# Learning Modelica Models from Component Libraries

Gregory Provan[1] Alex Feldman[2]

[1]Computer Science Department, University College Cork, Cork, Ireland, g.provan@cs.ucc.ie
[2]PARC Inc., Palo Alto, CA 94304, USA, afeldman@parc.com

## Abstract

The Modelica language is one of the most important languages for representing a large class of systems, ranging from vehicles to climate control systems in buildings. Component libraries, containing components like valves, motors, pumps, etc., have been built to facilitate the construction of complex systems, but at present model construction and parameter estimation are entirely manual. We have developed software for automating the process of constructing models that optimise a range of metrics (e.g., model-simulation accuracy or diagnostics accuracy) to tailor models to a range of applications, such as diagnosis and control. We describe initial experimental results using a well-known dynamical benchmark model.

*Keywords: Model Learning, Bayesian model selection*

## 1 Introduction

Modelica is an important language for simulating a large class of systems, ranging from vehicles (Brückmann et al., 2009) to physiology (Mateják et al., 2014) and climate control systems in buildings (Plessis et al., 2014; Wetter et al., 2014). Given the increasing use of Modelica as a language for systems modeling, it is being applied to a range of applications beyond system simulation, such as embedded control, diagnosis, and prediction (Casella et al., 2008). For these other applications, the models must satisfy requirements beyond just simulation accuracy. Consider the case of control models: these models must balance model fidelity (to represent fundamental dynamics necessary for control system performance) with simplicity, since most advanced control design techniques, e.g., model-predictive control, start to become intractable for systems of order greater than about ten. Creating tractable control models requires simplifying traditional simulation models, deleting any behaviours that do not significantly affect the equilibrium values and/or the control-relevant system dynamics (Casella et al., 2008).

For simulation purposes, the key criterion for a Modelica simulation model is accuracy, i.e., how closely does the model simulate the real system. In such cases, we can judge the quality of a Modelica model in terms of a criterion such as the mean squared error (MSE) measure, specified in terms of the difference between simulation output and data from the real system.

However, for applications like embedded control and diagnosis, other criteria are necessary. The computational complexity of evaluating the Modelica model is important, since inference involving control and diagnosis typically involves multiple simulations to determine an optimal system state. In addition, diagnostics accuracy may require different model properties than just simulation accuracy; for example, diagnosis involves computing whether the nominal simulation better matches the real data than a simulation of a failure behaviour, so we are comparing potentially multiple different simulation behaviours, rather than trying to make a single simulation as accurate as possible.

Component libraries, containing components like valves, motors, pumps, etc., have been built to facilitate the construction of complex systems. As useful as these libraries are, at present they have several deficiencies, including (1) model fidelity that may not be suitable for the application at hand, and (2) the fact that the component model construction and parameter estimation are entirely manual. It is well known that the models provided within Modelica component libraries, e.g., (Casella and Leva, 2003; Casella et al., 2006), are typically too detailed for control and diagnostics applications (Casella et al., 2008; Imsland et al., 2010). The models designed for control and diagnostics applications are referred to as reduced-order (RO) models (Antoulas et al., 2001) or balanced-complexity models (Elgsæter et al., 2012; Kittilsen et al., 2012).

The two approaches typically adopted for generating application-specific models are (a) reducing the order of a detailed simulation model (Antoulas et al., 2001), or (b) building the model from scratch (Kim et al., 2014; Provan, 2011). Approach (a) is challenging since model-reduction software is highly domain-specific and still requires further development to achieve commercial applicability. Approach (b) is expensive, and does not make use of component libraries in which considerable time and effort have been directed.

We present an alternative approach for generating application-specific models that does not require the use of model-reduction software, but which can make use of model libraries, albeit a more generalized view of model libraries. We propose an approach that can use a model library containing component models of multiple levels of fidelity to generate a system-level model that optimizes a particular model metric. This approach has two key novel aspects. First, we define a metric for evaluating the fitness of a model for particular applications. Second, we describe a software tool that can apply the metric for generating application-specific models. We demonstrate our approach in the area of diagnostics for process-control systems.

In this article, we address the issue of matching component model fidelity to the application at hand. Most existing domain-specific libraries, e.g., the building energy library (Wetter et al., 2014), are built in a hierarchical fashion on top of lower-level libraries, in this case the Modelica hydraulic library (Casella et al., 2006; Tummescheit and Eborn, 1998). Since the hydraulic library is highly detailed, the building library inherits this level of fidelity. For applications like control and diagnostics, this level of detail may be unnecessary, and may limit the applicability of the generated code due to slow inference speeds.

We have been developing libraries with components that are described in multiple levels of fidelity. Variable-fidelity models have been used for many years for the systems optimization process in engineering design (Schmit and Farshi, 1974). However, typically each model is system-level, rather than component-based.

In this article, we describe an approach for developing a component library $\mathscr{L}$ containing components described at multiple fidelity levels, e.g., as a non-linear system (high-fidelity model), linear system, or a qualitative system (low-fidelity model). Choosing the right component model for system simulation is a difficult task and requires a search in the space of all possible component-fidelity combinations.

We propose an automated method for computing a system model that optimizes a set of metrics (e.g., model-simulation accuracy or diagnostics accuracy) given a set of simulation scenarios. We describe initial experimental results showing the feasibility of this approach for diagnosis applications, using a tank benchmark to illustrate our approach.

