Proceedings
of the 4th International Modelica Conference,
Hamburg, March 7-8, 2005,
Gerhard Schmitz (editor)

R. Dorling
Advanced Dynamic Systems, Peterborough, U.K.
Model Validation and the Modelica Language
pp. 375-381

Paper presented at the 4th International Modelica Conference, March 7-8, 2005,
Hamburg University of Technology, Hamburg-Harburg, Germany,
organized by The Modelica Association and the Department of Thermodynamics, Hamburg University of Technology

Model Validation and the Modelica Language

Dr. Richard Dorling
Advanced Dynamic Systems
Datum House, Commerce Road, Lynch Wood, Peterborough, PE2 6LR, UK

Abstract

Model validation is a crucial aspect of the development of any dynamic system that uses computer aided engineering (CAE).

The ease of applying model validation techniques is dependent on the structure of the model, and this is often dependent on the CAE tool used.

The Modelica language is both well-structured, and independent of any CAE tool. As such it exhibits many features that make it ideal in the application of model validation techniques.

This paper considers various aspects of the Modelica language and how they relate to the implementation of a model validation tool.

The validation of a vehicle model is presented as an example of how the features of the Modelica language are used in the validation process.

Finally, future developments of the Modelica language that would enhance the performance of any model validation tool are identified.

1 Introduction

Model validation should be an essential ingredient in any dynamic system development that uses Computer Aided Engineering (CAE) methods.

Comprehensive checks against test data should be made for even the most rudimentary model in order to identify any errors and invalid assumptions in the model. These should then be corrected in order to gain sufficient confidence in the CAE results.

The term ‘model verification’ is used to describe the process by which the behaviour of the model is checked against test data.

The term ‘model validation’ is used to describe the process by which a model, and/or a real system, is corrected so that the model and system’s performance match.

Model verification is thus a prerequisite for, and an essential part of model validation.

The problems with implementing model validation stem from the limitations of existing model verification techniques. These revolve around the comparison of simulation results with test data.

1.1 Existing approach to model verification

The comparison of simulation results against test data can be carried out in two ways: either using time histories; or using frequency responses/statistical data.

The comparison of time histories is notoriously difficult due to the following drawbacks:

1. All inputs to the model must be known.
2. Any errors in a model get magnified during the simulation since they are integrated to generate future time histories.
3. Any error space, calculated by taking the difference between the real and virtual results, will not normally have a single minima.

The above drawbacks in model verification mean that any attempt at model validation must, at best, utilise slow global optimisation tools. The requirement for assumptions to be made for unknown inputs will also have a detrimental impact on the model validation process.

Although the comparison of frequency responses and statistical data generally overcome these drawbacks, both methods suffer from two further shortcomings:

1. Lack of resolution.
2. Concealment of non-linear effects by statistical processing or Fourier transformation.

These particular shortcomings result in poor model verification and therefore subsequent attempts at model validation will not characterise the complex behaviour of a dynamic system. This detailed information is vital when analysing system performance.

1.2 Example, a spring-mass system

Consider a mass, on a non-linear spring and damper, subject to a disturbance input at the free end as shown in figure 1. This system has a very simple mathematic representation, and yet highlights all the limitations of current model verification techniques and their impact on model validation.

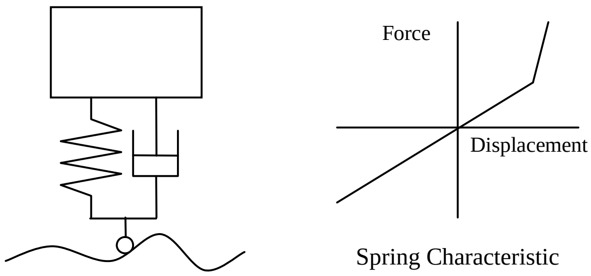

Figure 1: Example dynamic system

For the purpose of this example it will be assumed that the only measurement made during a test of the real system is that of the acceleration of the mass.

Considering the use of time history methods, and assuming the input to the system is known, it can immediately be shown that the error space between the measured and simulated accelerations does not have a single minima when the mass parameter is varied (figure 2).

