Chapter 23
High-Fidelity Simulation Surrogate Models for Systems Engineering

Alex Van der Velden

Abstract In this paper, we will present a method to approximate the behavior of high-fidelity simulation models with surrogates that are constructed from high-fidelity simulation results. High-fidelity N-code (FEA, CFD, logical) cosimulations can take as much as 1 h CPU-time for every real-time second of behavior prediction (Van der Velden et al. (2012) Probabilistic certificate of correctness for cyber physical systems. In: ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 12–15, Chicago, DETC2012-70135). We propose to speed up this process by four orders of magnitude, so that it is fast enough to be used in real-time applications, interactive design, multidisciplinary optimization, and verification of cyber–physical systems. These surrogate models can then be wrapped as a Functional Mock-up Unit (Functional Mock-up Interface: http://www.functional-mockup-interface.org/index.html) and deployed in a system simulation model such as Modelica® as well as in embedded hardware.

The present method converts time-series field data (as a function of design variables) to nonlinear ordinary differential equations for behavior at specific sensor locations using arbitrary surrogate (or meta) models (e.g., radial basis functions). The flexibility to choose any type of surrogate modeling technique maximizes the chance a highly predictive model can be found. The nonlinear ordinary differential equations are then solved efficiently with a numerical integration scheme. We verified the present method by comparing the analytical solution of the near chaotic motion of a double pendulum for given initial conditions with those of a surrogate model created from Modelica samples from different initial conditions.

The present method complements two other approaches to accelerate systems simulation. In the case of parameter estimation, the differential equations are known, but some of equation constants are not. In the case of model reduction, the full physical model is known a priori, but the complexity of the model is reduced. In the present method, the important (in and external) state variables

A. Van der Velden (✉)
SIMULIA, Dassault Systemes, Johnston, USA
e-mail: alex.vandervelden@3ds.com

A.M. Madni et al. (eds.), *Disciplinary Convergence in Systems Engineering Research*, https://doi.org/10.1007/978-3-319-62217-0_23

need to be known, but not the model form. The approach is limited by ability of the user-selected surrogate modeling technique to capture complex interactions.

The first example shows the effect of the activation of a controller to suppress airfoil flutter. The surrogate model was based on time-series responses for a 4-code Abaqus™ standard FEA, Abaqus CFD, control-build and Dymola cosimulation for various controller frequencies. Each of these cosimulating models solves either partial differential equations or ordinary differential equations that are coupled by a cosimulation engine.

The second example shows the dynamic motion and internal forces of a full 3D automotive vehicle as a function of arbitrary road surface boundary conditions. The surrogate model was based on time-series of 80 states computed for a specific virtual test track by the Abaqus™ explicit solver that took 24 h on 32 CPUs to complete. Subsequently, the fast surrogate model was successfully used to predict structural durability for arbitrary road conditions.

Keywords Approximation • Surrogate model • Meta-model • FEA • CFD • Cosimulation • Systems • Flutter • NVH

23.1 Introduction

N-code high-fidelity (FEA, CFD, logical) cosimulations can take as much as 1 h CPU time for every real-time second of predicted behavior. An example of an N-code cosimulation is an active control system based on wing trailing edge flap deflected by a servo actuator controlled by an electronic control unit [1]. Here, the Abaqus™ /FEA code is used for the structural analysis, the Abaqus™ /CFD code is used for the aeroloads, while the actuators are modeled with Dymola™ and the digital controller is modeled with ControlBuild.™ Each software code has its own specialized solver (ODE, PDE), while a cosimulation engine controls the time integration between the solvers.

Even though the high-fidelity cosimulation can predict highly accurate and highly refined behavior, it is typically too expensive to be used during highly iterative model-based system design phase. In addition, it can be very hard (or nearly impossible) to test the correctness of the simulations (e.g., in terms of numerical convergence) to verify the states during benchmark tests. The cosimulation cost can be reduced proportional by reducing the most expensive simulation using a multi-abstraction approach [1]. This reduction in simulation time [1] is a *necessary* condition to simulation-based design optimization and design verification.

Behavioral research [19] shows that "experience" is gained through real-time experimentation. If action causes a direct result in a short time interval, then "experience" is gained. Actions that have long-term effects only result in "experience" for a small group of people. Likewise, if there is no interactivity (i.e., no opportunity to change the experiment/simulation) little experience is gained.

