Calibration of Static Models using Dymola

Hans Olsson, Jonas Eborn*, Sven Erik Mattsson, Hilding Elmqvist
Dynasim *Modelon
Ideon Science Park, S-223 70 Lund, Sweden
{hans.olsson, svenerik, hilding.elmqvist}@dynasim.se, jonas.eborn@modelon.se

Abstract

A typical purpose of a static calibration is to tune static characteristics of components such pipes, valve, throttles, pumps, nonlinear resistors, frictions etc. Dymola's GUI supports setting up such a calibration without building a corresponding test rig model. The GUI allows simple redefinition of a variable to be an input. The measured data for such an input and of course also for an original input can then be specified to have a common value for all cases or to have a case dependent value read from a file.

Assume that we want to find a static relation from the variable v to w of the model and that we have measurements for v and w and all inputs of the model. First we need to decide a parameterized shape or in other words we need to come up with a function w = f(p, v) where p is a parameter vector. In many cases we can use polynomials. In general it is nontrivial to come up with a good function that fits the data well. However, in this case it is possible to use the model and the measured data to back calculate w for each case. Dymola supports the setup of such a calculation is in a straightforward way very similar to the setup of the transient calibration. Having w makes the relation much more explicit and easier to visualize and inspect. Just by plotting w against v, we can get a good estimate of the chances to get a good result.

If the plot shows that the points seem to be lying on a line, the chance is much better than when the plot looks like a random scatter. Such a plot may also give us good insight in what kind of functional relation we should use. Classic pen and paper approaches as plotting in lin-log or log-log diagrams can be used to find out if exponential or potential relations could be useful.

Keywords: parameter estimation, static models, dynamic models, Modelica

1 Introduction

Physical modeling is an important tool for investigating models without the need for building them and performing experiments that may be expensive, dangerous, or delay projects.

Ideally these models should be built from first principles and easily measurable quantities. This is, however, not always the case and thus one needs to calibrate physical models to the reality.

In many cases the unknown relationship is seen as a static correlation, and the correlation is parameterized in certain ways (e.g. polynomial functions between dimensionless variables).

The goal is to determine these correlations from static measurements. The component model can then be used in the complete model, and the complete model can be validated against transient measurements.

2 Mathematical description

The actual parameter estimation uses the same numeric method as the transient calibration (a non-linear least squares method) and fits into this framework to allow analysis of the setup, e.g. to find non-identifiable subsets [2].

The mathematical background of parameter estimation (also known as data fitting [3]) is that we have a number of measured inputs, v_i, outputs w_i, and a static relation between them

$$w_i = f(p, v_i)$$

The goal is find the best set of parameters, p, and to minimize the residuals

$$r_i(p) = f(p, v_i) - w_i$$

To combine all of the residuals into one scalar to be minimized we will use the least squares formulation and minimize

$$\sum_i r_i^2(p)$$

Equivalently we can minimize the square root of this, i.e. the 2-norm of the residual.

There are two reasons for preferring to minimize the least squares problem compared to other formulations: numeric efficiency and statistical interpretation.

2.1 Statistical interpretation

The statistical interpretation of is that the parameter values minimizing the least squares residuals are the maximum likelihood parameter values, *assuming* that all errors are measurement errors in the outputs, and that these errors are normally distributed with zero mean and identical variance. Furthermore the function must be sufficiently linear [3].

These assumptions are in general not satisfied, but they can still provide some guidelines, e.g. that we should aim for outputs with "errors" of similar size.

Allowing measurement errors in both inputs and outputs lead to a more complex total least squares problem, which explains why it is not used even though it could be argued that it is more realistic in many scenarios.

An implicit assumption is that the correlation function is of the correct type. We will in the back-calculation chapter discuss how we verify that the function is of the correct type, alternatively select an appropriate type of function.

There are also well-known issues with having too many parameters, or extrapolating a correlation function far from the calibrated region. There are several statistical methods for avoid over-parameterization, including Akaike's criterion and cross validation (and other computer intensive statistical methods).

2.2 Model errors

The errors above (measurement errors and incorrect correlation) are in one sense easy since they will (even for the optimal parameters) generate non-zero residuals, and based on this one could estimate a confidence interval for the parameters.

Modeling errors are a bit different, since we can in some cases get zero residual for a combination of incorrect model and incorrect parameters. To avoid the problem the model should be validated to ensure that it is sufficiently accurate – both for the measurement and for the actual behavior we want to study. These are normally two different validations for static calibrations, since we want to use a statically determined correlation function also for predicting transient behavior.