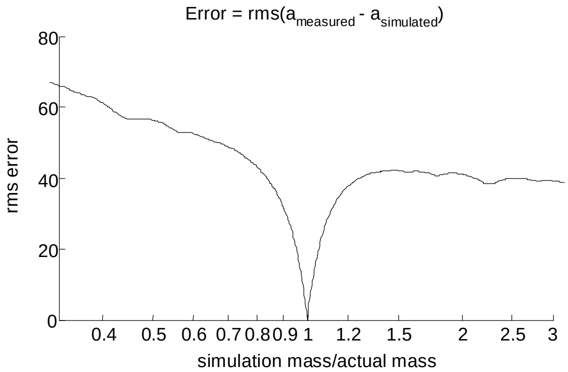

Figure 2: Verification error against mass

Considering frequency response methods, it can be seen that the frequency response of the system does not clearly represent the non-linearity of the spring characteristic, since it is lost within the measurement noise (figure 3).

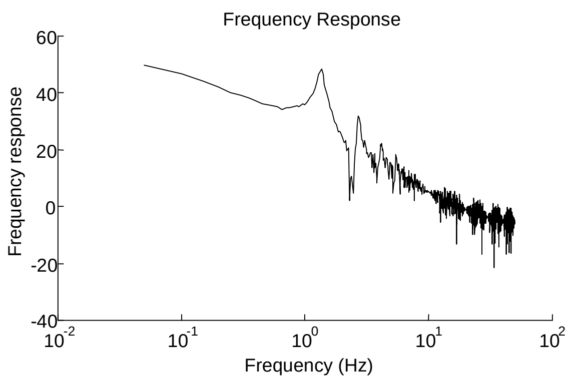

Figure 3: Measured frequency response of system

The more complex a system, the more problematic the limitations associated with current model verification techniques become.

These limitations mean that implementing a model validation technique for a complex dynamic system is almost impossible.

In order to develop a proper method for model validation, a robust model verification technique must first be developed that does not exhibit any of the drawbacks described above.

1.3 A new approach to model verification

The limitations of current model verification techniques all stem from the fact that it is the results of a model simulation that are compared to test data, rather than the model itself.

Any new approach to model verification must therefore compare the model itself to test data rather than the outputs of any analysis. This requires that the model must be independent of the analysis carried out on that model.

In order to achieve this, the model must be driven directly by the test data, and not by a set of inputs predetermined by the formulation of the model.

It is always possible to reformulate any model so that it can be driven by any arbitrary data, as long as there is sufficient data measured about the real system; it is just a matter of matching the number of equations to the number of pieces of information required to solve those equations.

When model verification is approached in this way, the requirement to know all inputs to the model is immediately removed. Instead, these inputs are derived as part of the model verification process.

In addition, the test data to be used can be differentiated and integrated prior to any analysis. As a result, errors do not get magnified by integration within the model and error spaces become well-behaved, exhibiting a single minima.

Finally as this approach is not statistical in nature, and does not use Fourier transforms, non-linearities are preserved, as is the resolution of the data.

The Modelica language has several features that are ideally suited to implementing a model verification technique based on the approach outlined above. Firstly, the equations of the model can be rearranged and manipulated as required. Secondly, the Modelica language includes only information about the model itself and not the analysis of that model.

1.4 A new approach to model validation

The model verification approach outlined in section 1.3 gives a measure of how well the model matches the physical system. This measure can be thought of as determining the 'error' between the model and the physical system, and as stated above this measure is 'well-behaved' exhibiting only a single minima at the point where the behaviour of the model and physical system agree.

The aim of any model validation process is to ensure that the behaviour of the model of a system matches that of its physical counterpart. In this approach, this is achieved by minimising the above error to an acceptable level.

In order to completely validate a model it is necessary to address two distinct issues: parameter identification and model structure development.

Parameter identification is concerned with tuning the parameters of the model to minimise the error between the model and physical system, whereas model structure development examines whether changes to the components or equations of the model would reduce this error.

By using the model verification approach discussed above, in conjunction with a fully parameterised model that allows replacement and updating of its individual components, it is possible to implement both parameter identification and model structure development techniques.

The Modelica language has several features that enable both parameter identification and model structure development to be implemented. This ensures that a robust and comprehensive method of model validation can be applied to Modelica models.

