Nonlinear Balanced Model Residualization via Neural Networks

Juergen Hahn, Sebastien Lextrait, and Thomas F. Edgar
Dept. of Chemical Engineering, The University of Texas at Austin, Austin, TX 78712

A variety of different model reduction techniques have been proposed for linear and nonlinear systems (Fortuna et al., 1992; Skogestad and Postlethwaite, 1997). One popular method for linear systems is balancing (Moore, 1981), where a transformation balances the observability and controllability gramians in order to determine which states have the greatest contribution to the input-output behavior. With the use of empirical gramians (Lall et al., 1999), this method has been extended to nonlinear systems such that the empirical gramians can be balanced by the same procedures developed for linear systems. A Galerkin projection is made onto the states corresponding to the largest singular values of the balanced gramians for the region of interest in state space. The reduction of the balanced system can be performed by either a truncation or residualization methods, where truncation is simpler to implement, but residualization usually results in a closer approximation (Hahn and Edgar, 2002). Since residualization results in a differential-algebraic equation (DAE) system and not all simulation packages are equipped with a robust DAE solver, it would be desirable to approximate this reduced model further in order to be able to make it simpler to compute. The next section describes how this reduction can be performed by the use of an artificial neural net. It is then shown that the reduced model can give an excellent approximation to the full-order one, depending on the choice of an optimum training set for the neural network.

The advantage that this hybrid approach has over replacing the whole model with a neural network is that due to balancing the structure of the model, it is preserved and only those states that contribute little to the input-output behavior of the system are replaced by the neural net. It is shown that this method results in reduced models that give a better approximation of the input-output behavior than can be achieved by nonlinear balanced truncation. While this approach results in reduced models that are almost as accurate in describing the input-output behavior of a model obtained from nonlinear balanced residualization, it has the advantage that the model obtained from residualization via neural networks does not require a DAE solver for its solution.

Nonlinear Model Reduction by Residualization

For the nonlinear system of the form given by Eq. 1, it is possible to find a linear state transformation (Eq. 2) such that the resulting system is approximately in balanced form (Hahn and Edgar, 2002)

$$\dot{x}(t) = f[x(t)] + g[x(t)]u(t) \tag{1}$$
$$y(t) = h[x(t)]$$

$$\bar{x} = Tx \tag{2}$$

In Eqs. 1 and 2 x refers to the original state variable, $\bar{x}$ is the state vector of the transformed system, and T is the transformation that balances the empirical gramians. These empirical gramians are an extension of the gramians of a linear system to stable control-affine nonlinear systems. They contain information about the input-to-state and the state-to-output behavior of the system under investigation. For a more detailed definition of the empirical gramians see Hahn and Edgar (2002) and Lall et al. (1999). Reduction of this balanced model by residualization given by Eq. 3 is based upon the idea that the derivatives of the less important states are approximated by zero while the rest of the system is unchanged. This results in the reduced system

$$\dot{\bar{x}}(t) = P_1Tf\left[T^{-1}\bar{x}(t)\right] + P_1Tg\left[T^{-1}\bar{x}(t)\right]u(t) \tag{3a}$$

$$0 = P_2Tf\left[T^{-1}\bar{x}(t)\right] + P_2Tg\left[T^{-1}\bar{x}(t)\right]u(t) \tag{3b}$$

$$y(t) = h\left(T^{-1}\bar{x}(t)\right)$$

where

$$\bar{x} = \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \end{pmatrix},\ P_1 = [I \quad 0],\ P_2 = [0 \quad I]$$

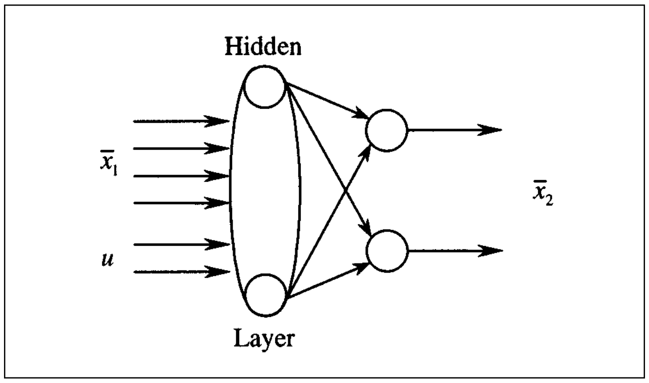

Figure 1. Neural network structure for nonlinear balanced residualization.

