2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

A novel nonlinear model reduction method applied to automotive controller software

Oskar Nilsson*, Anders Rantzer
Department of Automatic Control
Lund University
Box 118, SE-221 00 Lund, Sweden
{oskar,rantzer}@control.lth.se

Abstract— **The automotive industry is experiencing tightening emission legislations together with high demands on performance and driveability. As a counteraction, controller software tends to become more and more complex. Intricate controller software has several downsides, the large number of controller parameters yields an exhaustive calibration task, often performed through costly experiments. In addition, to guarantee reliability, validation and verification analysis is performed on the controller in combination with the engine. This task would also greatly benefit from a less complex controller structure.**

Here a novel model reduction method of nonlinear discrete-time systems is introduced and applied to an engine controller used in current production cars. The result is a nonlinear piecewise affine system with improved simulation speed.

I. INTRODUCTION

The current control design development process in automotive industry involves many expensive experiments and hand-tuning by experienced personnel. This process is time-consuming and even if only small changes have been done between two car models, many tuning tasks have to be repeated. Model-based development is a promising approach to reduce costs, development time and dependency of the undocumented knowledge possessed by experienced personnel. The key idea is to replace expensive experiments with simulation of mathematical models.

Modeling is a mayor undertaking when introducing a model-based development process. Models for various purposes yield different requirements on e.g. precision and simulation speed. A systematic method to reduce model complexity would be very useful tool to aid this process.

II. MODEL REDUCTION OF NONLINEAR SYSTEMS

Model reduction of nonlinear systems is a research area under heavy development. The currently available methods can be divided into the following categories.

A. Heuristic methods

Probably the most common way to simplify nonlinear models is through heuristic methods. For example, indirect model reduction is performed in all modeling-work when complexity is chosen to match the intended model purpose. There are three common ways to reduce complexity:

- To discard effects that by intuition or experience have a relatively weak impact on the dynamics of interest.
- Separation of time-scales and replacing relatively fast dynamics with static gains.
- Averaging several effects into one pseudo-effect.

All three approaches require great knowledge and intuition of the modeled object. However, attempts to perform these simplification steps in a systematic automatized manner has been investigated, see for example [4]. The second mentioned method can be applied in a more formal manner, it is commonly called the *singular perturbation method*, see [5].

B. Linear methods

Here, a linearization, around an equilibrium or trajectory, is made followed by the application of some linear model reduction method. The obvious downside of this procedure is that the end result will be a linear model that can only be expected to perform well in a region close to the mentioned equilibrium or trajectory. Further, the size of this region depends on how nonlinear the original system is.

C. Balancing of nonlinear systems

Balanced truncation is a popular method for model reduction of linear systems introduced in [1]. Recent research has extended this method to also cover the nonlinear case, see [2] and its discrete-time counterpart [3]. Here, a balancing nonlinear coordinate-change is applied followed by truncation of states. The method has strong mathematical support but due to the required numerical effort only models with very moderate size have so far been considered.

D. Pseudo-linear methods

These methods try to extend ideas of reduction of linear systems to the nonlinear case. Similar to the method mentioned in Section II-C they apply a coordinate-change followed by truncation, however here the coordinate change is linear. This restriction to linear subspaces makes applicability to large systems possible. The main difference between the following methods is how the coordinate-change is found.

A very commonly used method for nonlinear model reduction is the so called *Proper Orthogonal Decomposition*

*O. Nilsson is financially supported by Toyota Motor Corporation and Sweden Japan Foundation

method, introduced in [6], [7]. Here principal component analysis is performed on state data and the subspace that captures the majority of the variance is chosen. A common application is discretized partial differential equations, see [8]. The standard version of this method does not take any output-signal into consideration and can therefore be disadvantageous for control purposes.

A recent contribution is found in [9], the so called *empirical Gramian* approach extends ideas from *balanced truncation* of linear systems to the nonlinear case. Here state-space data are collected while impulse input-signals in different directions are applied. The data is then used to estimate a constant controllability Gramian matrix. Similarly, a constant observability Gramian matrix is constructed from simulation data generated by different initial values distributed on the unit sphere.

In [10] the so called *Trajectory Piecewise-Linear Approach* is presented. The method applies linear methods on linearizations distributed over one or several trajectories. Here the main focus is not only on reducing the number of states but also improving simulation speed.