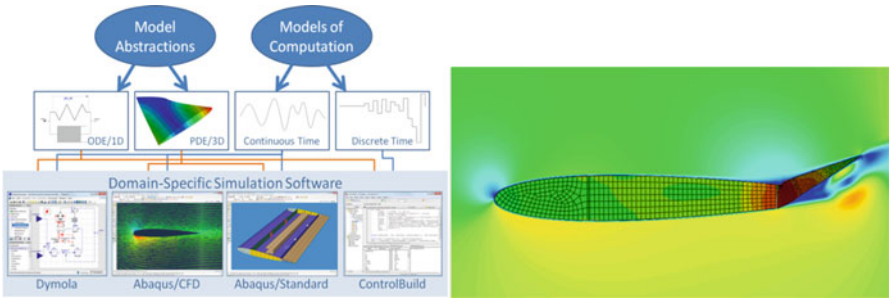

Fig. 23.1 N-code cosimulation of airfoil flutter (*left*). CFD flow field as predicted by Navier-Stokes for a partially separated flap during control system actuation (*right*) [1]

There are three existing approaches to reducing the computational burden and the solution complexity while maintaining solution accuracy over the domain of interest. The parameter estimation is probably the most widely used method. Pfeffer [7] defined the parameter estimation problem as follows: "Given a fixed structure for a dynamic model, an initial guess for the models parameters, and a set of frequency response data, find the parameters that make the model fit the frequency response data." The fitting operation is typically achieved by minimizing the errors between the frequency response data and the simulation data through optimization of the model parameters. A good example of the application of nonlinear parameter estimation on a problem similar to that of Fig. 23.1 is given by Klein [10]. A weakness of this approach is that the model form needs to be known a priori.

The second approach that is gaining interest is model reduction of nonlinear systems through the application of proper orthogonal decomposition by Karhunen [12] using Sirovich method [16] in combination with the trajectory piecewise linear [14] method to take into account nonlinearity. For chosen model states, linearizations of the reduced ODE model equations are stored along the solution trajectory. A benefit of this method is that it produces the same field information as the original high-fidelity simulation on which the reduced order model is based, whereas the parameter estimation method is limited to parametric data. However, this method does not allow for variations in the model as part of a scenario. In addition, the predicted values for individual sensor data (locations in the field) are not as accurate as they could be due to the inherent trade-off of minimizing the error for the entire field versus the error at discrete sensor locations.

However, if we are interested in creating fast running and accurate abstractions for design and verification, field data are typically not necessary. In model-based system engineering, behavior is only needed at a limited number of sensor locations. In addition, we want to leverage the utility of existing and widely used Design of Experiment (DOE) [15] and surrogate technology [13] to improve the capture of highly nonlinear systems and to study the impact of "design variables" in addition to dynamic states.

Such cosimulating surrogates can then be wrapped as a Functional Mock-up Units (FMUs) [2] and deployed in a system simulation tools such as well as in embedded hardware.

23.2 Present Method

Figure 23.2 shows a flowchart of the present method for authoring a surrogate for use in an interactive experience. The method begins by defining a simulation model representing a real-world system. The defined model includes system states and a design variable vector v. For example, the parametric state vector may include the position and angles of rotation (and their derivatives) of an object and the design variable vector may comprise the mass and the force (and its direction) on an object.