Here, P_1 is the projection with rank of the reduced-order system, and the rank of P_2 is equal to the number of states that were reduced. From the original state vector, $\bar{x}_1$ contains the states of the reduced system which are retained, and $\bar{x}_2$ represents the states that are reduced in the process. One of the main differences between residualization for linear and nonlinear systems is that in the nonlinear case we cannot in general solve explicitly for the states $\bar{x}_2$ in Eq. 3b. Instead, the original system of ODEs reduces to a DAE system that contains fewer differential equations than the full-order one (Hahn and Edgar, 2002).

Further reduction of the residualized system using neural networks

One way of obtaining an approximate solution to a DAE system is to replace the algebraic equations by a neural network that can be solved in a feedforward manner (see Figure 1). This approach is attractive, because the most important components of the system are contained in the remaining states and the neural network only corrects the system for the reduced states. This model reduction procedure preserves most of the structure of the original system, while reducing part of the model by replacing it with a neural network. The resulting system is given by Eq. 4, where the "sim" command refers to the outputs computed by the neural network for a set of inputs $\bar{x}_1$ and u

$$\begin{aligned}\dot{\bar{x}}_1(t) &= P_1Tf\left[T^{-1}\bar{x}(t)\right] + P_1Tg\left[T^{-1}\bar{x}(t)\right]u(t) \qquad (4)\\ \bar{x}_2(t) &= \text{sim}\left[\text{NeuralNet},\bar{x}_1(t),u(t)\right]\\ y(t) &= h\left(T^{-1}\bar{x}(t)\right)\end{aligned}$$

where

$$\bar{x} = \begin{pmatrix}\bar{x}_1\\ \bar{x}_2\end{pmatrix}, \; P_1 = [I \quad 0]$$

For reasons of simplicity, only one hidden layer in the neural net is used, as shown in Figure 1. The transfer functions are chosen to be hyperbolic tangents for the hidden layer and identity functions for the output layer.

The closeness of the approximation will largely depend on the training set for the neural network. Different methods for obtaining training sets for this specific application are illustrated in the following sections.

Training based on solving the algebraic equations on a uniform grid

One possibility for the training data is to solve only the algebraic equations given by Eq. 5 for $\bar{x}_2$ and find a best fit to these equations (Jang et al., 1997). The neural network can then be trained with the inputs of $\bar{x}_1$ and u and match $\bar{x}_2$ as a target, as shown in Figure 1

$$0 = P_2Tf\left(T^{-1}\bar{x}(t)\right) + P_2Tg\left(T^{-1}\bar{x}(t)\right)u(t) \qquad (5)$$

This should be done on a uniform grid of the space that is spanned by the inputs (u) and the remaining states ($\bar{x}_1$). This approach is easy to implement, since it involves no integration. Only the system of equations given by Eq. 5 has to be solved for the unknown $\bar{x}_2$. One possible implementation of this is to set it up as an optimization problem, where the performance index is given by the sum of the squares of the righthand side of Eq. 5. The goal is to find values for $\bar{x}_2$ that minimize this performance index. The data are then used to train a neural network that receives the values of $\bar{x}_1$ and u as inputs and has target values $\bar{x}_2$. The quality of the approximation for a correctly trained neural network will lie between the balanced truncation and balanced residualization.

Training based on the system response

An alternative approach is to generate data based upon the behavior of the real system, because it is desirable to collect a set that does not include any physically unrealistic points. This is done by exciting the system using the inputs and sampling data along the system trajectories. The algebraic equations can then be solved for points along these trajectories.

In order to collect a good training set, the following procedure should be applied. The system is excited with different magnitudes of all possible combinations of the inputs around their operating region. Then data along the trajectories is collected. Once the system is close to its new steady state, the inputs are reset to their original values. Due to this, the training data will include some trajectories that move the system away from steady state, as well as some that bring it back to the original operating point. After each control sequence, the system is reset to its initial steady state. The sequences that move the system to other equilibrium points can be eliminated in a post processing step, since these trajectories are usually outside of the region where the process will be controlled.