The method introduced in this paper also belongs to this class of reduction methods where a linear coordinate change is used. Further, as in [9] it applies the notion of Gramians and in similarity with [10], linearizations along trajectories are used.

III. Preliminaries

The method presented below is based on theory concerning linear time-varying systems. Consider the linear discrete-time time-varying system

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{aligned} \quad k \in [1, N],$$

where x_k is the state-vector, u_k the input-signal and y_k the output-signal at time k. Further, A_k, B_k, C_k and D_k are time-varying matrices of appropriate dimensions. In [2] the notion of so called energy functions is used. The *controllability energy function* is the amount of energy required in the input-signal to reach a specific state. In the linear-time varying case this can be stated as the optimal control problem

$$L_c(x^*, t) = \min_{\substack{u \in L_2(0,t) \\ x_1 = 0 \\ x_t = x^*}} \frac{1}{2} \sum_{k=1}^{t} ||u_k||^2. \quad (1)$$

That is, $L_c(x^*, t)$ is the minimal amount of energy in u required to reach a certain state x^* at time t, starting from the zero initial state.

Further, the *observability energy function* determines the energy induced in the output given a certain initial state and a zero input-signal. In this case it can be stated as

$$L_o(x^*, t) = \frac{1}{2} \sum_{k=t}^{N} ||y_k||^2, \quad x_t = x^*, \quad u \equiv 0. \quad (2)$$

That is, the amount of energy an initial state x^* at time t induces in the output-signal over the time-interval $[t, N]$. The concept of these energy function is illustrated in Fig. 1. The usefulness of these functions for model reduction is clear. If a large amount of energy is required to reach a certain state and if the same state yields a small output energy, this state is unimportant for the input-output behaviour of the system.

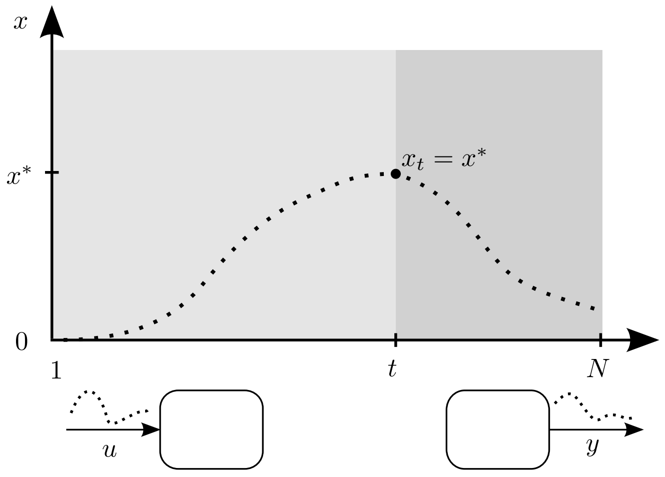

Fig. 1. Visualization of the energy functions. The left part illustrates the minimal input energy required to reach x^* at time t. In the right part the control signal is zero and the initial state x^* yields the mentioned output energy.

The energy functions can be determined trough the following Lyapunov equations

$$\begin{aligned} P_{k+1} &= A_k P_k A_k^T + B_k B_k^T, \qquad & k \in [1, N] \\ Q_k &= A_k^T Q_{k+1} A_k + C_k^T C_k, \qquad & k \in [1, N] \end{aligned}$$

with the boundary conditions $P_1 = 0$ and $Q_{N+1} = 0$. The matrices P_k and Q_k are commonly called the controllability Gramian and observability Gramian, respectively. Further, the solutions to (1) and (2) can be written as the quadratic forms

$$L_c(x^*, t) = \frac{1}{2} x^{*T} P_t^{-1} x^* \qquad L_o(x^*, t) = \frac{1}{2} x^{*T} Q_t x^*.$$

The Gramians P_k and Q_k, and their analogues for other system classes, are central to many model reduction methods. They show how strongly states are connected to the input and output and thereby supplies essential information of which state-subspace is of most significance.

IV. Method description

The method presented here is the discrete-time counterpart of the *average Gramian method* first presented in [11]. The system class is the general nonlinear discrete-time system

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k, u_k) \end{aligned} \quad (3)$$

where $u_k \in \mathbf{R}^l$, $x_k \in \mathbf{R}^n$ and $y_k \in \mathbf{R}^m$. The following sections explain the main steps involved in the method.