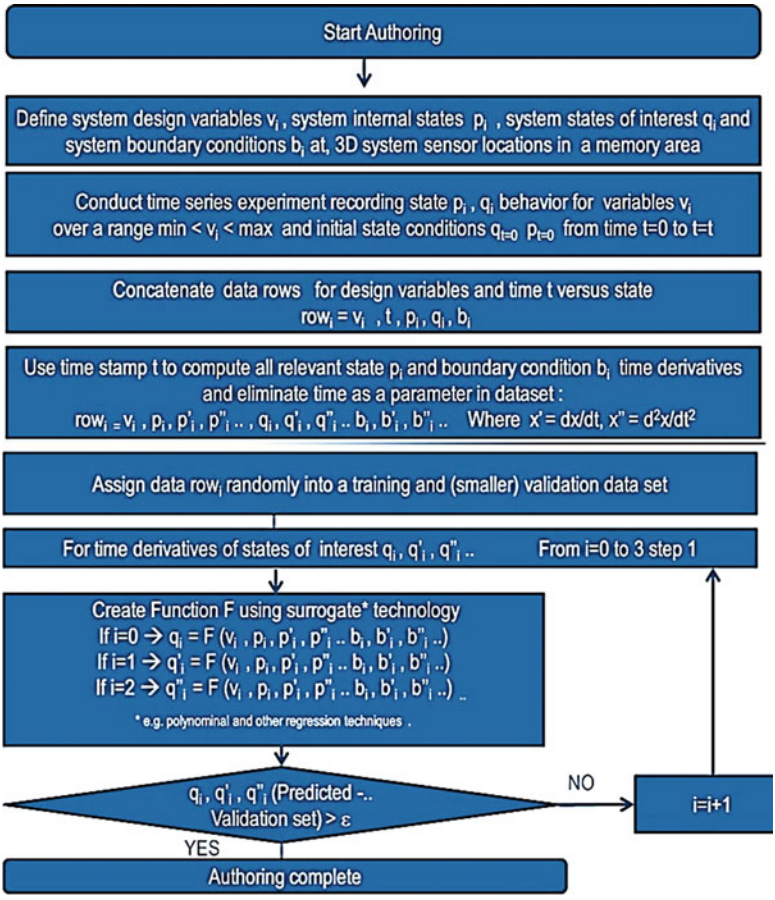

Fig. 23.2 Flowchart shows the present method of creating nonlinear ODE models for systems

The present method accelerates this realistic behavior modeling by first executing a physical or numerical high-fidelity experiment using a physical or numerical model and observing the response over time of the parametric state vector including internal states p and system states of interest q. This model can contain boundary conditions b such as discrete events (e.g., controller on/off) as well as externally forced conditions such as road excitations. Alternatively, before executing such an experiment, we may first reduce the dimensionality of the high-fidelity, high-dimensional numerical model using principal component analysis [6], or a similar technique [17], to a manageable number of parameters (e.g., approximately 100) before surrogation. This approach is repeated for instances of the design variable of interest v created by a Design of Experiment approach [15] to produce datasets as a function of (time, v, p, b, q) for a given set of initial conditions p(time = 0). The goal for the Design of Experiment technique is to produce evenly distributed samples in the hyperdimensional state and design space of interest. For the time-series, this is achieved in practice by using the initial state conditions as part of the DOE. However, it should be noted that this does not produce uniformly spaced training samples for the surrogate since trajectory traces are formed from each initial condition. Nevertheless, this approach is preferable to creating single (or excessively short) state time simulations, due to the simulation initialization cost. As such, there is a trade-off between the simulation initialization cost and the nonoptimality of long time-series simulations.

The method continues by concatenating the time-series datasets for each DOE and differentiating the variables with respect to time using a backward differentiation scheme and then removing time from the dataset, yielding (v, q,q',q"...,p,p', p'...', b,b'...). Then, the highest derivatives of the behavior of interest q is expressed a function of lower derivatives of q and the other vectors p, b, v and their derivatives. We can now leverage approximation techniques [13] such as radial basis functions, machine learning, Chebychev polynomials, and response surface methods with term reduction over the training set. The right predictive surrogate F is selected and its technique settings are optimized to reduce its variance and bias error and validated with the part of the dataset that is not used in the training.

The process is repeated for lower derivatives of the behavior of interest until an expression for F with the minimal sum of variance and bias error. When the error levels of the surrogate are similar for different derivatives of q, lower derivatives are preferred since they produce lower integration errors during execution.

Once the surrogates F are found for behavior q, it is now possible to numerical integrate q(t) from time = 0 to time = t for a given value v, given initial conditions p(time = 0) and variable boundary conditions b(t = t). These initial conditions of the state vector and design values are not limited to be the same as those of the original time-series experiments.

This model can in turn be exported to an FMU and be directly employed model-based systems to simulate realistic behavior in near real-time. Alternatively, it can be used directly in hardware, such as a flight simulator or an automotive electronic control unit (ECU).

23.2.1 Double Pendulum Model for Code Verification

The first example is principally for reproducible code verification. The equations of motion and their solution for the double pendulum can be found using Langrangian mechanics with generalized coordinates [18].

$$(m_1 + m_2)l_1\ddot{\theta} + m_2 l_2\ddot{\phi}\cos(\theta - \phi) + m_2 l_2\dot{\phi}^2 \sin(\theta - \phi) + (m_1 + m_2)g\sin(\theta) = 0$$
$$l_2\ddot{\phi} + l_1\ddot{\theta}\cos(\theta - \phi) - l_1\dot{\theta}\sin(\theta - \phi) + g\sin(\phi) = 0$$

This simple model, as shown in Fig. 23.3, exhibits extremely complex nonlinear behavior that is highly dependent on the initial conditions. For relatively small angles (<0.5 rad), an analytical solution can be found [18] as a function of initial conditions and time.