It should be noted that the completeness of the training set is very important, because the neural net should not be used for extrapolation. The following example illustrates the importance of choosing a well designed training set.

Example 1: Comparison of Three Different Neural Net Approaches. Consider a system of two CSTRs in series with one reaction $A \rightarrow B$. Each reactor has a mass, component, and energy balance, resulting in six nonlinear differential

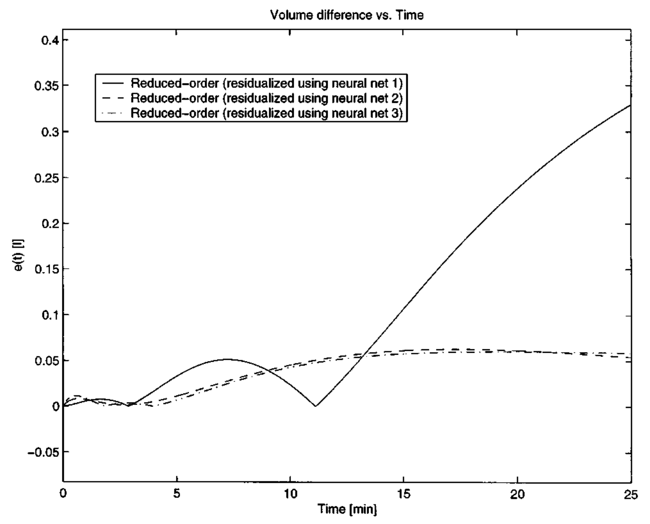

Figure 2. Difference for the volume of reactor 2 vs. time.

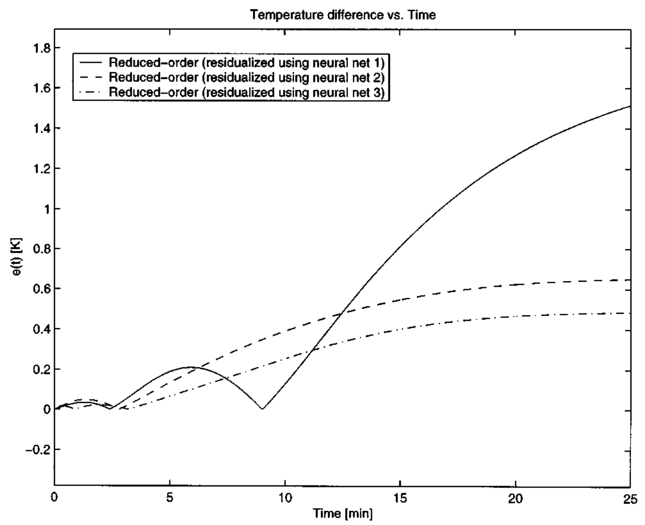

Figure 3. Difference for the temperature of reactor 2 vs. time.

equations. The volume and temperature of the second reactor can be measured. The valve position at the outlet of the second reactor, as well as the heat transferred to the first reactor, can be controlled. This results in a system with six states, two inputs and two outputs. The balanced empirical gramians suggest that a reduced system with four states will describe the model appropriately, because there is a difference of several orders of magnitude between the 4th and 5th Hankel singular value of the balanced empirical gramians. The dynamic response results from an 8% decrease in the valve position at the outlet of the second reactor.

In this example the difference in the behavior of three models of the form of Eq. 4 is compared. All graphs result from a balanced residualization, which is based upon empirical gramians. In all cases the algebraic equations are replaced by a feedforward neural net that consists of an input layer with six inputs (four states and two controls); a hidden layer with five neurons and two outputs. The only difference is the training set that is collected for the neural net. The training data set for the first approach was collected by solving the algebraic equations for all possible combinations of a perturbation of the $\bar{x}_1$ and u by 5%, up to a total of 10%, around their operating region. This resulted in a training set of $5^6 = 15{,}625$ input and output combinations. The error of the response generated by a system involving a neural network trained with this data set is given by the solid line in Figures 2 and 3.

