DOCUMENT CONTROL SHEET

	ORIGINATOR'S REF. NLR-TP-2003-583		SECURITY CLASS. Unclassified

ORGINATOR

National Aerospace Laboratory NLR, Amsterdam, The Netherlands

TITLE

Approximation of black-box system models in Matlab with direct application in Modelica

PRESENTED AT

The Modelica 2003 Conference, Linköping, Sweden, 3-4 November 2003.

AUTHORS W.F. Lammen, W.J. Vankan, R. Maas and J. Kos	DATE December 2003	PP 30	REF 12

DESCRIPTORS
Approximation
Black-box
System modelling
Data-fitting
MATLAB
Modelica

ABSTRACT
Modelica provides a hierarchical and object oriented approach for the modelling of dynamic systems. A system can relatively easily be composed by connecting a number of sub-system and component models together. The resulting integrated system model can be used in design and optimisation studies. The physical behaviour as defined in the sub-system and component models is a key determinant for the system behaviour. Therefore it is of importance that this physical behaviour is adequately modelled. It may however occur that some component models are, for various reasons, represented by no more than a data set of computational or experimental results of the component behaviour. If the precise physical basis for the behaviour of such components is not known or deliberately not taken into account, their behaviour can be considered as a "black-box" input-output relation. In such cases a black-box modelling approach is useful.

This paper describes a generic modelling approach based on approximation methods and applicable for black-box type models in Modelica. Various approximation techniques including polynomial methods, splines, neural networks and kriging models, are applied from a Matlab based graphical software tool with an automatic interface to Modelica. The complete process of model approximation and incorporation into Modelica system models is described and illustrated with a case study.

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2003-583

Approximation of black-box system models in Matlab with direct application in Modelica

W.F. Lammen, W.J. Vankan, R. Maas and J. Kos

This report is based on a presentation held at the Modelica 2003 Conference, Linköping, Sweden, 3-4 November 2003.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	I.1.A.1
Owner:	National Aerospace Laboratory NLR
Division:	Information and Communication Technology
Distribution:	Unlimited
Classification title:	Unclassified
	December 2003

Approved by author: [illegible] 4/12	Approved by project manager: [illegible] 4/12	Approved by project managing department: [illegible] 8/12-2003

Summary

Modelica provides a hierarchical and object oriented approach for the modelling of dynamic systems. A system can relatively easily be composed by connecting a number of sub-system and component models together. The resulting integrated system model can be used in design and optimisation studies. The physical behaviour as defined in the sub-system and component models is a key determinant for the system behaviour. Therefore it is of importance that this physical behaviour is adequately modelled. It may however occur that some component models are, for various reasons, represented by no more than a data set of computational or experimental results of the component behaviour. If the precise physical basis for the behaviour of such components is not known or deliberately not taken into account, their behaviour can be considered as a "black-box" input-output relation. In such cases a black-box modelling approach is useful.

This paper describes a generic modelling approach based on approximation methods and applicable for black-box type models in Modelica. Various approximation techniques including polynomial methods, splines, neural networks and kriging models, are applied from a Matlab based graphical software tool with an automatic interface to Modelica. The complete process of model approximation and incorporation into Modelica system models is described and illustrated with a case study.

List of acronyms

ANN	Artificial Neural Network
CFD	Computational Fluid Dynamics
DACE	Design and Analysis of Computer Experiments
DAE	Differential Algebraic Equation
EU	European Union
GUI	Graphical User Interface
NLR	Nationaal Lucht- en Ruimtevaartlaboratorium
POA	Power Optimised Aircraft
RMSE	Root Mean Square Error

List of symbols

$\mathbb{R}$	the set of real numbers
n	dimension (of input data set)
$\mathbb{R}^n$	*n*-dimensional set of real numbers
$\in$	is an element of (a set)
$\boldsymbol{x}$	(input) vector of *n* independent variables
y	output data element
$\boldsymbol{x}_i$	*i*-th (input) data vector
y_i	*i*-th (output) data element
$\hat{y}$	approximated (output) data element
$\hat{f}(\boldsymbol{x})$	approximation function
ε	approximation error
$\hat{e}(x)$	stochastically distributed error function

Contents

1	**Introduction**	7
2	**Approximation methods**	9
3	**The MultiFit GUI**	11
4	**Translation to Modelica**	13
5	**Verification of the translation to Modelica**	17
6	**Application example**	20
7	**Conclusions**	27
8	**References**	28

(29 pages in total)

1 Introduction

Numerical simulation of physical processes and optimisation of design objectives are commonly used in system design. Modelica is a powerful object oriented modelling language for hierarchical definition of dynamic systems [1]. Modelica system models usually consist of more than one "lower level" system models, resulting in a hierarchically integrated model that is suitable for system design studies. These lower level system models are divided into sub-system models and component models, where a component model is assumed to contain no lower level system models, and a sub-system model is assumed to consist of two or more component models or other sub-system models.

The physical behaviour of the sub-systems and components is a key determinant for the system behaviour. Therefore it is of importance that this physical behaviour is adequately modelled, both with respect to the component behaviour and with respect to system behaviour. Sometimes the model of the physical behaviour of a (sub-)system or component may be too complex to be simulated efficiently within the constraints of the integrated system model. Within such a system model one or a few component models of extreme complexity may exist among several relatively simple component models, resulting in an undesirable and unbalanced system model.

Alternatively, in collaborative development projects, as for example the EU project POA (Power Optimised Aircraft) [2], sub-system or component information may be supplied from one to another company and therefore proprietary constraints may prevent the use of detailed models of the physical behaviour of sub-systems or components. It may occur that some component models are, for example, represented by no more than a table with measurement results of the component behaviour.

In such situations a modelling approach based on alternative system representations is useful. Generic representations based on approximate models can then be applied to sub-system or component behaviour. Various approximate modelling approaches can be distinguished. For example an implementation in Modelica of an approach to system identification, where the focus is on estimation of coefficients of the differential-algebraic equations (DAE) of the time dependent sub-system or component behaviour, is described in [3]. The approach described in the present paper focuses on approximation based on steady-state sub-system or component behaviour.

The steady-state sub-system or component behaviour is assumed to be available as representative data sets, without the detailed mathematical models. These data sets, which represent the underlying system behaviour, may for example arise from series of complex system simulations (e.g. [4]) or physical experiments. If validation or optimisation of the integrated system is considered and consequently large numbers of system evaluations are typically required, approximate representations of the behaviour of the complex sub-systems and components can provide good possibilities for efficient evaluation at low computational cost and with adequate accuracy in Modelica.

Different methodologies are available for efficient approximate representation of system behaviour that is given by data sets. Matlab [5] provides a number of standard functions and toolboxes for approximation and curve fitting, such as the Spline, Curve Fitting and Neural Networks toolboxes, and in addition some other more specific Matlab programs are available, for example with an implementation of the kriging method [11]. In this study a number of these methods are investigated and applied to system simulations in Modelica.

This paper describes and illustrates the development of approximate models in Matlab and their application in Modelica system models. It starts out from data sets representing (sub-)system or component behaviour, which are approximated using a Matlab based tool called MultiFit [6]. An approximate model that has been generated with this tool can be automatically translated into Modelica code. This code can then be incorporated in a Modelica systems model. This complete process is described and illustrated with an example application of a standard engine model.

2 Approximation methods

A large variety of methods and tools is available for approximating system behaviour that is given by data sets. We limit this study to black-box type systems with one or more inputs and outputs. The most relevant approximation methods for such systems are considered and implemented in a generic Matlab based software tool. This tool, named MultiFit and developed by NLR, has been used in previous studies on approximate models of aircraft systems [4][6]. The tool provides a generic and intuitive graphical user interface (GUI) to approximation methods based on polynomial functions (in this case the approximation method is commonly referred to as response surface method [7]), splines [8], neural networks [9] and kriging models [10]. NLR has enhanced the MultiFit tool with the facility to automatically export approximate models from Matlab to Modelica code, see Figure 1.