We execute a Latin hypercube DOE where θ and ϕ are varied from −0.5 to 0.5 rad, and $\dot{\theta}$ and $\dot{\phi}$ are varied from −1 to 1 1 rad/s, whereas the l's are varied from 0.8 to 1.2. Figure 23.3 shows the double pendulum and its 500 DOE samples in the $\theta-\phi$ plane. As a response, values of θ'' and ϕ'' are calculated.

We find that this sample can be approximated extremely well with Chebyshev polynominals in Isight™. Figure 23.3 (right) compares a trajectory based on the analytical solution to an approximated trajectory using a full fourth order (including cross-terms) Chebyhevs F_1 and F_2 each with 256 terms:

$$F_1(\theta, \phi, \dot{\theta}, \dot{\phi}, l_1, l_2) = \ddot{\theta}$$
$$F_1(\theta, \phi, \dot{\theta}, \dot{\phi}, l_1, l_2) = \ddot{\phi}$$

For large angles an analytical solution is not available. However, we can simulate the nonlinear equations of motion in Dymola™ and compare with the present method. We are interested in expanding the space to a solution where θ and ϕ can be varied from $-\pi$ to π and the l's can be varied from 0.1 to 3. The ranges of $\dot{\theta}$ and $\dot{\phi}$ are determined by the motion simulation. For this purpose, we generated 1000 DOE initial condition samples with 500 time-step time-series of the double

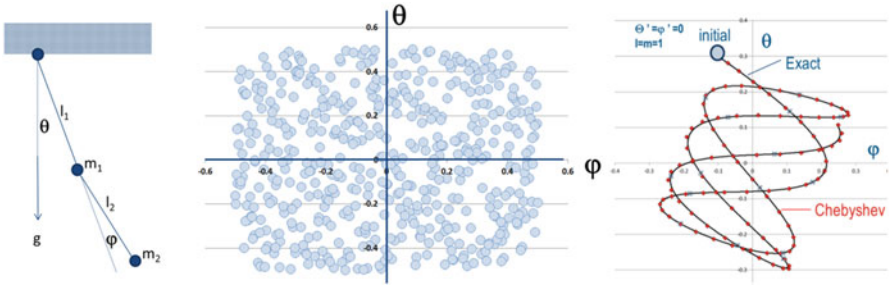

Fig. 23.3 Double Pendulum (*left*) and Latin hypercube samples (*center*) and analytical motion compared to present method

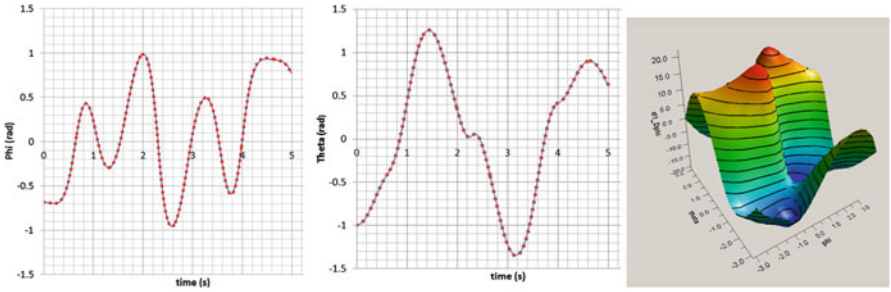

Fig. 23.4 Random validation trace of a large angle double pendulum simulation (*Red* = approximation, *Blue* = Dymola™) and partial depiction of state space F1

pendulum motion with Dymola™. This is more typical of the time-series samples we obtain from PDE-type simulations. However, this DOE produces a much less uniform distributed sample space, as the one shown in Fig. 23.3, and it is therefore much more challenging to reconstruct the state space from the samples. Also, very high accuracy is required for the forward integration and simple high polynomial models will fail on this problem. The problem was solved using a machine learning decision tree model with leaf regression. Figure 23.4 shows an example validation trace in the approximated space as well as a partial depiction of the state space. Note: The author can make this validation problem available to the reader for benchmarking.