A. Linearization along trajectory

The first step is to choose a so called training input-signal. This is an important step of the method of which the performance is highly dependent. As a general rule the input should be chosen to obey physical restrictions on the signal and to excite all relevant dynamics. To find such a signal might be a challenging task. However, one could also see this as an advantage of the method. If the reduced model is only going to be used for some restricted purposes, the model could probably be reduced to a greater extent. Through the choice of training input the user can show what behaviour is relevant for the reduced system to reproduce. Hence, the signal should be chosen corresponding to realistic usage of the model.

The system is then linearized along the state-trajectory the training input gave rise to. The result is a time-varying linear system

$$\begin{aligned} \Delta x_{k+1} &= A_k \Delta x_k + B_k \Delta u_k \\ \Delta y_k &= C_k \Delta x_k + D_k \Delta u_k \end{aligned} \qquad k \in [1, N]$$

where Δu, Δx and Δy denote deviations from the nominal trajectories. Further, A_k, B_k, C_k and D_k are time-varying matrices of appropriate size.

B. Compute the time-varying Gramians

Similar to balanced truncation the method uses the notion of Gramians. As mentioned, for time-varying systems the controllability Gramian can be computed according to the difference equation

$$P_{k+1} = A_k P_k A_k^T + B_k B_k^T, \quad k \in [1, N] \tag{4}$$

with $P_1 = 0$. Similarly, the observability Gramian is determined by

$$Q_k = A_k^T Q_{k+1} A_k + C_k^T C_k, \quad k \in [1, N] \tag{5}$$

with the boundary condition $Q_{N+1} = 0$.

C. Determine the average Gramians

The Gramians P_k and Q_k contain local information along the trajectory of how strongly states are connected to the input and output, respectively. In order to extract more overall information about which the important states are, one could use the *average Gramians*

$$\bar{P} = \frac{1}{N}\sum_{k=1}^{N} P_k \qquad \bar{Q} = \frac{1}{N}\sum_{k=1}^{N} Q_k. \tag{6}$$

These time-invariant matrices contain information of how strongly the states are connected to the input and output on average over the training trajectory. For example, if a certain linear state combination is unobservable from the output in all points of the trajectory, it will be revealed in $\bar{Q}$. Further, a rank deficiency of the matrix $\bar{P}\bar{Q}$ indicates that some states are obsolete and can be truncated from the model without changing the input-output relationship.

D. Find balancing coordinate-change

This step is performed to extract the relevant state subspace using the information gathered in the average Gramians. The chosen approach treats $\bar{P}$ and $\bar{Q}$ as if they belonged to a linear time-invariant system. By following the standard balanced truncation procedure for linear systems, a coordinate change $z = Tx$ can be found, see [12], such that the average Gramians become equal and diagonal with decreasing diagonal elements.

$$T\bar{P}T^T = T^{-T}\bar{Q}T^{-1} = \bar{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \tag{7}$$

The diagonal elements $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$ corresponds to the Hankel singular values in balanced truncation of linear systems, where they show how important states are for the input-output relationship. Although no error-bound is available, in contrast to the linear case, these values will be used to determine which model order to choose for the reduced system.

E. Truncate states

Truncating states corresponding to relatively small singular values and keeping $\hat{n}$ states is equivalent to removing rows and columns in T and T^{-1}, respectively.

$$\begin{aligned} T \in \mathbf{R}^{nxn} \quad &\Rightarrow \quad T_l \in \mathbf{R}^{\hat{n}xn} \\ T^{-1} \in \mathbf{R}^{nxn} \quad &\Rightarrow \quad T_r \in \mathbf{R}^{nx\hat{n}} \end{aligned} \tag{8}$$

Applying the truncated coordinate change to the original system formulation in (3) gives rise to the reduced order system

$$\begin{aligned} \hat{z}_{k+1} &= T_l f(T_r \hat{z}_k, u_k) \\ y_k &= g(T_r \hat{z}_k, u_k) \end{aligned} \tag{9}$$

where $\hat{z} \in \mathbf{R}^{\hat{n}}$. Deriving the reduced system through symbolical substitution in (9) is generally not an attractive option. Commonly, the original set of equations is sparse, i.e. all state equations do not involve all states. The sparsity is lost with a dense coordinate change and truncation of states. Therefore, the total computation time is not necessarily reduced for the right-hand-side functions, which can e.g. be seen in [15]. How to derive analytical expressions for the reduced system in (9) is highly dependent on the format the original model is implemented in. This matter will be further discussed in Section VI.