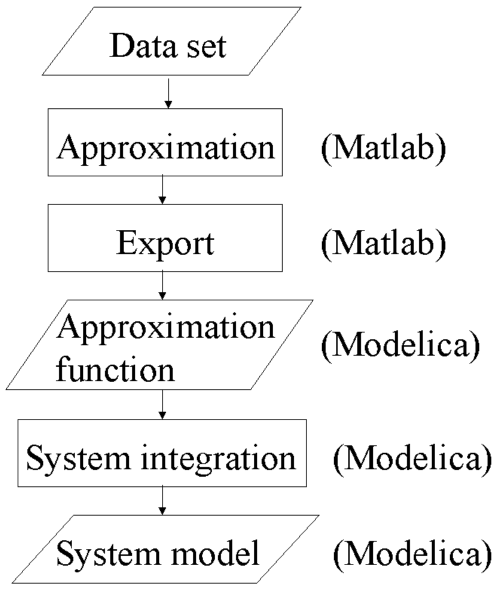


Figure 1 Process diagram of the creation and incorporation of an approximate model into a Modelica system model.

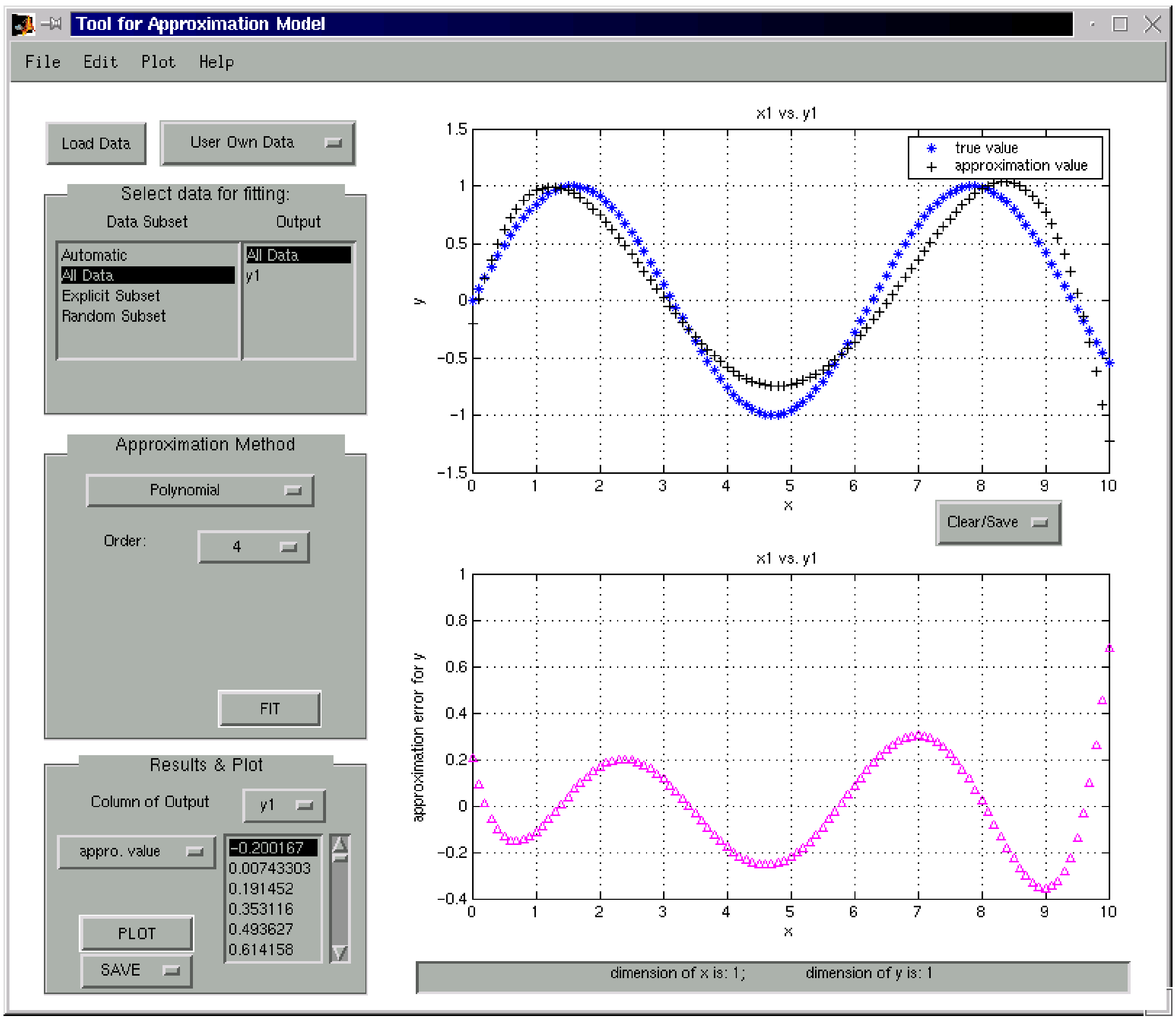
In the approximation tool MultiFit a set of efficient methodologies for approximate representation of data sets has been implemented. The tool makes use of a number of Matlab functions and toolboxes, such as the Neural Network toolbox [5], and other more specific Matlab programs like the DACE program [11] for the kriging method [10] for approximation. The output data that have to be approximated are identified by the variable y, are assumed to be scalar and depend on a vector $\boldsymbol{x}$ of n independent variables. The approximation is performed by fitting an approximate model, based on a specific approximation method, to a given data set $\{(\boldsymbol{x}_i, y_i),\ i = 1,2,\ldots,m\}$, which is referred to as training data set. As such this training data set consists of the discrete samples y_i at input values $\boldsymbol{x}_i$ and represents the real system. Let the approximation be defined as $\hat{y}=\hat{f}(\boldsymbol{x})$, where $\boldsymbol{x}\in\mathbb{R}^n$ and $y,\hat{y}\in\mathbb{R}$. The approximation error is expressed as $\varepsilon= y-\hat{y}$ and is measured in a set of $\boldsymbol{x}$-data points, which is also referred to as validation data set. The quality of the approximation model depends on the achieved accuracy, expressed in terms of the error ε by for example the *root mean square error* (RMSE). A brief description of the approximation methods used in MultiFit is the following:

1. Polynomials: A polynomial function in $\boldsymbol{x}$ is fitted to the data set using a standard least-squares regression technique. The order of the polynomial can be varied between 2 and 6.
2. Splines: Cubic Splines, provided in MultiFit are piecewise smooth polynomial approximations to the data. Both interpolating and smoothing cubic splines as available from Matlab (csapi and csaps) can be called from MultiFit.
3. Artificial Neural Networks (ANN): The ANN-type provided in MultiFit is a feed-forward ANN with one hidden layer. The number of hidden nodes is automatically determined within an interval supplied by the user, such that the approximation is optimised.
4. Kriging models: The kriging method is based on the formula $\hat{y}=\hat{f}(\boldsymbol{x})+\hat{e}(x)$, where $\hat{f}(\boldsymbol{x})$ is a polynomial regression function and a model of the deviation of the regression function $\hat{e}(x)$, which is stochastic with non-zero covariance [10]. In MultiFit kriging interpolation methods are used with a combination of a polynomial regression function of order zero, one or two, and an error model function based on a Gaussian, exponential or cubic spline correlation function, as in [11].