23.2.2 Wing Flutter Control System

In this example, we start with a time-series produced by the nonlinear N-code simulation [1] described in the introduction. The cosimulation computed a 10 s (10,000 sample point) time-series in about 4 h. Figure 23.5 shows the cosimulation points in red. The data points are originally in the following form:

$$F_1(\alpha, \dot{\alpha}, z, \dot{z}) = \ddot{\alpha}$$
$$F_2(\alpha, \dot{\alpha}, z, \dot{z}) = \ddot{z}$$

where t = time, α= airfoil pitch, and z = airfoil vertical displacement (heave) on a wing section. In this first example, there is no controller or flap motion. The highest state derivative can be written as a function of the lower derivatives. Once we fit the 800 snapshot cosimulation points to a first-order response surface we create the following easy to understand linear ODE with 10 terms:

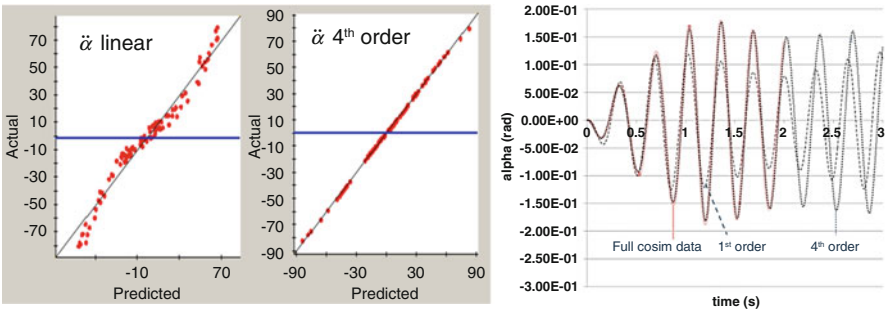

Fig. 23.5 Cross-Validation Error for the second time derivatives of α using a linear approximation (*left*) and a fourth-order approximation (*middle*). Time-series from Airfoil Flutter N-code simulation (*red*) approximated with a first- and fourth-order polynominal surrogate model (*right*)

$$\begin{bmatrix} \ddot{\alpha} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -1.4 \\ 0 \end{bmatrix} + \begin{bmatrix} -423.11 & -127.73 \\ 18.513 & -230.74 \end{bmatrix} \begin{bmatrix} \alpha \\ z \end{bmatrix} + \begin{bmatrix} 2.5370 & 26.374 \\ 0.19421 & -3.2897 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{z} \end{bmatrix}$$

By looking at the cross-validation errors of Fig. 23.5, we can see a priori that $\ddot{\alpha}$ is not well approximated. However, if we increase the polynominal to fourth order with all second-order cross-terms, we can achieve cross-validation errors below 1% with just 23 terms. The fourth-order approximation is obviously necessary to capture the flow separation physics shown in Fig. 23.1. The z-values are accurate even for the linear approximation. Using the ODE approximation, we can now time-step the solution as shown in Fig. 23.5 (right). The linear ODE approximation obviously does not capture the physics very well as expected, while the fourth-order approximation is right on target.

We now add samples (for different initial conditions) with a 20, 25, 33, 50, and 100 Hz digital controller actuating the flap. Figure 23.6 shows a trajectory for different initial conditions and controller bandwidth (42 Hz) that were not covered by the original samples. Again, over time some state error builds up due to numerical integration (euler time-steps) and approximation, which leads to an offset between the time-series. When the controller is turned on at 3.1 s the full cosimulation creates a smooth transition to very high pitch accelerations $\ddot{\alpha}$ (-120 rad/s^2) due to flap deflection. The RBF surrogate jumps from controller off (0 Hz) to 42 Hz in a single time-step. From that point on, the acceleration predicted by the surrogate is smooth, whereas the full cosimulation with digital controller has high-frequency discrete corrections to the pitch acceleration.

23.2.3 Vehicle Road Excitation

For the final example we will apply the present method to motion of a full Abaqus™ 3D car model excited by a four-post rig to simulate road conditions. The variable