## 2 Systems Model Specification

This section formalises the notion of model for two applications, simulation and diagnostics inference. As mentioned earlier, most uses of Modelica models are purely for simulation, but we will generalise that notion by defining a model that generates behaviours based on multiple modes, e.g., modes corresponding to normal behaviour, and modes corresponding to behaviour where a fault is present. For example, a valve can block flow when shut, but if it cannot close fully then it will not fully block flow when commanded shut. We first introduce the simulation and diagnosis tasks, and then define them more precisely.

### 2.1 Simulation and Diagnosis Tasks

We define a simulation task as follows.

**Definition 1 (Simulation)** *Given a model $\phi$ and initial conditions $x_0$ at $t_0$, generate the behaviour of the system over an interval $[t_0, \tau]$.*

Standard Modelica simulation models have been extended for diagnostics purposes by many authors, e.g., (Åkerlund, 2001; de Kleer et al., 2013; Bäck, 2008). The extensions focus on enabling the model to generate multiple possible behaviours (corresponding to nominal and faulty component conditions) rather than the single behaviour for a simulation model.

We formalise a diagnosis model as follows. Assume that we have a system $\mathscr{S}$ that can operate in a nominal state, $\xi_N$, or a faulty state, $\xi_F$, where $\Xi$ is the set of possible states of $\mathscr{S}$. We further assume that we have a discrete vector of measurements, $\tilde{\boldsymbol{Y}} = \{\tilde{y}_1, ..., \tilde{y}_n\}$ observed at times $t = \{1, ..., n\}$ that summarizes the response of the system $\mathscr{S}$ to control variables $\boldsymbol{U} = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_n\}$. Let $\boldsymbol{Y}_\phi = \{y_1, ..., y_n\}$ denote the corresponding predictions from a dynamic (non-linear) model, $\phi$, with parameter values $\boldsymbol{\theta}$: this can be represented by $\boldsymbol{Y}_\phi = \phi(x_0, \boldsymbol{\theta}, \xi, \tilde{\boldsymbol{U}})$, where $x_0$ signifies the initial states of the system at $t_0$.

Diagnostics takes as input anomalous data that disagrees with the system being in a nominal state, and isolates the fault causing the anomalous sensor readings. A range of definitions of diagnosis are possible, including the most likely diagnosis, all diagnoses, etc. We assume a fixed diagnosis task $\mathscr{T}$ throughout this article, e.g., computing the most likely diagnosis, or a deterministic multiple-fault diagnosis. Diagnosis is significantly more complex than simulation, as we now show.

We specify a diagnosis model as follows:

**Definition 2 (Diagnosis Model)** *We characterise a Diagnosis Model $\phi$ using the tuple $\langle \boldsymbol{V}, \boldsymbol{\theta}, \Xi, \mathscr{E} \rangle$, where*

- ***V*** *is a set of variables, consisting of variables denoting the system state (**X**), control (**U**), and observations (**Y**).*
- ***θ*** *is a set of parameters.*
- *Ξ is a set of system modes.*
- *$\mathscr{E}$ is a set of equations, with a subset $E_\xi \subseteq \mathscr{E}$ for each mode $\xi \in \Xi$.*

We assume that we can use a physics-based approach to hand-generate a set $\mathscr{E}$ of equations to specify a model. Obtaining good diagnostics accuracy, given a fixed $\mathscr{E}$, entails estimating the parameters $\boldsymbol{\theta}$ to optimise that accuracy.

The classical definition of diagnosis is as a state estimation task, whose objective is to identify the system state that minimises a residual vector. We refer to this as *accuracy-based diagnosis.*

**Definition 3 (Accuracy-Based Diagnosis)**

$$\xi^* = \underset{\xi \in \Xi}{\operatorname{argmin}} \mathscr{R}(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi), \tag{1}$$

where a residual vector $\mathscr{R}(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi)$ measures the difference between the actual and model-simulated system behaviour. An example of a residual vector is the mean-squared-error (MSE).

Diagnostics inference (as well as control-based inference) require not just accuracy but also computational efficiency, especially in real-time applications. As a consequence, we need to generalise the accuracy-based diagnosis notion to incorporate inference efficiency as well as accuracy, using an inference complexity measure as $\mathscr{C}(\tilde{\boldsymbol{Y}}, \phi)$.

**Definition 4 (Diagnosis Optimisation)** *A diagnosis task jointly minimises a function g that incorporates the accuracy (based on the residual function) and the inference complexity:*

$$\xi^* = \underset{\xi \in \Xi}{\operatorname{argmin}}\, g\left(\mathscr{R}(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi), \mathscr{C}(\tilde{\boldsymbol{Y}}, \phi)\right), \tag{2}$$

*where g specifies a loss or penalty function that induces a non-negative real-valued penalty based on the lack of accuracy and computational cost.*

Since this is a minimisation task, we typically need to run multiple simulations over the space of parameters and modes to compute $\xi^*$. We can abstract this process as performing model-inversion, i.e., computing some $\xi^* = \phi^{-1}(x_0, \boldsymbol{\theta}, \xi, \tilde{\boldsymbol{U}})$ that minimises $\mathscr{R}(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi)$.