3 The MultiFit GUI

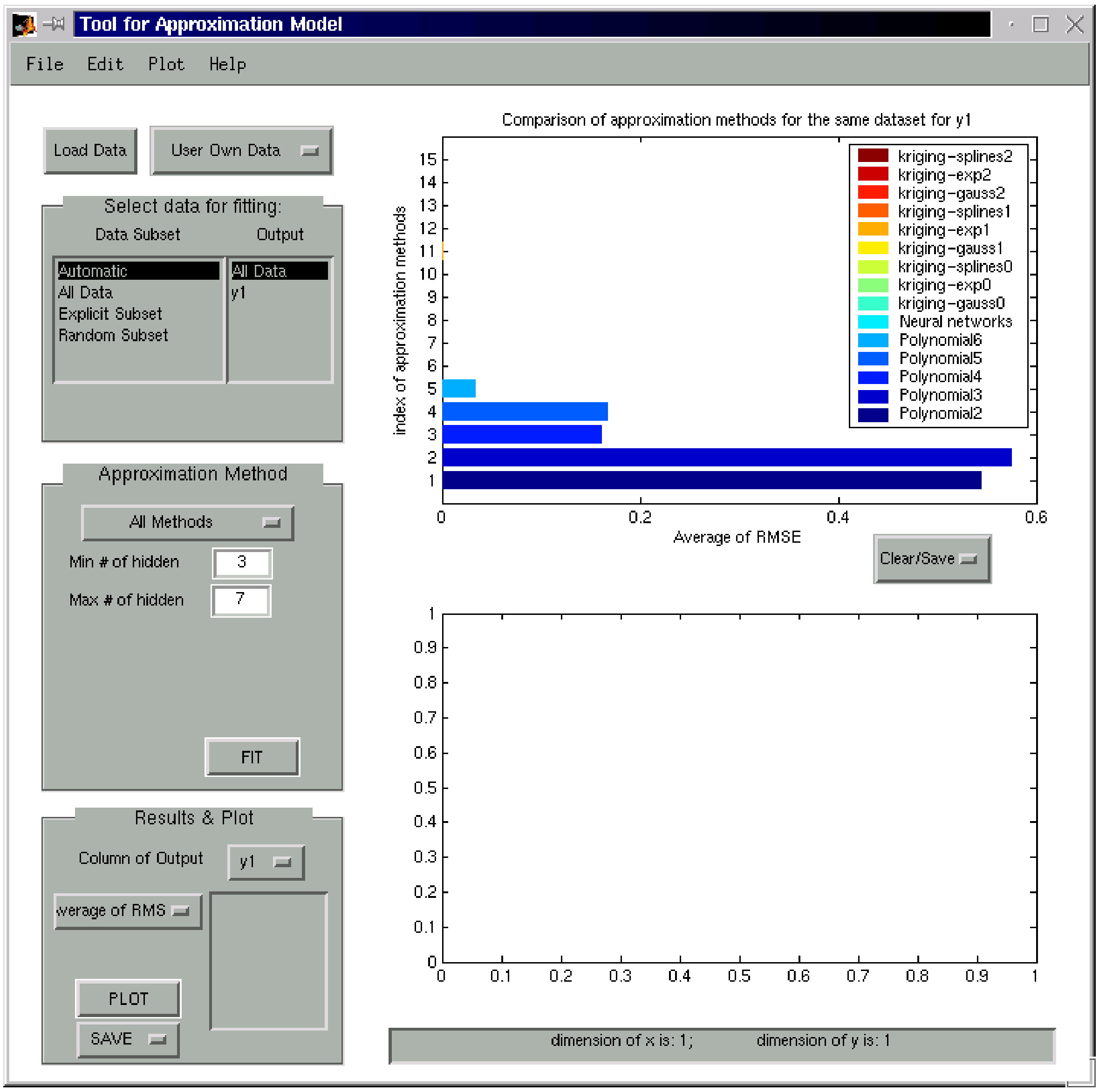
An example of the MultiFit GUI is given in Figure 2. As an illustration of the GUI functionality here a very basic example of a 4th order polynomial fit to a one-dimensional sinusoidal data set of 101 points is represented. The GUI directly presents to the user the data set (in a 2D or 3D plot) that is selected for the fit and the available approximation methods for which the user can make appropriate selections from dynamically generated pop-up menus.

An approximation method that is the best for one data set is not always suitable for another data set. Therefore MultiFit provides an automatic approximation assessment based on RMSE values that gives information about the quality of the fit of the different methods (Figure 3). This way it is possible to select the optimal fit for a given data set.



```
-0.200167
0.00743303
0.191452
0.353116
0.493627
0.614158
```

Figure 2 Example of the MultiFit GUI with a 4th order polynomial fit to a sinusoidal data set (upper graph in the GUI) and the approximation error (lower graph).



Min # of hidden 3
Max # of hidden 7

dimension of x is: 1; dimension of y is: 1

Figure 3 Assessment of fits on a data set using all available approximation methods.

4 Translation to Modelica

MultiFit takes advantage of existing MATLAB approximation functionality. Therefore the process of fitting takes place in MATLAB. A fit result is a function (the approximation function), which can be represented in Modelica format. MultiFit has a functionality that facilitates automatic translation of the resulting approximation function to Modelica code, which can be easily called from the MultiFit file menu (Figure 4).

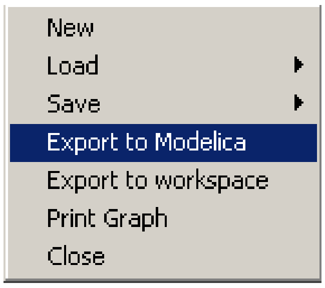
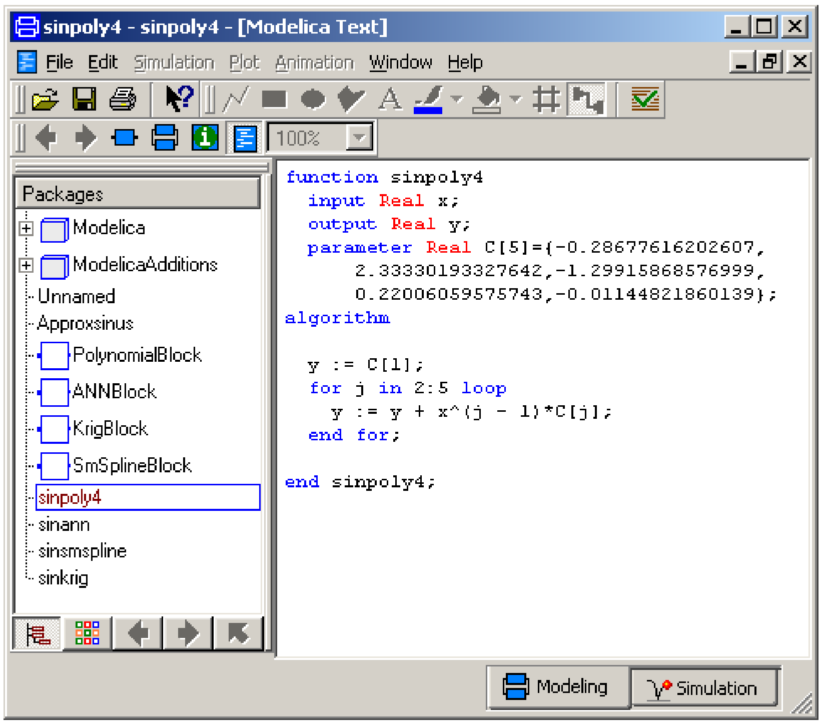


Figure 4 The MultiFit GUI offers easy automatic transfer of approximation models to Modelica, directly from the file menu.

The process is as follows: After an approximation method has been satisfactorily fitted to a data set with MultiFit, this approximation function can be translated to Modelica with the MultiFit export facility. The actual translation is achieved by literal translation of the Matlab expression of the approximation function to Modelica syntax. All the relations between inputs and outputs are explicitly translated with highly accurate export of all real valued parameters.