2 Features of the Modelica language relevant to Model Validation

There are many features of the Modelica language that make it ideal for implementing the approach to model verification and model validation described above. These features are discussed under six headings. The first four relate to requirements identified in sections 1.3 and 1.4:

- For model verification it is important that the model is separate from any analysis of that model.
- For parameter identification the model must be fully parameterised.
- For model structure development components must be replaceable with other representations of that component.
- For model structure development the equations of the model must be available in an accessible form.

The following additional features are necessary for the control of the model validation process:

- In order to produce a meaningful result from the model verification analysis, weighting factors must be applied to quantities within the model
- In order to control the model validation process, attributes need to be associated with the quantities, components and equations of the model.

2.1 Separation of the model and analysis

A model in which simulation and/or integration are integral parts of the model is of limited use when implementing model validation technologies, as these models will inherently suffer from the problems outlined in section 1.1.

Although the Modelica language has been designed to produce models that will be simulated (integrated against time) the formulation of these models does not explicitly require this. Furthermore there is no necessity for the integrators to be included within the formulation of the model.

Modelica models can therefore be analysed in ways other than by simulation techniques, such as techniques based on the model verification and validation approaches described above.

2.2 Model parameterisation

Full model parameterisation requires that any parameter of the model can be given a new value and the model re-executed, without the need for extensive recompilation. In this way the sensitivity of the model to parametric changes can be quickly assessed by the model validation analysis.

Parameters within a Modelica model are identified as such by the keyword **parameter,** used in the definition of that quantity. This allows such quantities to be treated appropriately, and the model checked for consistency.

This information can also be used to produce a full parameterisation of the model. Such a parameterisation would require that each parameter of the model was stored independently of the equations of the model, rather than hard coded into those equations.

There is no specific requirement in the Modelica language for models to be parameterised, and therefore this becomes an implementation issue; the language itself carries all the necessary information for a full parameterisation of the model.

2.3 Replaceable components

Being able to replace the model of a component with different variations of that model allows different component models to be assessed as to their suitability to model a particular physical sub-system.

This, in turn, allows intelligent trade offs to be made between the simplicity of individual sub-component models and the accuracy of the model as a whole.

Replaceable components have been part of Modelica since its first documented version. These allow components or sub-components of a model to be replaced with different variations of the same component. For example a model of a resistor can be replaced with a model of temperature sensitive resistor.

2.4 Availability of model equations in an accessible form

In a Modelica model, components are described by a series of equations. These equations are written in a standard form that can be interpreted, simplified and re-arranged as necessary.

The accessibility to these equations means that changes to the basic formulation of the model can be made, and the impact of such changes assessed with respect to the behaviour of the model.

In Modelica models, equations are grouped together under the definitions of the components that they represent. This makes the user-selection of equations to be assessed straightforward.

2.5 The association of weighting factors with quantities within the model

In order to generate a representative error from the model verification process described in section 1.3, it is important that all the errors across the model are scaled appropriately during the model verification process. This ensures that all measurement errors and all modelling errors are treated equivalently, rather than giving emphases to those quantities with a high nominal value.

For example, in a car, if both suspension force and wheel movements were measured as part of a test, it would be incorrect to assume that an error of 1N in the suspension force was equivalent to 1m of deflection in the suspension. In fact, it may be more appropriate to make 5kN of suspension force equivalent to 10mm of suspension deflection. Weighting factors of 5000 and 0.01 would therefore be applied to these quantities respectively. In fact, in most cases it makes sense to weight a given quantities error by the nominal value of that quantity.

Such scaling values can usually be determined from the nominal attribute of quantities within the model, but this is sometimes not the case, as will be discussed in section 4.

2.6 Attributes for control of the model validation process

Control attributes are required to mark quantities, components and equations as being included or excluded from the model validation process.

In certain situations it does not make sense for a quantity, a component or an equation of a component to be assessed as part of a model validation exercise. For example, it does not make sense to change an equation that directly implements a physical law, such as Newton’s Second Law.

In such cases, it is useful to mark these quantities, components, and equations within the model, so that they can be automatically excluded from any model validation analysis.

Annotations, with their relatively free form, are an ideal way of marking such quantities, equations and components, and feeding extra information about the model into the model validation process.

Using annotations in such a way enables all the important information about the validation process that relates to the model to be contained within the model itself.

3 Example: Validation of a model of a Racing Car.

The validation of a model of a racing car against test data is presented. The technique used is based on the model validation approach described above.