During this diagnostics inference task, a model $\phi$ can play two roles: (a) simulating a behaviour to estimate $\mathscr{R}(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi)$; (b) enabling the computation of $\xi^* = \phi^{-1}(x_0, \boldsymbol{\theta}, \xi, \tilde{\boldsymbol{U}})$. It is clear that diagnostics inference requires a model that has good fidelity and is computationally efficient for performing these two roles.

In forward simulation, a model $\phi$, with parameters $\boldsymbol{\theta}$, can generate multiple observations $\tilde{\boldsymbol{Y}} = \{\tilde{y}_1, ..., \tilde{y}_n\}$. The diagnostics task involves performing the inverse operation on these observations. Our objective thus involves optimising the state estimation task over a future set of observations, $\tilde{\boldsymbol{Y}} = \{\tilde{\boldsymbol{Y}}_1, ..., \tilde{\boldsymbol{Y}}_n\}$. Our model $\phi$ and inference algorithm $\mathscr{A}$ have different performance based on $\tilde{\boldsymbol{Y}}_i, i = 1, ..., n$: for example, (Feldman et al., 2008) shows that both inference-accuracy and -time vary based on the fault cardinality . As a consequence, to compute $\xi^*$ we want to optimise the *mean* performance over future observations. This notion of *mean* performance optimisation has been characterised using the Bayesian model selection approach, which we examine in section 3.3.

## 2.2 Running Example: Three-Tank Benchmark

We now describe our running example, which is the well-known tank benchmark. We will first describe a tank component, and then connect three tanks in series to form a tank system.

### 2.2.1 Tank Component

A component consists of a set of interfaces (inputs and outputs), a set of functions to map inputs to outputs, and a set of parameters. We use a tank to illustrate this. A tank is a component that has an inflow of fluid (qIn), and a regulated outflow (qOut) through a hole at the bottom of the tank, which is controlled by a valve. The height $h$ of liquid in the tank is governed by $h \propto (qIn - qOut)$.

A tank component has four interfaces (connectors in Modelica): *qIn* for input flow, *qOut* for output flow, *pressure* for providing pressure measurements, and *valve-control* for setting the position of the valve at the outlet of the tank.

Within Modelica, we assume that we will create connector classes as follows. First, we have a class *val* for reading the pressure level (Figure 1):

```
connector Measure "Measuring pressure"
            Real val(unit="m");
end Measure;
```

**Figure 1.** Modelica class for pressure level measurement

We create a class *ActSignal* for setting valve position, and a class *lflow* for the liquid flow at inlets and outlets.

The key equation regulating the behaviour of the tank is the mass balance equation, where the change in height of fluid is given by the difference in input and output flows (assuming constant pressure). Figure 2 shows the Modelica code for this component.

```
model Tank
        parameter Real A=1 "Ground area of tank in m^2";
        parameter Real a=0.2 "Area of drain hole in m^2";
        constant Real g=Modelica.Constants.g_n;
        Real h(start=0.0,unit="m") "Tank level";
        Real pressure(start=0.0,unit="Nm") "Pressure";
        Interfaces.Measure pressure "Connector, sensor
                reading pressure (Nm)";
        Interfaces.LiquidFlow qIn "Connector, flow (m3/s)
                through input valve";
        Interfaces.LiquidFlow qOut "Connector, flow (m3/s)
                through output valve";
        Interfaces.ActSignal valve-control "Connector,
                actuator controlling output flow";
equation
        der(h)=(qIn.lflow - qOut.lflow)/A;
        qOut.lflow = valve-control.ActSignal *
                sqrt(max(0,2*g*h))*a;
        pressure.val=pressure;
end Tank;
```

**Figure 2.** Modelica code for Tank component

Given this tank component, we can then generate a system with multiple tanks in series simply by connecting the output flow connector of one tank to the input flow connector of the succeeding tank.

### 2.2.2 Three-Tank System

In this paper, we use the three-tank system shown in Fig. 3 to illustrate our approach. The three tanks are denoted as $T_1$, $T_2$, and $T_3$. Each tank has the same area $A_1 = A_2 = A_3$. For $i = 1,2,3$, tank $T_i$ has height $h_i$, a pressure sensor $p_i$, and a valve $V_i$, $i = 1,2,3$ that controls the flow of liquid out of $T_i$. We assume that gravity $g = 10$ and the liquid has density $\rho = 1$.

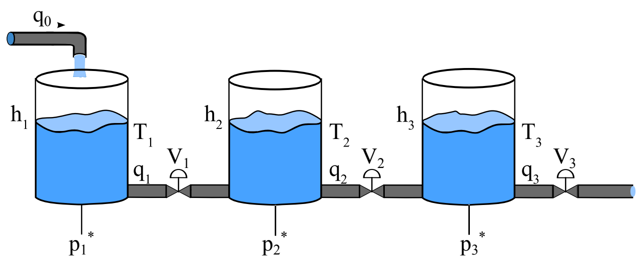

**Figure 3.** Diagram of the three-tank system.

Tank $T_1$ gets filled from a pipe, with measured flow $q_0$. Using Torricelli's law, the model can be described by the following non-linear equations:

$$\frac{dh_1}{dt} = \frac{1}{A_1}\left[-\kappa_1\sqrt{h_1-h_2}+q_0\right], \quad (3)$$

$$\frac{dh_2}{dt} = \frac{1}{A_2}\left[\kappa_1\sqrt{h_1-h_2}-\kappa_2\sqrt{h_2-h_3}\right], \quad (4)$$

$$\frac{dh_3}{dt} = \frac{1}{A_3}\left[\kappa_2\sqrt{h_2-h_3}-\kappa_3\sqrt{h_3}\right]. \quad (5)$$

In eq. 3, the coefficient $\kappa_1$ denotes a parameter that captures the product of the cross-sectional area of the tank $A_1$, the area of the drainage hole, a gravity-based constant ($\sqrt{2g}$), and the friction/contraction factor of the hole. $\kappa_2$ and $\kappa_3$ can be defined analogously.