Related to model errors are unidentifiable sub-sets of parameters. These are handled by fixing some parameters so that remaining ones can be accurately identified. (The fixed parameters can be given reasonable values, ignored by setting them to zero or infinity, or automatically handled by static calibration). Also in this case it is important to ensure that the actual behavior does not depend on the fixed parameters. By applying the tools for finding unidentifiable sub-sets and parameters sweeps [3] to the actual behavior one can automatically verify this, or otherwise complement the static calibration with transient calibration.

2.3 Numeric solution procedure

The numeric efficiency for the solution of the non-linear least squares problem is due to the Gauss-Newton method that linearizes the residual and as one step minimizes the linear least squares problem:

$$\sum_i \left(r_i(p_0) + \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0} (p-p_0) \right)^2$$

This is a standard problem that can be solved using QR-factorization, and by iterating this linear procedure we get the solution for the original non-linear problem. The solution is given by the normal equations:

$$\sum_i \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0} \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0}^T (p-p_0) = -\sum_i \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0} r_i(p_0)$$

In practice some parameter sub-sets might be non-identifiable, or the ignored non-linear part could cause the new parameter estimate to have larger residual. We guard against both of these problems by modifying the algorithm to use the Levenberg-Marquardt method [3]:

$$\left(\nu I + \sum_i \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0} \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0}^T \right)(p-p_0) = -\sum_i \left.\frac{\partial r_i}{\partial p}\right|_{p=p_0} r_i(p_0)$$

We use an automatic control of the iteration parameter, ν (which is non-negative). This problem can also be solved using the QR-factorization, and the modification gives robustness at the cost of slower convergence for problems where some parameters cannot be identified.

3 Modifying models

The previous work with parameter estimation from dynamic measurements parameter in [2] could be used. However, it is not an ideal solution since:

- Special test setup for the component model must be built, which increases the likelihood of errors.
- Either each measurement point requires one simulation (which will be slow), or a faked time is introduced leading to interpolation of the measurements (which causes slow-downs, and the model might be invalid in those regions).

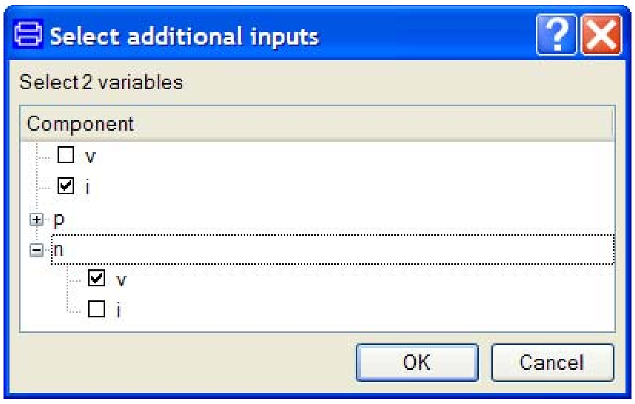
The static calibration framework handles both of these, the first by modifying the model and the second by running all parameter cases in a special way. These two functionalities can be used independently of each other, and thus static calibration can be performed without modifying the model (useful if a test bench already has been set up), or modify models for other experiments.

The static calibration function handles both the case when the model is completely state-less, such as valve characteristics, and more complex steady-state cases for which either dynamics are ignored or each case is simulated until steady-state is obtained. For completely static models special code is generated, combining the model equations and the sweep over calibration cases, which gives a very efficient solution method. The more complex steady-state cases require individual simulations of each point, which gives longer solution times.

A future work is instead of selecting the component model directly select the component, which would allow the calibration function to directly update the parameters for the component. A related possibility is to automatically construct a new class extending from the base and with the updated parameters as modifiers. This allows estimation of parameters in read-only models.

3.1 Simple Example

To give a concrete example we consider a circuit with an electric resistive component. Instead of building test-rig with a source we just select to calibrate the component model Resistor. Dymola does then not default connect the two top-level pins (removing two equations), but instead asks us to transform two variables into inputs as shown in Figure 1 (to ensure that we have the same number of equations and variable):



Select 2 variables

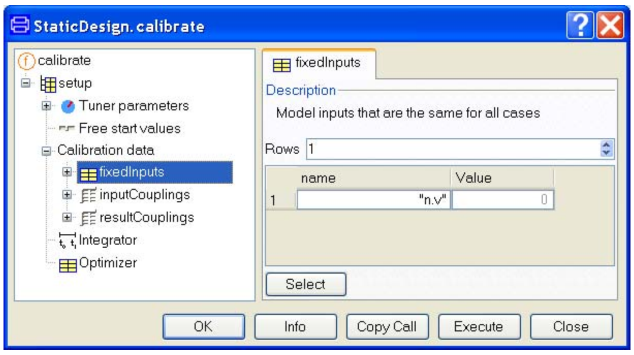
- ☐ v
- ☑ i
- p
- n
  - ☑ v
  - ☐ i

Figure 1. Selecting additional inputs, to be connected to measured data.