V. Model description

The model considered in this work consists of software used for online air-path dynamics estimation in current production cars. In particular, the model estimates the air charge in a spark ignition engine, i.e., the amount of air the cylinder is loaded with when the inlet valve closes. The amount of fuel to inject is then determined from this value in order to achieve a certain air-fuel ratio. The exhaust treatment system requires a precise air-fuel ratio, therefore high fidelity of the estimation is crucial since a mismatch would yield suboptimal performance. For further information see [13].

The model is implemented in *MATLAB®/Simulink®* and can be compiled through *Real-Time Workshop®*. The resulting binary runs in real-time in the Engine Control Unit(ECU), shown in Fig. 2.

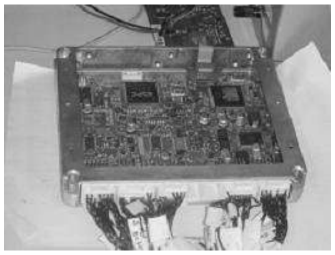

Fig. 2. Engine Control Unit with the embedded controller software.

The model is devised for real-time purposes with limited hardware resources. For example, only discrete-time components are present and fixed-point arithmetic is used. The most common arrangement, used in automotive industry, to achieve high performance for a wide area of operating conditions is to divide the problem into regions and perform local tuning of variables. The model therefore contains a large amount of logical branches and look-up tables.

The model is a so called mean-value model, see [14], but details concerning model implementation are proprietary information and are therefore not disclosed.

The model estimates several variables using various measurements. As a proof of concept, only one input-output pair is treated here. The chosen input-signal is the throttle angle measured in degrees and the output-signal is the air charge given in percentage, as illustrated in Fig. 3.

Model reduction of the Simulink control algorithm implementation would ideally yield a binary file that runs faster and uses less memory. Hence, smaller hardware resources would be required yielding a lower controller hardware cost. In addition, formal validation and verification of the controller in combination with the engine would be facilitated. Moreover, the original model structure is hard to overview and it might be easier to understand the reduced model's behaviour and visualize its components.

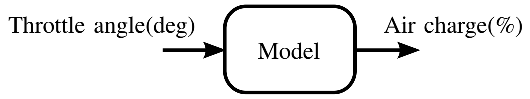

Fig. 3. Chosen input-output pair for model reduction.

VI. MODEL REDUCTION APPLIED TO THE CONTROLLER MODEL

With some slight modification the controller software fits within the model class of (3). The state-vector x represents the ECU memory used to store data between samples. The controller turned out to contain six states. The input-signal u is the throttle angle and the output y is the air charge. Simulink provides tools for extraction of simulation data and linearizations. With this data available the model can be reduced following the procedure described in Section IV.

As stated in Section IV-A, the first step was to choose a training trajectory. Here, a simple ramp-like throttle opening profile was chosen, see Fig. 4, starting from closed throttle and then linearly increasing until fully open. Of course, different choices are possible depending of the purpose of the reduced model. Notice the nonlinear effect in Fig. 4, the output has almost settled after half a second while the input continue to increase half a second more. This is due to that the flow over the throttle is much more sensitive to an increase in throttle angle when it is almost closed, see e.g. [13].

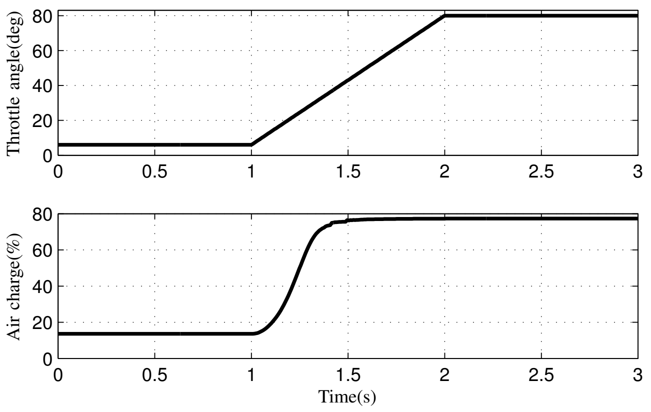

Fig. 4. The ramp-like training input-signal(Throttle angle) and the corresponding output-signal(Air charge).