The exact interfaces of the approximate model in the Modelica environment are not known beforehand. Therefore a modular representation of the approximation code is required. The approximation function is written in a separate Modelica source file as a Modelica *function* (e.g. *sin_poly4* in Figure 5) with the approximation function expression included as a Modelica *algorithm.*



```
function sinpoly4
  input Real x;
  output Real y;
  parameter Real C[5]={-0.28677616202607,
      2.33330193327642,-1.29915868576999,
      0.22006059575743,-0.01144821860139};
algorithm

  y := C[1];
  for j in 2:5 loop
    y := y + x^(j - 1)*C[j];
  end for;

end sinpoly4;
```

Figure 5 Example of the Modelica source code of a 4th order polynomial approximation of a 1-D sinus function as translated from Matlab to Modelica by MultiFit.

The Modelica approximation function can be included in a component model by inserting an equation of the type *y=<functionname>(x)* (e.g. *y=sin_poly4(x)*) in the Modelica code of the component model that should be approximated. In this way several approximation modules can be used to fit a sinus function.

To illustrate the use of the approximated sinus functions in Modelica, these approximate functions are evaluated in Dymola [12] and compared to the exact Modelica sinus function. In Figure 6 the overall Modelica model is shown, which simulates a sinusoidal signal as a function of time and calls four different approximation functions (4th order polynomial, smoothing spline, ANN and kriging with constant regression and Gaussian correlation) that were automatically generated from MultiFit using the sinusoidal data set of 101 data points as was shown in Figure 2.

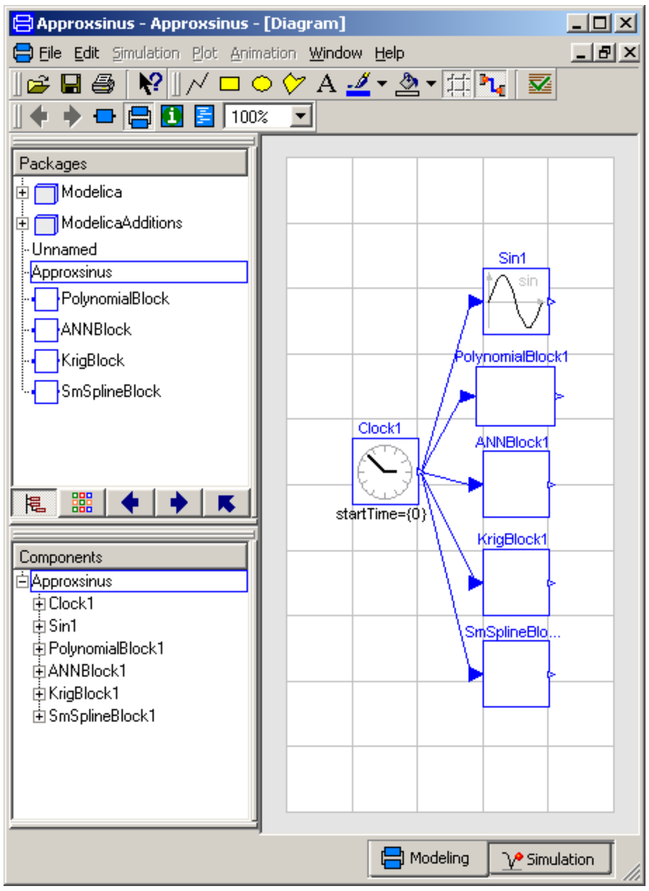


Figure 6 Modelica model with multiple modules to approximate a sinus function.

Figure 7 llustrates the fit results compared to the Modelica sinus signal and confirms the MultiFit information that the polynomial fit has a relatively large approximation error, compared to the other methods.

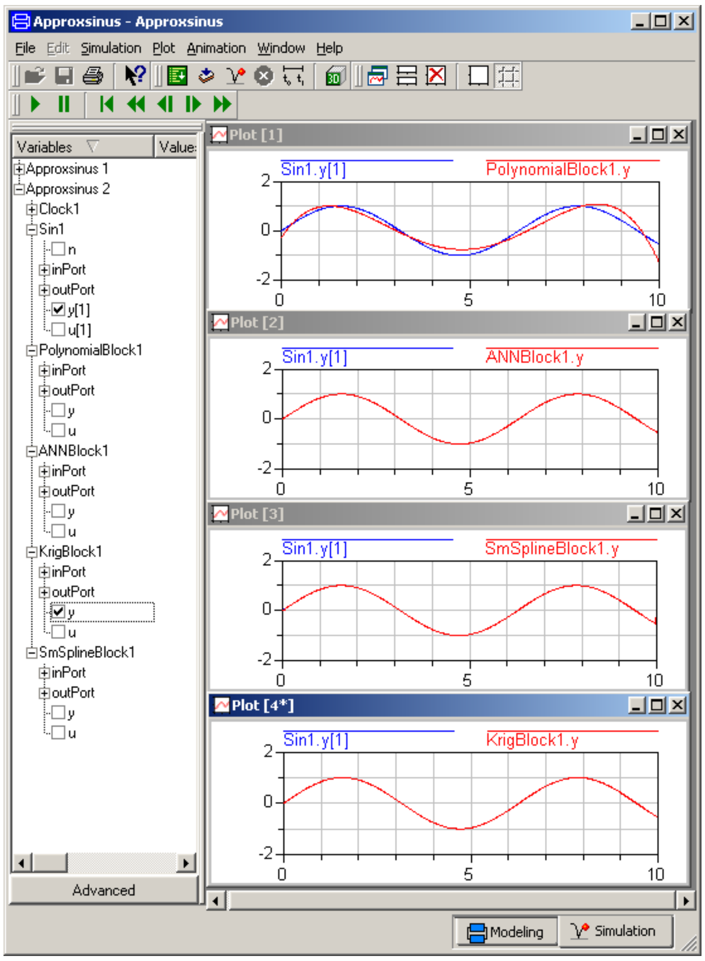


Figure 7 *Dymosim simulation results with sinus approximations by (from top to bottom) 4th order polynomial, kriging (constant regression Gauss corr.), ANN, smoothing spline.*

5 Verification of the translation to Modelica

To illustrate the translation process and demonstrate the validity of the result, a small verification study of the translation from Matlab to Modelica of all the MultiFit approximation methods is presented. In order to investigate the validity of the translation process not just in a theoretical case with a data set based on some analytical function, a more realistic case is considered where the data set is based on a numerical experiment of the behaviour of an aircraft air-conditioning system. This experiment is based on samples of the local temperature in the aircraft cabin as predicted by CFD simulations, as a function of different settings of the air-conditioning system in terms of inflow temperature and velocity [4], see Figure 8. This data set consists of 121 data points, each of which with a scalar output value (i.e., cabin temperature value).