We can then set “n.v” to zero (arbitrary grounding) and can then use measurements for “i” and “v” to calibrate the resistance.

This corresponds to constructing a test-circuit with inputs for “i” and “n.v” and without the default connect, i.e. connecting a current source to the resistor and grounding the circuit.

The remainder of the setup is similar to the setup in [2], and the main differences are that one measurement file contains all of the cases and the possibility to provide fixed values for variables such as “n.v”, as shown in Figure 2 below.



Description: Model inputs that are the same for all cases

Rows: 1

	name	Value
1	"n.v"	0

Figure 2. Assigning a fixed input signal to one potential, corresponding to grounding the resistor.

3.2 Back-calculation

Since the goal is to determine the correlations one can start without any correlation. By running the static calibration cases one gets data so that one can plot the correlations and either try to find correlations or simply verify that the selected correlation is appropriate.

For the resistor this would mean having a generic resistive load without any correlation, in that case one has to give three additional inputs: “v”, “i” and “n.v”, and can then plot voltage against current to find that Ohm’s law is valid, or that the resistive load is non-linear.

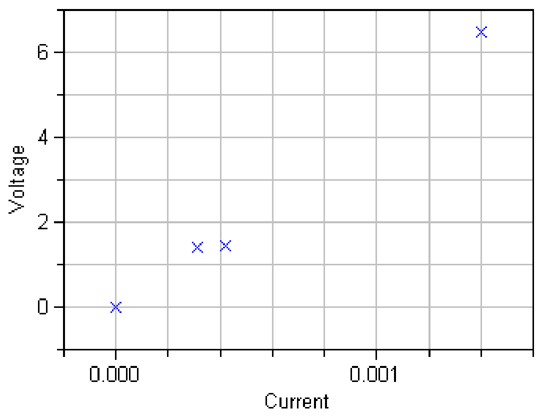


Figure 3. Simple validation of Ohm's law (errors likely due to inaccurate measurements).

The resistance can be calibrated by using the Resistor class and selecting "R" as the tuner parameter (similarly as in [2] and it is the only possible choice).

4 Application example

4.1 Calibrating steady-state compressor characteristics

One of the most important characteristics in a vapor compression cycle for refrigeration or air conditioning is the static characteristic of the compressor. This map usually describes flow rate directly as a function of compressor speed n and pressure ratio ($\pi = p_d/p_s$), or indirectly through efficiency functions, e.g. volumetric efficiency λ_{eff} and isentropic efficiency η_{is}, [4, 5]. As an example volumetric efficiency for an Obrist swash-plate compressor using CO_2 as refrigerant has been fitted to the functional form used in the AirConditioning library, the data is taken from [5] and is also available in the library. The function used for calibration is given below, for more details see 7.

$$\lambda_{eff} = \left(\pi_0 - \frac{p_d/p_s}{\pi_0 - 1}\right)^2 \left(\frac{x - x_0}{1 - x_0}\right) \left(a_2 n^2 x + a_1 n x + a_0\right)$$

where p_d and p_s are discharge and suction pressure, respectively. The relative displacement, x, was not part of the data set, so it is held fixed at x=1. The calibrated parameters are the maximum pressure ratio, π_0, and the coefficients, a_i, of a first or second order polynomial. The graph below shows the result for λ_{eff} over the 30 data points.

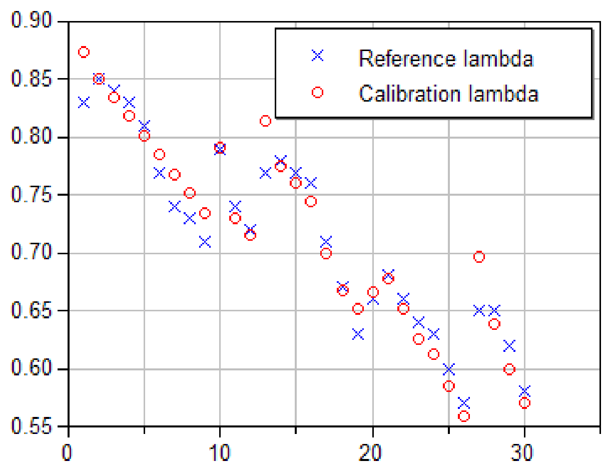


Figure 4. Calibrated volumetric efficiency over 30 data points.