With the training input-signal chosen, the model was linearized around the corresponding state-space trajectory. Doing so gives rise to the linearizations (A_k,B_k,C_k,D_k), for this model the D_k matrix is zero for all k. Following the procedure, the time-varying Gramians P_k and Q_k were computed through (4) and (5). The average Gramians $\bar{P}$ and $\bar{Q}$ were then obtained by (6). The singular values in (7) are shown in Fig. 5, three of the six values turned out to be exactly zero and are not plotted. The relative size of these values indicate the importance of the states. Here, there is a factor 10^4 difference between the largest and second largest value. In model reduction of linear systems one could easily reduce to one state. Despite the absence of a formal error-bound this will be done for the nonlinear system as well. Calculating the balancing coordinate-change and truncating to one state according to (8) yield the two matrices T_l and T_r.

The next and final step of the method consists of applying the coordinate-change to the original nonlinear system. In this case, the functions f and g were not explicitly available but embedded in the Simulink program. Hence, symbolical manipulation of the functions is not a straight forward process. One could consider using the piece-wise affine approach described in [10], where f and g are reconstructed through a weighted sum of linearization points along a training trajectory. However, using linearizations as basis

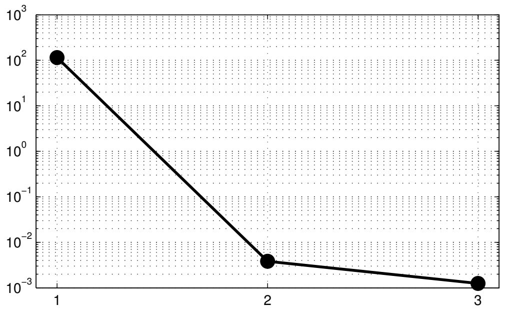

Fig. 5. The three largest Hankel singular values, notice the 10^4 drop between the first and second value.

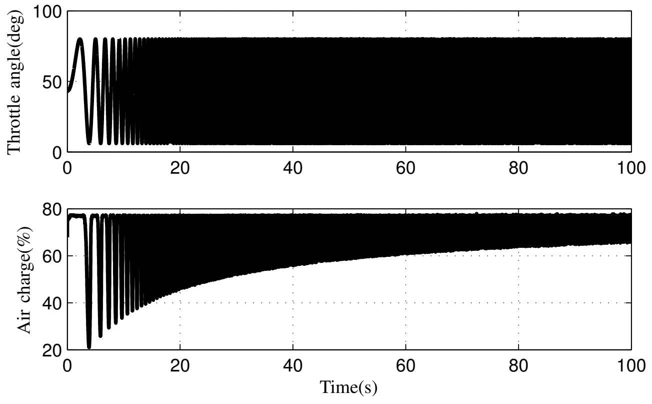

Fig. 6. The chirp input-signal used for map generation and the corresponding output-signal.

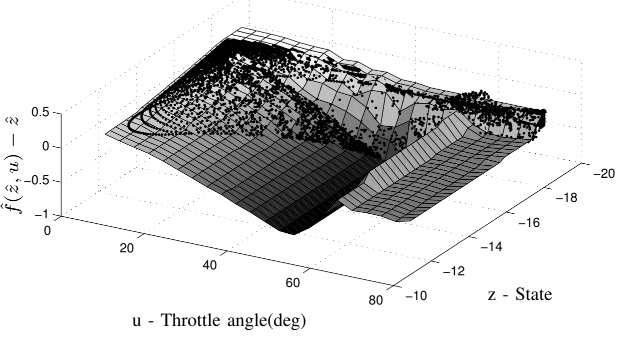

Fig. 7. The data-points and generated surface representing the right-hand-side function $\hat{f}(\hat{z}, u)$, here visualized in incremental form.

functions is not tractable in this case. Due to the logical branches and non-smooth look-up tables the function f becomes "noisy". Therefore, a local linearization provides inadequate information about the neighbouring state space and an unreasonably large number of linearizations would be required.