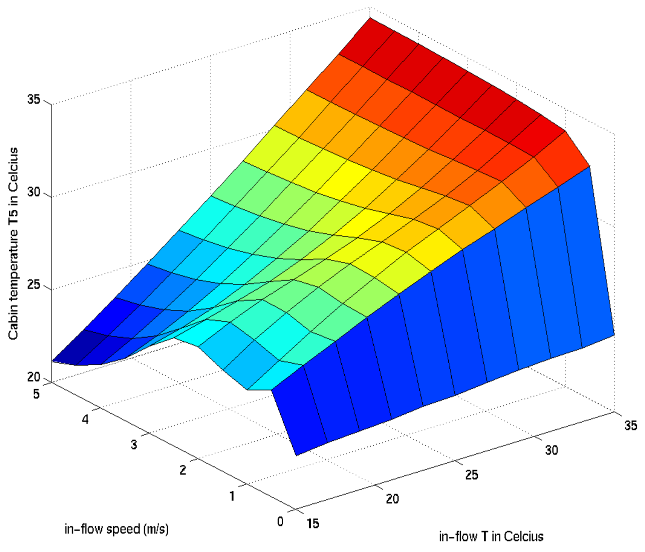
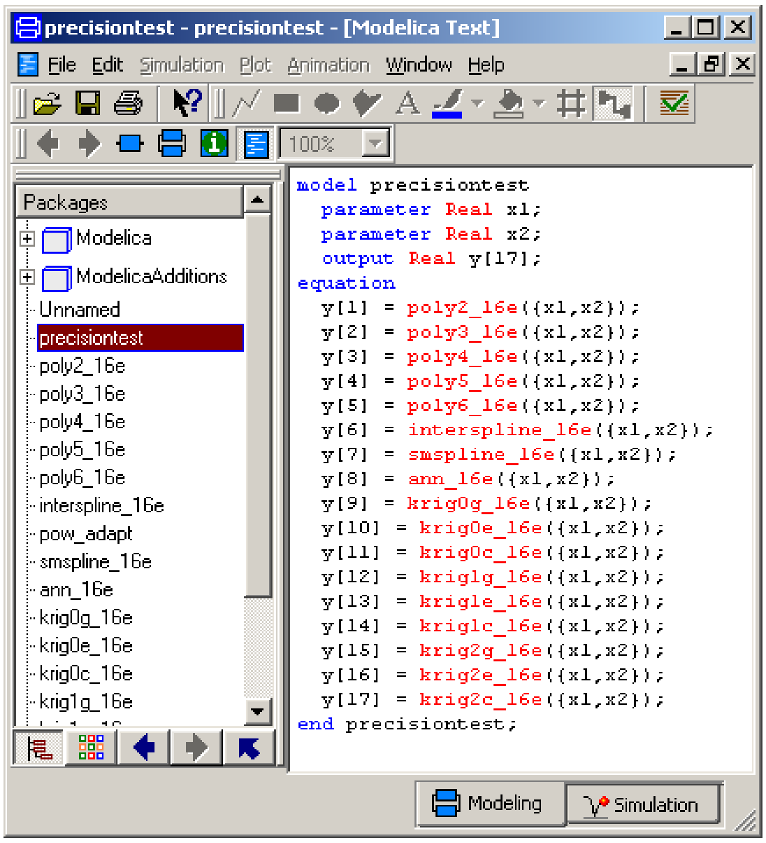


Figure 8 Example data set of local cabin temperature as a function of inflow temperature and velocity, obtained from aircraft cabin CFD simulations

This verification intends to assess the validity and accuracy of the translation process as follows: the approximation methods are first fitted to the data set with MultiFit. Then all the resulting approximation functions are translated to Modelica with the MultiFit export facility. The resulting Modelica code then contains the mathematical expressions of the approximation functions, which should be identical to the expressions in Matlab. It should be noted that in the translation process all real values (of parameters etc.) are exported in *%.16e* format in order to avoid any truncation errors.

The different approximation functions are stored in different Modelica *function* objects and are called from one single Modelica *model* object shown in Figure 9. Then the Modelica approximation functions are compiled in Dymola and the resulting *dymosim* executable is evaluated in a test set of input points (i.e. x-values). In this verification study this test set consists of the training data set that was used to create the approximation functions in MultiFit. Then the resulting output values of the approximation functions are transferred back to Matlab (again in *%.16e* format) and compared to the output values of the original approximation functions in Matlab. For each approximation function, the maximum value of the difference between Modelica and Matlab output arrays is considered and shown in Figure 10.



```
model precisiontest
  parameter Real x1;
  parameter Real x2;
  output Real y[17];
equation
  y[1] = poly2_16e({x1,x2});
  y[2] = poly3_16e({x1,x2});
  y[3] = poly4_16e({x1,x2});
  y[4] = poly5_16e({x1,x2});
  y[5] = poly6_16e({x1,x2});
  y[6] = interspline_16e({x1,x2});
  y[7] = smspline_16e({x1,x2});
  y[8] = ann_16e({x1,x2});
  y[9] = krig0g_16e({x1,x2});
  y[10] = krig0e_16e({x1,x2});
  y[11] = krig0c_16e({x1,x2});
  y[12] = krig1g_16e({x1,x2});
  y[13] = krig1e_16e({x1,x2});
  y[14] = krig1c_16e({x1,x2});
  y[15] = krig2g_16e({x1,x2});
  y[16] = krig2e_16e({x1,x2});
  y[17] = krig2c_16e({x1,x2});
end precisiontest;
```

Figure 9 Example of the Modelica model that calls all the approximation functions that have been translated to Modelica.

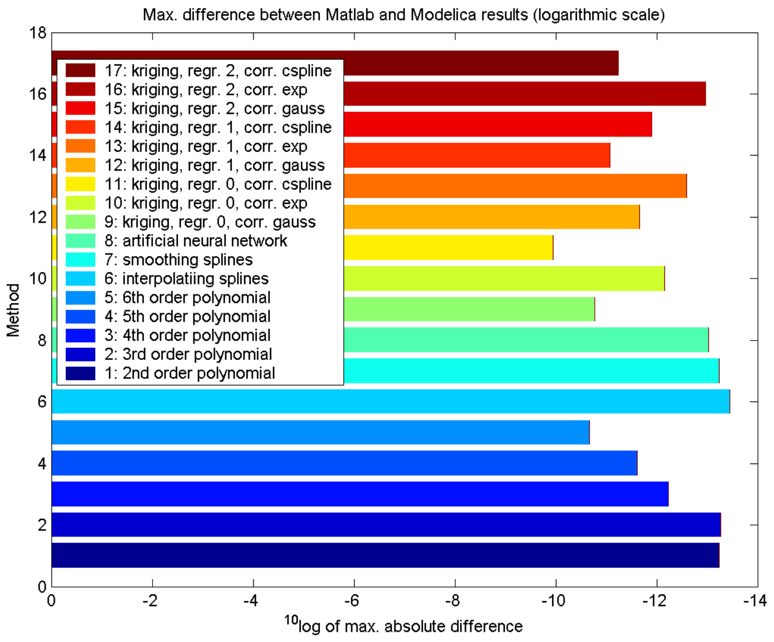


Figure 10 Histogram plot of the maximum difference values for each approximation method.

The differences shown in Figure 10 are, as expected, not far from machine precision (~10^{-16}) for most of the fits. However, some of the fits, in particular the higher order polynomial and some of the kriging models, have much larger differences between the Matlab and Modelica computations (up to ~10^{-10}). For the polynomials this effect is due to the rather large x-values (temperatures of up to 35 ^{o}C) that exist in the considered data set and give very large values in the polynomial arithmetic (i.e., up to 35^6 ~= 1.8 10^9 in the 6th order polynomial) that amplify the possible difference in the exported parameter values. It should be noted that the polynomials have been implemented in a straightforward way, without the re-scaling of variables.

For the kriging models the differences between the Matlab and Modelica computations are due to the very large values of some of the coefficients and arithmetic operations in the approximation functions in these cases. In particular in the kriging models with exponential and Gaussian correlation functions (fits 9, 11, 12, 14, 15 and 17) coefficients of order 10^3 and 10^4 occur.

The following can be concluded from this verification study. The accuracy of the translation is very high, as is shown for the realistic case in this section. Machine precision is not achieved for the maximum values of the differences between the fits in Matlab and Modelica but the discrepancies between the fits can be explained, as shown in the above example. Moreover, these differences are negligible when compared to the errors (e.g. RMSE) of the approximation functions (both in Matlab and in Modelica) in validation data points, which are in the orders of 10^{-1} to 10^{-3}.

6 Application example

As explained in the introduction of this paper, the reasons for applying approximate models of (sub-) system or component behaviour are various. In this section we will consider the case where a sub-system model is available in Matlab/Simulink and has to be integrated into a system model in Modelica because of the ease of multi-physical modelling and sub-system model integration in Modelica. However, the effort of translating the complete model of the physical behaviour as implemented in the Matlab/Simulink model into Modelica code is not intended and therefore an approximate modelling approach is applied.