Using compressor speed in rpm as independent variable gives a better view of the calibration result. The three lines in Figure 5 correspond to three different operating points at pressure ratio, π= 2, 2.7 and 4, respectively.

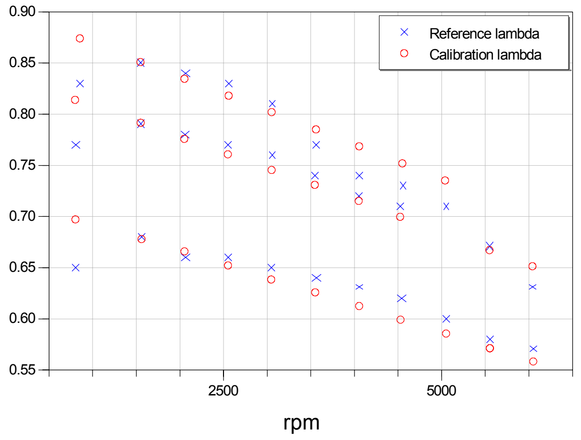


Figure 5. Volumetric efficiency as a function of rpm, showing three data sets at different pressure ratios.

5 Comparisons

Model calibration was previously available in Dymola using external optimization functions, e.g. the BasicOptimizer package using a simplex-method. This was often quite cumbersome to set up since a special model for calibration needed to be created that integrated:

- Model equations and parameter tuners
- Measurement data and residual function
- Evaluation of value function to be optimized

The case in section 4.1 is part of the example package ACWorkbench.CompressorOptimization included in the commercial AirConditioning library. Using the new Dymola calibration GUI most similar examples are set up without creating a task-specific model. For the volumetric efficiency function, a simple wrapper model needs to be created since direct optimization of functions is not yet supported in the GUI. The basic least squares optimizer has replaceable function and additional data and could be used without any model wrapper, but due to current restrictions (in Modelica and in Dymola) it uses packages with several replaceable classes that would no be convenient to directly support in the GUI.

The Modelica code for the wrapper is provided below.

```
model CalibrateVolumetricEfficiency
  "wrapper model for calibrating lambda"
  import CompEff = ThermoFluidPro.
SubComponents.CompressorEfficiencies
  input SIunits.Frequency n "speed in Hz";
  input SIunits.Pressure pi "pressure ra-
tio";
  output Real lambda "vol. efficiency";
  parameter CompEff.EfficiencyPars
    ce "compressor coefficients";
equation
  lambda =
    CompEff.EffectiveVolumetricEfficiency(
      n, pi*1e5, 1e5, 1, ce);
end CalibrateVolumetricEfficiency;
```

The wrapper contains definitions of the inputs and output. They are connected to measurements via the GUI, where the data values can be linearly scaled. In the case of multiple outputs the residuals can also be weighted. The tuners are found in the record class EfficiencyPars together with other coefficients that are not used. Which parameters that are active during calibration, as well as starting and min/max values are also entered in the GUI.

Also the execution of the calibration is significantly faster using the new calibration GUI. The example here is simple and calibrating the two parameters of a first order polynomial takes a few seconds. The calibration result in Figure 4 took less than five seconds and 15 model evaluations to generate. This can be compared with 78 evaluations in approximately 20 seconds using the BasicOptimizer package.

6 Complex steady-state cases

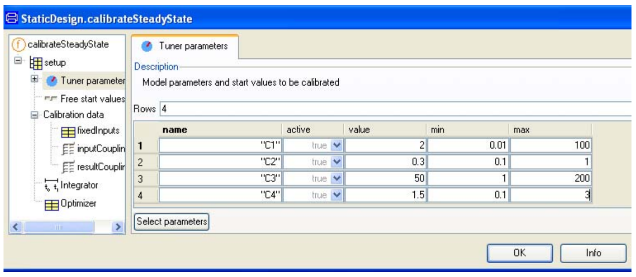
The calibration method has also been tested on a case fitting values to unknown coefficients of the heat transfer and pressure drop correlations on the air side of an automotive condenser. This is a common task in the industry, and often requested by AirConditioning users. In the example two standard correlations from the AirConditioning library are used, both power laws with unknown multiplier and exponent:

$$\mathbf{Nu} = C_1 \times \mathrm{Re}^{C_2} \times \mathrm{Pr}^{1/3}$$

$$\mathbf{\Delta p} = C_3 \times (\dot{m}/\dot{m}_0)^{C_4}$$

The first expression gives Nusselt number, **Nu**, which is used in the condenser model to calculate heat transfer. The second expression gives the pressure drop as a function of mass flow. Reynolds number, Re, depends on air flow and geometry. Prandtl number, Pr, depends on the medium properties. The unknown coefficients are C_1, C_2, C_3 and C_4.