By reducing to one state, the right-hand-side function in (9)

$$\hat{z}_{k+1} = \hat{f}(\hat{z}_k, u_k) = T_l f(T_r \hat{z}_k, u_k)$$

becomes a two-dimensional map $\hat{f} : \mathbf{R}^2 \to \mathbf{R}$. An alternative approach, used below, is to let the map $\hat{f}$ be generated from simulation data. State-trajectories induced by some input-signal could be projected with T_r and provide values for $\hat{z}_k$. Value triplets of $\hat{z}_{k+1}$, $\hat{z}_k$ and u_k supply point-wise information of the map. A drawback is that an input-signal rich enough to make sufficient state-space coverage in $(\hat{z}_k, u_k)$ is needed and the choice can be nontrivial. The difficulties relate to the choice of training input in the first step of the model reduction procedure and although the purpose is different the same input-signal could be used. A chirp signal with maximal amplitude, shown in Fig. 6, was chosen as the exciting input-signal.

For simulation of the reduced model, point-wise information of the map is not sufficient, an analytical expression of $\hat{f}$ is needed. Trough local averaging followed by linear interpolation and extrapolation, a piece-wise affine surface was generated to approximate the data-cloud, see Fig. 7. To clarify the structure of the map the incremental form $\hat{f}(\hat{z}, u) - \hat{z}$ instead of $\hat{f}(\hat{z}, u)$ was used. As the map is piece-wise affine, so is the reduced system. One can notice the rough areas in the upper part of the map, their origin is most probably the non-smooth look-up tables present in the model.

In general the same procedure would be applied to the output function g. Here, the output function $g(T_r \hat{z}_k, u_k) = \hat{g}(\hat{z})$ turned out to be nearly affine and a least-square fit showed to be an adequate approximation.

With analytical expressions for $\hat{f}$ and $\hat{g}$ at hand, simulation of the reduced system is possible. Fig. 8 shows a validation result where the original and reduced model were simulated with a ramp-like opening and closing of the throttle. The initial value of $\hat{z}$ was arbitrarily set, hence the initial mismatch between the two output-signals. In contrast, after 0.5 seconds the worst case error was less than 1.2% air charge.

In Fig. 9 the validation trajectory together with the data-points used for map generation are shown. The smaller dots are from the chirp-signal simulation in Fig. 6 and the connected dots from the validation simulation in Fig. 8. As can be seen, the validation trajectory is covered by the look-up table. For a different validation scenario this might not necessarily be the case. However, if necessary, a richer input-signal could be designed to overspread a larger area and a more general map could be generated. Due to the low dimensionality of this case, the coverage could be graphically examined. However, in a more general setting with higher dimensionality it may be hard to verify.

A compact Simulink® implementation of the reduced model is depicted in Fig. 10. The model runs more than 100 times faster than the full original model. However, to carry out a fair comparison, the reduced model should also be equipped with the same amount of input and output-signals.

VII. Conclusions

A novel model reduction method of nonlinear discrete-time systems has been presented together with its applicability to an engine controller used in current production cars. The number of states was reduced from six to one

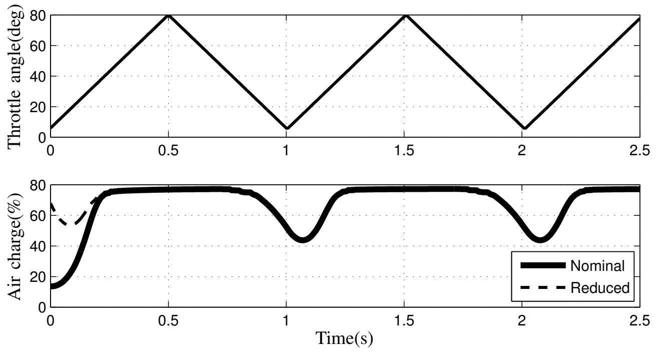

Fig. 8. Validation result of the reduced model. The lower plot shows the output-signal of the original model together with the reduced one. The initial output error is due to an unmatched initial condition.