Finally, the pressure at the bottom of each tank is obtained from the height: $p_i = g\,h_i$, where $i$ is the tank index ($i \in \{1,2,3\}$).

We emphasize the use of the $\kappa_i$, $i = 1,2,3$ because we will use these parameter-values as a means for "diagnosing" our system in term of changes in $\kappa_i$, $i = 1,2,3$. Consider a physical valve $R_1$ between $T_1$ and $T_2$ that constraints the flow between the two tanks. We can say that the valve changes proportionally the cross-sectional drainage area of $q_1$ and hence $\kappa_1$. The diagnostic task will be to compute the true value of $\kappa_1$, given $p_1$, and from $\kappa_1$ we can compute the actual position of the valve $R_1$.

We now characterise our nominal model in terms of Definition 2:

- variables $\boldsymbol{V}$ consist of variables denoting the system state ($\boldsymbol{X} = \{h_1,h_2,h_3\}$), control ($\boldsymbol{U} = \{q_0,V_1,V_2,V_3\}$), and observations ($\boldsymbol{Y} = \{p_1,p_2,p_3\}$).
- $\boldsymbol{\theta} = \{\{A_1,\ A_2,\ A_3\},\{\kappa_1,\ \kappa_2,\ \kappa_3\}\}$ is the set of parameters.
- $\Xi$ consists of a single nominal mode.
- $\mathscr{E}$ is a set of equations, given by equations 3 through 5.

Note that this model has a total of 6 parameters.

**Fault Model** In this article we focus on valve faults, where a valve can have a blockage or a leak. We model this class of faults by including in equations 3 to 5 an additive parameter $\beta$, which is applied to the parameter $\kappa$, i.e., as $\kappa_i(1+\beta_i)$, $i = 1,2,3$, where $-1 \leq \beta_i \leq \frac{1}{\kappa_i} - 1$, $i = 1,2,3$. $\beta > 0$ corresponds to a leak, such that $\beta \in (0, 1/\kappa - 1]$; $\beta < 0$ corresponds to a blockage, such that $\beta \in [-1, 0)$. The fault equations allow faults for any combination of the valves $\{V_1,V_2,V_3\}$, resulting in system modes $\Xi = \{\xi_N,\xi_1,\xi_2,\xi_3,\xi_{12},\xi_{13},\xi_{23},\xi_{123}\}$, where $\xi_N$ is the nominal mode, and $\xi_\cdot$ is the mode where $\cdot$ denotes the combination of valves (taken from a combination of $\{1,2,3\}$) which are faulty. This fault model has 9 parameters.

## 3 Modelling Metrics

This section describes the metrics that can be applied to estimate properties of a diagnosis model. We describe two types of metrics, dealing with accuracy (fidelity) and complexity.

Given a space $\Phi$ of possible models, we can define our model selection task as follows:

$$\phi^* = \underset{\phi\in\Phi}{\operatorname{argmin}}\, g_1\left(\tilde{\boldsymbol{Y}},\boldsymbol{Y}_\phi\right) + g_2\left(\mathscr{C}(\tilde{\boldsymbol{Y}},\phi)\right), \quad (6)$$

adopting the simplifying assumption that our loss function $g$ is additively decomposable. We now briefly examine the accuracy ($g_1$) and complexity ($g_2$) metrics.

## 3.1 Model Accuracy

Model accuracy concerns the ability of a model to mimic a real system. Since model fidelity for a multi-mode approach varies depending on the mode, it is important to explicitly consider the *mean performance* of $\phi$ over the entire observation space $\mathscr{Y}$ (the space of possible observations of the system). In this article simulation accuracy is defined using an expected residual:

**Definition 5 (Expected Residual)** *given a space $\mathscr{Y} = \{\tilde{\boldsymbol{Y}}_1, ..., \tilde{\boldsymbol{Y}}_n\}$ of observations, the expected residual is the average over the n observations, e.g., as given by: $\bar{\mathscr{R}} = \frac{1}{n}\sum_{i=1}^{n} \mathscr{R}(\tilde{\boldsymbol{Y}}_i, \boldsymbol{Y}_\phi)$.*

Diagnostics accuracy is different, in that it involves computing the correct diagnosis without delays in fault isolation from the pair $(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi)$. If we denote the tuple $(\omega, t)$ as the correct diagnosis occurring at time $t$, and $(\hat{\omega}, \hat{t})$ as the computed diagnosis identified at time $\hat{t}$, then our $g_1$ function must capture some difference measure for accuracy, e.g., $\|\hat{\omega} - \omega\|$, and some difference measure for isolation delays, e.g., $\|\hat{t} - t\|$. We can represent this abstractly as:

**Definition 6 (Diagnosis Accuracy)** *Given measures for isolation accuracy ($g'$) and isolation delay ($g''$), diagnosis accuracy can be defined as*

$$g_1(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}_\phi) = g'(\|\hat{\omega} - \omega\|) + g''(\|\hat{t} - t\|).$$

## 3.2 Model Complexity

At present, there is no commonly-accepted definition of model complexity, whether the model is used purely for simulation or if it is used for diagnostics or control. Defining the complexity of a model is inherently tricky, due to the number of factors involved.