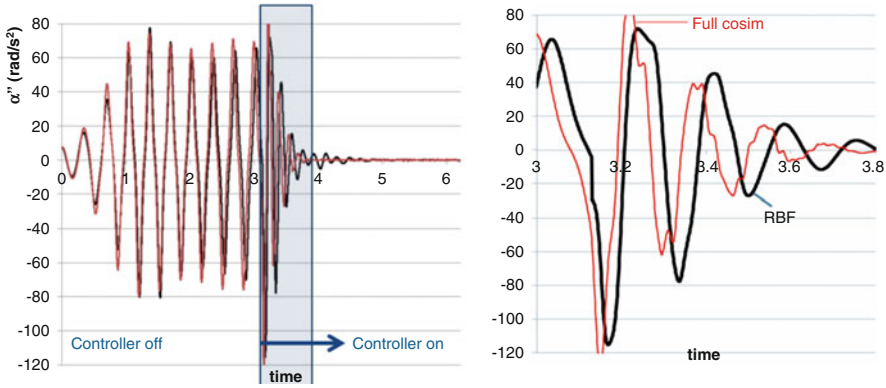

Fig. 23.6 Time-series from Airfoil Flutter N-code simulation (*red*) approximated with a fourth-order RSM with controller off and RBF with controller on at 3.1 s and 42 Hz. Figure 23.6 on the left on right represents the zoomed box of figure on *left*

road geometry is modeled as boundary conditions over time. Sensors, as shown in Fig. 23.7, are located on the 3D car model. The suspension system connects the wheels through the front rackhouse, so the wheels do not move independently on a bumpy road. The main conduit for the suspension forces to the vehicle structure are the damper tops. As shown below, the force at the top left damper is measured by a sensor in the simulation.

The challenge with this model is the very long simulation time. The body-in-white has a relatively inexpensive implicit FEA, but the realistic tire models are simulated with explicit FEA. A single 40 s simulation consists of 8000 time-steps, takes over 24 h on 32 CPUs, while recording over 80 states as a function of time. As a consequence, only one time-series simulation was made for this paper, which meant that we could not investigate the impact of design variables as we did in the previous example.

Figure 23.7 shows the trace of the rotational acceleration of the center of gravity as the car drives over different simulated road surfaces. The trace was divided into three pieces. First, the training piece of 10–30 s, next a secondary training piece from 30 to 40 s. The primary training time-series was used to find the approximation form. The secondary training piece was used to select its options by computing the error between the model based on the primary training set and the secondary training set data. The behavior in the first 10 s (as shown in Fig. 23.7) seems to be different from the rest of the time trace. We will use this time segment for validation.

This surrogate was then used to predict the sensor state time-series from 1 to 10 s on the basis of initial conditions on road boundary conditions alone. The author found that the response surface method (polynomial fit) with strong Efroymson term reduction [20] produced the best results on the secondary training set. The high number of input parameters resulted in over 1000 possible terms for a full second-order model. The R-values for this primary training set alone were typically 1.00,

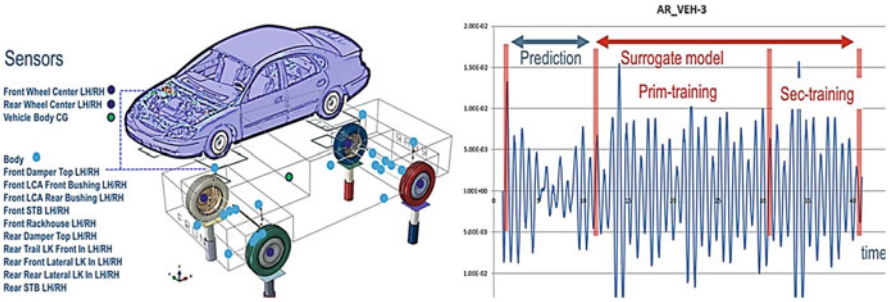

Fig. 23.7 Full Abaqus vehicle model with sensor locations. Dashed line is the location of the top front damper force sensor (*left*). Simulated time-series of the rotational acceleration of the center of gravity (*left*)

suggesting an excellent fit. However, the comparison to the secondary training set was typically poor due to variance error. This is probably because we do not have enough points in the primary training set. We were able to reduce the variance error by reducing the allowable maximum number of terms in the term selection. This provided significant (order of magnitude) additional benefits over the statistical method of [20] based on the primary training set alone.

It was found that first-order nonlinear ODEs best approximated the motion of the vehicle. The author is not entirely sure as to why this is, but it is probably because the damping forces dominate the problem and dropping some of the second-order effects improved the variance errors. Figure 23.8 shows the comparison of the model prediction versus the set-aside points from the simulation trace. There is drift due to cascading integration and approximation errors, but in general the validation is good.