The case was set up in a standard AirConditioning test bench model, supplying fixed outlet pressure and refrigerant subcooling as well as inflow boundary conditions on both the air and refrigerant sides of the heat exchanger. The condenser was set up with the correct geometry and the correlations corresponding to the ones showed above selected for the air side heat transfer and pressure drop. This completes the model setup, with a model corresponding to the same test bench that would be used for simulated validation or manual tuning of the condenser to measured data. The only difference is that a variable for air inlet velocity needs to be included, to connect to the corresponding measurement. Also the parameters to be calibrated were propagated to the top-level for convenience.



Description

Model parameters and start values to be calibrated

Rows 4

	name	active	value	min	max
1	"C1"	true	2	0.01	100
2	"C2"	true	0.3	0.1	1
3	"C3"	true	50	1	200
4	"C4"	true	1.5	0.1	3

Figure 6. Dymola dialog for calibration setup of the heat exchanger case.

Using the new Dymola calibration GUI, the case was easily set up, as shown in Figure 6. On different tabs first the model is selected, then the parameters to be tuned, third the data file is supplied, and last columns with measured data connected to the corresponding inputs and variables in the model. Once setup is completed the model is translated and simulated multiple times. For a steady-state case like this each iteration requires #data points × #parameters × 2 simulations. Since the test bench model in question only required 1 second simulation to solve, the function converges in a few minutes. During the function iterations, a plot shows the residual between the calculated and measured air side power and pressure drop respectively. The final residuals for power are displayed in Figure 7. The final residuals are below 100 W in each point, corresponding to less than 1% relative error. The shape of the residual suggests that the final measured value, at the highest air flow velocity, is either an outlier or that the chosen correlation is not suited for high flow speeds. The final residuals in pressure drop are around 1 Pa, and are not shown.

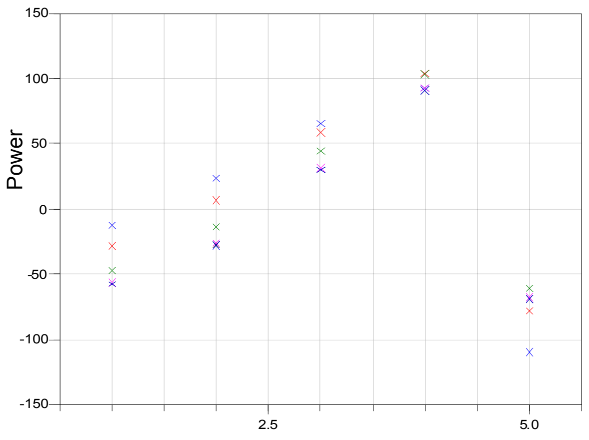


Figure 7. Power[W] residual for the last five iterations.

7 Conclusions

The above shows that Dymola is able to go from static measurements, via plots of correlations, to calibration of parameterized static correlations.

This is done in user-friendly way, without having to build additional models, and furthermore this is useful for real problems, and works more efficiently than the previous optimizer.

Furthermore the static calibration and the complex steady state cases both use the same underlying optimizer as in [2] – demonstrating its usefulness.

References

[1] Dymola User's Manual, www.dynasim.com

[2] Elmqvist, H., H. Olsson, S.E. Mattsson, D. Brück, C. Schweiger, D. Joos, M. Otter, Optimization for Design and Parameter Estimation. In *Proceedings of the 4th International Modelica Conference*, Hamburg, Germany, 2005.

[3] Fletcher, R., *Practical Methods of Optimization*, 2nd edition, John Wiley&Sons, 1987.

[4] Tummescheit, H., J. Eborn, and K. Prölß. AirConditioning – A Modelica Library for Dynamic Simulation of AC Systems. In *Proceedings of the 4th International Modelica Conference*, Hamburg, Germany, 2005.

[5] Försterling, S. *Vergleichende Untersuchung von CO_2-Verdischtern in Hinblick auf den Einsatz in mobilen Anwendungen*, Ph.D.-thesis, TU Braunschweig, Germany, 2004.