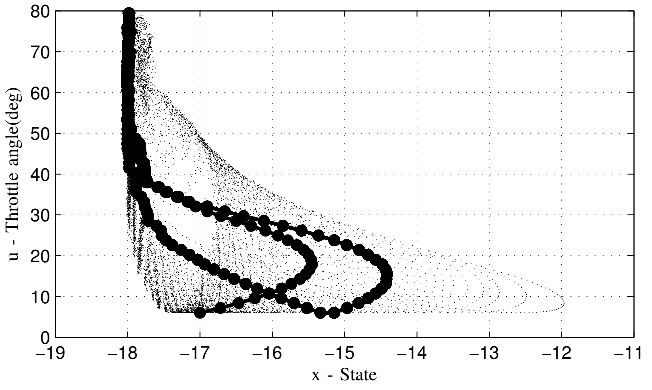

Fig. 9. The validation trajectory(the connected dots) is covered by the span of data-points(the smaller dots) used for the look-up table $\hat{f}(\hat{z}_k, u_k)$.

and the resulting nonlinear piece-wise affine system showed a 100-fold improved simulation speed, with little loss of accuracy. Despite the initial model's complexity in terms of look-up tables and logical switches the method demonstrated its applicability. The method also provided information for analysis of overall controller behaviour, such as the software visualisation in Fig. 7.

Further research is required regarding the extension to models with several inputs and outputs. In particular, derivation of the map in state-input space in a robust manner.

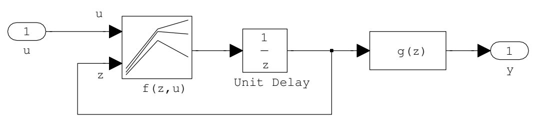

Fig. 10. Simulink implementation of the reduced model.

REFERENCES

[1] B. Moore, "Principal component analysis in linear systems: controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. 26, pp. 17–32, February 1981.

[2] J. Scherpen and K. Fujimoto, "Nonlinear balanced realization based on singular value analysis of Hankel operators," in *Proceedings of the 42nd IEEE Conference on Decision & Control*, (Maui, USA), pp. 6072–6077, IEEE, December 2003.

[3] J. Scherpen and K. Fujimoto, "Balancing and model reduction for discrete-time nonlinear systems based on Hankel singular value analysis," in *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems*, (Leuven, Belgium), July 2004.

[4] J. Broz, C. Clauss, T. Halfmann, P. Lang, R. Martin, and P. Schwarz, "Automated symbolic model reduction for mechatronical systems," in *Proc. of 2006 IEEE International Symposium on Computer-Aided Control Systems Design*, (Munich, Germany), pp. 408–415, IEEE, Oct. 2006.

[5] H. K. Khalil, *Nonlinear systems*. Upper Saddle River, New Jersey: Prentice Hall, third ed., 2002.

[6] M. Loève, "Fonctions aléatoires de second ordre," *Comptes Rendus Acad. Sci. Paris*, pp. 220–469, 1945.

[7] K. Karhunen, "Zur spektraltheorie stochastischer prozesse," *Ann. Acad. Sci. Fennicae*, vol. Ser.A1, no. 34, 1946.

[8] P. Astrid, *Reduction of process simulation models: a proper orthogonal decomposition approach*. PhD thesis, Technische Universiteit Eindhoven, Netherlands, 2004.

[9] S. Lall, J. Marsden, and S. Glavaski, "A subspace approach to balanced truncation for model reduction of nonlinear control systems," *International Journal of Robust and Nonlinear Control*, vol. 12, pp. 519–535, 2002.

[10] D. Vasilyev, M. J. Rewieński, and J. White, "Macromodel generation for biomems components using a stabilized balanced truncation plus trajectory piecewise-linear approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, Feb. 2006.

[11] O. Nilsson, "Modeling and model reduction in automotive systems," Licentiate Thesis ISRN LUTFD2/TFRT--3242--SE, Department of Automatic Control, Lund University, Sweden, Dec. 2006.

[12] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Upper Saddle River, New Jersey: Prentice Hall, 1998.

[13] J. Heywood, *Internal Combustion Engine Fundamentals*. New York: McGraw-Hill, 1988.

[14] E. Hendricks, A. Chevalier, M. Jensen, S. Sorenson, D. Trumpy, and J. Asik, "Modelling of the intake manifold filling dynamics," No. 960037, SAE Technical Paper, 1996.

[15] Z. Liu and J. Wagner, "Nonlinear model reduction for dynamic and automotive system descriptions," *Journal of Dynamic Systems, Measurement, and Control*, vol. 124, pp. 637–647, December 2002.