Figure 11 shows a Matlab/Simulink [5] demonstration model of a combustion engine. Although this is a relatively simple representation of an engine, the emphasis is on the fact that the steady-state behaviour of this model can be integrated into a Modelica system model without completely translating the internal model logic by using the approximate modelling approach. Therefore the model will be treated as if it were a black-box. The inputs of the model are the fuel throttle angle and the torque load. From these values the model calculates the crank speed of the engine in radials per second. If constant values are inserted for the torque load and the throttle the crank speed quickly converges to a steady-state value, see Figure 12.

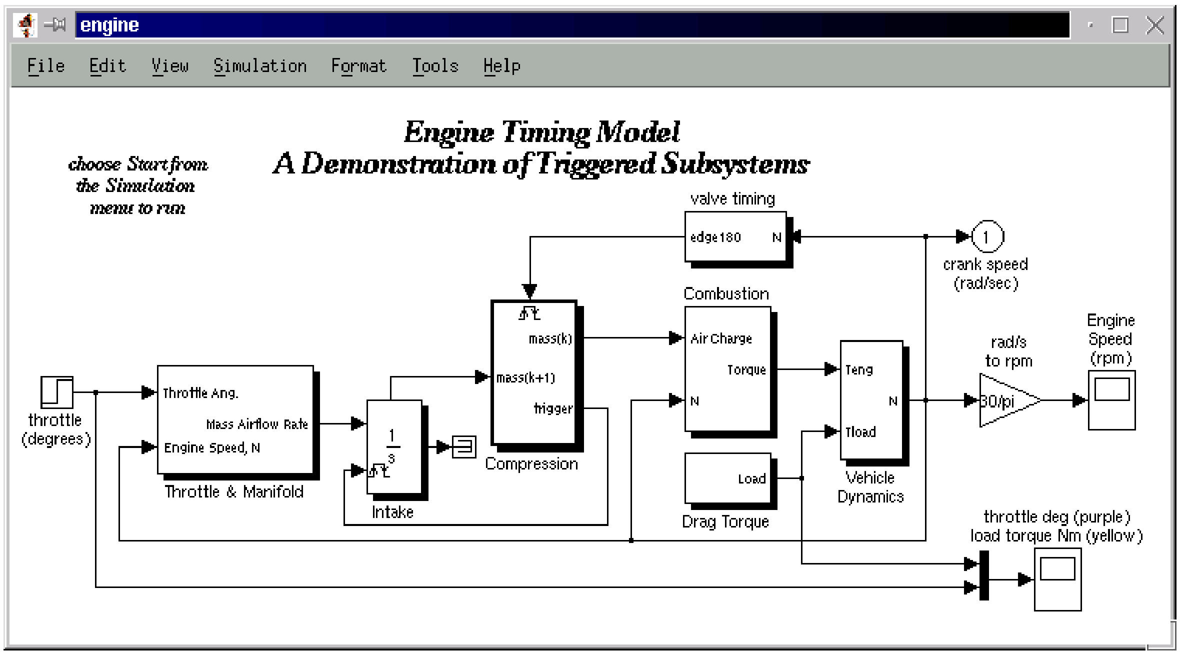


Figure 11 Simulink demo model of an engine [5].

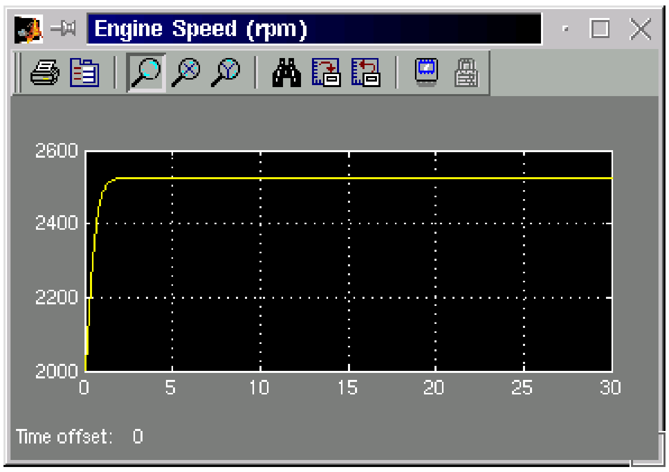


Figure 12 Simulation results for the calculated crank speed (converted to revolutes per minute) with constant torque (20 Nm) and throttle (10 degrees) values

In order to create a data set of the behaviour of the engine, a design study has been performed where the throttle angle and the torque load have been varied from 0 to 20 degrees and from 0 to 50 Nm, with steps of 2 deg and 5 Nm, respectively. For each torque-throttle combination a simulation of 30 seconds has been performed. For some combinations, i.e. a relatively large torque load compared to the throttle angle, the crank speed drops far below zero and the simulation becomes unstable and the crank speed is not defined. Therefore the model is considered inadequate for these input combinations and these points are excluded from the training data set. The resulting training data set contains 94 valid points and is plotted in Figure 13.

All fit methods offered by MultiFit have been applied to this data set, except the spline methods, since this data set is "gridded" and therefore not appropriate for the applied spline implementation [5]. To determine the best fit method a separate validation set has been generated beside the data set. The validation points are also plotted in Figure 13.

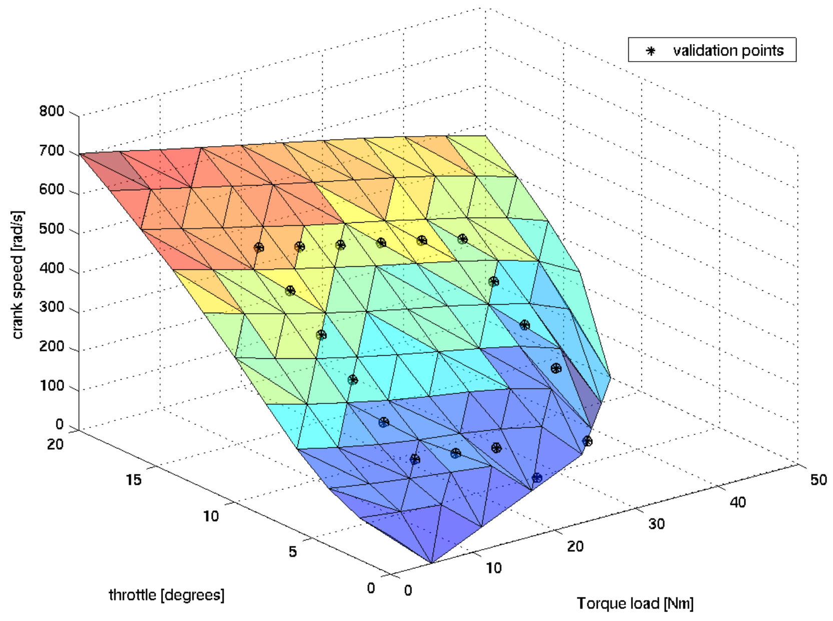


Figure 13 Data set generated from Simulink engine model, with validation data points

The RMSE of the approximate model in the validation points is used as the criterion to determine the optimal fit method for this data set. The results are plotted in Figure 14.

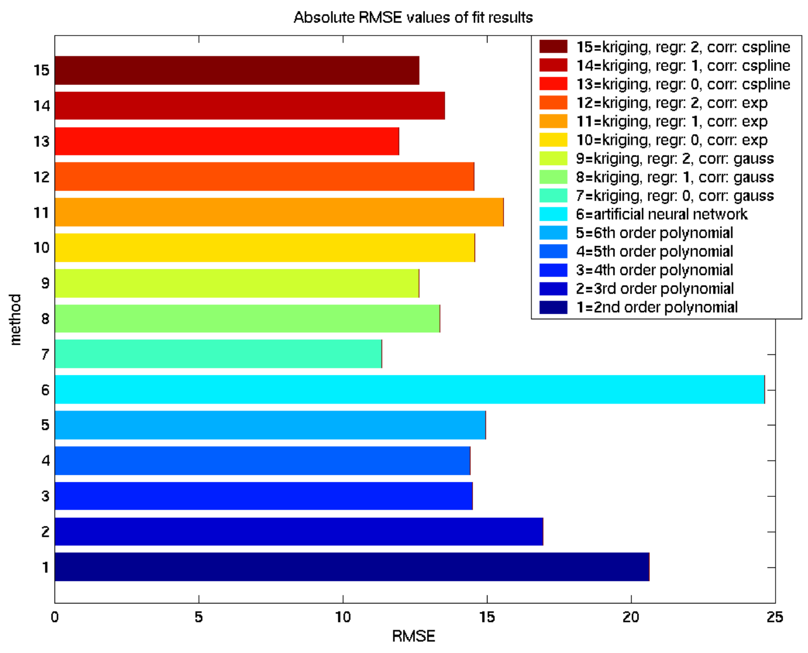


Figure 14 Absolute RMSE values based on a 18 point validation set of the engine model.