Less complex models are often preferred either due to their low computational simulation costs (Keating et al., 2010), or to minimise model over-fitting given observed data (Pande et al., 2009; Schoups et al., 2008). Given the task of simulating a variable of interest conditioned by certain future values of input (control) variables, overfitting can lead to high uncertainty in creating accurate simulations. Overfitting is especially severe when we have limited observation variables for generating a model representing the underlying process dynamics. In contrast, models with low parameter dimensionality (i.e. fewer parameters) are considered less complex and hence are associated with low prediction uncertainty (Pande et al., 2014).

Several approaches have been used, based on issues like (a) number of variables (Kunz et al., 2006), (b) model structure (Provan and Wang, 2007), (c) number of free parameters (Pande et al., 2014), (d) number of parameters that the data can constrain (Spiegelhalter et al., 2002), (e) a notion of model weight (Du, 2014), or (f) *type* and *order* of equations for a non-linear dynamical model (Antoulas et al., 2001), where type corresponds to non-linear, linear, etc.; e.g., order for a non-linear model is such that a $k$-th order system has $k$-th derivates in $\mathscr{E}$.

Factors that contribute to the true cost of a model include: (a) model-generation; (b) parameter estimation; and (c) simulation complexity, i.e., the computational expense (in terms of CPU-time and memory) needed to simulate the model given a set of initial conditions Rather than try to formulate this notion in terms of the number of model variables or parameters, or a notion of model structural complexity, we specify model complexity in terms of a measure based on parameter estimation, and inference complexity, assuming a construction cost of zero.

A thorough analysis of model complexity will need to take into consideration the model equation class, since model complexity is class-specific. For example, for non-linear dynamical models, complexity is governed by the *type* and *order* of equations (Antoulas et al., 2001). In contrast, for linear dynamical models, which have only matrices and variables in equations (no derivatives), it is the order of the matrices that determines complexity. In this article, we assume that models are of appropriate complexity, and hence do not address Model order reduction techniques (Antoulas et al., 2001), which aim to generate lower-dimensional systems that trade off fidelity for reduced model complexity.

## 3.3 Bayesian Model Complexity

The Bayesian model complexity approach, which is based on the model likelihood, measures whether the increased "complexity" of a model with more parameters is justified by the data. The Bayesian approach chooses a model (and a model structure) from a set of competing models (from the set of corresponding model structures, respectively) such that the value of a Bayesian criterion is maximized (or prediction uncertainty in choosing a model structure is minimized).

The Bayesian approach addresses prediction uncertainty by specifying an appropriate likelihood function. In other words, it specifies the probability with which the observed values of a variable of interest are generated by a model. The marginal likelihood of a model structure, which represents a class of models capturing the same processes (and hence have the same parameter dimensionality), is obtained by integrating over the prior distribution of model parameters; this measures the prediction uncertainty of the model structure (Vrugt and Robinson, 2007).

The idea is that by adding parameters to a model we obtain improvement in fit, but at the expense of making parameter estimates "worse"' because we have less data (i.e., information) per parameter. In addition, the computations typically require more time. So the key question is how to identify how complex a model works best for a given problem.

In the statistics literature several decomposable approximations have been proposed. We now review the best-known methods. (Spiegelhalter et al., 2002) have proposed a well-known such decomposable framework, termed the Deviance Information Criterion (DIC), which measures the number of model parameters that the data can constrain.: $DIC = \overline{D} + p_D$, where $\overline{D}$ is a measure of fit (expected deviance), and $p_D$ is a complexity measure, the *effective* number of parameters. The Akaike Information Criterion (AIC) (Akaike, 1974, 1981) is another well-known measure: $AIC = \mathscr{L}(\hat{\boldsymbol{\theta}}) + 2k$, where $\hat{\boldsymbol{\theta}}$ is the Maximum Likelihood Estimate (MLE) of $\boldsymbol{\theta}$ and $k$ is the number of parameters.

Another computationally more tractable approach is the Bayesian Information Criterion (BIC) (Schwarz, 1978): $BIC = -2\mathscr{L}(\hat{\boldsymbol{\theta}}) + klogn$, where $k$ is the number of *estimable* parameters, and $n$ is the sample size (number of observations). BIC was developed as an approximation to the log marginal likelihood of a model, and therefore, the difference between two BIC estimates may be a good approximation to the natural log of the Bayes factor. Given equal priors for all competing models, choosing the model with the smallest BIC is equivalent to selecting the model with the maximum posterior probability. BIC assumes that the (parameters') prior is the unit information prior (i.e., a multivariate normal prior with mean at the maximum likelihood estimate and variance equal to the expected information matrix for one observation).