The second and third set are generated by exciting the system with all possible combinations of the inputs by 0%, 5%, and 10% around their operating point. Data are collected along the trajectory and the sets consist of 15,200 points each. The difference between the second and third training set is that the second one receives its target values by solving the set of algebraic equations (Eq. 5) at each point, whereas the third target consists of the reduced states themselves. The error of a system that uses the second training set is represented by the dashed line in Figures 2 and 3 and the dash dotted line corresponds to the error produced by a system based upon the third training set.

These figures compare the absolute value of the difference between the original and the reduced model of all three systems, because it is hard to distinguish among the model responses. This difference is given by Eq. 6, where the subscript i stands for the ith state and the one norm is used

$$e_i(t) = \| x(t) - T^{-1}\bar{x}(t) \|_i \tag{6}$$

The reason that the solid line in Figures 2 and 3 exhibits nonsmooth behavior is that the trajectory of the reduced system at these points crosses over the full-order one. Although the difference between the reduced system and the original model is zero at these points, it is not an excellent approximation at these points since the derivatives do not match.

As can be seen from Figures 2 and 3, the response based upon the first training set (neural net 1) results in the largest error. This is due to the fact that the majority of the data points in this training set correspond to points that the system cannot be driven to given the available control action. Both systems that are based upon training sets that are collected along system trajectories (neural nets 2 and 3) result in smaller errors than a model based upon the first set. It can be concluded that the response based upon the third training set (neural net 3) performed better than the one based upon the second one. This is due to the fact that the algebraic equations that are used for the second set only approximate the real system behavior, whereas the third one is based upon the trajectories themselves and therefore better approximates the behavior of the full-order model.

Application to not completely observable/controllable models

Most complex models contain states that are not observable or controllable. Therefore, these states do not contribute to the input-output behavior of the system. The balancing procedure identifies the unobservable/uncontrollable parts of the model, so that they can be reduced. When a balanced residualization procedure is used for the reduction step, these states are still included in the reduced model in the form of algebraic equations or a neural network representation, as shown in this article. However, since these states do not contribute to the input-output behavior, they can be set to their steady-state values and the residualization is based on less important, but observable/controllable states. This results in a system that is given by Eq. 7

$$\begin{aligned}\dot{\bar{x}}_1(t) &= P_1Tf\left[T^{-1}\bar{x}(t)\right] + P_1Tg\left[T^{-1}\bar{x}(t)\right]u(t)\\ \bar{x}_2(t) &= \mathrm{sim}\left[\mathrm{NeuralNet},\ \bar{x}_1(t),\ u(t)\right]\\ \bar{x}_3(t) &= \bar{x}_{3,ss}(0)\\ y(t) &= h\left[T^{-1}\bar{x}(t)\right]\end{aligned} \tag{7}$$

where

$$\bar{x} = \begin{pmatrix}\bar{x}_1\\ \bar{x}_2\\ \bar{x}_3\end{pmatrix},\ P_1 = [I \quad 0]$$

In Eq. 7, the $\bar{x}_1$ are the states of the reduced system that are described by differential equations. The state vector $\bar{x}_2$ consists of states of the reduced system that are described by the neural network model. The vector $\bar{x}_3$ contains all states that were truncated during the reduction procedure. $\bar{x}_3$ should at least contain all states that are either unobservable or uncontrollable. Furthermore, it can include some less important states that are controllable and observable.

Example 2: Model Reduction of a Binary Distillation Column. Consider a distillation column with 30 trays for the separation of a binary mixture studied by Horton et al. (1991). The column has 32 states and is assumed to have a constant relative volatility of 1.6, and symmetric product compositions. The feedstream is introduced at the middle of the column on stage 17 and has a composition of $x_F = 0.5$. Distillate and bottoms purities are $x_D = 0.935$ and $x_B = 0.065$, respectively. The reflux ratio which can be controlled is set to 3.0, and the purity of the distillate is measured.