It was found that the kriging method with constant regression and Gauss correlation has the lowest RMSE (Figure 14) and gives the best approximation. Therefore, the corresponding approximation function has been translated to Modelica, see Figure 15.

In Modelica an engine model has been created that calls the approximate function to define the relation between the throttle, torque load and crank speed, see Figure 16. Note that the case in which the throttle-torque combination would exceed the domain of the original Simulink model is interpreted as an engine shut-down. In this case is the crank speed is set to zero.

```
algorithm

  // scale x
  for i in 1:2 loop
    sx[i] := (x[i] - Ssc[1, i])/Ssc[2, i];
  end for;
  // make correlation vector
  for i in 1:94 loop
    corr := 0;
    for j in 1:2 loop
      corr := corr - theta[j]*(sx[j]-S[i, j])^2;
    end for;
    corr := exp(corr);
    r[i] := corr;
  end for;
  //rescale y
  sy := gamma*r;
  sy[1] := 1*beta[1] + sy[1];
  y := Ysc[1] + Ysc[2]*sy[1];
```

Figure 15 Kriging (constant regression, Gauss correlation) approximation model of Simulink engine in Modelica code

```
model Engine "engine model based on approximation"
  Real w_rad;
  output Modelica.SIunits.Power P_generated=flange.tau*w_rad
    "Power supplied to loads";
Modelica.Blocks.Interfaces.InPort Throttle "Throttle angle"
Modelica.Mechanics.Rotational.Interfaces.Flange_b flange

Equation

  If noEvent((Throttle.signal[1] < 6 and 5*Throttle.signal[1] >= flange.tau)
    or (Throttle.signal[1] >= 6 and 2.5*Throttle.signal[1] + 15 >= flange.tau))
  then
    w_rad = engdata1_kri0G({flange.tau,Throttle.signal[1]});
  else
    w_rad = 0;
  end if;
  der(flange.phi) = w_rad;
end Engine;
```

Figure 16   Modelica implementation of the wrapping engine model

The engine model has a rotational mechanical connector (the engine shaft) on one side and a signal input (throttle angle) on the other side. Therefore the model can be connected to other Modelica components like signal generators and gear-boxes. The engine model has been integrated, together with some of these other components, into a Modelica system model, as shown in Figure 17.

In combination with prescribed torque load and throttle signal the engine model has been tested, where for the sake of simplicity a gear-box ratio equal to one was used, see Figure 18.

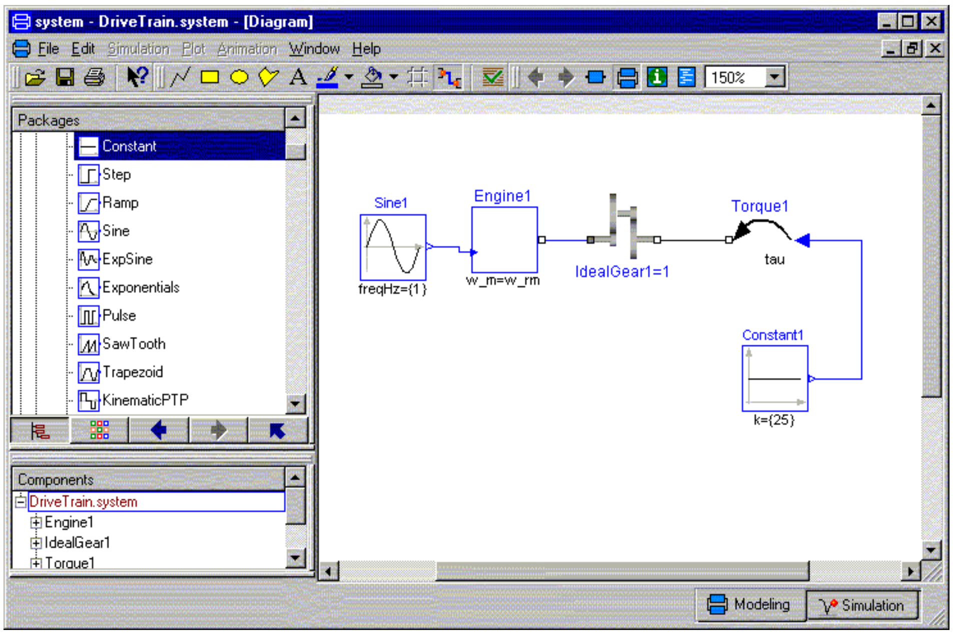
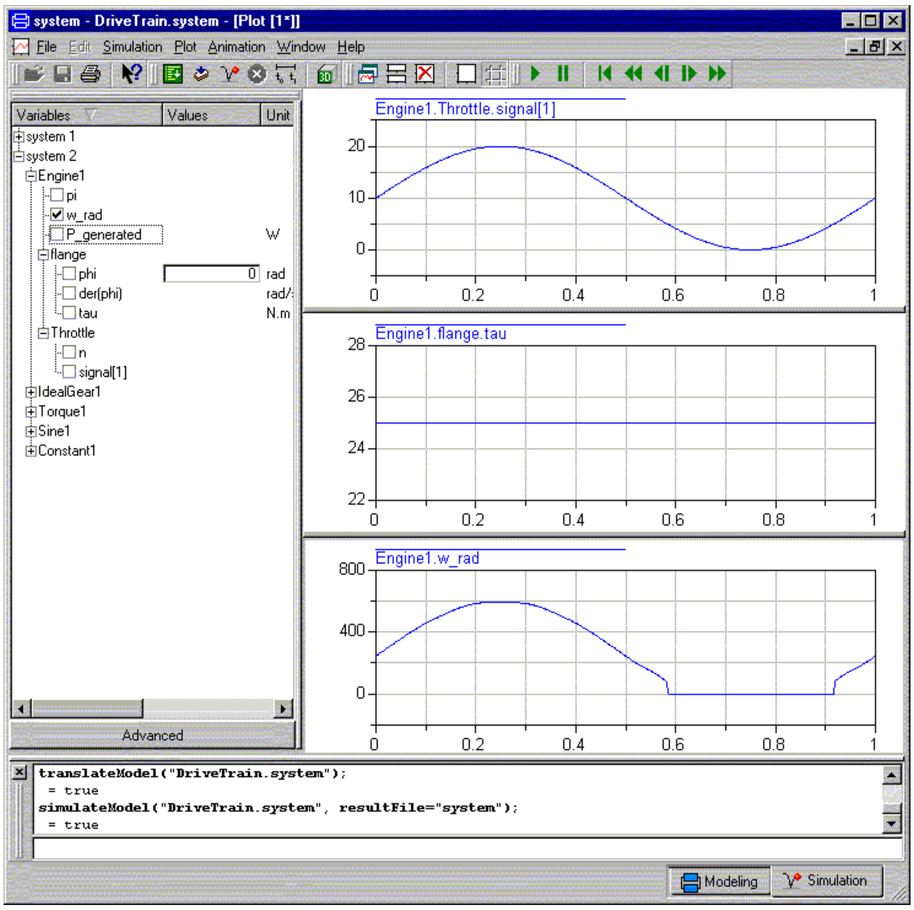