The accuracy of this cosimulating surrogate model is good enough for durability studies and system suspension tuning. Figure 23.9 (left) shows the fatigue life of the body structure near the top damper due to the full FEA simulation and Fig. 23.9 (right) shows the fatigue life as determined cosimulating surrogate model of the top damper force. The full FEA model for the tire, suspension, and body is very expensive (1 h per simulated second) to simulate for arbitrary roads, whereas the reduced order model of the body FEA and the cosimulating surrogate model can be computed near real time (1 s per simulated second).

23.3 Conclusion

In this paper we propose a new method to create (co)simulation surrogates for highly nonlinear systems with active design variables that can be used during the model-based system design and verification phase. This method was designed to be

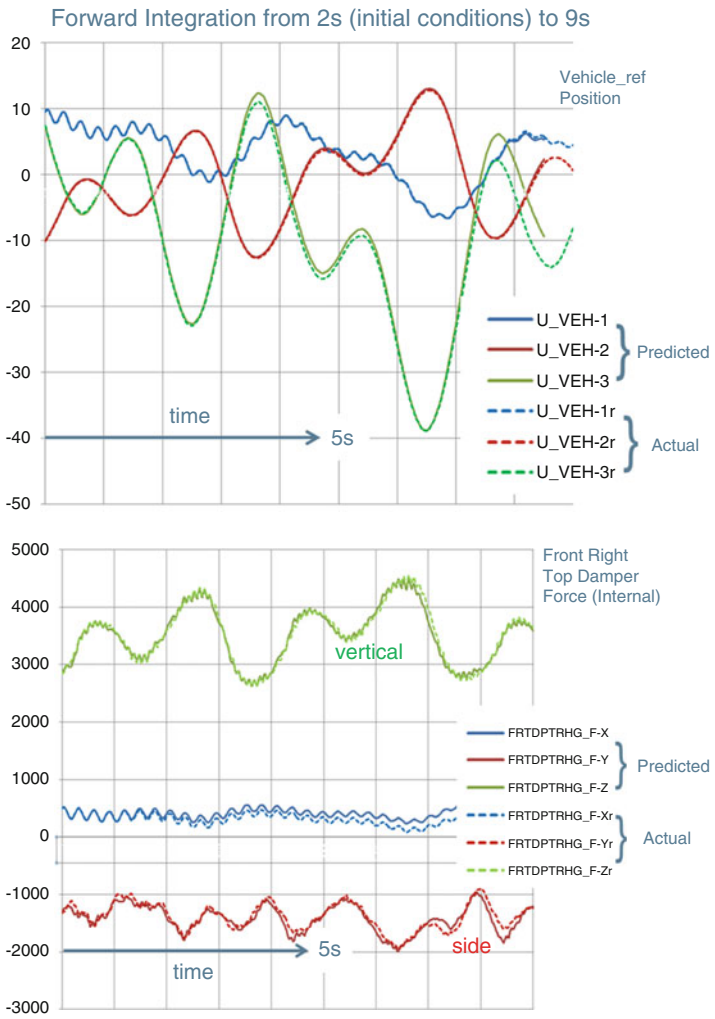

Fig. 23.8 Prediction of the vehicle motion (*top*) and vehicle internal damper force (*bottom*) for the simulation time-series of Fig. 23.7 from 2 to 9 s

broadly applicable to any simulation, but we focused on applying the method to mechanical simulations in this paper.

We used a combination of classical DOE [15] and Sirovich snapshot samples [16] to create nonlinear surrogates representing nonlinear ordinary differential equations for design variables and state variables. The ability to have choice [13] in the type of approximation seems critical to achieve the right approximation accuracy in the nonlinear region. Typically, the average error of the surrogate should not exceed 0.1%. to avoid instability in the numerical integration.

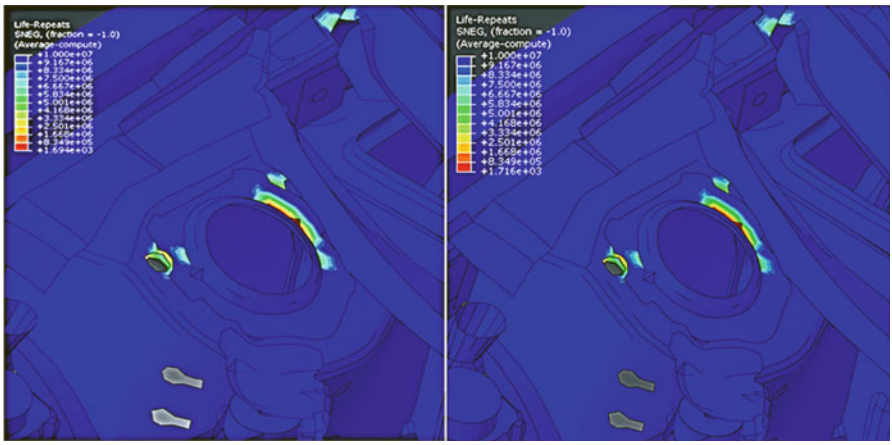

Fig. 23.9 Fatigue life of top damper structure using full FEA (*left*). Fatigue life of top damper structure using cosimulating surrogate model of top damper forces (*right*)