In this example two models are compared. One was produced using the full-order model with 32 states and the other results from a reduced model, where the first three states are described by differential equations. Furthermore, this model includes a neural network with two additional outputs and all the remaining states are set to their steady-state value. The training set is generated by the trajectory approach involving the actual values of the trajectories. Figure 4 compares the two graphs resulting from a 10% change in the reflux ratio. It can be concluded that the residualized model with three states provides an excellent approximation to the full-order one.

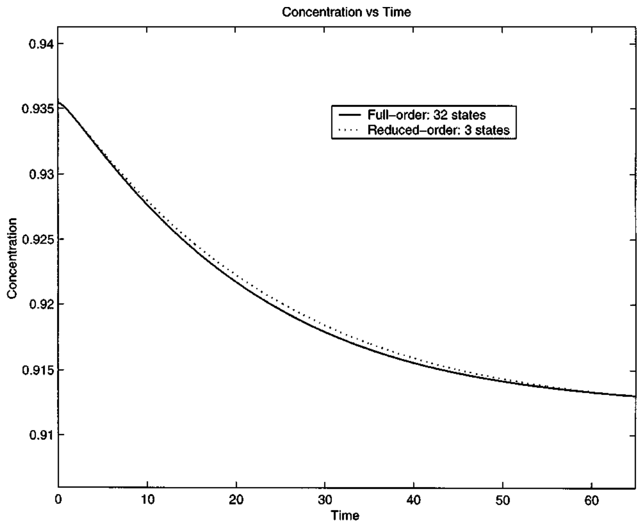

Figure 4. Comparison of model behavior for a 10% change in the reflux ratio.

Conclusions

This article presents a reduction procedure for nonlinear models. The reduction scheme is an extension of balancing to nonlinear systems, using the concept of empirical gramians. These gramians are balanced by simple matrix computations, and a projection is used to map the nonlinear system to the reduced states. Balanced truncation, as well as residualization, can be used as methods for the reduction step. The former will lead to a reduced system of ODEs, whereas the latter will result in a DAE system, which has the same steady-state behavior as the original model. The algebraic equations of the DAE system can be replaced with a neural network that exhibits similar behavior, but is computed in a feedforward manner. This DAE approximation is well-suited for models that were derived using nonlinear balancing. The approximation is appropriate, because the most important components of the system are kept by the remaining states and the neural network only corrects the system for the reduced states. This procedure results in a good approximation, while, at the same time, yields models that are easier to solve than those obtained from nonlinear balanced residualization alone.

Three different training sets were used to train the neural net. The first one was based upon solving the algebraic equations of the DAE system at points on a grid in state space. The second training set was based upon the solution of the algebraic equations along the trajectories of the system response. The third set contained the same inputs as the second one, but its target values were given by the reduced states along the trajectories. It can be concluded from the data that the two training sets generated by the system response give a better approximation, because they contain no data points that are physically unrealistic to generate. Of the two methods that used the trajectory approach, the one that utilized the response data itself to train a neural network performed better than the one that solved the algebraic equations for the given data points. This is the case, because it approximates the behavior of the real system, whereas the other method approximates the residualized system behavior, which is an approximation to the real system itself.

Literature Cited

Fortuna, L., G. Nunnari, and A. Gallo, *Model Order Reduction Techniques with Applications in Electrical Engineering*, Springer-Verlag, London (1992).

Hahn, J., and T. F. Edgar, "An Improved Approach to the Reduction of Nonlinear Models Using Balancing of Empirical Gramians," *Computers Chem. Eng.*, in press (2002).

Horton, R. R., B. W. Bequette, and T. F. Edgar, "Improvements in Dynamic Compartmental Modeling for Distillation," *Computers Chem. Eng.*, **15**, 197 (1991).

Jang, J. S. R., C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, Upper Saddle River, NJ (1997).

Lall, S., J. E. Marsden, and S. Glavaski, "Empirical Model Reduction of Controlled Nonlinear Systems," *IFAC World Cong.*, Beijing, China (1999).

Moore, B. C., "Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction," *IEEE Trans. on Automatic Control*, **26**, 17 (1981).

Skogestad, S., and I. Postlethwaite, *Multivariable Feedback Control*, Wiley, New York (1997).

Manuscript received Dec. 11, 2000, and revision received Jan. 2, 2002.