Figure 17 Engine in combination with torque load

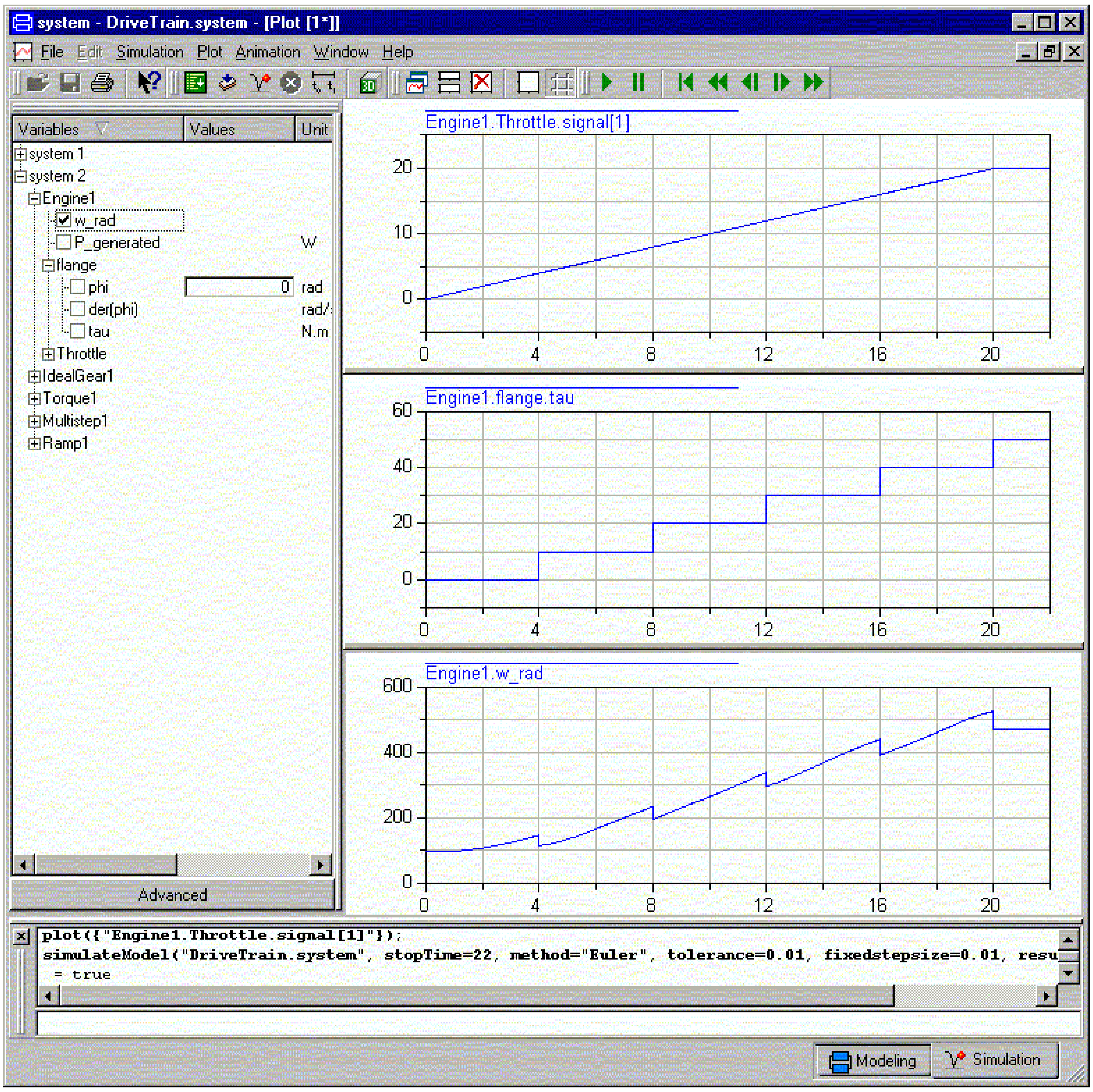


```
translateModel("DriveTrain.system");
 = true
simulateModel("DriveTrain.system", resultFile="system");
 = true
```

Figure 18 Dymola simulation results with a constant torque load and a varying throttle signal

Figure 18 shows that below a certain throttle value the engine crank speed will be zero if a constant torque is applied. Figure 19 also shows how both the torque and the throttle can be varied, e.g. when simulating a strongly simplified automatic gearbox based on the steady state engine behaviour.

Since the engine model has a mechanical interface it can be connected to other physics models e.g. a generator, hydraulic and pneumatic pumps etc. to simulate integrated multi-physics systems.



```
plot({"Engine1.Throttle.signal[1]"});
simulateModel("DriveTrain.system", stopTime=22, method="Euler", tolerance=0.01, fixedstepsize=0.01, resu
 = true
```

Figure 19 *Dymola simulation results with a step-wise increasing torque load and a linear increasing throttle angle*

7 Conclusions

An approximate system representation in Modelica has been presented, which is complementary to Modelica's multi-physics modelling paradigm. This complementary system representation is based on input-output data sets. This system representation is particularly useful if the physical behaviour of a considered (sub-)system is too complex, not known, computationally expensive, protected, or just not available.

The approximate system representation in Modelica can be generated with the Matlab based tool MultiFit that was developed at NLR. This tool provides easy and common access to several approximation methods, since each of these methods has its specific merits and there is no "globally best" method available.

It can be concluded that the combination of the NLR tool MultiFit with Modelica supports the full process from approximation of data sets to the integration with other multi-physics components for system optimisation. Specifically in the cases that models have restricted information or are computationally complex the approximation approach promises to be useful for integrated system design.

8 References

[1] Modelica Association, http://www.modelica.org.

[2] Power Optimised Aircraft, contract G4RD-CT-2001-00601 under the European Communities 5th framework Programme for Research - Competitive and Sustainable Growth - Key Action, New Perspectives in Aeronautics. http://www.poa-project.com..

[3] Sjöberg J., Fyhr F., and Grönstedt T., Estimating parameters in physical models using MathModelica, 2nd International Modelica Conference 2002.

[4] Vankan, W.J. and R. Maas: Approximate modelling and multi objective optimisation in aeronautic design, CMMSE 2002 Conference, Alicante, Spain, 2002, NLR-TP-2002-386.

[5] The MathWorks: Developers of Matlab and Simulink, http://www.mathworks.com.

[6] Vankan, W.J., J. Kos, W.F. Lammen: Approximation Models for Multi-Disciplinary System design - Application in a Design Study of Power Optimised Aircraft, Eurogen 2003 Conference, Barcelona, Spain, 2003, NLR-TP-2003-369.

[7] Myers, R.H. and D.C. Montgomery: Response surface methodology: Process and product optimization using designed experiments, Wiley, New York, 1995.

[8] de Boor, C.: A Practical Guide to Splines, Springer-Verlag, 1978.

[9] Caudill, M.: Neural Networks Primer, San Francisco, CA: Miller Freeman Publications, 1989.

[10] Simpson, T.W., T.M. Mauery, J.J. Korte and F. Mistree: Kriging models for global approximation in simulation based multi-disciplinary design optimization, AIAA Journal, 39(12), 2001.

[11] Lophaven, S.N et al. DACE: A Matlab Kriging Toolbox, Technical University of Denmark, IMM-TR-2002-12, http://www.imm.dtu.dk/~hbn/dace.

[12] Dynasim AB, http://www.dymola.com.

Acknowledgements: The presented work has partially been performed within the framework of the POA project, under contract G4RD-CT-2001-00601. Furthermore the support of Ms. C. Shen, of the University of Amsterdam, in the work on the approximation modelling, and of Dr. H. van der Ven, of NLR, and Prof. A. Veldman, of the University of Groningen, in the work on the cabin airflow CFD simulations, are gratefully acknowledged.