The model used for the validation exercise is shown graphically in figure 4.

The data used for the validation is taken from a test of the car on a chassis dynamic test rig and consists of data for:

- Forces applied by aerodynamic loading actuators
- Displacements and Forces applied by road input actuators

- Accelerations of the body and wheels
- Forces in the four pushrods
- Displacements of the four dampers
- Height, roll and pitch of the body of the car

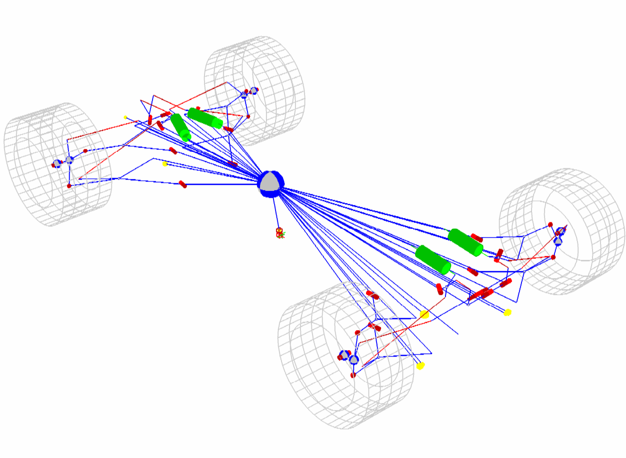

Figure 4: Model of a racing car.

The model validation technique used here is, in general, an iterative process where the largest causes of error are removed first, followed by the second largest, and so on. Each time a source of error is removed, other smaller errors become apparent in the model.

In this example, one iteration of parameter identification and one iteration of model structure development is presented, together with the overall model validation results.

3.1 Parameter Identification

Using the model verification technique described in section 1.3, a time history of the error between the model and measured data can be derived.

This error is analysed with respect to the model, using a cause and effect analysis, to determine which parameters are most likely to be causing the error between the model and the physical system. Furthermore the sensitivity of each parameter to the error is also calculated.

Table 1 shows a sample of results of such an analysis.

Table 1: Results of Parameter Identification analysis

Parameter	Correlation	Sensitivity
Car.Front.AntiRollBar.Stiffness	0.945	165.7
Car.Front.Suspension.Stiffness	0.856	1443
Car.Rear.AntiRollBar.Stiffness	0.832	-40.54
⋮	⋮	⋮

The values in the column labelled ‘correlation’ determine the likelihood that a particular parameter is causing the error between the model and the physical system. In this case therefore, it is most likely that the error is caused by the front anti-roll bar, however it may also be caused by the front springs or the rear anti-roll bar.

The choice of parameter can usually be made using the sensitivity index and engineering judgement. This parameter can then be adjusted, either automatically, or by the user, to a new and better value.

A new table of results similar to table 1, but with one of the modelling errors removed can then be calculated and the next iteration made.

3.2 Model Structure Identification

As an example of the development of the model structure, the rear dampers of the vehicle will be considered.

The model of these non-linear dampers includes an equation that specifies the force generated by each damper due to the velocity across the dampers.

The characteristics of this equation can be checked by removing the equation from the model, or turning the equation off, and treating the force generated by the damper as a continuous input to the model.

The model verification process will then generate a damper force against time that minimises the error between the model and real system.

The true characteristic of the non-linear damper can be examined by plotting the damper force against damper velocity. The form of the original equation can be checked and revised by considering the shape of this characteristic versus that generated by the equation it replaced.

The results for the model validation example are shown in figure 5, with both the left and right damper characteristics plotted.

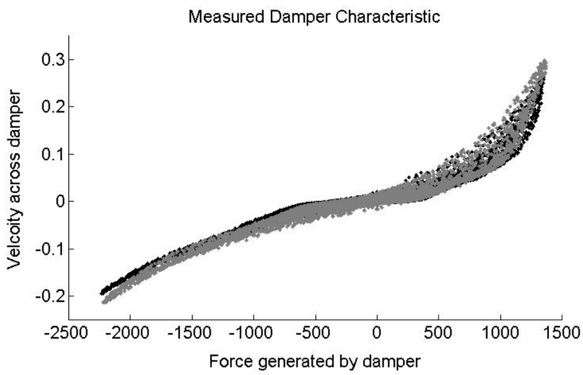

Figure 5: Identified Damper characteristic

There is good agreement between both the left and right dampers, and the expected characteristic.