(Wagenmakers, 2007) shows that one can convert the BIC measure to $BIC = n\ log \frac{SSE}{SS_{total}} + k\ logn$, where SSE is the sum of squares for the error term. In our experiments, we assume that the non-linear model is the "correct" model (or the null hypothesis $H_0$), and either the linear or qualitative models are the competing model (or alternative hypothesis $H_1$). Hence what we do is use BIC to compare the non-linear to each of the competing models.

## 4 Experimental Design

This section compares three tank benchmark models according to various model-selection measures. We adopt as our "correct" model the non-linear model. We will examine the fidelity and complexity trade-offs of two simpler models over a selection of failure scenarios.

The diagnostic task will be to compute the fault state of the system, given an injected fault. This translates to different tasks given the different models.

**non-linear model** estimate the true value of $\kappa_1$ given $p_1$, which corresponds to a most-likely failure mode assignment of one of $(\xi_N, \xi_B, \xi_P)$.

**linear model** estimate the true value of $\kappa_1$ given $p_1$, which corresponds to a most-likely failure mode assignment of one of $(\xi_N, \xi_B, \xi_P)$.

**qualitative model** estimate the failure mode assignment of one of $(\xi_N, \xi_B, \xi_P)$.

### 4.1 Alternative Models

This section describes the two alternative models that we compare to the non-linear model, a linear and a qualitative model.

#### 4.1.1 Linear Model

We compare the non-linear model with a linearised version. We can perform this linearised process in a variety of ways (Spanos, 1977). In this simple tank example, we can perform the linearisation directly through replacement of non-linear and linear operators, as shown below.

Nominal Model We can linearise the the non-linear 3-tank model by replacing the non-linear sub-function $\sqrt{h_i - h_j}$ with the linear sub-function $\gamma_{ij}(h_i - h_j)$, where $\gamma_{ij}$ is a parameter (to be estimated) governing the flow between tanks $i$ and $j$. The linear model has 4 parameters, $\gamma_{12}$, $\gamma_{12}$, $\gamma_{23}$, $\gamma_3$.

Fault Model The fault model introduces a parameter $\beta_i$ associated with $\kappa_i$, i.e., we replace $\kappa_i$ with $\kappa_i(1+\beta_i)$, $i = 1,2,3$, where $-1 \leq \beta_i \leq \frac{1}{\kappa_i} - 1$, $i = 1,2,3$. This model has 7 parameters, adding parameters $\beta_1$, $\beta_2$, $\beta_3$.

#### 4.1.2 Qualitative Model

Nominal Model For the model we replace the non-linear sub-function $\sqrt{h_i - h_j}$ with the qualitative sub-function $M^+(h_i - h_j)$, where $M^+$ is the set of reasonable functions $f$ such that $f' > 0$ on the interior of its domain (Kuipers and Åström, 1994).

The tank-heights are constrained to be non-negative, as are the parameters $\kappa_i$. As a consequence, we can discretize the $h_i$ to take on values $\{+, 0\}$, which means that $M^+(h_i - h_j)$ can take on values $\{+, 0, -\}$. The domain for $\frac{dh_1}{dt}$ must be $\{+, 0, -\}$, since

POSITION first generates a model where $T_1$ is non-linear, $T_2$ is non-linear, and $T_3$ is non-linear. In the second call NEXTMODELCOMPOSITION changes $T_1$ from non-linear to linear. Depending on the implementation of NEXTMODELCOMPOSITION the next candidate can be either $T_1$ and $T_2$ non-linear while $T_3$ linear or $T_1$ non-linear, $T_2$ linear, and $T_3$ non-linear.

The greedy search approach guarantees that Algorithm 1 will terminate with $O(n)$ model evaluations, but at the cost of optimality. Optimality is also limited by the "quality" of the component models and the set of test cases used for testing the metrics.

## 6 Experimental Results

This section describes our experimental results, summarising the data first and then discussing the implications of the results.

### 6.1 Model Comparisons

We have empirically compared the diagnostics performance of several multi-tank models. In our first set of experiments, we ran a simulation over 500 seconds, and induced a fault (valve $V_1$ at 50%) after 250 s. The model combinations involved a non-linear (NL) model, a model (denoted M) with tank $T_1$ being linear (and other tanks non-linear), a fully linear model (denoted L), and a Qualitative model (denoted Q).

To compute a measure of diagnostics error (or loss), we use the difference between the true fault (which is known for each simulation) and the computed fault. We denote the true fault existing at time $t$ using the pair $(\omega,t)$; the computed fault at time $t$ is denoted using the pair $(\hat{\omega},\hat{t})$. The inference system that we use, LNG, computes an uncertainty measure associated with each computed fault, denoted $P(\hat{\omega})$. Hence, we define a measure of diagnostics error over a time window $[0,T]$ using

$$\gamma_1^D = \sum_{t=0}^{T} \sum_{\xi \in \Xi} |P(\hat{\omega}) - \omega|, \tag{7}$$

where $\Xi$ is the set of failure modes for the model.

