Modelling Dependence with Copulas and Applications to Risk Management,* in Handbook of Heavy Tailed Distributions in Finance, ed. S. Rachev, Elsevier, Chapter 8, pp. 329-384, 2003.