It should also be noted that sometimes we found that the inability to create a surrogate was due to the fact that the original simulation had model errors. Once these model errors were removed, we were able to successfully create a surrogate. As such, the surrogate approach can be an additional verification of the underlying data. This method was successfully applied on the practical example of a wing flutter control system and a vehicle NVH/durability model. In these cases, speedups of three orders of magnitude were achieved over the full-fidelity FEA models with little loss in predictive accuracy.

Looking forward, we expect that the present method surrogates can be combined with validated first principles models in order to fill gaps in modeling knowledge rather than using it to build the entire model.

Acknowledgments The author would like to thank Jeff Haan for generating the sample airfoil flutter sample data, as well as Fig. 23.1; and Youngwon Hahn for generating the sample data on the automotive NVH case, as well as providing Figs. 23.7 and 23.9.

References

1. Van der Velden A, Koch P, Devanathan S, Haan J, Naehring D, Fox D (2012) Probabilistic certificate of correctness for cyber physical systems. In: ASME 2012 international design engineering technical conferences & computers and information in engineering conference, August 12–15, Chicago, DETC2012-70135
2. Functional Mock-up Interface. http://www.functional-mockup-interface.org/index.html
3. Hardy RL (1971) Multiquadratic equations of topography and other irregular surfaces. J Geophys Res 76:1905–1915
4. Broomhead DS, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355

5. Kansa EJ (1999) Motivation for using radial basis functions to solve PDE's (unpublished: author kansa@IrisINTERNET.net)
6. Jolliffe IT (2002) Principal component analysis, series: Springer series in statistics, 2nd edn. Springer, New York
7. Pfeffer LE (1993) The RPM toolbox: a system for fitting linear model to frequency response data. In: 1993 Matlab Conference, October 18–20, Cambridge, MA
8. Dyson F (2004) Turning Points a meeting with Enrico Fermi. Nature 427:297. doi:10.1038/427297a
9. Adcock J (1987) Analyzer synthesizes frequency response of linear systems. Hewlett-Packard-J, January.
10. Klein V (1998) Aerodynamic parameters of high performance aircraft estimated from wind tunnel and flight test data, NASA-98-AGARD
11. Herkt S (2008) Model reduction of nonlinear problems in structural mechanics. PhD thesis, University Kaiserslautern. Heers B, Chin C, Kim M, Belsky V (2013) Computational advances in substructure generation and their implication of system-level analysis. NAFEMS World Congress
12. Berkooz G, Holmes P, Lumley JL (1996) Turbulence, coherent structures, dynamical systems and symmetry, Cambridge monographs on mechanics. Cambridge University Press, Cambridge
13. Devanathan S, Koch P (2011) Comparison of meta-modeling approaches for optimization. In: Proceedings of the ASME 2011 International Mechanical Engineering Congress IMECE2011-65541
14. Rewienski M (2003) A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems. PhD thesis, MIT
15. Box GE, Hunter JS, Hunter WG (2005) Statistics for experimenters: design, innovation, and discovery, 2nd edn. Wiley. ISBN 0-471-71813-0. Taguchi, G (1995) Quality engineering (Taguchi methods) for the development of electronic circuit technology. IEEE Trans Reliab (IEEE Reliab Soc) 44(2):225–229. doi:10.1109/24.387375. ISSN 0018-9529
16. Sirovich L (1987) Turbulence and the dynamics of coherent structures I–III. Quart Appl Math 45(3):561–590
17. Jagenberg T (2008) Dimensional analysis in data approximation. Masters thesis, Institut for Flugzeugbau University of Stuttgart
18. Double Pendulum (2013) http://www.math24.net
19. Kahneman D (2011) Thinking fast and slow. fsgbooks
20. Efroymson MA (1960) Multiple regression analysis. In: Ralston A, Wilf HS (eds) Mathematical methods for digital computers. Wiley, New York