Our second metric covers the fault latency, i.e., how quickly the model identifies the true fault $(\omega,t)$:

$$\gamma_3 = t - \hat{t}. \tag{8}$$

Table 1 summarises our results. The first columns compare the number of parameters for the different models, followed by comparisons of the error ($\gamma_1$) and the CPU-time ($\gamma_3$). The data show that the error($\gamma_1$) does not grow very much as we increase model size, but it increases as we decrease model fidelity from non-linear through to qualitative models. In contrast, the CPU-time (a) increases as we increase model size, and (b) is proportional to model fidelity, i.e., it and decreases as we decrease model fidelity from non-linear through to qualitative models.

| Tanks | | 2 | 3 | 4 |
|---|---|---|---|---|
| # Parameters | NL | 7 | 9 | 11 |
| | M | 6 | 8 | 10 |
| | L | 5 | 7 | 9 |
| | Q | 2 | 3 | 4 |
| $\gamma_1$ | NL | 242 | 242 | 242 |
| | M | 997 | 1076 | 1192 |
| | L | 1236 | 1288 | 1342 |
| | Q | 3859 | 3994 | 4261 |
| $\gamma_3$ | NL | 10.59 | 23.7 | 39.5 |
| | M | 8.52 | 17.96 | 34.6 |
| | L | 6.11 | 10.57 | 32.0 |
| | Q | 4.64 | 7.31 | 26.4 |

**Table 1.** Data for 2-, 3-, and 4-tank models using Non-linear (NL), Mixed (M), Linear (L) and Qualitative (Q) representations

In a second set of experiments, we focused on multiple model types for a 3-tank system, with simulations running over 50s. The model combinations involved a non-linear (NL) model, a model with tank 3 linear (and other tanks non-linear), a model with tanks 2 and 3 linear and tank 1 non-linear, a fully linear model, and a qualitative model. Table 2 summarises our results.

The data show that, as model fidelity decreases, the error $\gamma_1$ increases significantly and the inference times $\gamma_3$ decrease modestly. If we examine the outputs from $AIC_c$, we see that the best model is the mixed model ($T_3$-linear). BIC indicates the qualitative model as the best; it is worth noting that BIC typically will choose the simplest model.

| | $\gamma_1$ | $\gamma_3$ | $AIC_c$ | BIC |
|---|---|---|---|---|
| Non-Linear | 0.97 | 23.7 | 29.45 | 43.7 |
| $T_3$-linear | 3.12 | 17.96 | 26.77 | 42.9 |
| $T_2,T_3$-linear | 21.96 | 13.21 | 31.12 | 39.56 |
| Linear | 77.43 | 10.57 | 35.76 | 37.55 |
| Qualitative | 304.41 | 9.74 | 43.01 | 29.13 |

**Table 2.** Data for 3-tank model, using Non-linear, Mixed, Linear and Qualitative representations, given a fault (valve $V_1$ at 50%) after 25 s

### 6.2 Discussion

Our results show that MBD is a complex task with several conflicting factors.

- The error $\gamma_1$ is inversely proportional to model fidelity, given a fixed diagnosis task.
- The error $\gamma_1$ increases with fault cardinality.
- The CPU-time $\gamma_3$ increases with model size (i.e., number of tanks).

We have shown how one can extend a set of information-theoretic metrics to combine these competing factors in diagnostics model selection. Further work is necessary to identify how best to extend the existing information-theoretic metrics to suit the needs of diagnostics inference.

It is important to note that we conducted experiments with un-calibrated models, and we have ignored the cost of calibration in this article. The literature suggests that linear models can be calibrated to achieve good performance, although performance inferior to that of calibrated non-linear models. This class of qualitative models does not possess calibration factors, so calibration will not improve their performance.

This article has introduced a framework that can be used to trade off the different factors in MBD. It is likely that the "best" model may be domain- and task-specific; further work is needed to answer this notion.

## 7 Conclusions

This article has presented a framework for generating Modelica models for a range of applications, such as simulation, control and diagnosis. Our approach can tailor the fidelity of the model to the application, using an application-specific metric for judging model "quality". The metric aims to balance competing properties of models, e.g., fidelity and computational complexity.

We demonstrated our approach on the domain of model-based diagnosis (MBD). We have proposed some Bayesian metrics for MBD model evaluation, and conducted some preliminary experiments to show how these metrics may be applied. This work thus constitutes *a start* to a full analysis of model performance. Our intention is to initiate a more formal analysis of modeling and model evaluation, since there is no framework in existence for this task. Further, the experiments are only preliminary, and are meant to demonstrate how a framework can be applied to model comparison and evaluation.

Significant work remains to be done, on a range of fronts. In particular, a thorough empirical investigation is needed on diagnostics modeling. Second, the real-world utility of our proposed framework needs to be determined. Third, a theoretical study of the issues of mode-based parameter estimation and its use for MBD is necessary.

## References

Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

Hirotugu Akaike. Likelihood of a model and information criteria. *Journal of econometrics*, 16(1):3–14, 1981.

Per Åkerlund. *Model-based diagnosis using MathModelica.* PhD thesis, MasterâĂŹs thesis, Linköping University, SE-581 83 Linköping, 2001.

AC Antoulas, DC Sorensen, and S Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary mathematics*, 280:193–220, 2001.

Olof Bäck. *Modelling for diagnosis in Modelica: implementation and analysis.* PhD thesis, MasterâĂŹs thesis, Linköping University, SE-581 83 Linköping, 2008.