Comparing the verification errors for the model with the damper equation turned on and turned off enables the error attributable to that component to be quantified.

3.3 Model Validation Results

In this example model validation, an initial error of 0.3 was reduces to a final error of 0.01. The model validation process was stopped at this point as it was decided that the model was of sufficient accuracy for its intended purpose.

The reduction in error can be broken down into 3 categories, listed in table 2.

Table 2: Errors with model

Error due to incorrect calibration of sensors	0.18
Error due to incorrect parameters	0.07
Error due to invalid assumptions	0.04

4 Future Developments of the Modelica Language for model validation

Although the Modelica language has many features that make it an excellent choice for modelling systems that will be validated against test data, it was originally designed for modelling systems to be simulated. As such there are two areas where small changes or developments of the language would improve the validation of Modelica models; these are extensions to the Real attributes and annotation specification.

Two further issues are discussed: the interpretation of Modelica models and the pre-resolution of constraint equations. The resolution of these issues would benefit both model validation and simulation tools.

4.1 Extension of the Real type, to include a sensitivity attribute

Although weighting factors can be currently entered using the **nominal** attribute of Real quantities, there is no possibility to enter sensitivity values. There are two cases in which sensitivity is an issue that need to be properly addressed when applying model validation techniques.

Firstly, some quantities have a large value, but a small range; for example, damper lengths as measured on racing cars. A damper may have a static length of approximately 20 cm, but a variation in this length of between 18 and 22 cm. In this example, the nominal value of 20 cm should not be used as a weighting factor for the error in the damper length; instead a value of 4 cm should be used. If it was possible to attribute a sensitivity of 4 cm to this quantity, this could be used to weight any errors attributed to the damper length.

Secondly, it is possible to have quantities that although have a large range, are very sensitive to variations in their value. Good examples of such quantities are the wheel speeds of a racing car and the forward velocity of that car. When calculating the tyre force, using standard models such as Pacejka, the difference is taken between the forward velocity of the car, and the velocity of the tread of the tyre relative to the vehicle. This gives the slip of the tyre on the ground. Changes in slip of only 1% of the speed of the vehicle can result in large forces being generated by the tyre. In this case the nominal value for the velocity of the car would be 50 m/s, whereas the sensitivity should be 1% of this at 0.5m/s.

For simulation tools sensitivity has not been a problem as the equations are all solved to 0, and tolerances, both relative and absolute are used to determine accuracy. However, such sensitivities may be useful for giving the user a greater degree of control of tolerances in specific situations.

4.2 Specific annotations for model validation

In section 2.6, attributes for equations and components were discussed. It was noted that the **annotation** mechanism in Modelica is suited to the requirements of indicating which equations were to be included in any model validation analysis.

As with the drawing of graphics and icons for components, it would be useful to include any specification of annotations within the Modelica specification, so that all model validation tools could use a common source of models.

4.3 Interpretation of Modelica models

Another area of development of the Modelica language is to consider whether it is possible to interpret (or Just-In-Time compile) Modelica models rather than pre-compile them.

For the purposes of model validation, interpretation offers many benefits over compilation as changes to equations within the model can be assessed more quickly without the need for recompilation of the model.

4.4 Pre-resolution of constraints

It is possible in some cases to pre-solve constraint equations and generate lookup tables for their solutions.-

For example, a double wishbone suspension on a race car, if modelled with solid elements, will have 5 states and 4 constraints. These constraint equations can be easily removed and lookup tables inserted. This reduces the size of matrices within the model validation tool and speeds up the entire model validation process.

5 Conclusions

Model validation should be an essential ingredient in any dynamic system development that uses Computer Aided Engineering (CAE) methods. The choice of CAE tools should reflect this, and so it is important with any modelling language development, such as that of the Modelica language, to consider whether the underlying structure of the language is suitable for the implementation of such techniques.

The Modelica language uniquely combines several features that make it an excellent base for the implementation of model validation techniques.

References

[1] Dorling R.J. ADS Trackside User Manual. Advanced Dynamic Systems Limited, 2004.

[2] Modelica Association. Modelica – A Unified Object-Orientated Language for Physical Systems Modeling – Language Specification, Version 2.1