H Brückmann, J Strenkert, U Keller, B Wiesner, and A Junghanns. Model-based development of a dual-clutch transmission using rapid prototyping and sil. In *International VDI Congress Transmissions in Vehicles*, 2009.

Francesco Casella and Alberto Leva. Modelica open library for power plant simulation: design and experimental validation. In *Proceedings of the 3rd international Modelica conference*, 2003.

Francesco Casella, Martin Otter, Katrin Proelss, Christoph Richter, and Hubertus Tummescheit. The modelica fluid and media library for modeling of incompressible and compressible thermo-fluid pipe networks. In *Proceedings of the Modelica Conference*, pages 631–640, 2006.

Francesco Casella, Filippo Donida, and Marco Lovera. Beyond simulation: Computer aided control system design using equation-based object oriented modelling for the next decade. In *2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, page 35, 2008.

Johan de Kleer, Bill Janssen, Daniel G Bobrow, Tolga Kurtoglu, Bhaskar Saha, Nicholas R Moore, and Saravan Sutharshana. Fault augmented modelica models. In *The 24th International Workshop on Principles of Diagnosis*, pages 71–78, 2013.

Jing Du. The "weight" of models and complexity. *Complexity*, 2014.

S Elgsæter, Pal Kittilsen, and Svein Olav Hauger. Designing large-scale balanced-complexity models for online use. In *Proc. IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, Trondheim, Norway*, pages 157–162, 2012.

Alexander Feldman, Gregory M Provan, and Arjan JC van Gemund. Computing observation vectors for max-fault min-cardinality diagnoses. In *AAAI*, pages 919–924, 2008.

Alexander Feldman, Helena Vicente de Castro, Arjan van Gemund, and Gregory Provan. Model-based diagnostic decision-support system for satellites. In *Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, USA*, pages 1–14, March 2013.

Lars Imsland, Pal Kittilsen, and Tor S Schei. Model-based optimizing control and estimation using modelica model. *Modeling, Identification and Control*, 31(3): 107–121, 2010.

Elizabeth H Keating, John Doherty, Jasper A Vrugt, and Qinjun Kang. Optimization and uncertainty assessment of strongly nonlinear groundwater models with high parameter dimensionality. *Water Resources Research*, 46(10), 2010.

Eui-Jong Kim, Gilles Plessis, Jean-Luc Hubert, and Jean-Jacques Roux. Urban energy simulation: Simplification and reduction of building envelope models. *Energy and Buildings*, 84:193–202, 2014.

Pål Kittilsen, Svein Olav Hauger, and Stein O Wasbø. Designing models for online use with modelica and fmi. In *Proceedings of the 9th International Modelica Conference. Munich, Germany*, pages 197–204, 2012.

Benjamin Kuipers and Karl Åström. The composition and validation of heterogeneous control laws. *Automatica*, 30(2):233–249, 1994.

Martin Kunz, Roberto Trotta, and David R Parkinson. Measuring the effective complexity of cosmological models. *Physical Review D*, 74(2):023503, 2006.

Marek Mateják, Tomáš Kulhánek, Jan Šilar, Pavol Privitzer, Filip Ježek, and Jiří Kofránek. Physiolibrary-modelica library for physiology. In *10th International Modelica Conference*, pages 499–505. Linköping University Electronic Press Lund, Sweden, 2014.

S Pande, L Arkesteijn, HHG Savenije, and LA Bastidas. Hydrological model parameter dimensionality is a weak measure of prediction uncertainty. *Natural Hazards and Earth System Sciences Discusions, 11, 2014*, 2014.

Saket Pande, Mac McKee, and Luis A Bastidas. Complexity-based robust hydrologic prediction. *Water resources research*, 45(10), 2009.

Gilles Plessis, Aurélie Kaemmerlen, and Amy Lindsay. Buildsyspro: a modelica library for modelling buildings and energy systems. In *Proceedings of the International Modelica Conference*, 2014.

Gregory Provan. Generating reduced-order diagnosis models for hvac systems. In *Itnl. Workshop on Principles of Diagnosis, Murnau, Germany*, 2011.

Gregory M Provan and Jun Wang. Automated benchmark model generators for model-based diagnostic inference. In *IJCAI*, pages 513–518, 2007.

Lucien A Schmit and B Farshi. Some approximation concepts for structural synthesis. *AIAA journal*, 12(5): 692–699, 1974.

G Schoups, NC Van de Giesen, and HHG Savenije. Model complexity control for hydrologic prediction. *Water Resources Research*, 44(12), 2008.

G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6:461Ű466, 1978.

Pol D Spanos. *Linearization techniques for non-linear dynamical systems.* PhD thesis, California Institute of Technology, 1977.

David J Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4): 583–639, 2002.

Hubertus Tummescheit and Jonas Eborn. Design of a thermo-hydraulic model library in modelica. In *12th European Simulation Multiconference*, 1998.

Jasper A Vrugt and Bruce A Robinson. Treatment of uncertainty using ensemble methods: Comparison of sequential data assimilation and bayesian model averaging. *Water Resources Research*, 43(1), 2007.

Eric-Jan Wagenmakers. A practical solution to the pervasive problems ofp values. *Psychonomic bulletin & review*, 14(5):779–804, 2007.

Michael Wetter, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. Modelica buildings library. *Journal of Building Performance Simulation*, 7(4):253–270, 2014.